Problem Statement

Climbing Mont Blanc and Scalability

Climbing Mont Blanc (CMB) is a system for evaluation of programs executed on modern heterogeneous multi-cores such as the Exynos Octa chips used in, e.g., Samsung Galaxy S5 and S6 mobile phones, see `https://www.ntnu.edu/idi/card/cmb`. CMB evaluates both performance and energy efficiency and provides the possibility of performance ranking lists and online competitions. A first version of the system is available and under trial use. This master thesis project is focused on improving the system with increased scalability so that the system can handle more user submissions per hour.

The project involves the following subtasks:

1. Study the existing solution in CMB for automatic system monitoring and recovery and suggest improvements.
2. Describe and implement a dispatcher in the current CMB system that will allow the use of multiple XU3 backends to serve concurrent user submissions.
3. Test the dispatcher with two or three XU3 boards. Implement a simple script for generating a synthetic load (simulating users and submission) to be able to evaluate the scalability of a new CMB variant using the developed dispatcher.
4. Describe how the dispatcher can be used to allow different boards than the XU3 to be used together with XU3 boards. Discuss what effects this will have on other parts of the system.

If time permits:

5. Propose possible throttling techniques for cases where CMB gets too many/frequent submissions from a single user, or in total from all active users.
6. Propose compilation/Makefile improvements, and the possibility of a makefile that can be edited by problem-setter through admin-interface.
7. Propose how more detailed statistics such as performance counter values can be given as feedback to the CMB user.
8. Propose the addition of more programming languages and libraries.
9. Propose an extension of the dispatcher into a load-balancing broker.
10. Implement some of the proposed solutions after approval by, and in collaboration with the CMB team.

The master thesis project is part of the EECS Strategic Research project at IME (`www.ntnu.edu/ime/eecs`).

Supervisor: Prof. Lasse Natvig

Abstract

This thesis details a proposed system implementation upgrade for the CMB system, accessible at `climb.idi.ntnu.no`, which profiles C/C++ code for its energy efficiency on an Odroid-XU3 board, which utilises a Samsung Exynos 5 Octa CPU, and has an ARM Mali-T628 GPU. Our proposed system implementation improves the robustness of the code base and its execution, in addition to permitting an increased throughput of submissions profiled by the system with the implementation's dispatcher which allows the system to utilise several Odroid-XU3 backends for the energy and timing measurement profiling. Our tests show that our implementation can achieve more than 4x speedup with "Hello World" submissions using a parallelized web server, and around 2x speedup with "Shortest Path" submissions using a serial web server.

Preface

I have many I want to acknowledge for all their help, support, and comradeship during my time in academia (not just for the duration of this master project, at the end of the line), but individuals of note include the following:

- Lasse Natvig, my supervisor for this Master project, for his patience, wisdom, and guidance through the course of this project.
- IDI's Technical group, especially Arne Dag Fidjestøl, Jan Grønsberg, and Erik Houmb, for their support and tips in the development of this project's proposed system implementation.
- Sindre Magnussen, for his help in understanding and learning the workings of the CMB system.
- Dag Frode Solberg, Christoffer Viken, (and the rest of the crowd from NTNU's PVV), for all those hours, spent as my rubber ducks and coming with tips and guidance when my technical competence was insufficient.
- Finn Inderhaug Holme, for saving my bacon when the power connections of the equipment in Trondheim had to be disconnected and reconnected after I had moved to Oslo due to the delays incurred during this project.
- My family for their support and love throughout.
- And anyone else whose notable assistance may have (temporarily!) been forgotten during the time of this writing.

Note: This report is rather long and was not condensed down as is the norm, due to the time constraints described on the next page. However, I want to make it clear that if you care about trees, you should not print this report in its entirety. Chapters 5, 6, and Appendixes A, C, D, are all rather long, and especially boring to read on paper.

The main challenge faced during this project

At the outset of this 21-week, contractually decided period allotted for this master project, Celery was chosen by the author of this project with the support of the supervisor, Lasse Natvig, to be a promising and efficient way to solve a majority of the challenges of this project.

However, on the 15th of March, in a meeting with a member of the institutes's (IDI's) Technical Group (Arne Dag Fidjestøl), and the other master student currently working on the CMB project (Sindre Magnussen), it was concluded that Celery, while fit for the task, was introducing more complexity into the system, than what was currently needed. This conclusion was based on the fact that the earlier project future use estimates of the CMB system were too ambitious, and thus the CMB system did not need to support so potentially high frequencies and concurrently submitted submissions by its user base.

Therefore, in week 10 of this project (start of the project's 21-week allotted time was the 11th of January), it was decided that a proposed implementation based upon the use of Celery, would be of little use to future iterations of the CMB system, at this time. Thus, the author of this project has reversed all efforts of implementing the use of Celery into CMB and has instead landed upon (and implemented) the proposed system implementation described in Chapter 4.

In addition to compensation for a three week documented sick-leave, this project has received an extension of two additional weeks to compensate for this setback. On top of all that, the author of this paper had to move residence from Trondheim to Oslo in the last four weeks of this project, as the move had been planned and scheduled before the delays happened and the compensation was given.

The author of this paper has had to omit goals and desired implementations/improvements of this project (and paper research), due to this time-constraining setback, but wants to state that had this project been granted 4-6 more weeks, the complete (or near-complete) implementation of both the database, and automatic system monitoring and recovery (described in Chapters 4 and 9) may well have been realized.

Abbreviations and Glossary

ARM	=	Advanced RISC Machine (`http://www.arm.com/`)
BSC	=	Barcelona Supercomputing Center
CMB	=	Climbing Mont Blanc `https://climb.idi.ntnu.no/`
HPC	=	High-Performance Computing
MB	=	Mont-Blanc (The EU Project: `https://www.montblanc-project.eu/`)
SoC	=	System(s)-on-Chip
VM	=	Virtual Machine

Table of Contents

List of Tables

List of Figures

List of Listings

Chapter 1

Introduction

This chapter will first explain the Motivation for this Thesis in Section 1.1, before detailing the Project Goals of the Master Project in Section 1.2. Finally, Section 1.3 lists the structure of this Thesis.

1.1 Motivation

With supercomputing clusters (colloquially known as *High-Performance Computing* (HPC) centres) being a viable, albeit prohibitively expensive, alternative for computationally intensive workloads, there is a lot of money invested in HPC. However, while initial costs for HPC centres are often staggering, they also tend to accrue an equally staggering cost in electrical bills, just for keeping the system online and running (Subramaniam and Feng, 2010).

This has prompted the HPC community to search for new, more "energy efficient" solutions. The search for more architectures viable for HPC has spawned many efforts, such as the *Mont-Blanc* (MB) EU project (Rajovic et al., 2013).

The MB project aims to design a new type of computer architecture capable of setting future HPC standards worldwide, built from energy efficient solutions used in embedded and mobile devices. The MB EU project started this quest with Rajovic et al. (2013), aiming to answer the question of whether mobile, ARM-based, *System(s)-on-Chip* (SoC) can help reduce the cost of HPC, due to their proliferate abundance in embedded devices such as smartphones. The MB project base this question on the premise that the *x86* architecture still dominated the TOP500 list of supercomputers in the world in June 2013.

During the literature research for this project, it seems that hardware is the primary platform on which the focus of energy efficiency currently resides.

However, while there is a considerable amount of effort put into finding the next generation energy efficient hardware, software will always remain an important "half" of energy efficient HPC. Energy efficient software is simply software which, while competitively achieving the same results as the "traditional" software, also consumes less energy during execution.

The *Climbing Mont Blanc* (CMB) project (Støa and Follan, 2015), based on the MB project, attempts to aid the search for energy efficient software. Støa and Follan (2015), along with Natvig et al. (2015), built a system for measuring the energy efficiency of code, which runs on an ARM-based architecture also used in Samsung Exynos (smartphone) processors. This system permits the user to upload their code via a web-browser user interface[1], have their code run on an *Odroid-XU3* development board, and have the *timing* and *energy consumption* readings returned to them upon successful execution.

The plans for CMB, going forward (Magnussen, 2015), include:

- To handle a larger user base:
  - Support a higher *frequency* of submissions.
  - Support a higher *concurrency* of submissions.

Currently, CMB consists of one web server, which utilises a single Odroid-XU3 card as a backend, to execute all submitted code profilings, one at a time. While there exist a plethora of web technologies to enable better load-balancing of web servers (such as Gunicorn (2010)), we are currently not aware of a technology which would permit us with relative ease and reliability load-balance the code profilings/executions on multiple Odroid-XU3 cards.

Technologies like Vagrant (Gajda, 2015) and Docker (Merkel, 2014) might have been of use. However, without extensive testing, we doubt the reliability of the energy consumption readings of a system implementing these technologies, due to the added complexities/overhead incurred by either of these.

1.2 Project Goals

In this section, we summarise the goals we have set for this project, based upon the tasks listed in the Problem Statement, and inform the reader where in the report the detailing of their realisations are located.

[1] `https://climb.idi.ntnu.no`

1.2.1 Automatic System Monitoring and Recovery

With the Motivation from Section 1.1 in mind, the Problem Statement lists four subtasks, and an additional six subtasks if time permits. Of the four mandatory subtasks, only the first one does not mention/involve a dispatcher. Said subtask says "Study the existing solution in CMB for automatic system monitoring and recovery and suggest improvements".

Currently, the system relies on the administrators with access to the server and backend to log in and manually restart any component(s) that crash/go down. Thus, a goal of this project is to simplify the start-up process, such that a service like Upstart (Upstart, 2006) or Systemd (Poettering et al., 2010) may monitor the processes of the system, and restart them as required.

The creation/implementation of an automatic monitorization and recovery system through the use of Systemd or Upstart was not accomplished in the allotted time. The reason for this is elaborated in the Preface, while the efforts made are described in Section 4.2 and a potential solution is described in Section 9.1.

1.2.2 The Dispatcher

Subtasks 2-4 from the Problem Statement either specify the creation/addition of a "dispatcher" to CMB, or rely on an existing one. The job of the dispatcher is to dispatch code submissions users have uploaded to CMB, to the backends which perform the code execution/profiling. And hence the creation/addition of such a dispatcher in the CMB codebase is one of the goals we have set for this project.

While Celery (Solem, 2009) at first was considered to be an apt solution for parallelizing CMB's backend[2], it was decided by the people behind CMB in week 10 of this contractually allotted 21-week project that introducing Celery into CMB would introduce too much complexity into the system[3].

Thus, the parallelization of the backends in the proposed system solution is realised through extending the REST API of the server, and having the backends added polling this API every so often. The realisation of this feature is further detailed in Section 4.5.

1.3 Thesis Structure

With the very technical emphasis of this master project, the structure of this report stands a little out from the perceived norm. In this report, we focus on presenting a

[2] Hence, a non-negligible amount of time and effort was spent in the attempt of implementing the use of Celery in CMB, in the first half of this project. See the Preface for more information.

[3] Celery and its complexities are detailed in Section 2.4.

robust, dependable proposed system implementation, with which we hope to fulfil as many goals set in the Problem Statement and in Section 1.2 as possible.

With this in mind, we first introduce the CMB project, and a bit about its history in Chapter 2, where we also introduce and describe some of the tools/implementations considered during this project.

Following, we continue with Chapter 3, where we list related works we were able to find, for the CMB system, and the parallelization project, at the heart of this master project.

After that, we introduce and detail our proposed system solution/implementation in Chapter 4, and describe the proposed implementation changes, and their benefits to the CMB system.

What follows in Chapters 5 and 6, are detailed and technical install instructions, for the proposed system implementation described in Chapter 4. The reason for not having these two chapters as appendixes is that we consider them to be relevant[4] to much of the report, besides the aforementioned fact that this is a very technical report, for a very technical master project. (Compared to how the norm of master projects often involve proving/disproving the veracity of newer, novel ideas, the efforts of this project are based on already proven laws of software parallelization).

We then list the methodology of our tests in Chapter 7, where we also state the hypotheses of our project, before reporting (and discussing) the results of said tests in Chapter 8.

Finally, we detail what improvements we were unable to complete, due to the time constraints described in the Preface, in addition to our thoughts on how the CMB system could further be improved by any future CMB system developers in Chapter 9, before we conclude in Chapter 10.

[4] And they're referenced repeatedly throughout the report.

Chapter 2

Background

This chapter starts with a summary of both the EU Mont-Blanc project in Section 2.1, and the Climbing Mont Blanc system in Section 2.2. A detailing of the Odroid-XU3 hardware used as backend follows in Section 2.3. Section 2.4 details software alternatives considered for the implementation of the dispatcher introduced in Section 1.2.

2.1 Mont-Blanc, The EU Project

The *Barcelona Supercomputing Center* (BSC) coordinates the Mont-Blanc project (BSC, 2011), which, since October 2011, has had the aim to design a new type of computer architecture capable of setting future global HPC standards, built from energy efficient solutions used in embedded and mobile devices. Their long-term goal is to provide Exascale performance using 15 to 30 times less energy than current architectures.

In 2013, phases 1 and 2 of the MB project were given a budget of 22 million €, of which 16 million € were granted by the European Commission. The time extension provided by the final 8 million € from the European Commission in 2013, permitted BSC to extend Mont-Blanc project activities until September 2016.

The third phase of the MB project (coordinated by Bull (2016), the Atos brand for technology products and software), started in October 2015 got funded by the European Commission under the Horizon 2020 programme. Its aim is to design a new high-end HPC platform that can deliver a new level of performance/energy ratio when executing real applications.

2.2 The Climbing Mont Blanc System

In 2012, the *Faculty of Information Technology, Mathematics and Engineering* (IME) at the *Norwegian University of Science and Technology* (NTNU) had the *Energy Efficient Computing Systems*[1](EECS) Strategic Research Area projects running. *Lasse Natvig* (from IME, NTNU), proposed at HiPEAC3 2012, in Gothenburg Sweden, that the masses of young students and programmers could be utilised in the quest for knowledge wrt. energy efficient computing.

In the fall of 2014, Simen Støa and Torbjørn Follan began the development of *Climbing Mont Blanc* (CMB), under the supervision of Lasse. CMB (Støa and Follan, 2015), is a system with a web-frontend which permits a user to upload code to be executed and profiled for time and energy consumption on an Odroid-XU3 development board. Since January 2015, each school semester at NTNU has had one or more subjects/activities utilising (and some relying), on the CMB system for competitions and/or homework.

CMB utilises a Python Flask (Ronacher, 2010) web server, with an added JS frontend built with AngularJS (Green and Seshadri, 2013), which serves web browsers the user-interface of `https://climb.idi.ntnu.no`. The Python-Flask server, running on an Ubuntu 14.04 LTS Linux OS, is a REST API, which utilises an SQL database for its data, in addition to one Odroid-XU3[2] development board for profiling/executing uploaded code submissions. See Figure 2.1 for a graphical overview of CMB's system/architecture.

Thus, the CMB system permits:

1. The creation of User accounts.

2. The creation of Administrator accounts which can create problems to which Users can upload/submit code to in attempts to solve.

3. Administrator accounts to view all submissions made by Users on the system.

4. A ranking system based on the timing/energy consumption of submitted code, per problem, available for all to see (global).

5. Users to belong to groups, and the groups may have individual ranking lists (private).

[1] `http://www.ntnu.edu/ime/eecs`
[2] For more information, see Section 2.3.

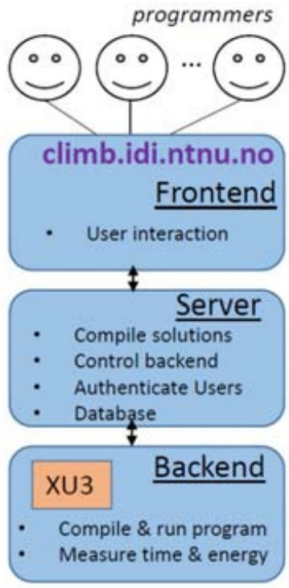


Figure 2.1: CMB System Architecture. Source: Natvig et al. (2015)

2.3 Odroid-XU3

The Odroid-XU3 (www.hardkernel.com, 2016) currently serves as the backend of the CMB system. (Støa and Follan, 2015) report that it has a Samsung Exynos 5 Octa (5422) chip, which has four ARM Cortex-A15 and four Cortex-A7 cores, making it a heterogeneous multi-processing platform using ARM big.LITTLE technology. The ARM big.LITTLE technology reportedly enables seamless and automatic movement of workloads to appropriate CPU cores based on performance needs.

(Støa and Follan, 2015) also report that the Odroid-XU3 has an ARM Mali-T628 GPU, which supports OpenGL ES 3.0/2.0/1.1 and OpenCL 1.1.

Additional details of the board can be found in (Støa and Follan, 2015) and (www.hardkernel.com, 2016).

2.4 Concurrency Softwares Considered

This Section describes the different software frameworks/packages considered when realising the Project Goals(listed in Section 1.2) of this project.

The different software/frameworks were considered with the following points in mind:

1. How would it enable concurrency among the backends?
2. How would/could it support differentiation between the backends wrt. to factors like hardware architecture and/or installed/available programming languages/libraries?
3. How stable (reputed stability/usage) does the concurrency technology/implementation seem to be?
4. How supported does any utilised (implemented) software seem to be, by its developers? How reliable does the software's future support appear?

As such, ZeroMQ and Celery were the only alternatives found and seriously considered, with the above points in mind.

2.4.1 ZeroMQ

ZeroMQ (ZeroMQ, 2011) is a distributed messaging framework, which allows you to implement your own messaging infrastructure. There are many different usage examples/implementations to be found, but it was quickly decided that we would rather look for an alternative which required less implementation effort.

The reason why a messaging infrastructure would be needed is that at the outset of this project, there was no two-way communication going between the CMB server and the attached backend. All interactions between the two were Bash scripts executed on the one, which in turn executed another Bash script through an SSH tunnel on the other.

If at any point during the executions of the Bash failed, crashed, or got stuck; the CMB system often got so unstable that it had at best to be restarted, at worst debugged, before it could continue to operate.

In parallel with this master project, Sindre Magnussen continues with his work on the frontend from the fall of 2015 (Magnussen, 2015). In his master project, running concurrently with the master project of this report, he has implemented the use of SocketIO (Rai, 2013) into his development Git branch of the CMB system. As such, any future combining of the efforts of his master project and this one, discussed in Section 9.4, could perhaps capitalise on this, if needed.

ZeroMQ have several protocols which may have suited CMB (such as the Majordomo protocol[3]), but again, it would require more effort than we were interested in spending to utilise it in CMB.

The strength of ZeroMQ lies in its versatility and seems to be used widely enough (and sufficiently supported) to be a candidate for CMB. However, this versatility comes at the cost of having to implement our own messaging infrastructure.

[3] `http://rfc.zeromq.org/spec:7`.

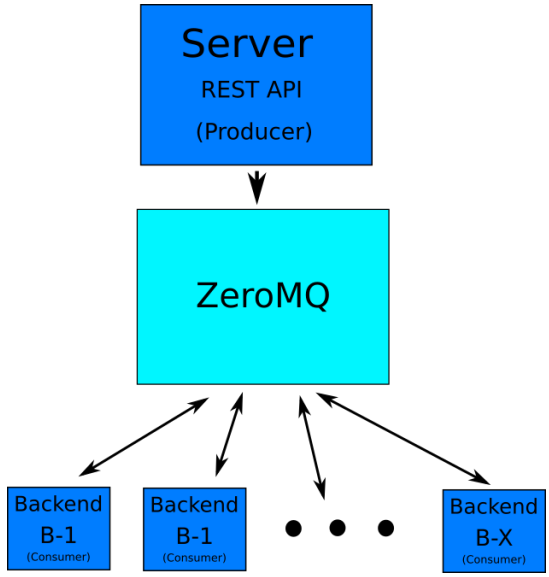


Figure 2.2: An illustration of how ZeroMQ could be utilized.

2.4.2 Celery

Celery is a Python framework offering distributed task queues. These task queues get tasks submitted to them by *producer(s)*. Tasks submitted to queue(s) will be executed by *consumer(s)* listening to said queue(s). By default, a task submitted to a queue will only be executed *once* by *one consumer* listening to said queue.

Celery was during the first ten weeks of this project assumed to represent the best course of action for implementing the concurrency goals of this project.

Celery works by having functions assigned to the Celery framework trigger the creation of tasks to be queued and executed by the aforementioned queued and consumers, respectively. The way Celery uniquely identifies functions is by using the Python task (function) signatures, which are the result of the absolute import package path, and the task's (function's) function definition. The Python PATH is where these signatures are defined.

As a consequence, identical task signatures must be present in both the producer and consumer, and the developer must be aware of any differences in the body of the function to which the task signature corresponds. Thus, the use of Git makes it straightforward, to have multiple copies of the same code base (and thus identical task signatures) on different machines.

Thus, when a producer submits a task to a queue to which one or more consumers are

listening, Celery will (with its default set-up) ensure that the task is only executed once by one consumer. If the task is submitted to multiple queues, it will be executed once per queue, perhaps even by the same consumer, if said consumer is listening to the relevant queues.

If so desired, a task can also be submitted multiple times to one queue. Celery ensures that each tasks-submission in the queue gets uniquely identified by the relevant systems, and with default settings, each task-submission in the queue will still only be executed *once* by a consumer listening to said queue.

Figure 2.3 illustrates how a CMB implementation using Celery would rely on the different components which gives Celery its complexity, e.g. RabbitMQ.

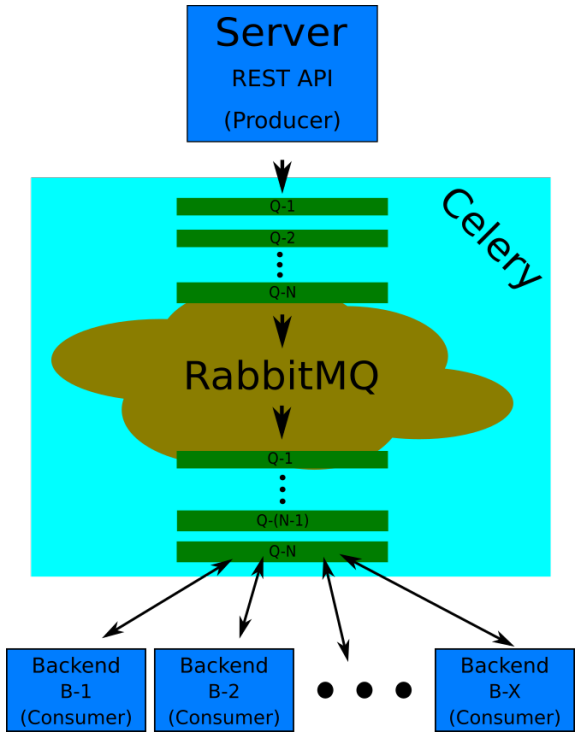


Figure 2.3: An example of how CMB could utilize Celery.

However, Celery needs a message broker (which ZeroMQ could have been implemented as) on which to create/maintain its queues and their tasks. Celery itself supports several different brokers[4] and backends (backends used for the results of the tasks executed).

As the reader can see from Figure 2.3, Celery would encompass, and ensure the functioning

[4] `http://docs.celeryproject.org/en/latest/getting-started/brokers/`.

of, everything within the turquoise square. RabbitMQ would be the *message broker*, which Celery would rely upon and utilise to realise the functionality described with its consumers, producers, and queues.

So while ZeroMQ offers neither the framework of consumers, producers, and queues (nor the transport/message layer of RabbitMQ), it instead offers the flexibility of more "open" slate.

As such, Celery gives for free that which ZeroMQ does not, automatic dispatching which ensures a submission is only executed once by a target backend (through the use of queues), in an already existing and documented Python framework, in use by several industry giants, like Opera, Google, and Facebook.

Chapter 3

Related Work

In this chapter, we list the related work we were able to find in Section 3.2, but first we give a bit more background on what CMB is can (and perhaps should?) be considered as, and what else like it there is in the world.

3.1 Online Judges - Websites for Competitive Programming

Crowdsourcing can be defined as, "*the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call*" (http://www.crowdsourcing.com/, 2016).

With that definition in mind, the case that CMB represents a system enabling the crowdsourcing for more energy efficient software can be made. However, CMB is not the only system or platform to which program submissions can be sent/uploaded, and evaluated.

Due to the time constraint detailed in the Preface, we were unable to complete the research for related work of Online Judges (OJs) to a satisfactory extent. However, one of the previous projects on the CMB system (Magnussen, 2015) references multiple other Online Judges which have varying degrees of popularity. Of these, the CMB system itself can be said to have been (at least in part) inspired by the likes of Kattis (Kattis, 2016), which is yearly used by the International Collegiate Programming Contest (ICPC).

However, like most of the others referenced by (Magnussen, 2015), Kattis focuses on the timing efficiency of a program submission and pay little to no regards for the energy consumption of the submission.

Kattis, and many of the other OJs that are out there (including CMB) work very similarly, with a straight-forward process:

1. Create a set of problems, to which program submissions can be submitted to the system to solve.
2. Have a system through which teams or individuals can upload their submissions, and keep track of which submissions was uploaded by whom.
3. Have the OJ be able to measure each submission (consumption of time/energy/memory/something else), and store the measurements with a relationship connecting the measurement to its submission.
4. Additionally, most, if not all OJs include the following:
    - The ability to show results on a scoreboard, publicly or otherwise available for judges/contestants.
    - Deadlines within which submissions must be uploaded, so that the measurements of the submissions will be valid for any potential competition/scoreboard.

3.2 Backend Parallelization Projects

As previously stated in Section 3.1, in the time remaining for this project, we were unable to find references to other backend parallelization projects in computer science literature.

However, it is the opinion of the author of this report that there may not be that much publicly available out there, even for scholars looking searching through literature behind pay-walls.

The reasoning behind this is two-fold. First, it is the belief of this author that with current attitudes of not allowing potential competitors, nor anyone who might represent a security risk, gain insight into the workings of backends of most complicated IT systems.

Second, there seems to be little literature to be found at all, regarding technical implementations of parallelization in backends of systems. (Qian, 2012) lists multiple parallelization tools, who without specifying the scope of parallelization tools to be evaluated, lists no backend software implementations per say, but rather discusses the automation of programming tools which are designed to write parallelized software (and the ability of compilers to parallelize code). As stated in Chapter 1, this report is a very technical report, with little to no focus on any new/novel ideas and their merits.

With this second reason in mind, it seems to the author of this project that it might be very well plausible that there exist few, if any papers in computer science research literature that can be described as “related work”.

However, several videos of different developers at different companies lay claim on YouTube (YouTube, 2016) that they make use of Celery. In fact, three companies (Instagram, Mozilla, and AdRoil) all pride the bottom of the front-page of Celery's home website `http://www.celeryproject.org/` under the heading of "Who is using celery".

We were, unfortunately, unable to find any written, publicly available technical reports on any such efforts.

Chapter 4

Proposed System Solution

In this chapter, we detail the implementation of the proposed system solution/ implementation. The proposed system implementation has been developed with the intent to fulfill the Project Goals listed in Section 1.2 in a robust, and dependable manner, with which the CMB system can grow.

First, we describe our perceived outset state of CMB, from when this master project was started. Then, we continue with describing how the proposed system implementation supports the Project Goal of Automatic System Monitoring and Recovery. The chapter then continues with the efforts expended in this project to upgrade the CMB code base from Python 2.7 to Python 3.4, to not only provide more utilities for the rest of this project but also to help future-proofing the CMB project.

After that, the chapter continues with describing the database changes necessitated to support the development of the Dispatcher in this proposed system implementation, before detailing the implementation of the actual Dispatcher itself, first introduced in Section 1.2.

Finally, we also detail the implementation efforts made on the code base for the Backends (`cmb-board` Git repository) of the CMB system.

4.1 The outset state of CMB

At the outset of this project, there were issues with the CMB system we felt had to be addressed before our work could begin in earnest. These were:

- The random spread of where environment variables necessary for CMB's successful

execution were located.
- The lack of a simple and robust configuration system which could easily (and with proper oversight) permit configuration changes.
- A simplified and more robust start-up script (and process), so that automatic system recovery (and monitoring) could be implemented in an efficient manner.

All of the above points are related to each other, the improvement of one helps the improvement of the others. The outset state of CMB has environment variables necessary for its start-up and execution located in the following locations:

Table 4.1: A table showing which environment variables were located in what file, and at what location, at the outset of this project.

Location	Environment Variables
~/.bash_profile	• APPLICATION_SETTINGS (Basically a config file) • CMB_MAIL_USERNAME • CMB_MAIL_PASSWORD • CMB_TOKEN_SECRET • CMB_SECRET_KEY
~/cmb/server/cmb-flask/server.cfg (This is the config file APPLICATION_SETTINGS points to).	• SERVER_PORT • BOARD_IP • MALI_DIR • FLASK_DIR • FRONTEND_DIR • UPLOAD_FOLDER • MAIL_SERVER • MAIL_PORT • MAIL_USE_TLS • MAIL_USE_SSL • GUNICORN_LOG_LEVEL • VERSION ("dev" or "prod" for Production/development)
~/cmb/server/cmb-flask/crontab.txt	• APPLICATION_SETTINGS (Hardcoded to refer to the above server.cfg).
~/cmb/server/cmb-flask/scripts/init_cmb.sh	• logfile (Where the CMB processes, such as Gunicorn, log their output to.).

Location	Environment Variables
~/cmb/server/cmb-flask/scripts/gunicorn_start	• NAME • USER • GROUP • NUM_WORKERS

In addition to the the five locations listed above, there are more variables which might qualify as "environment variables" in the Python code files located in **~/cmb/server/cmb-flask/source/*.py**, but we chose to leave those for another effort, another time. All of the environment variables in Table 4.1 can be put into one of three categories, as shown in Table 4.2.

Additionally, examples of both the config files used in the testing of the proposed system implementation (with their included environment variables used for the tests detailed in Chapter 7), and the bash script which easily lets one source the needed environment files (as demonstrated in both Chapters 5 and 6), can be found in Appendix A.

Table 4.2: A table showing how every environment variable listed in Table 4.1 belongs to one of three categories, with the omission of APPLICATION_SETTÌNGS.

Category	Environment Variables
Machine-specific	• SERVER_PORT • FLASK_DIR • VERSION ("dev" or "prod" for Production/development) • USER (Name of OS user, process(es) is(are) executed with). • GROUP (Name of OS group process(es) is(are) executed with).

Category	Environment Variables
Processes-specific	• GUNICORN_LOG_LEVEL • logfile/GUNICORN_LOG_FILE (Where the CMB processes, such as Gunicorn, log their output to. It has thus been renamed to "GUNICORN_LOG_FILE" in this proposed system implementation). • NUM_WORKERS • MAIL_USE_TLS • MAIL_USE_SSL
Secret / Sensitive	• MAIL_PORT • MAIL_SERVER • CMB_MAIL_USERNAME • CMB_MAIL_PASSWORD • CMB_TOKEN_SECRET. • CMB_SECRET_KEY

4.1.1 Proposed solution to environment variables/settings

Thus, the proposed system solution has consolidated the spread of these environment variables/settings. In the proposed system solution, a new folder has been added to the **cmb-flask** (and **cmb-board**) directory; **configs**.

The proposed concept is that in this (these) folder(s), any files with the string "secret" in the name, will be ignored by Git, and thus never added (as it never should be) to the Git repository/commit history. Meanwhile, machine-specific environment variables/settings, as well as process-specific ones can also reside here, and be copied/spread to new machines through Git at a developer's wish. As such, all other scripts, processes, and programs in the CMB project will always know where to look for any setting they may need.

As an example, **cmb-flask/configs** may (and should) contain the equivalent of the following:

- *crontab.txt*
- *secrets.cfg*
  (May (should?) be named something else than just "secrets", so as to differentiate between "dev", "test", and "prod").
- *machine-settings.cfg*
  (May (should?) be named something else than just "machine-settings", so as to differentiate between "dev", "test", and "prod").

- *gunicorn-config.cfg*
  (May (should?) be named something else than just “gunicorn-config”, so as to differentiate between “dev”, “test”, and “prod”).

Having consolidated the location for all environment variables required by the CMB system’s processes makes the start-up not only simpler but also more robust. Also, this system makes it clearer where what environment variables should be located, and what environment variables each file represents and should contain. As long as the directory structure inside **cmb-flask** (and **cmb-board**) is upheld, the excerpts of the start-up scripts listed in Subsection 5.3.1 show how the new system would work, compared with the old.

Note, for instance, that with the new system, the initiating user of the CMB-processes does not need to remember to source five (5) specific files (residing in differing locations) him-/her- self, nor activate the virtual environment path variables.

These efforts simplify the implementation of the automatic process monitoring and recovery substantially, further detailed in Section 4.2.

4.2 Automatic system monitoring and recovery

With the strategy regarding environment variables laid out in Subsection 4.1.1, initiating the CMB processes on e.g. the server is much simplified.

An automatic monitoring and recovery implementation needs to be able to achieve two things:

a) *Monitor when the system either crashes or becomes unresponsive,*

b) *and start the system as needed.*

As stated in Subsection 4.1.1, the commands required to initiate the start-up of the CMB processes(es), can be found in Sections 5.3 for the proposed server implementation, and Section 6.3 for the proposed backend implementation.

For comparison purposes, Listing 4.1 shows the start-up script used for the CMB system at the outset of this project. This file is with the outset state of CMB only usable by one specific user on the machine, which all developers / maintainers of the CMB system must be able to access.

Unfortunately, due to the time constraints detailed in the Preface, the proposed system implementation in this report does not include an analogous start-up script, for neither the server nor backend, nor an equivalent of the “checkOnline.sh” crontab[1] script. However,

[1] Citation: help.ubuntu.com/community/ (2016).

Section 9.1 describes how these things may be achieved, in the future of the CMB system.

Note that with the "init_cmb.sh" file shown in Listing 4.1, environment variables such as APPLICATION_SETTINGS must already have been set (sourced) for the script to work. (*This is why, with the outset state of the CMB system, that the script can only be initiated by a user who has the needed environment variables stored in its* `~/.bash_profile`).

In contrast with the start-up script in Listing 4.1, the `source` commands listed in Sections 5.3 and 6.3 can be input into a start-up script, completely eliminating the need to log into a system as a particular user (which is arguably a security risk), and gives the CMB system the potential to not only initiate the start-up of CMB processes by different users on a system/machine, but also have a log of who initiated what, when.

Listing 4.1: The CMB start-up script used for both starting and stopping CMB, at the outset of this project.

```
1  #!/bin/bash
2  #FRONTEND_DIR and FLASK_DIR is defined in server.cfg
3  . $APPLICATION_SETTINGS
4  logfile="/srv/climber/cmb/server/cmb-flask/logs/startup.log"
5  set -e
6  function start_cmb {
7          if screen -list | grep -q "cmb"; then
8                  echo "CMB allready running. Try stopping before starting"
9                  exit 0
10         fi
11 # screen -d -m -S cmb
12 # screen -S cmb -X stuff "cd $FRONTEND_DIR && gulp $VERSION
13 #"
14         cd $FRONTEND_DIR && gulp maintenance 2>&1 >> $logfile
15         #sleep 2
16         echo "starting server..."
17         sleep 1
18 # screen -S cmb -X screen $FLASK_DIR/scripts/gunicorn_start
19         { $FLASK_DIR/scripts/gunicorn_start 2>&1 >> $logfile &} > /dev/null
20         disown
21         sleep 5
22         echo "starting push..."
23         #screen -S cmb -X screen
24         #sleep 1
25         #screen -S cmb -p 2 -X stuff "export LC_ALL=''
26         screen -d -m -S cmb
27         screen -S cmb -X stuff "source $FLASK_DIR/../venv/bin/activate && cd
               $FLASK_DIR/source && python $FLASK_DIR/source/push.py
28 "
29         echo "starting frontend..."
30         cd $FRONTEND_DIR && gulp $VERSION 2>&1 >> $logfile
```

```
31          echo "CMB started"
32  }
33
34  function stop_cmb {
35  # echo "stopping screen..."
36          set +e
37          pkill gunicorn
38          set -e
39          screen -X -S cmb quit
40          sleep 1
41          echo "CMB stopped"
42  }
43
44  case "$1" in
45
46  start) start_cmb
47  ;;
48  stop) stop_cmb
49  ;;
50  restart) stop_cmb
51          sleep 1
52          start_cmb
53  ;;
54  *) echo "Please pass 'start', 'restart' or 'stop' as argument"
55  esac
```

Finally, on the subject of automatic process monitoring, the aforementioned "checkOnline.sh" script can be found in Listing 9.1. While simple in principle, there are a several "gotcha's" with the Bash implementation which seems to have gone unnoticed by the developers of this script.

- *It is with the intent of avoiding such pitfalls commonplace with Bash code, that there was (and still is) a strong desire at the outset of this project by all who have recently worked on the CMB system, to convert the Bash scripts running on both the server and backend into Python code.*

The realization of a monitoring system in the proposed system implementation of CMB is already partly implemented, with there being a REST API call implemented in the file `cmb-flask/source/routes/backends.py`, which can be utilized (or further modified) to have the server report how long ago since a backend polled the server. The code for this REST API call can also be expanded to more carefully take into consideration that a backend which hasn't recently polled, may be busy with an assigned submission to profile.

4.3 Upgrade from Python 2.7 to 3.4

Python[2] the programming language has had a checkered history, with regards to its evolution.

As with any programming/scripting language, its developers want to balance the wish for greater features implemented into the language while ensuring backward compatibility. The two extremes can often be mutually exclusive for any software product (not just programming languages), and focusing too much on one (even if it's just "tidying up" or improving the internals of the product/language), may nevertheless often end up neglecting users.

And with Python 3.0 being released in 2008, and the final 2.7 version released in mid-2010, the jump from Python 2 to 3 had been made with less regard for backward compatibility, due to a wish to clean up Python 2.7 properly[3]. Unfortunately for Python, this created somewhat of a split between Python 2 and Python 3, with the user base split on which version they wanted to use.

The Python 2 user base wanted to continue using the no-longer-receiving major updates Python 2 due to all of the scripts, programs, and efforts spent in Python 2, and the non-negligible cost in the effort of upgrading all existing Python 2 code to Python 3. Meanwhile, the Python 3 user base wanted to capitalize on the better Unicode support (all text strings now being Unicode by default), saner bytes/Unicode separation, in addition to many other improvements and utile additions.

Two areas utile for CMB in the effort of replacing the bash-scripts executing code profilings, which got improvements in Python 3, was the "OS" library[4] and "Subprocess" library[5].

The Python OS library (*module* in Python terminology) offers tools for file manipulation, with greater reliability and utility than the Python 2 version does. Things such as os.makedirs() now being able to construct all non-existing leaf-folders necessary, and permitting os.chmod() to accept a file descriptor as input, in addition to following symlinks, and more.

While in the Python Subprocess module offers the tools for spawning new processes, obtaining their return codes, and connecting to their input/output/error pipes. All of which CMB dearly needs for replacing as much as possible of the execution of the code submission profilings. What is new in version 3.3, (and 2.7 does not have), is the support for giving spawned processes a timeout limit. This has long (and often) been an issue for CMB with its implementation at the outset of this project.

[2] `https://www.python.org/about/`
[3] `https://wiki.python.org/moin/Python2orPython3`.
[4] `https://docs.python.org/3.4/library/os.html`.
[5] `https://docs.python.org/3.4/library/subprocess.html`.

During this project, great effort has been expended to upgrade not only the code base from Python 2 to Python 3 but also to ensure that the unit tests already written for this project also worked as intended in Python 3.

4.3.1 Git submodules

Thus, with the added realization that a lot of the code needed on the server, would also be of use to the backend (such as a shared function permitting the spawning of new processes with an optional timeout), the proposed implementation involved converting the **cmb-flask/source/cmb_utils** folder and its Python contents into a Git *submodule*.

A Git submodule is its own wholly valid Git repository, but it is also simultaneously acting as a "sub-directory" (hence the name "submodule") of another Git repository. This permits us only to have to deal with one set of code, instead of having copy/paste-like duplicates between both. As earlier stated, the majority of the functionality available in **cmb-flask/source/cmb_utils/*.py**'s files would contain functionality utile for both the server and the backend(s).

An added advantage of this implementation is that if there's a bug found, the bugfix only needs to be implemented once, and can then be pulled[6] into the other repositories also using it as a submodule.

4.4 Database changes

With the implementation of the Dispatcher[7], code submissions uploaded by users can be run on any eligible backend for profiling its energy efficiency and timing. Thus, it would behoove the administrators of CMB to know which profiling was run on which backend (Odroid-XU3 board). This is particularly the case if a future CMB system wishes to support different backend architectures, as stated in the Problem Statement.

The addition of rows in tables, or manipulation of relationships between them, can be modified both through scripted or interactive Python code (exemplified in the `cmb-flask/sources/init_db.py` file), or through the admin interface, created by the Flask web server.

In Figure 4.1, the white rectangles represent the database tables which CMB had in its implementation from the outset of this project. It is worth mentioning that while the database schema permits an uploaded code submission to have multiple runs, this functionality is utilized by the CMB today (nor in this reports proposed system implementation, though we attempted to make it easier for future developers to enable it).

[6] Through use of the command `git pull`.
[7] Detailed in Section 4.5.

This is discussed further in Section 9.3.

In summary, a Problem may have anywhere from 0 to N Submissions, and each Submission necessitates one (and at most one) User. Each Submission may (permitted through the database schema, not the server-code implementation) have 0 to N Runs, of said uploaded code submission executed on a backend, as illustrated in Figure 2.1.

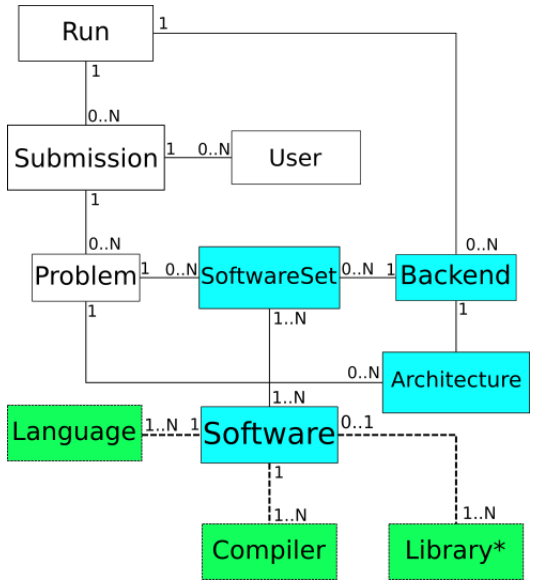


Figure 4.1: Proposed CMB Database Schema.

4.4.1 Changes necessitated by the Dispatcher

Before we continue describing Figure 4.1 and the changes it represents made in the proposed system implementation, we want to make clear that due to the time constraints detailed in the Preface; there is one discrepancy between the figure and the proposed system implementation. The relationship between the SoftwareSet table and the Backend table does not exist. Instead, the Backend table has a "many-to-many" relationship with the Software table. Note that the proposed system implementation takes this into consideration, and testing shows that it works as intended, however inelegant and undesired this alternative is to what's presented in Figure 4.1.

The light blue rectangles in Figure 4.1 represent the tables added in the proposed system implementation of this project.

Thus, a Run now needs to associate with a Backend, while a Backend may have 0 to N Runs associated with it. Each Backend, in turn, needs to associate with (and at most) 1 Architecture, but several Backends may share the same Architecture. An example of an

Architecture would be “Odroid-XU3”, signifying the 32-bit ARM big.LITTLE CPU and ARM Mali GPU cores[8].

Also, a SoftwareSet consists of 1 to N Softwares (which in turn are unique by the combination of name and version), with the restriction that no two SoftwareSets can be identical. Every Backend must support a SoftwareSet, and several Backends may support the same SoftwareSet.

Likewise, a Problem must require a SoftwareSet, and multiple Problems may require the same SoftwareSet. With the final additional requirement of Problems needing a target Architecture, it is through this logic that a newly spawned Run checks whether or not a Backend querying for its next job is eligible or not.

4.4.2 Potential changes for software language support

Subtasks 8 (and to a small extent 6) in the Problem Statement ask for a proposal on how the CMB system could be expanded so as to support for code submissions in different programming languages.

The green rectangles represent database tables which have not been implemented, which if implemented could easily facilitate support for multiple languages/libraries in CMB. This is further discussed in Section 9.3.

4.5 The Dispatcher

As stated in the motivations listed in Section 1.1, CMB currently executes all uploaded code submissions sequentially, on one set of hardware. To enable code submissions to be profiled/executed concurrently, the implementation of a “dispatcher”, which can dispatch submissions to different backends, has been requested.

At first, Celery[9] was considered to be an apt tool for implementing the dispatcher. However, as first mentioned in the Preface, in week 10 out of the 21 weeks of the project's duration, it was decided in a meeting with Lasse Natvig and IDI's IT dept. representative Arne Dag Fidjestøl that Celery introduced too much complexity[10], in addition to having to the backends communicate directly with the MySQL database.

CMB is currently switching from using a sqlite3 database to a MySQL database hosted on a separate machine/server, and thus at first the combination of RabbitMQ and MySQL

[8] Detailed in Section 2.3.

[9] Introduced in Subsection 2.4.2.

[10] With the need for a broker such as RabbitMQ, in addition to the backends having to communicate with something like the database for results persistence.

as a broker (message transport), and results backend (respectively) appeared to be a good fit for this project.

With the decision of *not* to utilize Celery having been made half-way into the project[11], it was instead decided to expand and utilize on CMB's REST API, to realize the "dispatching" mechanism. This decision was in part sparked by the fact that implementing the Dispatcher through the Flask REST API, instead of a Celery implementation, leaves the Flask web server the sole agent in Figure 4.2 interacting with the database.

Figure 4.2 shows how the different actors of the system, the users, CMB administrators, backends, database, and web server interact with one another. What the dotted line denotes is that the user base can in principle perform HTTP Requests to the Flask web server REST API, as long as the firewall settings of the web server permit it.

Otherwise, all interactions with the system go through the Flask web server, with the Flask web server being the only agent (as previously stated) interacting with the Database, and each of the backends. (The backends in turn only communicating separately with the Flask web server). The front-end is a process currently running on the same VM/machine as the Flask web server (though it does not have to), and it serves the HTML/JavaScript web content accessible to the user base through the `climb.idi.ntnu.no` URL endpoint.

Additionally, in Figure 4.2, the arrows denote which actors communicate with one another (as already detailed), with the arrowhead pointing *from* the active agent, and *to* the agent the active agent requests information from (or updates with information).

[11] At the meeting described in the Preface.

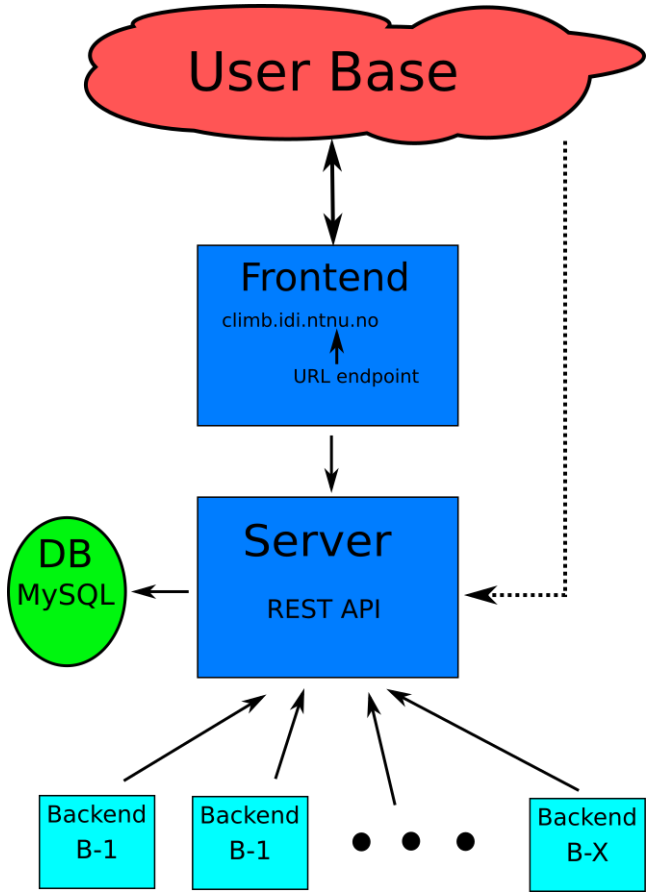


Figure 4.2: Diagram showing the activity relationships between the actors in the proposed CMB implementation.

4.6 The Backend

With the REST API implementation introduced and described in Section 4.5, each backend needs to pro-actively poll the Flask web server's REST API, to see whether or not there are any queued submissions awaiting energy and timing execution profiling.

With the added motivation of moving away from Bash scripts with insufficient reliability already stated in Section 4.2, efforts were made in this project to write the pro-active software on the backend in Python code which would be as stable as possible, and the least prone to unexpected errors, as possible.

Listing 4.2 illustrates this, through showing the "main infinite while-loop" running on the

backend. It is this code which is continuously executed when the steps in Section 6.3 are followed. It is the `try/except` and `try/except/finally` code blocks which ensure that no matter what errors or crashes occur in the Python code itself, the process running on the backend will not end until so told by outside influence.

Listing 4.2: The infinite while-loop of the backend process, polling the Flask web server for submissions to profile.

```
77 def main():
78     while True: # "main" infinite loop
79         status_code, polling_request = poll_request()
80         while status_code != 200:
81             sleep(SLEEP_PERIOD)
82             status_code, polling_request = poll_request()
83 
84         print(getCurrentTimeString() + "Parsing code submission received from server...")
85         try:
86             request_json_data = polling_request.json()
87         except Exception as e:
88             print(getCurrentTimeString() +
89                   "Following Exception occured during parsing of received request's JSON
                          buffer:\n\n{}\n"
90                   "\tRe-fetching code submission from server.\n".format(e))
91             continue
92         print(getCurrentTimeString() + "Code submission received from server parsed:")
93         print(request_json_data)
94 
95         try:
96             print(getCurrentTimeString() + "Preparing backend code profiling run...")
97             profile_dict = run_backend_profiling_run(request_json_data, sleep_period=
                   SLEEP_PERIOD)
98         except Exception as e:
99             print(getCurrentTimeString() +
100                  "Following Exception occured during execution of backend profiling run:\n\
                         n{}\n"
101                  "\tRe-fetching code submission from server.\n".format(e))
102            continue
103        finally:
104            cleanup(request_json_data['data']['run_id'])
105 
106        # Return data
107        push_results(profile_dict)
```

Figure 4.3 shows the control-flow of the code (including the code in the infinite while-loop shown in Listing 4.2) running on the backend. The green, white, yellow and red boxes represent actions which should be easily recognized in Listing 4.2. All the steps in the blue boxes however, are all inside the `run_backend_profiling_run()` function call on line 97 of Listing 4.2.

The stippled lines going from each of the blue boxes, and to the red box, represent the code written to support the abortion of the profiling run at the end of any of the steps represented by a blue box.

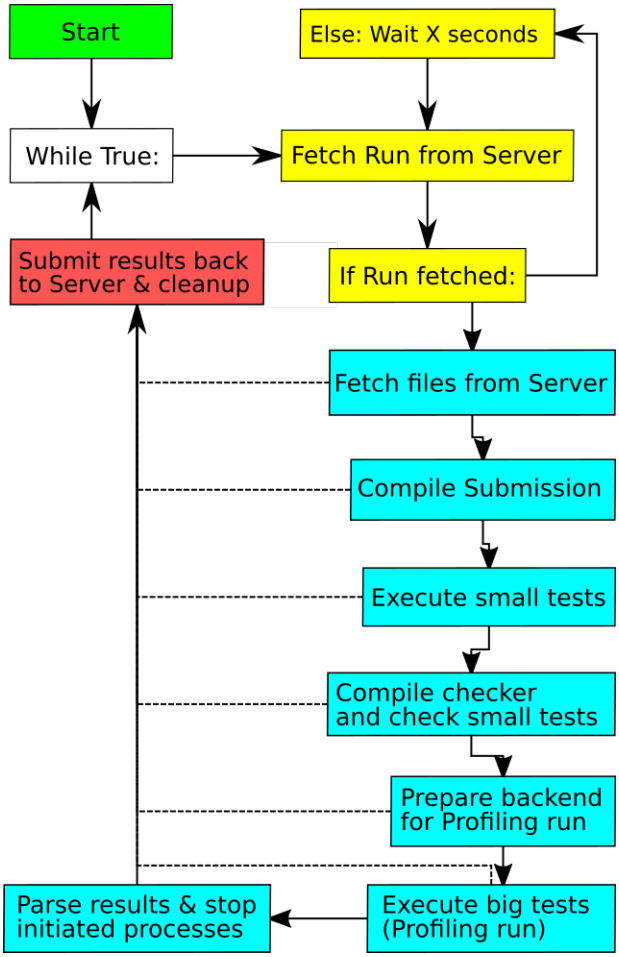


Figure 4.3: Diagram of Program Flow on Backend.

Finally, we wish to make it known, that with the folder structure in `cmb-board`, the intent is that it should be easy to add implementations for new architectures/backends in folders analogous/parallel to the `cmb-board/odroid-xu3` folder.

Thus, only the code represented by the blue boxes of Figure 4.3 need be replaced.

An important note regarding the backends used in the tests of Chapters 7 and 8 in this report:

Due to a misunderstanding between the author of this thesis, and the supervisor of this project, Lasse Natvig, it was discovered that Lasse Natvig had intended for us to completely re-format the backends given to run tests on. However, we had understood that we were to take caution with the system, disturbing it the least possible, so that if something were to go wrong, with either the existing system or the proposed system implementation, the backends given could more easily be reverted into use with the existing CMB system.

Unfortunately, this misunderstanding was uncovered too late[12] for there to be sufficient time remaining to completely re-format the given backends and re-install them as described in Chapter 6.

[12] See the Preface for more details of the delays and time limitations of this project.

Chapter 5

Server Installation Instructions

This chapter (introduced in Section 1.3) lists and describes how to install the server components of the proposed system implementation of this master project (discussed in Chapter 4). The chapter is written so as to work as an install manual for the server-side services and code for the proposed CMB implementation of this report.

(Magnussen, 2015) and (Støa and Follan, 2015) have both detailed the need for the different tools and components required by the CMB software, and any need for any components or tools inadequately described in this report, can be found in their papers.

Any new tools or components required by this proposed system implementation will either be adequately explained here in the install instructions, or the chapter detailing/discussing said tool or component.

Keep in mind the following when reading the instructions detailed within:

- At the outset of this project, CMB had *one* Virtual Machine (VM) server communicating with *one* backend.
- At *both* backend and server, a user was created to execute the services of CMB.

NB: The default/standard password(s) have been *changed* by adding "2" at the end of the current password string(s) used on all VMs (servers) and Odroid-XU3 cards (backends) which implement this proposed system solution.

5.1 Getting the Code

With the changes to the folder structure described[1] and discussed[2] in the report, we recommend that the folders for the frontend and server CMB codebases (both of which currently reside on the server), be located in the same directory.

Thus, it's recommended to make one `cmb/` folder, for example either in `~/` or `/Documents/`, in which both the two below folders get located.

1. cd into the folder you want them to end up in, and execute the following commands:
   git clone git@bitbucket.org:climbingmontblanc/climbing−mont−blanc.git cmb−js
   git clone git@bitbucket.org:climbingmontblanc/cmb−flask.git cmb−flask
   - Both having a website on the service hosting the `cmb-js` and `cmb-flask` Git repositories respectively:
     `https://bitbucket.org/climbingmontblanc/climbing-mont-blanc`
     `https://bitbucket.org/climbingmontblanc/cmb-flask/`
2. Inside the `cmb-flask`[3] folder, execute the following command to also git pull the needed submodule repository[4]:
   git submodule update −−init −−recursive

At the time of writing, there are several divergent branches in the different Git repositories, due to there having been several different Master Projects working on/with the system simultaneously, with at least two of those modifying the same codebases.

- Thus, make sure you select the correct Git branches you want, and that they are compatible with each other. The default Git branch master should be compatible with all the other master branches. This can be confirmed as needed with Lasse Natvig and Sindre Magnussen[5].

- If you want to utilize the branches (commits) which run this report's proposed system implementation, you can execute the following commands in the two folders when their installation is complete:
  cd cmb−flask/
  git checkout test−chrischa−branch [6]
  cd cmb−js/

[1] Section 4.1.
[2] Section 9.4.
[3] `cmb-flask` and `cmb-js` can be named anything, as long as the names of the two directories differ. They will be referred to as `cmb-flask` and `cmb-js` in the rest of this chapter.
[4] Discussed in Subsection 4.3.1.
[5] `lasse@idi.ntnu.no` and `sindrma@stud.ntnu.no`, respectively
[6] Git commit hash: `7ed3e1d8958a98b4e5cf7d5c713f9e5636b8ef3b`

`git checkout project_structure_rewrite` [7]

5.2 Install Instructions and Pre-Requisites

For the setup of the server, the following instructions list the pre-requisites and remaining install instructions:

a) A machine-specific `<X>-secrets.cfg`[8] config file located in `cmb-flask/configs/<X>-secrets.cfg`. It is recommended to copy the one used on a previously working system, and modify it as needed on the new system.

b) A MySQL Server.

- The username, password, and database name to be used on the MySQL server stored in CMB_MYSQL_USER, CMB_MYSQL_PASSWORD, and CMB_MYSQL_DATABASE, environment variables located in `<X>-secrets.cfg`, respectively.

c) A Ubuntu 14.04 (or equivalent/derivative) server-machine[9], which will need the following:

1. The IP's of all the backends which will be used (minimum one) stored in a comma-separated string in the BOARD_IPs environment variable in `<X>-secrets.cfg`.
2. A user (normally just named "climber"), which must be able to `ssh` to each and every backend without being requested for password. `ssh-copy-id` [10] can be used to achieve this[11].

   If the install happens on a IDI VM machine, it's recommended to edit the user's UID in `/etc/passwd` to an available number under 1000, so as to enable the `sudo passwd climber` command to run without being hindered by IDI/NTNU's Kerberos.
   .
3. The equivalent of `sudo apt-get install` -ing the following:
   `build-essential gcc-5 g++-5`
   `libmysqlclient-dev libffi-dev`
   `git python3 python3-dev python-virtualenv realpath`

[7] Git commit hash: `910d4f7d91c60d5e5e5283e1aa4c93d4eacf7cbb`
[8] `<X>` being say "prod", "dev", or "test3", as discussed in Section 4.1.
[9] These install commands have been tested on Ubuntu 14.04 LTS, 15.10, and 16.04 LTS.
[10] See Section 6.1 for an example of how to.
[11] If the Python code struggles with SSH during execution, see Appendix B.

```
openssh-server openssh-client fail2ban unattended-upgrades
```

Important note: (Støa and Follan, 2015; Magnussen, 2015) both report that the only requirement to have security updates automatically installed is to install the package `unattended-upgrades`. In the duration of this project, it has repeatedly been noticed that the backends have stated they have "X security updates pending", without this reported number ever diminishing. Some research into the tool revealed this[12] URL, which documents that there is additional set-up required (other than just installation through the package manager), for the tool to automatically install security updates on the machine.

4. And running these commands to ensure the security of the machines:
   ```
   sudo ufw allow 22
   sudo ufw allow 80
   sudo ufw allow 443
   sudo ufw enable
   ```

d) The following commands executed in `cmb-js`:

- Need to `sudo apt-get-install`:
  ```
  npm nodejs-legacy
  ```
- And then (in the root folder of CMB frontend):
  ```
  sudo npm install
  sudo chown -R climber:climber node_modules
  ```
  [13]

e) The following commands executed to finish the install of `cmb-flask`:

```
cd cmb-flask/
virtualenv -p python3 venv
venv/bin/pip install -r requirements.txt
source scripts/<X>-source_cmb_envvars.sh
cd source/ && ../venv/bin/python init_db.py
```

The last two commands sets up the database as needed, and you can read `cmb-flask/source/init_db.py` for what else it does if it's a "dev" install.

f) Finally, copy the `Mali_OpenCL_SDK_v1.1.0` folder and following contents with the following commands into the same directory where `cmb-flask` is located:

```
cd cmb-flask/../
```

[12] `https://help.ubuntu.com/community/AutomaticSecurityUpdates`

[13] Replace "climber:climber" with whatever "<user>:<usergroup>" that's applicable for your installation, as needed.

```
mkdir Mali_OpenCL_SDK_v1.1.0
```

On a machine which already has CMB running successfully, enter the `Mali_OpenCL_SDK_v1.1.0` folder, and run the following command:

```
scp -r common/ include docs/ lib/ \
Mali_OpenCL_SDK_v1.1.0_Documentation.html platform.mk samples/ \
climber@<new-machine>:Documents/cmb/Mali_OpenCL_SDK_v1.1.0/
```

5.3 Starting the Server

- To start the Flask web server in production mode, execute the following commands:

```
cd cmb-flask/source
source ../scripts/<X>-source_cmb_envvars.sh
../scripts/gunicorn_start.sh
```

- To start the Flask web server in develpment mode, execute the following commands:

```
cd cmb-flask/source
source ../scripts/<X>-source_cmb_envvars.sh
../venv/bin/python manager.py runserver -r -d -h $CMB_SERVER\
-p $SERVER_PORT
```

- To start the frontend (and not just the REST API), execute the following commands:

```
cd cmb-js/
node_modules/.bin/gulp local-dev
npm start
```

5.3.1 Start-up Script Differences

Chapter 4 details how the start-up of the CMB system functioned at the outset of the system and describes the proposed changes to simplify and make the process more robust. Appendix A contains the sourcing script file used for the test-system used for the tests described in Chapter 7, and most of the changes suggested in Chapter 4 can be recognized in the script file.

Chapter 6

Backend Installation Instructions

This chapter (introduced in Section 1.3) lists and describes how to install the backends and their constituent/ needed components of the proposed system implementation of this master project (discussed in Chapter 4). The chapter is written so as to work as an install manual for the backend-side services and code for the proposed CMB implementation of this report.

(Magnussen, 2015) and (Støa and Follan, 2015) have both detailed the need for the different tools and components required by the CMB software, and any need for any components or tools inadequately described in this report, can be found in their papers.

Any new tools or components required by this proposed system implementation will either be adequately explained here in the install instructions, or the chapter detailing/discussing said tool or component.

Keep in mind the following when reading the instructions detailed within:

- At the outset of this project, CMB had *one* server communicating with *one* backend.
- At *both* backend and server, a user was created to execute the services of CMB.

NB: The default/standard password(s) have been *changed* by adding "2" at the end of the current password string(s) used on all VMs (servers) and odroid-xu3 cards (backends) which implement this proposed system solution.

6.1 Install Instructions and Pre-Requisites

The backends used in this proposed system implementation and so far in the CMB project have been Odroid-XU3 cards, detailed in Section 2.3. These (and any new hardware used as backend) will need to be able to execute bash and Python code with reliable accuracy with regards to energy and timing measurements. Thus, the CMB project has so far used Ubuntu or Ubuntu OS derivatives as the OS to run on the backends; all gathered from `http://www.hardkernel.com/main/main.php`.

The latest install on a fresh Odroid-XU3 used Ubuntu 15.10[1], and was executed on an Ubuntu 15.10 system following the below install instructions:

1. Download the Ubuntu OS derivative image of your choice, made available for install on the XU3 at `http://www.hardkernel.com/`.

2. The Odroid-XU3 board can either install boot its OS from the MicroSD card, or the eMMC module. To install the downloaded OS image onto the eMMC module, an eMMC module reader is needed. Likewise, if the OS image is to be installed onto from the MicroSD card, a MicroSD card reader is required.

   It is important to ensure that the OS image chosen and downloaded supports the ARM EnergyMonitor program.

3. Execute the following Unix terminal instructions to install the OS image onto whichever card was chosen as destination[2]:

   The following command flushes the card chosen as destination for the OS, overwriting all its data in $4M$ blocksizes of zeroes.
   ```
   sudo dd if=/dev/zero of=/dev/path/to/chosen/card bs=4M conv=fsync
   ```

   The remaining commands describe the actual install of the OS image onto the destination card:
   ```
   unxz <chosen OS>.img.xz
   sudo dd if=<chosen OS>.img of=/dev/path/to/chosen/card bs=4M conv=fsync
   sync
   ```

4. After attaching the MicroSD card or eMMC module (loaded with the OS image) to the board, boot it and connect it to a monitor through a mini-HDMI or DisplayPort connection. An automatic login will appear for the user "odroid", open a terminal command window and execute the following commands:
   ```
   sudo adduser climber --home /home/climber --shell /bin/bash
   ```

[1] Acquired from `http://forum.odroid.com/viewtopic.php?f=95&t=18375`, last accessed February 2016.

[2] `http://odroid.com/dokuwiki/doku.php?id=en:xu3_bootmode_configuration` describes how to switch between booting from the eMMC module and MicroSD card.

```
sudo adduser climber sudo
sudo useradd -s /usr/sbin/nologin -M -N -g climber -K UID_MAX=999 worker
```

These two commands will create the "climber" and "worker" users, and giving "climber" sudo-powers. When prompted for a password after entering the command for the "climber" user, enter the password decided upon by the CMB team. (There'll be no password request prompt for the "worker" user, as intended). Use the above `sudo adduser climber <group>` command to add climber to all the groups of the "odroid" user is a member of, to ensure its working in the installed OS. Both users are needed to execute the backend profiling executions on the backend.

Additionally, the following line should be added to the file opened by the `sudo visudo`, to enable the worker to execute programs compiled by the "climber" user:

```
(...)
climber ALL=(worker) NOPASSWD: ALL
(...)
```

5. Thereafter, log in with the "climber" user, and `sudo apt-get install` the following Ubuntu packages:

   These first two lines of package names represent the tools needed for the execution of the CMB software, and the preparation and compilation of it.

   ```
   build-essential gcc-5 g++-5 gfortran
   git python3 python3-dev python-scipy python-virtualenv realpath
   ```

   The next line of package names represent the tools used for both control, access, and safety of the backend:

   ```
   openssh-server openssh-client fail2ban unattended-upgrades
   ```

   Finally, this last line of package names represent the libraries needed for compiling the CMB code, in addition to the libraries needed by the CMB code during execution:

   ```
   libblas-dev liblapack-dev libffi-dev libatlas-base-dev qt4-default libqwt-dev
   ```

6. The following commands represent the absolute minimum necessary to set-up the necessary SSH connections for CMB to work:

   First, log in with the "climber" user.
   Then, execute the following commands in a terminal, using all defaults when prompted (by pressing <enter>):

   ```
   ssh-keygen -t rsa -b 4096
   ```

`ssh−copy−id climber@<the CMB server backend will communicate with>` [3][4]

From this point on, it is recommended to execute the remaining commands through an SSH terminal from another machine on the NTNU network (presuming that the backend is also located within the NTNU network).

Thus, all I/O peripherals (with the exception of Ethernet and the eMMC module/MicroSD card) can be disconnected from the Odroid-XU3 board. Whenever a board in shutdown mode is connected to power, it will automatically boot.

7. `fail2ban` should work out-of-the-box, and is used to enhance the safety and security of the backend.

8. As stated in Chapter 5, `unattended−upgrades` has been reported by both (Støa and Follan, 2015; Magnussen, 2015) to work without any required additional set-up, beyond what is done by the package manager with which it was installed. However, with the observation that the backends keep stating that they have "X security updates pending", and with this number only having increased during this project, some research into the tool revealed this online resource: `https://help.ubuntu.com/community/AutomaticSecurityUpdates`. This resource explicitly states that there's additional set-up required post package manager installations, for this tool to automatically install security updates. For anyone utilizing this install guide, it is recommended to follow the recommendations of this online resource when installing `unattended−upgrades`.

   `ufw` should already be installed, and the following commands will ensure that only connections from IP addresses within the NTNU network can connect to the backend:
   `sudo ufw allow from 129.241.0.0/16` [5]
   `sudo ufw enable`

9. Next, you need to copy over the files and folders required, which are not included in the Git repository due to Intellectual Property (IP) reasons:

   `ssh` onto another CMB Odroid-XU3 backend which has successfully completed its install, and execute the below commands inside of its `cmb−board/` folder:
   `scp −r common/ EnergyMonitor_v3/ include/ lib/ climber@<new backend>`

10. Finally, to ensure more accurate and stable energy readings, remove the Ubuntu install's UI with the equivalent of the following commands:
    `sudo service lightdm stop`

[3] Replace "climber" as needed with whatever user name is used on the CMB server.
[4] If the Python code struggles with SSH during execution, see Appendix B.
[5] This being the IPv4 range of which NTNU holds ownership of. Any IPv4 addresses in this range will be an NTNU address.

```
sudo apt-get purge lightdm
```

6.2 Getting the code

The installation of the backend(s) differ little from the described proceedings in previous iterations of CMB (Støa and Follan, 2015; Magnussen, 2015). Thus, while the instructions in this chapter will re-iterate these, it will put a heavier focus on the changes necessitated by the proposed system implementation of this master project.

All of the following instructions will only apply to one *backend, but the instructions are identical for each additional backend added.*

Differing from the server install instructions in Chapter 5, the necessary code will only have to reside in one single folder. After the backend (the Odroid-XU3 board) has gotten its OS and tools installed[6], execute the following steps to download and prepare the necessary code onto and on the backend:

1. `cd` into the `$HOME` folder of the "climber" user (or equivalent) created on the backend, and execute the following command within:
   ```
   git clone git@bitbucket.org:climbingmontblanc/cmb-board.git cmb-board
   ```
   [7]
   ```
   cd cmb-board/
   ```

   - This Git repository has a website on the service hosting the `cmb-board` repository at the following URL:
     `https://bitbucket.org/climbingmontblanc/cmb-board/`

NB: At the time of writing, there are several divergent branches in the different Git repositories, due to there having been several different Master Projects working on/with the system simultaneously, with at least two of those modifying the same codebases.

- Thus, make sure you select the correct Git branches you want, and that they are compatible with each other (server and backend(s)). The default Git branch `master` should be compatible with all the other `master` branches. This can be confirmed as needed with Lasse Natvig and Sindre Magnussen[8].

- If you want to utilize the branches (commits) which run this report's proposed system implementation, you can execute the following commands in the two folders when their installation is complete:

[6] Detailed in Section 6.1.

[7] `cmb-board` can only be named something else if this change is reflected in the `cmb-flask/source/*.py` files of the server which the backend will be used with.

[8] `lasse@idi.ntnu.no` and `sindrma@stud.ntnu.no`, respectively

```
cd cmb-board/
git checkout test-chrischa-branch
```
[9]

2. The remaining steps detail how to copy the needed folders and their contents into `cmb-board`, which could not be added to the Git repository due to Intellectual Property reasons:

   On a CMB server which already has CMB running successfully, enter the `Mali_OpenCL_SDK_v1.1.0` folder, and run the following command:
   ```
   scp -r common/ include docs/ lib/ \
   Mali_OpenCL_SDK_v1.1.0_Documentation.html platform.mk samples/ \
   climber@<new backend>:~/cmb-board/
   ```

   On another CMB backend which already has CMB running successfully, enter the `cmb-board` folder, and run the following command:
   ```
   scp -r EnergyMonitor_v3/ climber@<new backend>:~/cmb-board/
   ```

3. The following instructions/commands are necessary for the compilation of the CMB code and its tools:

   This first command is a binary needed for the compilation/execution of EnergyMonitor and the `cmb-flask/Makefile` all user submitted code submissions currently use:
   ```
   sudo ln -sf /lib/ld-linux-armhf.so.3 /lib/ld-linux.so.3
   ```
   [10]

   Thereafter, add the below line to the `/etc/rc.local` file, but ensure you add it *above* the line saying `exit 0`.
   ```
   chmod +r /sys/devices/10060000.tmu/temp
   ```

4. Next, follow the below instructions/commands to compile the needed tools for the CMB software's execution:
   `cd` into `cmb-board/EnergyMonitor_v3/` and execute the following commands:
   ```
   qmake
   make *
   ```

   Thereafter, `cd` into `cmb-board/mountBlanc`, and execute the following commands:
   ```
   g++ -O2 ./dropCache.cpp -o dropCache
   chmod 4710 ./dropCache
   ```

[9] Git commit hash: `aacf2e562f007280270a149bb375e85f51c6ed45`

[10] Your mileage may vary. During some installs, symlinks have been required. These instructions will mark the following command(s) where it requirement has occurred with a "*". All the required binaries (targeted by the symlinks) have always been present when following the instructions detailed in this chapter. All the needed symlinks have been needed in `/usr/lib`.

5. Inside the `cmb-board` folder, execute the following command to also `git pull` the needed submodule repository[11]:
   `git submodule update --init --recursive`

6. Finally, to complete the installation of the CMB software, `cd` into `cmb-board/` and execute the following commands:
   `virtualenv -p python3 venv`

   Now, be warned that the following two commands may take quite a long time to execute (surplus of 20 minutes). It is recommended[12] to append `| less` to each command, if the terminal seems to freeze on a weird symbol mid-install or mid-compilation of a dependency:
   `venv/bin/pip install numpy==1.11.0`
   `venv/bin/pip install scipy==0.17.1`

   The above two packets are present in the below `requirements.txt` file, but `pip` still hasn't managed to solve `scipy`'s dependency of `numpy` in a good fashion, thus we install those two ourselves, manually, first[13].

   Finally, install the rest of the Python dependencies needed by the CMB backend software with the below command:
   `venv/bin/pip install -r requirements.txt`

6.3 Starting the Backend

At the outset of this project (as described in Section 4.5), there was no active agent or process running on the single backend the CMB system previously ran with. However, with the implementation described in Chapter 4, and with the install (and execute) commands listed in this chapter, the proposed system implementation of this report now requires the backends to become active agents in the CMB system. It should be relatively simple and straight forward, to have both a logging to file of the process running in the background, and a simple start-up script, combining the below three commands.

- To start the indefinitely running CMB backend Python process, execute the following commands:
  `cd cmb-board/`
  `source scripts/<X>-source_odroid_backend_variables.sh`
  `venv/bin/ipython service.py`

[11] Discussed in Subsection 4.3.1.
[12] If the commands are executed manually.
[13] This is why the packages `libblas-dev liblapack-dev libatlas-base-dev` are necessary.

Chapter 7

Methodology

In this chapter, we describe the hardware, software, configurations, and set-up used for the experiments this thesis reports the results of in Chapter 8. Hypotheses will be stated throughout the chapter, wherever writing relevant to the hypotheses reside.

First, we detail the Hardware used in Section 7.1, before we list the Software and Configurations used in Section 7.2. Following that, we justify the use of the problem submissions we have used in our testing.

Finally, we detail the Benchmark Tests which show how a CMB implementation with only *one* backend would behave with regards to timings (sequentially) in Section 7.4, and complete the chapter with describing the Parallelization Tests which show the timings of the (parallelized) implementation of our proposed system in Section 7.5.

7.1 Hardware & Hardware Set-Up

As previously stated in this report, the design of CMB is based on the premise of having one server, with $N \in [1, \rightarrow X)$[1] backends polling it at intervals. Thus, for testing the system, a server and backends were required.

The following two Subsections (7.1.1 & 7.1.2) detail what hardware was used for the tests described in this chapter.

[1] Where $X > 1$, up to the number of backends a CMB server can handle simultaneously. This is discussed further in Subsection 9.2.2.

7.1.1 CMB Server

The CMB Server on which these tests have been executed is a VM supplied by Teknisk Gruppe (Tg) at IDI. Tg is also the ones who have provided the previous VM and Database resources for the CMB project. Hence, an identical VM to the already existing ones in production, and the one reported in (Støa and Follan, 2015; Natvig et al., 2015; Magnussen, 2015) was requested.

Table 7.1: Representative values of the VM running the CMB test-server.

Resource	Available CPU cores	CPU Core Speed	BogoMIPS	Main Memory (RAM)	Total Disk Memory	CPU op-mode(s)
Amount / Value	3 (three)	2000.001 MHz	4000.00	2 338 692 KiB	15 GiB	32-bit, 64-bit

Table 7.1 lists a selection of key stats of the VM used. The remaining output data is found in Appendix C. All the stats listed were given by the following commands, executed in a terminal on the VM:

- `lscpu`
- `blockdev`
- `cat /proc/meminfo`
- `cat /proc/cpuinfo`

7.1.2 Backends

The backends used in the testing are the Odroid-XU3 boards detailed in (Støa and Follan, 2015) and Section 2.3.

Each card used utilizes the same hardware, with an *Samsung Exynos 5422* ARM *big.LITTLE* 32-bit Processor, an *ARM Mali T628 (MP6)* GPU, with 2 043 084 KiB of LPDDR3 RAM. The only hardware difference between the backends used in testing is that of the three cards used, one (named "*dev3*") has a faulty eMMC port. Thus, the "*dev3*" backend utilizes an inserted MicroSD memory card for "Disk Memory", as opposed to "*dev1*" and "*dev2*", which use the eMMC.

While all eMMC modules and MicroSD cards in use by the test backends are of the same GiB capacity, any difference in "Disk Memory" capacity between the backends is presumed to have negligible effect on the tests, since:

- There's sufficient space for the OS.
- There's sufficient space for the for the CMB software itself.
- There's sufficient space for the Software tools utilized by the CMB implementation.
- And any additionally required amount of storage space available, dependent on the size of the submissions profiled by the backend.
- The CMB implementation (like the one proposed in this report), saves no persistent data on backends as a result of executing a profiling run.
  (*For accuracy and completeness, we also remind the reader that there are currently no logs saved on the backend, beyond what the OS and tools offer by default. Nevertheless, any logs can be very easily implemented, for example in combination with the Linux terminal program* `screen`.)

However, size limitations aside, what's more relevant is the read/write speed differences between the eMMC cards and the MicroSD card. Table 7.2 lists the averaged read speed values collected by running the Linux memory device program `hdparm`[2] with the following command:

```
sudo hdparm -Tt /dev/mmcblk0
```

The average was found by running the command in an ssh terminal manually 15 times on each device, and linearly averaging the results of the last ten executions, with no more nor fewer processes running on the device than when the backend runs CMB. Table 7.2 also lists the variance (σ) of the dataset used for the linear average, to give an idea of the stability of the results. *dev3* denotes the backend which utilizes a MicroSD card as its storage device, while both *dev1* and *dev2* utilize their eMMC5.0 Modules as their storage device.

Table 7.2: Linux command "`hdparm`" device & cache read averaged (and the dataset's variance) benchmarking results of backends used in testing. Backend devices without underline are running on their eMMC Module, and the one(s) with are running on their MicroSD card.

Backend	*dev1*	*dev2*	*dev3*
Avg. cached read speed	1010.303 MB/sec	999.823 MB/sec	964.761 MB/sec
Avg. cached MB/s variance ($\sqrt{\sigma^2}$)	5.430	6.077	7.786
Avg. buffered disk read speed	101.570 MB/sec	149.765 MB/sec	18.045 MB/sec
Avg. buffered disk MB/s variance ($\sqrt{\sigma^2}$)	0.166	9.964	0.010

CMB has no limitations (beyond what the installed OS offers) on the memory available to the programs executed on the backends during profiling. What we surmise from Table 7.2, is that while the buffered disk reading speeds differs much more between the backends

[2] Version 9.43.

running on an eMMC Module, and the one running on a MicroSD card, the differences in cache read speeds are comparatively much smaller.

As long as the combined running of the test submission programs and the CMB systems on the backend don't surpass the main memory capacity of the backend, we can safely conclude that there's no paging to the disks on any device. The baseline memory usage of the backends, without CMB software processes running, is approximately between 300 to 400 MiB. We note that $400 + 627$ MiB is $< \simeq 2.0$ GiB. Hence, this gives our first hypothesis:

Hypoth. I *That the combined base memory usage of the backend, and the memory usage of the test submission programs executed by the CMB software processes, do not exceed the main memory (RAM) capacity of the Odroid-XU3 backends.*
(Thus, no paging to eMMC Module or MicroSD Card required).

(Fu et al., 2015) state that the eMMC5.0 HS400 module can theoretically achieve a max data transfer of 400 MB/s, which seems congruent with the results measured in Table 7.2. They also write that "*The design goal of this system is to achieve the read/write speed of eMMC array as 400/200 MB/s.*"

Meanwhile, (www.sdcard.org, 2016) write that MicroSD UHS-1 cards have a bus speed of 50 to 104 MB/s, depending on whether it's an "SDR50" or "SDR104" implementation. (www.hardkernel.com, 2016) does not specify which of the two implementations come with the Odroid-XU3 board, and we were unable to find another source detailing this. Nevertheless, a 50 to 104 theoretical max speed is congruent with the results listed in Table 7.2.

Note that the measured speeds reported in Table 7.2 fractional differences from the aforementioned theoretical max speeds vary only $\sqrt{\sigma}^2 = 10.111$ percentage points from one-another, if we compare with the "SDR104" implementation. If we compare with the "SDR50" implementation for the MicroSD Card, the variance is only $\sqrt{\sigma}^2 = 6.601$ percentage points.

7.2 Software & Configurations

The CMB Server and Backends both utilize (and rely upon), a Debian/Ubuntu OS to execute. (Støa and Follan, 2015) implemented the first iteration of the CMB system with Ubuntu 14.04 on all utilized machines, and in this report, we have implemented the server with Ubuntu 16.04, and a new backend with Ubuntu 15.10.

Due to the time constraints detailed in the Preface, we were unable to get the time to attempt a backend install using Ubuntu 16.04. Thus, while both CMB backends and servers use a similar (if not same) OS, the tools, and configurations of both server(s) and

backend(s) differ to some extent. In the two following subsections, we detail where more information about the software (OS, tools, and CMB) and configuration of the software can be found, so as to accurately reproduce the circumstances of our tests.

7.2.1 CMB Server

Table 7.3 lists a few key stats of the server with which the tests, whose results are reported in Chapter 8, were executed. The software installed/required to run the server implementation of CMB proposed in this report is listed in the install instructions detailed in Chapter 5, and the full output of the commands which gave the data for Table 7.3 can be found in Appendix C.

Table 7.3: Key OS and Software stats of the CMB test-server.

Software	**OS Version**	**Kernel Version**	**Default C/C++ Compiler**
CMB test-server	Ubuntu 16.04 LTS	Linux 4.4.0-22-generic x86_64 GNU/Linux	gcc (Ubuntu 5.3.1-14ubuntu2.1) 5.3.1 20160413

Beyond the software, the only differences from a new server created with the instructions in Chapter 5, is the configuration settings found in the:

1. Environment variables in file `cmb-flask/configs/test-server.cfg`,
2. and the environment variables in file `cmb-flask/configs/test-secrets.cfg`.
    - Censored versions of the two above-mentioned files can be found in Appendix A.

7.2.2 Backends

Table 7.4 lists some key stats of the backends with which the tests, whose results are reported in Chapter 8, were executed. The software required to run the backend implementation of CMB proposed in this report is listed in the install instructions in Chapter 6, and the full output of the commands which gave the data for Table 7.4 can be found in Appendix D.

Beyond the software, the only differences from a new backend created with the instructions in Chapter 6, is the configuration settings found in the:

1. Environment variables in file `cmb-board/configs/odroid-xu3.cfg`,
2. and the environment variables in file `cmb-board/configs/test-secrets.cfg`.
    - Censored versions of the two above-mentioned files can be found in Appendix A.

Table 7.4: Key OS and Software stats of the CMB test-backends.

Software	**OS Version**	**Kernel Version**	**Default C/C++ Compiler**
dev1	Ubuntu 14.04.4 LTS	Linux 3.10.54+ #1 SMP PREEMPT armv7l armv7l armv7l GNU/Linux	gcc-4.9.real (Ubuntu/Linaro 4.9.3-8ubuntu2 14.04) 4.9.3
dev2	Ubuntu 14.04.4 LTS	Linux 3.10.69 #1 SMP PREEMPT armv7l armv7l armv7l GNU/Linux	gcc-4.8.real (Ubuntu/Linaro 4.8.5-2ubuntu1 14.04.1) 4.8.5
dev3	Ubuntu 15.10	Linux 3.10.96-78 #1 SMP PREEMPT armv7l armv7l armv7l GNU/Linux	gcc (Ubuntu/Linaro 4.9.3-5ubuntu1) 4.9.3

7.3 Upload and Profiling Test-Problems

To test the proposed system implementation, we decided to use the already existing "Hello World" and "Shortest Path" problems in the existing CMB system. Not only will this save us the time and effort of creating new ones, and thus save us from having to ensure both their stability and reproducibility, but "Shortest Path" has also been used as a benchmark in (Støa and Follan, 2015).

Therefore we feel confident in the assertion that "Hello World" will represent something close to a *minimum-baseline-resources-needed* problem submission, while "Shortest Path" can be used to get more realistic results from both measuring the system performance during execution/testing, and energy profiling results.

Additionally, all tests whose results are listed in Chapter 8 are executed after the CMB system has already run tests (without stopping its processes in-between), but always right after having reset the CMB system (again without stopping its processes). This means that for as long as the backends and the problem used between tests (described in Sections 7.4 and 7.5) remain the same, said tests will not have the database reset, nor have all files previously uploaded to the test-server deleted.

If either the backends in use are changed, or a switch from "Hello World" to "Shortest Path" or vice versa occurs, the database is reset, and all files belonging to previous submissions are deleted on the server[3].

[3] Remember that no files/data of submissions profiled are stored on backends.

The CMB test-server will for each and every test always have been started through the use of the `cmb-flask/scripts/gunicorn_start.sh` bash script.

Each backend, when running, polls the server immediately after start, and immediately after completing a submission profiling. If the backend receives a "No more submissions to profile at this time" response from the test-server, it will wait 12 seconds before asking again.

7.4 Benchmark Tests

To test the implemented Dispatcher[4], we needed to have a benchmark of the sequential performance of the proposed system implementation, with which to compare. We get this benchmark by having only one backend running (polling the CMB test-server for uploads to profile) at a time, and executing 78 uploads and profilings of each of the "Hello World" and "Shortest Path" problem test-submissions on said backend.

Hypoth. II *The addition of N extra identical submissions, to be submitted to the CMB system to process simultaneously with the rest, should increase the amount of time it takes the CMB system to complete profile all the submissions simultaneously submitted linearly per submission added. This should hold true as long as there's only one backend polling the system, and it's the only doing so from the submissions are uploaded, until the system is done with them.*

Each of these 78 uploads and profilings were divided into twelve sets, where the Nth set ($N \in [1, 12]$) has N "concurrent" uploads and profilings. Thus, the first set has 1 upload and profiling which is executed on the CMB system, before the second set starts its 2 uploads and profilings, and so on, up until the twelfth set, which has 12 uploads and profilings executed on the CMB system.

All uploads and profilings of the previous set N_{X-1} are completed (reported as failed or successful by the CMB system), before the test continues with set N_X. The purpose of this wait is to time the amount of time it takes for the CMB system to handle the concurrent number of uploads/submissions of a set. This will help us get an idea of the user-experienced wait-time as the system gets more than one submission to profile at once (and thus testing the capacity and capabilities of the dispatcher).

Thus, we propose three tests, one for each of the backends;

Test 1: *$2 \times N$ sets, with $N \in [1, 12]$ of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev1*, with each set's submissions submitted concurrently to the test system.*

[4] Detailed in Section 4.5.

Test 2: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev2*, with each set's submissions submitted concurrently to the test system.*

Test 3: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev3*, with each set's submissions submitted concurrently to the test system.*

7.4.1 Benchmark Tests Setup

We list the average time it takes for a submission of "Hello World" and a submission of "Shortest Path" to execute from a user-experience perspective in Section 8.1, per backend.

Additionally, if time permits, we will also attempt to find and show the timing of the different steps a uploaded submission goes through when uploaded to and executed in the CMB implementation.

7.4.2 Challenge due to timing difference between backends

To test the Dispatcher of the proposed system implementation, we use the given three Odroid-XU3 cards as backends, two of which already had an OS installed and had been employed by the CMB project, in addition to an entirely new one (without OS installed)[5]. These backends have been referred to as *dev1*, *dev2*, and *dev3* respectively, in this chapter already.

Due to the differences between the backends described in Section 4.6, throughout our experimentation, development, and testing with the three backends, we have discovered timing differences between all three, even though hardware wise only *dev3* stands out from *dev1* and *dev2* in its use of a MicroSD card instead of the eMMC5.0 Module. Therefore, we have run the "$1 - 12$" benchmarks with both "Hello World" and "Shortest Path" on each backend, with only said backend in use by the CMB test-system.

7.5 Parallelization Tests

For testing the concurrency capabilities and capacity of the Dispatcher[6] of the proposed system implementation, we intend to execute the $2x78$ "Hello World" and "Shortest Path" test submission sets (upload and profile) with combinations of all three *dev1*, *dev2*, and *dev3* backends simultaneously polling the CMB test-server for new submissions to profile.

[5] As explained in Section 4.6.
[6] Detailed in Section 4.5.

Since we have three backends available for the tests of this project, testing with only one backend (as proposed in tests $1, 2, 3$ in Section 7.4), in addition to two, and then finally three backends concurrently polling the CMB test-system during the execution of the tests, gives us an idea of how the system performance behaves when adding additional backends.

7.5.1 Tests

Thus, we propose the following tests, to not only test the capabilities of the proposed system Dispatcher but also to see how the system performance changes with differing amounts ($[1, 3]$) of backends polling the system for submissions to profile:

Test 4: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path", with each set's submissions submitted concurrently to the test system, and all sets executed sequentially on the test-system with all three backends polling it.*

Test 5: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path", with each set's submissions submitted concurrently to the test system, and all sets executed sequentially on the test-system with backends dev1 and dev2 polling it.*

Both of tests **4** and **5** will tell whether or not the dispatcher manages to parallelize the execution of the submission's profilings. Our intent with test **5** is to bridge the gap between 1 and 3 backends polling the system during test execution. Thus, the results of test **5** can give an idea as to the trend of the system's behavior, when an additional backend is added. We chose backends *dev1* and *dev2* to be the backends used in test **5**, due to the arithmetically averaged differences between the average runtimes per set ($Avg(Set_N)$) between *dev1* and *dev2* being smaller than the difference between either of these and *dev3*.

With these tests in mind, we want to state our main hypothesis of this project, which we have regarding the capabilities of the Dispatcher:

Hypoth. III *The Dispatcher will have the effect that for N simultaneous, identical submissions (of the same problem) submitted to the CMB test-system, with M backends polling the server for submissions to execute, and with;*

a) $1 \leq N \leq M$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach the upper bound duration of 1 (one) analogous submission of said problem.

b) $1 \leq M \leq N$ and $N \bmod M = 0$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach the upper bound duration of $\frac{N}{M}$ analogous serially executed submissions of said problem.

c) $1 \leq M < N$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach

the upper bound duration of $\lceil \frac{N}{M} \rceil$ analogous serially executed submissions of said problem.

However, due to the bug discussed in Subsection 9.2.3, we run the web-server in the "Shortest Path" halves of tests **4** and **5** with the

```
../venv/bin/python manager.py runserver -h $CMB_SERVER -p $SERVER_PORT
```

command, instead of the `cmb-flask/scripts/gunicorn_start.sh` bash script.

Chapter 8

Results

In this chapter, we list the results of the proposed tests described in Chapter 7 and we discuss the results in Section 8.3. First, we report on the results of the Benchmark tests, described in Section 7.4 in Section 8.1, and follow it up with the same for the results of the Parallelization tests, introduced in Section 7.5 in Section 8.2. Finally in Section 8.3 we discuss the results listed Sections 8.1 and 8.2, in addition to the hypotheses stated in Chapter 7, and how they measure up to the results listed.

All reported averages in this chapter are arithmetic averages.

8.1 Benchmark Tests

Section 7.4 specifies tests **1**, **2** and **3**. These three propose the execution of "Hello World" and "Shortest Path" tests with the $N \in [1, 12]$ sets set-up, executed three times, once with each backend singly polling the test-server.

Test 1: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev1*, with each set's submissions submitted concurrently to the test system.*

Test 2: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev2*, with each set's submissions submitted concurrently to the test system.*

Test 3: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path" executed sequentially on backend* dev3*, with each set's submissions submitted concurrently to the test system.*

Table 8.1: Average runtime for "Hello World" submissions in each Nth set, executed with only one backend polling the test-server at a time.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	39.51	-
2	52.65	13.14
3	63.80	11.15
4	76.62	12.82
5	88.58	11.96
6	101.30	12.72
7	114.10	12.80
8	126.10	12.00
9	138.30	12.20
10	150.40	12.10
11	165.60	15.20
12	177.00	11.40

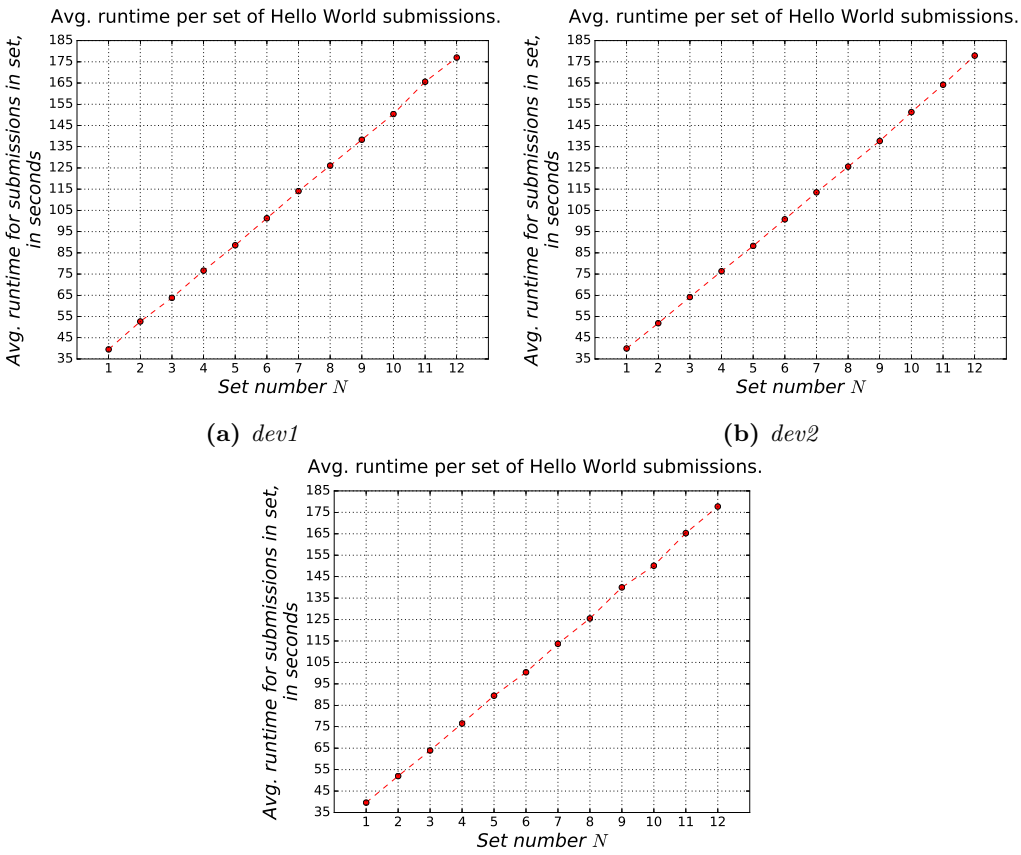
(a) Average runtimes for submissions per set in seconds, executed on *dev1*, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	39.94	-
2	51.85	11.91
3	64.17	12.32
4	76.35	12.18
5	88.26	11.91
6	100.80	12.54
7	113.50	12.70
8	125.60	12.10
9	137.70	12.10
10	151.30	12.60
11	164.20	12.90
12	177.90	12.70

(b) Average runtimes for submissions per set in seconds, executed on *dev1*, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	39.61	-
2	52.00	12.39
3	63.93	11.93
4	76.52	12.59
5	89.48	12.96
6	100.40	10.92
7	113.70	13.30
8	125.50	11.80
9	140.00	14.50
10	150.10	10.10
11	165.30	15.20
12	177.70	12.40

(c) Average runtimes for submissions per set in seconds, executed on *dev3*, and the absolute difference between the current and previous set's average.



(a) *dev1* **(b)** *dev2*

(c) *dev3*

Figure 8.1: Average runtime for "Hello World" submissions in each Nth set, executed once with each backend singly polling the test system.

First, in Table 8.1 we list the average runtimes of each set of the "Hello World" tests executed on each backend. These tables also include the absolute time increments between sets N_x and N_{x-1} of the arithmetic average runtimes of all submissions in each set.

Figure 8.1's Subfigures 8.1a, 8.1b, and 8.1c show the graphs corresponding to the growth of each set's average submission runtime, with the same numbers as the middle column of Subtables 8.1a, 8.1b, and 8.1c, respectively.

Following, Table 8.2 shows the same number from similar tests as 8.1, except the results listed in Table 8.2 stem from the tests having been executed with the "Shortest Path" problem submission.

Figure 8.2's Subfigures 8.2a, 8.2b, and 8.2c show the graphs corresponding to the growth

of each set's average submission runtime, with the same numbers as the middle column of Subtables 8.2a, 8.2b, 8.2c, respectively, analogous to Figure 8.1 and Table 8.1.

Table 8.2: Average runtime for "Shortest Path" submissions in each Nth set, executed with only one backend polling the test-server at a time.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	95.18	-
2	135.40	40.22
3	174.20	38.80
4	214.70	40.50
5	252.00	37.30
6	293.30	41.30
7	335.10	41.80
8	375.20	40.10
9	411.60	36.40
10	452.20	40.60
11	499.10	46.90
12	540.30	41.20

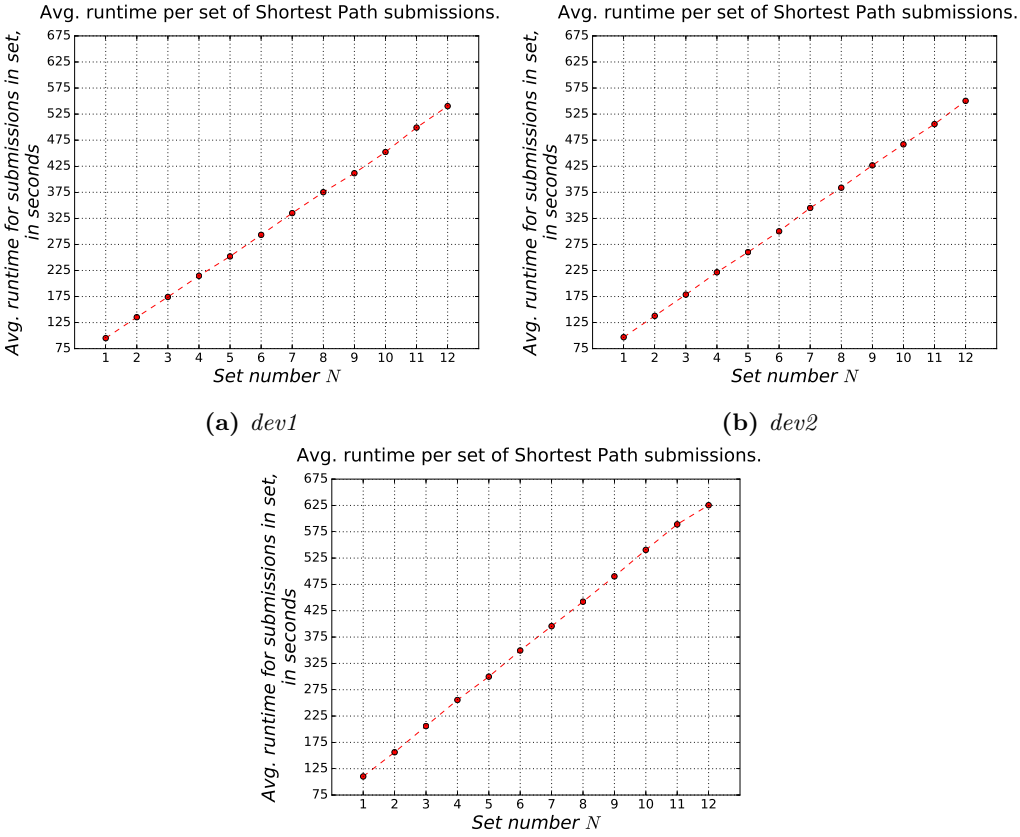
(a) Average runtimes for submissions per set in seconds, executed on *dev1*, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	97.00	-
2	137.90	40.90
3	178.70	40.80
4	221.60	42.90
5	260.20	38.60
6	300.20	40.00
7	344.90	44.70
8	383.80	38.90
9	426.70	42.90
10	467.00	40.30
11	505.80	38.80
12	550.40	44.60

(b) Average runtimes for submissions per set in seconds, executed on *dev2*, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	110.30	-
2	156.30	46.00
3	206.00	49.70
4	255.50	49.50
5	299.80	44.30
6	349.50	49.70
7	395.70	46.20
8	442.20	46.50
9	490.30	48.10
10	540.50	50.20
11	589.10	48.60
12	625.40	36.30

(c) Average runtimes for submissions per set in seconds, executed on *dev3*, and the absolute difference between the current and previous set's average.



(a) *dev1* **(b)** *dev2*

(c) *dev3*

Figure 8.2: Average runtime for "Shortest Path" submissions in each Nth set, executed once with each backend singly polling the test system.

8.2 Parallelization Tests

In Subsection 7.5.1, tests **4** and **5** are specified. Like tests **1-3**, they propose the execution of "Hello World" and "Shortest Path" with the $N \in [1, 12]$ sets set-up, with three and two backends concurrently in use by the CMB test-system, respectively.

Test 4: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path", with each set's submissions submitted concurrently to the test system, and all sets executed sequentially on the test-system with all three backends polling it.*

Test 5: $2 \times N$ *sets, with* $N \in [1, 12]$ *of first "Hello World", and then "Shortest Path", with each set's submissions submitted concurrently to the test system, and all sets executed sequentially on the test-system with backends dev1 and dev2 polling it.*

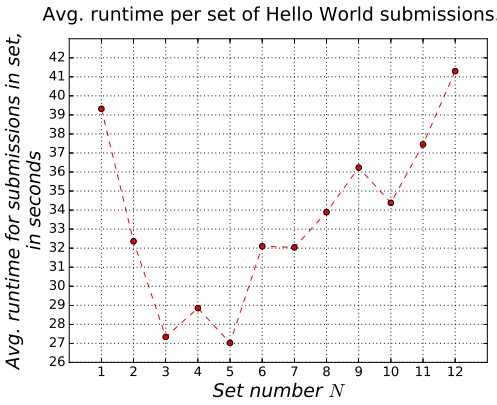
Table 8.3: Average runtime for "Hello World" and "Shortest Path" submissions of each Nth set, executed with all three backends polling the test-system simultaneously, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	39.32	-
2	32.36	6.96
3	27.34	5.02
4	28.85	1.51
5	27.02	1.83
6	32.10	5.08
7	32.04	0.06
8	33.89	1.85
9	36.23	2.34
10	34.38	1.85
11	37.45	3.07
12	41.30	3.85

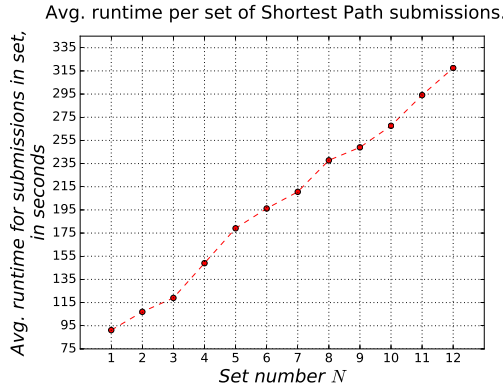
(a) "Hello World" test results.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	91.21	-
2	107.00	15.79
3	119.00	12.00
4	149.00	30.00
5	179.10	30.10
6	196.20	17.10
7	210.60	14.40
8	237.80	27.20
9	249.00	11.20
10	267.60	18.60
11	294.10	26.50
12	317.50	23.40

(b) "Shortest Path" test results.



(a) "Hello World" test results.



(b) "Shortest Path" test results.

Figure 8.3: Average runtime for "Hello World" and "Shortest Path" submissions in each Nth set of test **4**, when executed with all three backends polling the test-system during the test.

Figure 8.3's Subfigure 8.3b shows how the averaged user-experienced run-time for all submissions in a "Shortest Path" set increase with the amount of submissions in each set, and Subfigure 8.3a shows the same for "Hello World", with all three backends polling the system during the tests' execution. Together, Subfigures 8.3a and 8.3b represent the results of **Test 4**, listed in Table 8.3's Subtables 8.3a and 8.3b.

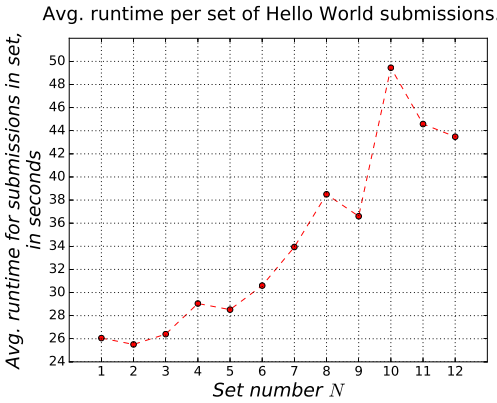
Table 8.4: Average runtime for "Hello World" and "Shortest Path" submissions of each Nth set, executed with backends *dev1* and *dev2* polling the test-system simultaneously, and the absolute difference between the current and previous set's average.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	26.06	-
2	25.51	0.55
3	26.40	0.89
4	29.05	2.65
5	28.52	0.53
6	30.60	2.08
7	33.94	3.34
8	38.50	4.56
9	36.60	1.90
10	49.45	12.85
11	44.58	4.87
12	43.48	1.10

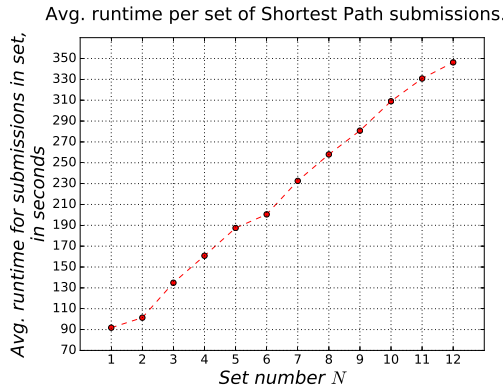
(a) "Hello World" test results.

Nth set	Avg.	$\lvert Avg(N_x) - Avg(N_{x-1})\rvert$
1	91.80	-
2	101.40	9.60
3	134.90	33.50
4	160.80	25.90
5	187.40	13.10
6	200.50	32.10
7	232.60	25.40
8	258.00	27.20
9	280.80	22.80
10	309.00	28.20
11	330.80	21.80
12	346.30	15.50

(b) "Shortest Path" test results.



(a) "Hello World" test results.



(b) "Shortest Path" test results.

Figure 8.4: Average runtime for "Hello World" and "Shortest Path" submissions in each Nth set of test **4**, when executed with backends *dev1* and *dev2* polling the test-system during the test.

Figure 8.4's Subfigure 8.4a shows how the averaged user-experienced run-time for all submissions in a "Hello World" set increase with the amount of submissions in each set, and Subfigure 8.4b shows the same for "Shortest Path", with backends *dev1* and *dev2* polling the system during the tests' execution. Together, Subfigures 8.4a and 8.4b represent the results of **Test 5**, listed in Table 8.4's Subtables 8.4a and 8.4b.

8.3 Discussion

In this section we discuss the results of Sections 8.1 and 8.2, and we evaluate these results against the hypotheses introduced in Chapter 7.

8.3.1 Benchmark Tests

The graphs of Figure 8.1 and Figure 8.2 both show near-identical trends as the amount of submissions in a set increase linearly.

Nonetheless, we are unable to find consistency in the sets averaged differences listed in Table 8.1 and Table 8.2. While the variance in-between these differences may be explained by the uncertainties of networks (even though all backends and VMs/servers are within IDI, NTNU's LANs), we hesitate to put all the responsibility for the variance on the network connections.

The backends wait 12 seconds before polling the server anew for a new submission to profile if the previous poll did not result in any submission from the server, as stated in Section 7.3. However, this should only affect the first three submissions of any set $N, N \geq 3$, and we can see discrepancies of up to $(|Avg(10) - Avg(9)|) = 10.10$, vs $(|Avg(11) - Avg(10)|) = 15.20$ in the serial "Hello World" test, and $(|Avg(12| - Avg(11)|) = 36.3$, vs $(|Avg(10) - Avg(9)|) = 50.20$ on *dev3*.

While *dev3* differs with both the OS it runs with, and the hardware device it uses as its HDD (detailed in Section 4.6 and Section 7.1 respectively), backends *dev1* and *dev2* also have analogous discrepancies in their variance between the averaged runtime of submissions per set.

(Støa and Follan, 2015) report very stable runtime on "Shortest Path" program executions on the backend, and that matches our observations during tests with the proposed system implementation. This causes us to suspect the CMB software implementation itself for the variances reported in the timing results. This is further discussed in Section 9.4.

8.3.2 Serial Hypotheses

Hypoth. I *That the combined base memory usage of the backend, and the memory usage of the test submission programs executed by the CMB software processes, do not exceed the main memory (RAM) capacity of the Odroid-XU3 backends.*
(Thus, no paging to eMMC Module or MicroSD Card required).

Hypothesis I proposes that there should be no part of the CMB system on the backends during the profiling of test submissions, which require more memory than the capacity of

the main memory of the backends, such that they have to page memory to their HDDs. (And thus eliminating the need to account for the MicroSD vs. eMMC5.0 hardware differences).

Executing a "Shortest Path" problem submission to the CMB test-system, with only one backend polling the CMB test-server, shows through the execution of the CMB software on the backend with the `/usr/bin/time -v <execute-tests-bash-command>` Linux/Ubuntu terminal command, that the maximum (peak) "resident size" (memory usage) of the backends is $\simeq 627$ MiB.

Hence, if there's at minimum > 630 MiB available main memory to each backend when the CMB backend software is not running, that should indicate no need to page memory to a backend's "HDD" device. After completing a 78 submissions test-round of "Shortest Path", each backend reports[1] 225, 229, and 239 MiB out of $1,990/1,990/1,995$ MiB available main memory (RAM), respectively.

Thus, we consider this as supporting evidence for Hypothesis I, that the MicroSD card is not what makes backend *dev3* consistently perform slower than *dev1* and *dev2* (unless the OS for some reason decides to take advantage of the MicroSD card when there should be sufficient RAM available). Instead, we suspect the reason to lie in the Kernel and OS differences, or the packages installed on the different systems, listed in Table 7.4 and Appendix D respectively.

Hypoth. II *The addition of N extra identical submissions, to be submitted to the CMB system to process simultaneously with the rest, should increase the amount of time it takes the CMB system to complete profile all the submissions simultaneously submitted linearly per submission added. This should hold true as long as there's only one backend polling the system, and it's the only doing so from the submissions are uploaded, until the system is done with them.*

Hypothesis II proposes that for a CMB system, with the proposed system implementation of this report that has only one backend polling it, for any set of simultaneously identical submissions submitted to it, it should only be a linear increment of time for the system to complete the set, for any additional identical submission added to the set.

We consider the graphs in Figure 8.1 and Figure 8.2 to support Hypothesis II, with what appears to be very linear developments for between each set. Additionally, accounting for the noise in the results listed in Table 8.1 and Table 8.2, the values in the third columns of each subtable show

[1] In the Linux/Ubuntu terminal program `htop`.

Backend	"Hello World" σ	"Hello World" μ	"Shortest Path" σ	"Shortest Path" μ
dev1	1.176	12.499	7.396	40.465
dev2	0.119	12.360	4.990	41.218
dev3	2.121	12.554	15.818	46.827

Table 8.5: Average (μ) and variance (σ) of the average runtime differences values from Benchmark tests.

Table 8.5 takes values in the third columns of the subtables in Tables 8.1 and 8.2, and lists the arithmetic mean (μ) and variance (σ) of these numbers, and its numbers show us a few things:

1. *dev1* requires the smallest average time increment when adding an additional submission to a set executed with only this backend polling the test-server.
2. *dev2* has the lowest variance in the averaged runtime values of the sets executed with only this backend polling the test-server.
3. *dev3* has both the worst variance and arithmetic mean of the two values mentioned above.

We feel it's important to stress that our results are from just executing each test *once* and that any indications of trends between the cards should be taken with a bit of salt, especially when it comes to the variances of the tests on each backend.

However, we nonetheless feel that the data supports Hypothesis II, especially when the greatest difference $max(\frac{\sigma}{\mu})$ of the same test and backend in Table 8.5 is $\simeq 0.33$, which is notable, but not high enough to change the trends shown that easily.

8.3.3 Parallel Tests

The results in Section 8.2 clearly show that the proposed system implementation offers benefits to the timing of multiple problem submissions submitted to the CMB system.

While there is a lot of "noise" in the results (particularly the results of the "Hello World" test submissions), Table 8.6 clearly show us that there are benefits to be gained by adding an additional backend to the system when processing multiple concurrently submitted submissions.

Though the numbers in Subtable 8.6a vary to a larger extent than those of Subtable 8.6b, the "Hello World" numbers must take into account the use of Gunicorn, with three workers, as detailed in Appendix A. The "Shortest Path" numbers, however, are gained with the CMB test-server process running with a single thread, single process, making its behavior completely serial when multiple backends communicate with it.

We speculate that this fact may be why there's such a big difference in the parallelization speedup in Subtables 8.6a and 8.6b. Attempts to raise the `GUNICORN_WORKER_TIMEOUT` value in `cmb-flask/source/scripts/gunicorn_start.sh` to 150 seconds, but that did not have any effect on the problems mentioned in Subsection 7.5.1, as this was our best guess as to what could cause these errors from occurring.

Table 8.6: The speedup of the average runtimes per set in tests **4** and **5**, divided by *dev3*'s Benchmark tests' average timings per set.

Nth set	$Avg_{dev12}(N_x)/Avg_{dev3}(N_x)$	$Avg_{dev123}(N_x)/Avg_{dev3}(N_x)$
1	1.520	1.007
2	2.038	1.607
3	2.422	2.338
4	2.634	2.652
5	3.137	3.312
6	3.281	3.128
7	3.350	3.549
8	3.260	3.703
9	3.825	3.864
10	3.035	4.366
11	3.708	4.414
12	4.087	4.303

(a) "Hello World"

Nth set	$Avg_{dev12}(N_x)/Avg_{dev3}(N_x)$	$Avg_{dev123}(N_x)/Avg_{dev3}(N_x)$
1	1.202	1.209
2	1.541	1.461
3	1.527	1.731
4	1.589	1.715
5	1.600	1.674
6	1.743	1.781
7	1.701	1.879
8	1.714	1.860
9	1.746	1.969
10	1.749	2.020
11	1.781	2.003
12	1.806	1.970

(b) "Shortest Path"

Figure 8.5 show the trends of the numbers listed in Subtables 8.6a and 8.6b. The triangles denote values of the rightmost column of these two subtables, and the filled circles denote the values from the middle column.

Albeit the fact that the variance in the numbers for "Hello World", the trend in Figure 8.5 is quite apparent; The "Hello World" tests, utilizing Gunicorn on the test-server, have not stabilized with the 12 submissions of set $N = 12$ enough to give an idea as to the graphs asymptote.

The "Shortest Path" lines in Figure 8.5 do not depict trends equally positive. However, we consider the omission of running tests **4**'s and **5**'s "Shortest Path" submissions without Gunicorn to be an important point to keep in mind.

In the attempts to perform them *with* Gunicorn, results like $Avg_{dev12}(N = 12) = 280.70$ seconds were achieved, with at most 1 submission reported as "failed" by the CMB test-server's big-tests-results check.

Re-calculating the bottom middle number of Subtable 8.6b with this result, gives a speedup of 2.228, which is demonstrably more positive when it comes to the capabilities of the proposed system implementation's dispatcher than 1.806.

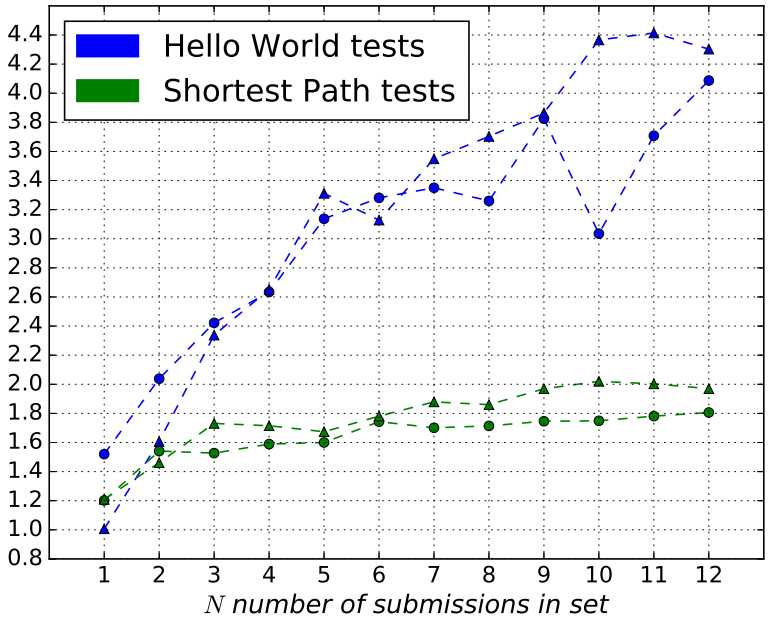


Figure 8.5: Parallelization speedup trends from tests' averages.

With the results reported herein, we conclude that the proposed system implementation can achieve a lower bound of 1.5x increased speedup when using two Odroid-XU3 backends instead of one, and can potentially approach an upper bound of $>$ 4x better speedup under ideal circumstances.

Keep in mind that the numbers used to get the numbers contained in Table 8.6 and

depicted in Figure 8.5 come from the slowest backend of the test-system, *dev3*.

Any system with faster backends will potentially be able to gain even better speedup with this fixed workload.

8.3.4 Parallel Hypothesis

Hypoth. III *The Dispatcher will have the effect that for N simultaneous, identical submissions (of the same problem) submitted to the CMB test-system, with M backends polling the server for submissions to execute, and with;*

a) $1 \leq N \leq M$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach the upper bound duration of 1 (one) analogous submission of said problem.

b) $1 \leq M \leq N$ and $N \bmod M = 0$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach the upper bound duration of $\frac{N}{M}$ analogous serially executed submissions of said problem.

c) $1 \leq M < N$, the total time the CMB system requires from the first upload until last submission's profiling is reported as completed, will at the most approach the upper bound duration of $\lceil \frac{N}{M} \rceil$ analogous serially executed submissions of said problem.

What Hypothesis III proposes is that with the proposed system implementation, we should see a decrease in the time necessitated by the system per uploaded submission, for each backend, added, and that the decrease should approximate the expressions in a), b), and c).

The first subpart of Hypothesis II; a), only says that for a single submission, and with multiple backends polling the CMB system, the system should never take longer to complete the submission, than it would with just one backend polling it.

dev1 has the overall fastest execution of the serial tests, and it completed "Shortest Path" $N = 1$ in 95.18 seconds, while both of tests **4** and **5** cleared it faster than that.

However, the timing results for "Hello World" do not support Hypothesis IIIa), but instead show us, with the rest of the measurements depicted in Figure 8.3a, that there's a lot of variance in the required execution time for problem submissions which execute fast, for the proposed system implementation.

The case that Hypothesis IIIa) still holds true, but cannot be confirmed due to the variances in timing with the proposed system implementation can be made. This is further discussed in Section 9.4.

Nevertheless, Hypothesis IIIb) is supported by the values depicted in the figures, and listed in the tables of this chapter. At no point, does a set with N ($Nmod\,3 = 0 \rightarrow [3, 6, 9, 12]$)

submissions take longer in tests **4** or **5**, than they do in any of the serial Benchmark tests. We thus consider the evidence presented to support Hypothesis IIIb).

Finally, what Hypothesis IIIc) attempts to cover, is that which a) and b) have not, that the time required for the system in these configurations to complete N submissions (when $N mod\ 3 \neq 0$), does not need more time than it would by adding 1 or 2 more submissions, making $N mod\ 3 = 0$. In other words, that the system does not spike multitudes of the time increment an additional submission would incur, when $N mod\ 3 \neq 0$.

With the exception of the results from first set $N = 1$ of tests **4** and **5** "Hello World" tests, which we've already stated we suspect vary so much due to the variances in the timing required by the CMB system (aside from the time required by the actual submission), we consider the evidence given in this chapter to support Hypothesis IIIc).

We feel safe with that claim, due to none of the remaining sets $N \in [2, 12]$ ever approaching the time it took the fastest backend *dev1* to execute "Hello World" with $N = 2$.

Chapter 9

Future Work

In this chapter, we discuss all the things we did not get the time or capacity to do, in addition to any suggestions we feel worthwhile to consider for the future of CMB.

First we discuss the completion of the Automatic Monitoring and Recovery system implementation, the first contribution introduced in Section 1.2, before continuing with discussing potential improvements for the second contribution introduced in Section 1.2, the Dispatcher.

After that, we discuss expanding the functionality of the proposed system implementation so as to support multiple language problems/submissions. Finally, we discuss the rest of the non-Problem Statement introduced improvements we would like to propose that future developers of CMB keep in mind when continuing the development of this system.

9.1 Completing the Automatic System Monitoring and Recovery implementation

As already briefly touched upon in Section 4.2, the work with the automatic monitoring and recovery system was not completed due to the time constraints incurred by the events described in the Preface. However, it was always our intent to implement a robust system for this project goal, using the likes of Systemd (Poettering et al., 2010) or Upstart (Upstart, 2006).

Listing 9.1 shows the crontab script running every 15 minutes, which is how the CMB system currently (and at the outset of this project) performs its "automatic system recovery".

Listing 9.1: The CMB crontab script used for monitoring the server processes of CMB, at the outset of this project.

```
1  #!/bin/bash
2  . $APPLICATION_SETTINGS
3  lines=‘ps aux | grep "push.py" | wc -l‘
4  sendMail=false
5  msg=""
6  pushDown=0
7  if [[ $lines == 1 ]]; then
8          sendMail=true
9          msg="PUSH DOWN!"
10         pushDown=1
11 fi
12
13 guniDown=0
14 lines=‘ps aux | grep "gunicorn" | wc -l‘
15 if [[ $lines == 1 ]]; then
16         sendMail=true
17         msg="$msg GUNICORN DOWN!"
18         guniDown=1
19 fi
20
21 #check board
22 ping -q -w 2 $BOARD_IP > /dev/null
23 test=$?
24 boardDown=0
25 if [[ $test != 0 ]]; then
26         sendMail=true
27         msg="$msg BOARD DOWN"
28         boardDown=1
29 fi
30
31 obj="{\"pushDown\":$pushDown, \"guniDown\":$guniDown, \"boardDown\":
       $boardDown, \"date\":\"‘date‘\"}"
32 echo $obj >> /srv/climber/cmb/server/cmb-flask/logs/systemStability.txt
33
34 #send mail is done by crontab
35 if [[ $sendMail == true ]]; then
36         cd /srv/climber/cmb/frontend
37         /usr/local/bin/gulp maintenance > /dev/null
38         echo $msg
39 fi
```

There are a few critiques worth mentioning as pitfalls to avoid in a future implementation of this for the CMB system:

1. Their use of the Linux terminal command `grep` to regex after the process names given by the `ps aux` command, runs the risk of giving false positives.
   - This is detailed in a post on `stackoverflow.com`, which has an analogous use of `grep`: `http://stackoverflow.com/a/3510850/1503549`
2. The hard-coded location and file-name of where to save the JSON object created for logging purposes at the end of each execution of the script.
   - It is our opinion that these things belong in a *Machine-specific* environment variable, as discussed in Subsection 4.1.1.
3. The omnipresent (and mandatory for the successful execution of this script) use of the APPLICATION_SETTINGS environment variable.

Listing 9.2 shows how far the efforts regarding automatic monitoring in the proposed system implementation came, and the concept/premise we intend for its future use.

Listing 9.2: The proposed system implementation of the automatic monitoring of backends.

```
290 @backend_routes.route('/api/backends/status/', methods=['GET'])
291 @backend_routes.route('/api/backends/status/<int:optional_time_limit>/', methods=['GET'
        ])
292 @myWrappers.sameIP_required
293 def check_last_query_of_backends(optional_time_limit=20):
294     backends = Backend.query.all()
295 
296     late_backends = [("backend.id", "backend.last_query")]
297     should_have_polled_since = datetime.now()
298     should_have_polled_since -= timedelta(seconds=optional_time_limit)
299     for backend in backends:
300         # Late_backend = namedtuple('backend_id', 'last_query')
301         if (backend.last_query is None or
302             backend.last_query < should_have_polled_since):
303             # late_backends += Late_backend(backend.id, backend.last_query)
304             late_backends += [(backend.id, backend.last_query)]
305 
306     # If one or more backends are late:
307     if len(late_backends) > 1:
308         busy_backends = []
309         for backend in late_backends:
310             runs_assigned_to_backend = Run.query.filter(
311                 Run.backend_assigned == backend,
312                 Run.dequeued.isnot(None),
313                 Run.finished.is_(None)).first()
314             if runs_assigned_to_backend:
315                 busy_backends += backend.id
316 
317         return jsonify(
318             returncode=1, late_backends=late_backends,
319             time_limit="{} second(s).".format(optional_time_limit),
320             current_time_and_date="Current time and date on server: "
321             "{}".format(getCurrentTimeString()),
322             backends_currently_busy_with_a_code_profiling=busy_backends,
```

```
323                 message="Some backends have not polled within the given time limit.")
324
325     # If no backends were late:
326     return jsonify(
327         returncode=0,
328         time_limit="{} second(s).".format(optional_time_limit),
329         current_time_and_date="Current time and date on server: "
330         "{}".format(getCurrentTimeString()),
331         message="All backends have polled within the given time limit.")
```

As can be seen in Listing 9.2, it is incomplete in its functionality, but the tools for creating a more thorough check of which backends are unresponsive are in place. In addition, it is very easy, as exemplified in almost all of the Python files in `cmb-flask/source/routes` and `cmb-board/service.py`, to have an independent Python script run every 15 minutes (like the crontab script of Listing 9.1), and report by mail or otherwise to the administrators if something is amiss.

This, combined with an implementation like what Systemd (Poettering et al., 2010) offers to ensure that the Flask web server's process(es) are always running, would constitute both an automatic monitoring and recovery implementation.

9.2 Future improvements to the Dispatcher

The Dispatcher, as implemented in the proposed system implementation, simply assigns the next submission to be profiled to the first backend polling which happens to be eligible.

Thus, it's merely acting as a First-In, First-Out (FIFO) queue handler, with no regards to load-balancing. As such, there's no code ensuring erroneous behavior, like the same backend repeatedly polling while the others are silent, is handled correctly (besides what's mentioned in this chapter).

Hence, as more backends are added, especially backends with differing performance, code which helps load-balancing the system better than FIFO should be simple enough for any developer familiar with the code base to implement, given the proposed system implementation changes to both how the backends poll the Flask web server REST API for new submissions to profile, and the database changes suggested in Section 4.4.

9.2.1 Expanding the dispatcher into a broker

One desire stated in the Problem Statement is for suggestions on how the Dispatcher could be expanded into a broker.

As previously stated, with the proposed system implementation, it is our opinion that

there's little need for a broker. Not only is every backend (with the exception of what's described in Sections 4.6 and 7) identical, but with the current "Master Makefile" (detailed in Subsection 9.3.1) being used in every compilation of every uploaded submission, it's ensured that (insofar as the server and backends remain constant/identical) nothing is different between any two identical profiling executions.

However, if some backends at some point in the future are "put aside" for special purposes, or a change is made so that a problem may be run on different architectures as long as the software required is present, then a broker may be of use.

The changes to the database schema, and the code deciding the behavior the CMB system with regards to backends should be simple enough to expand upon for any developer familiar with the code base. Especially considering how consolidated the database schema and backend behavior is in `cmb-flask/source/database/models.py` and `cmb-flask/source/routes/backends.py`, respectively.

9.2.2 Discovering the upper limit of backends a server can handle

Parallel to expanding the Dispatcher into a broker, an additional point of future interest for the Dispatcher is finding out the limit after which adding a new backend does not help improve the concurrency capabilities of the CMB system.

This, however, ties closely in with the implementation of the server, which with the differing threads of Gunicorn (given that there are sufficient processing cores available for the Gunicorn "worker" threads), can more easily be parallelized, as long as important sections of code is written so as to hinder race-conditions, such as shown in Listing 9.3.

It is from line 45 and until line 67 the atomic lock in Listing 9.3 ensures that no Gunicorn "worker" thread/process executes the code within simultaneously with another. The race-condition avoided by this implementation is to avoid that a scheduled run, not yet assigned to a backend, gets assigned to two backends so that later the Flask web servers REST API will receive POST requests from both backends having run the same submission.

Listing 9.3: How the proposed system implementation handles the potential race-condition of multiple backends polling for submissions from the web server running with several Gunicorn "worker" threads.

```
29 atomic_db_access_lock = Lock() # Used to ensure atomic retrieval of potential runs
30 backend_routes = Blueprint('backend_routes', __name__, None)
31
32
33 def check_for_available_runs_by_backend(backend, LIFO_order=False):
34     # Check first if a run is already assigned to backend
35     already_assigned_run = Run.query.filter(
36         Run.backend_assigned == backend,
37         Run.dequeued.isnot(None),
```

```
38          Run.finished.is_(None)).first()
39
40      if already_assigned_run is not None:
41          return already_assigned_run
42
43      # If not, see if there's a new one to assign
44      untaken_runs = []
45      atomic_db_access_lock.acquire() # Acquire mutex lock
46      if LIFO_order:
47          untaken_runs = Run.query.\
48              filter_by(backend_assigned=None).\
49              order_by(desc(Run.enqueued)).all()
50      else:
51          untaken_runs = Run.query.\
52              filter_by(backend_assigned=None).\
53              order_by(Run.enqueued).all()
54
55      for run in untaken_runs:
56          if (run.submission.problem.architecture == backend.architecture and
57              run.submission.problem.software_required <= set(backend.softwares)):
58              # Update run that it's going to get processed
59              run.backend_assigned = backend
60              run.dequeued = datetime.now()
61              db.session.add(run)
62              db.session.commit()
63              atomic_db_access_lock.release() # Release mutex lock
64              return run
65
66      # If no run was found in for-loop:
67      atomic_db_access_lock.release() # Release mutex lock
68      return None
```

As parting thoughts on the discussion of this subsection, it is our opinion that the inherently serial check of whether or not an uploaded submission successfully compiles should be delegated to the backends, instead of the server.

This notion is spawned due to the premise to moving any inherently serial work the "single-point-of-contact" (the Flak web server) between all the elements of the CMB system performs, to the backends, which with proposed system implementation of this report there may be many attempting concurrent communication with the Flask web server.

Amdahl's Law of parallelization (Mark D. Hill and Michael R. Marty, 2008) foretells this to be a potential choke point for future scalability of the CMB system with the current behavior.

9.2.3 Fixing the undiscovered Gunicorn bug

Listing 9.4 shows lines 147−191 and 216−252 of `cmb-flask/source/routes/backends.py`, which handle the case when the submission profiled on a backend is returned, and either failed at some point on the backend or fails the final big correctness test on the server.

Listing 9.4: The lines of Python code where we believe the Gunicorn bug can occur.

```
147     # Deal with the case that it didn't finish successfully on backend
148     if not json_data['was_profiling_successfully_completed?']:
149         submission_of_run.state = "failed"
150         current_run.msg = json_data['msg']
151 
152         check_step = ProfilingStepReady(0)
153         # While-loop for future readability, even though
154         # it would've been "faster" to just create the
155         # ProfilingStepReady corresponding to the value of
156         # json_data['last_successful_step'] instead...
157         while check_step < json_data['last_successful_step']:
158             check_step += 1
159 
160         failed_step = check_step + 1
161         print(getCurrentTimeString() + "Profiling failed during step {} on backend.".format(
                failed_step))
162         # If the failed step is during checking correctness of small tests,
163         if failed_step > 4:
164             submission_of_run.msg = (
165                 "Profiling of uploaded code failed on "
166                 "backend between '{}' and '{}'".format(
167                     check_step.status, failed_step.status))
168         else:
169             submission_of_run.msg = current_run.msg
170 
171         # Save and report
172         db.session.add(backend)
173         db.session.add(current_run)
174         db.session.add(submission_of_run)
175         db.session.commit()
176         response = jsonify(message="Ack")
177         response.status_code = 200
178         return response
179 
180     # Copy over tmpfile with big test(s)'s output from backend,
181     # and save file in local folder cmb-flask/problems/<problem_name>/problemIO/
182     remote_path = json_data['output_file_path']
183     local_path = secure_filename(current_run.submission.problem.name).lower()
184     local_path += "/problemIO/{}_solution.txt".format(run_id)
185     local_path = op.join(op.expandvars(os.environ['UPLOAD_FOLDER']), local_path)
186     try:
187         cmb_ssh_client = connectSSHClient(
```

```
188                 hostname=backend.ip_address, port=22,
189                 username=os.environ['CMB_USER'], private_key_pw=os.environ['
                        CMB_SERVER_SSH_PW'])
190             print(getCurrentTimeString() + "\tMade SSH connection client.")
216         except Exception as e:
217             error_message = "\tFollowing Exception occured during SSH/SCP Operations:\n"
218             error_message += "\t{}".format(e)
219             print(getCurrentTimeString() + error_message)
220         finally:
221             print(getCurrentTimeString() + "\tClosing SSH connection client.")
222             cmb_ssh_client.close()
223 
224             if not op.exists(local_path):
225                 # Should never happen...
226                 print("\n\tFILE '{}' WAS NOT SUCCESSFULLY SCP'ED FROM BACKEND:\
                        n\t\t{}\n".format(remote_path, backend))
227 
228         # Run checker executable (which should be located in directory to which the
229         # tempfile was copied).
230         os.chdir(op.dirname(local_path))
231         execute_chckr_cmd = ["./checker", "input.txt", local_path, "answer.txt"]
232         execute_chckr_dict = execute_os_command(cmd=execute_chckr_cmd, timeout=None)
233 
234         # Parse checker output, see if output passed it or not
235         submission_of_run.state = "finished"
236         chckr_output = execute_chckr_dict['stdout'].split("\n")
237         if (execute_chckr_dict['returncode'] != 0 and str(chckr_output[0]).lower() != "ok"):
238             print(getCurrentTimeString() + "Profiling big test(s) output failed correctness test.")
239             submission_of_run.msg = "Failed big correctness test"
240             submission_of_run.state = "failed"
241 
242             # Save and report
243             backend.last_query = datetime.now()
244             db.session.add(backend)
245             db.session.add(current_run)
246             db.session.add(submission_of_run)
247             db.session.commit()
248             response = jsonify(message="Ack")
249             response.status_code = 200
250             return response
251 
252         os.remove(local_path)
```

We show this code here, to show the reader the only place (and what happens just before) in the proposed system implementation, the feedback message "Failed big correctness test" is set. This is the error which is shown when logging into the web interface of the test-system, after running the Flask web server with Gunicorn and more than one backend polling the server of the test-system.

For any reader with experience reading Python code, it should be evident that there is nothing inherently serial with the code on lines $230 - 250$ of Listing 9.2.3. As such, these lines of code should offer no challenge for Gunicorn workers to execute, no matter what amount of sibling-workers are active, as long as each worker handles the received HTTP request alone.

As Chapter 7 specifies, the tests are run with only "Hello World" or "Shortest Path" problem submissions between each database/system re-set, and each $N \in [1, 12]$ set submits 78 identical uploads of said problem submission. Yet, $1 - 3$ of every 12 sets of 78 "Shortest Path" test submissions fail with the "Failed big correctness test" error.

Thus, since this bug only occurs with "Shortest Path" submissions when the Flask web server is run with Gunicorn, we do not consider it to be likely that the root cause of this bug occurs on the backend.

However, we consider this a rather critical issue to consider for future work on the proposed CMB system implementation of this report.

9.3 Expanding CMB to support language-specific problems/submissions

Section 4.4 proposes changes to the database, to easily facilitate the future support for submissions written in other programming languages than C/C++ to be uploaded. The premise is based on the fact that the administrators of the CMB system must update the database, letting it know what software is present on what backend, and through the software entered into the database, tell the system which backends supports which programming languages.

Additonally, the proposed scheme detailed in Figure 4.1 also facilitates the relationship between a Problem and the permitted languages with which to upload submissions for said Problem, through the SoftwareSet table/relationship with the Problem table. One tip for future developers of the CMB system is to look into how the "abc" Python module can be of help: `https://docs.python.org/3/library/abc.html`

9.3.1 Permitting problem creators to edit C/C++ Makefile

One challenge not detailed in this report, which should also involve any developers working on the front-end user-interface of CMB, is the adding a "Makefile template" for C/C++ Problems in the CMB system. In the past year, it has often been the wish and desire of multiple parties interested in the CMB system to modify what is currently a "Master" Makefile, located in `cmb-flask/Makefile`. This Makefile is used both to test whether or not a submission compiles on the server, in addition to compile and execute the uploaded

submission code on the backends.

As such, it'd be more ideal if some effort could be spent on discovering what compilation parameters are mandatory for a successful profiling of an uploaded submission, and through the admin's web-interface permit CMB administrators to modify the remaining parameters of the Makefile to their desire.

Examples of such differences of desire could be the use of OpenCL vs OpenMP. Or even in a future implementation of CMB, MPI. However, this would require the storing of the strings of which the different Makefiles consist in a central location, such as the database currently in use by the CMB system, on a per-problem basis.

9.4 Remaining future potential improvements

This section details our remaining thoughts on what aspects of the CMB system future developers of the CMB system ought to keep in mind. Both for the sake of making their efforts in developing the system simpler and easier, but also for the robustness and potential future-proofing of the CMB system.

9.4.1 Folder re-structuring

The ulterior motive behind the changes of the proposed system implementation described in Subsection 4.1.1, is to not only consolidate the configuration files/variables of the CMB system into a simple-to-find location but also to provide example as to how the rest of the system might be structured.

For instance, that the `configs` folder should be located in the top-level root-folder of both the `cmb-board` and `cmb-flask` Git repositories. (Likewise with the `scripts` folder).

As such, it's our recommendation that new folders, and files, are not added in "happenstance" locations within the Git repositories mentioned in this report, but that instead a semblance of common structure is strived for.

9.4.2 Adding new architectures/backends to the proposed system implementation

With the detailing of the proposed changes to the backends from Section 4.6, we want to stress that if it ever becomes relevant to add software support for executing profiling of uploaded submissions on a different type of backend/architecture, the `cmb-board` folder structures is designed with that intent in mind.

The import statement at the top of `cmb-board/service.py` can be switched out with different import statements, as long as the Python code containing the new code follow the below two guidelines:

1. The Python code supporting the profiling on a different architecture/backend is located in a folder parallel to the `cmb-flask/odroid_xu3` folder.
2. The Python code supporting the profiling on a different architecture/backend, contains the same function names/analogous steps, as the ones located in the `cmb-flask/odroid_xu3`, so that the only change required is the switching of the import statements at the top of the `cmb-board/service.py` file.

9.4.3 Improving and completing the DB schema in a future-proofing manner

As stated in Section 4.4, we were unable to complete the intended changes for the database schema, especially regarding the relationship between the Software and Backend table, which we intended to be replaced by a relationship between the SoftwareSet and Backend table, as illustrated in Figure 4.1.

Our main struggle with implementing the necessitated changes to the DB schema is that there seems to be a lot of misleading, if not erroneous documentation online, as to how Python code utilizing SQLAlchemy should be written/implemented.

Thus, it is our recommendation for any future developers of the CMB system, to spend the time and effort to learning the difference between implementing Python SQL code statements the *Flask* way, vs. the *SQLAlchemy* way. In our efforts, we experienced time and time again, that the numerous, easy-to-find, examples and tutorials showing how to write SQLAlchemy code in a Flask Python project, often made things more difficult, instead of facilitating the changes desired.

This is also something we have noticed being a recurring issue, in much of the `cmb-flask/source/*.py` code base, and is something we highly recommend future developers of the CMB system to keep in mind when developing CMB. Python is a language in which "the intention is that there's preferably only *one* way to write code correctly" (Peters, 2016), as opposed to Perl, where the notion is that "there are many different ways to correctly solve the problem" (Wall et al., 2000).

9.4.4 Combining the efforts of Sindre Magnussen and this project

As previously mentioned in this report, Sindre Magnussen, the author of (Magnussen, 2015), also worked on his master in parallel with this project, on the CMB system.

While his master focused on the user-interface and user-experience of CMB, the focus

of the master project was to implement changes permitting the CMB system to profile multiple uploaded submissions concurrently. As it stands now, each master resides in separate, divergent Git branches in the same Git repositories.

The Git repositories being `cmb-flask`, and `cmb-board`, links can be found in Chapters 5 and 6. The proposed system implementation of this report, also includes the Git submodule `cmb_utils`[1], introduced in Subection 4.3.1, while Sindre Magnussen's master project implements the use of SocketIO into the communication between the Flask web server and the frontend shown in Figure 4.2.

Ideally, at some point in the future, both our master projects' efforts can be combined, and thus strengthening the CMB system as a whole.

Additionally, as final word on this subject, the Git commit history of this master project (in Git branch "test-chrischa-branch"), will show common history with the current "master" branch in the `cmb-flask` Git repository. This is a consequence of our early efforts to `git pull` changes made by Sindre Magnussen in the CMB project, for his master, into the code base of this master project, so that a future merging of the two will be more easily facilitated.

The majority of the merge efforts were mainly done with regards to the database interactions in the Python code, in addition to the `cmb-flask/source/server.py` and `cmb-flask/source/manager.py` files.

9.4.5 Stabilizing time requirements of the CMB software

Another potential for future work we would like to mention is to minimize the variance in timing in the different steps of the Flask web server.

Figure 8.3a illustrates the problem very well. While the "Hello World" test submission does nothing more than print "Hello World!", all the submissions are identical, so a linear trend like what's depicted in Figure 8.3b is what we expected.

Potential areas of interest to research for these variances can be any of the following non-exhaustive list:

1. Database calls to the MySQL database, located on another machine and communicated with via the LAN network at IDI, NTNU.

2. Idiosyncrasies of Gunicorn with the proposed system implementation.

3. Uncontrollable delays in the live and otherwise-in-use LAN networks at IDI, NTNU.

[1] Whose URL endpoint is `https://bitbucket.org/climbingmontblanc/cmb_utils/`.

4. Weird OS behavior on CMB server or backends, enforcing the use slower memory other than the RAM memory, which in our tests have been shown to have sufficient capacity.

Of the above four potential areas, we consider the first two to be the most likely ones where improvements can be found. The above subsection on improving the DB schema can be kept in mind when considering improving (and making more effective use of) calls to the DB for data in the Python code on the server (`cmb-flask`).

9.4.6 Improving server storage efficiency

Multiple times during the execution of tests for Chapter 8, the VM offered as CMB test-server by IDI's Technical Group got filled up when it approached the 12 to 14 hundreds of uploaded submissions.

It was this that first prompted us to write scripts to easily facilitate the reset of the DB between tests, and deleting all previously uploaded submission files on the CMB test-server.

However, considering that this is a platform that has been used for mandatory assignments counting towards the final grade of University-level courses, and is currently being evaluated as a platform used for C/C++ programming exams for students, we want to suggest a hybrid system.

The hybrid system we envision would unpack and unzip all uploads, automatically remove OS X file system files and other similarly irrelevant files (this could be a hardcoded or periodically/annually-/biannually-updated list), and generate a checksum for all the files in the uploaded submission, much like Git.

Then, if a user submits the same files over and over, e.g. to get averaged energy measurements from the system, the CMB system could just re-utilize the same zip, ignoring the ones the user upload after the first one, due to storing and repeatedly referring to the checksum and the upload that originally created it.

Likewise, this checksum could then also be used to check whether any users (students) upload the same zip, from different accounts.

Additionally, if done right, it could perhaps be possible for the checksum to alert the CMB system how *similar* two uploads are. If this could be achieved, it could help warning the creators of the problem in the CMB system of the submissions whose code is very similar, so that they perhaps warrant an extra close, manual look.

9.4.7 Coverage testing

Finally, again due to the time constraints made by the events described in the Preface, we were unable to write sufficient tests to achieve a test coverage of at minimum 90% of the added Python code.

During the efforts made in this project of upgrading the Python code base from Python 2 to Python 3, the existing unit/integration tests were (all but one - the one integration test written by (Magnussen, 2015) -), working too, and this was after the making (splitting) of the `cmb_utils` folder into a Git submodule. Thus, most of the code now in the submodule, with the exception of some of the recursive code, was also covered by the tests (with a coverage higher than 90%).

As a concluding remark, it is our belief that there is a good chance (though not 100% certain), that the test's code coverage could have increased to at least 75% of the Python code in `cmb-flask`, if this project could have continued for another 4 to 6 weeks as stated in the Preface.

Writing unit tests for the Python code in the `cmb-board` code, may, however, be very difficult, if the goal is to achieve $> 90\%$ code coverage with these. It's our suspicion that it may very well be more trouble than it's worth, to attempt this goal without relying on implementation tests, as opposed to unit tests.

Chapter 10

Conclusion and contribution

This report has documented, described, and proposed a new system implementation for the CMB system, so as to minimize the user-experienced wait-time when there are multiple submissions submitted to the CMB Flask web server, in a short period of time.

The proposed system implementation has been tested with three backends, tests, and backends both described in Chapter 7. Chapter 8 lists and discusses the results of the tests, and hypotheses detailed in Chapter 7.

Our test results from Chapter 8 show that our proposed system implementation does indeed improve the throughput of the amount of submissions the CMB system is able to handle concurrently, with parallelization speedup of two "Shortest Path" submissions giving $\simeq 1.461$ speedup, and ten submissions giving 2.020 speedup, when having three backends in use by the test-system.

Likewise, measured parallelization speedups ranged from 1.541 speedup with two "Shortest Path" submissions using the two fastest backends, up to 1.806 speedup achieved with twelve "Shortest Path" submissions.

These results, though hindered by the troubles of having submissions that should run through the system with the system reporting them as correct instead of erroneous when test-server is executed with Gunicorn, has left us with the belief that a future effort to implement the CMB system which can scale almost linearly with the available resources, can be achieved without gargantuan efforts.

It is our opinion that the CMB system could be very useful in the quest for facilitating the search for more energy efficient software and algorithms, but the CMB system does have potential to improve, as discussed in Chapter 9.

10.1 Contribution

This report details several contributions to the CMB project under Lasse Natvig:

1. The implementation of a Dispatcher, permitting more than one backend to concurrently profile an uploaded problem submission.
   (Described in Section 4.5).
   - And thus re-writing the majority of the bash-scripts causing errors on the CMB system as it was at the outset of this project, into Python 3 code, so as to handle errors and exceptions more smoothly.
2. The upgrade of the code base to Python 3, from Python 2.
   (Described in Section 4.3).
3. The re-structuring of sensitive/secrets such as IP addresses and passwords, as well as the "tidying up" of where these and any other environment variables are located. This to more readily permit current and future developers of the CMB system to quickly find the environment variables (and their data) that they need, and for them to know where new ones should be put.
   (Described in Subsection 4.1.1).
4. Simplified the start-up for `cmb-flask`, and created a similarly simple start-up for `cmb-board`. Also added a simplified start-up sequence for the frontend of the CMB system, for development mode.
   (Shown in Sections 5.3 and 6.3).

Bibliography

Barrett, D. J., Silverman, R. E., 2001. SSH, The Secure Shell: The Definitive Guide. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

BSC, 2011. The Mont Blanc project. `https://www.montblanc-project.eu/`, accessed Jan. 21st 2016.

Bull, 2016. An Atos brand for technology products and software. `http://www.bull.com/`, accessed Jan. 21st 2016.

Fu, N., Li, Y., Liu, B., Xu, H., Zhang, Y., Nov 2015. Realization of controlling emmc 5.0 device based on fpga for automatic test system. In: IEEE AUTOTESTCON, 2015. pp. 251–255.

Gajda, W., 2015. Pro Vagrant, 1st Edition. Apress, Berkely, CA, USA.

Green, B., Seshadri, S., 2013. AngularJS, 1st Edition. O'Reilly Media, Inc.

Gunicorn, 2010. Gunicorn, An Open-Source Project. `http://gunicorn.org/`, accessed Mar. 22nd 2016.

help.ubuntu.com/community/, 2015. SSH/TransferFiles. Accessed Jul. 6th 2016.

help.ubuntu.com/community/, 2016. The Cron Daemon System. `https://help.ubuntu.com/community/CronHowto`, accessed Jul. 2nd 2016.

http://www.crowdsourcing.com/, 2016. http://www.crowdsourcing.com/. `http://www.crowdsourcing.com/`, accessed Jul. 7th 2016.

Kattis, 2016. Kattis. `http://www.kattis.com/`, accessed Jul. 6th 2016.

Magnussen, S., 2015. Improvements, Stability and Handover of Climbing Mont Blanc.

Mark D. Hill and Michael R. Marty, 2008. Amdahl's Law in the Multicore Era. IEEE Computer 41, 33–38.

Merkel, D., Mar. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment.
URL http://dl.acm.org/citation.cfm?id=2600239.2600241

Natvig, L., Follan, T., Støa, S., Magnussen, S., García-Guirado, A., 2015. Climbing Mont Blanc - A Training Site for Energy Efficient Programming on Heterogeneous Multicore Processors. CoRR abs/1511.02240.
URL http://arxiv.org/abs/1511.02240

Peters, T., 2016. The Zen of PythonAccessed Jul. 7th 2016.

Poettering, L., Sievers, K., Hoyer, H., Gundersen, D. M. T., Herrmann, D., 2010. Upstart, An Open-Source Project. https://freedesktop.org/wiki/Software/systemd/, accessed Mar. 22nd 2016.

Qian, Y., 2012. Automatic parallelization tools. In: Proceedings of the World Congress on Engineering and Computer Science. Vol. 1.

Rai, R., 2013. Socket. IO Real-time Web Application Development. Packt Publishing Ltd.

Rajovic, N., Carpenter, P. M., Gelado, I., Puzovic, N., Ramirez, A., Valero, M., 2013. Supercomputing with commodity cpus: Are mobile socs ready for hpc? In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. SC '13. ACM, New York, NY, USA, pp. 40:1–40:12.
URL http://doi.acm.org/10.1145/2503210.2503281

Ronacher, A., 2010. Flask, a Python micro web-framework. (Open-Source Project). http://flask.pocoo.org/, accessed Mar. 22nd 2016.

Solem, A., 2009. Celery, a distributed Task based event queue. (Open-Source Project). http://celery.readthedocs.org/en/, accessed Mar. 22nd 2016.

Støa, S., Follan, T., June 2015. Climbing Mont Blanc. Master's thesis, Department of Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), Norway.

Subramaniam, B., Feng, W.-c., 2010. Understanding Power Measurement Implications in the Green500 List. In: Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing. GREENCOM-CPSCOM '10. IEEE Computer Society, Washington, DC, USA, pp. 245–251.
URL http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.140

Upstart, 2006. systemdd, An Open-Source Project. http://upstart.ubuntu.com/, accessed Mar. 22nd 2016.

Wall, L., Christiansen, T., Orwant, J., 2000. Programming Perl - there's more than one way to do it (3. ed.). O'Reilly.

www.hardkernel.com, 2016. ODROID-XU3. http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127, accessed Jun. 30th 2016.

www.sdcard.org, 2016. Bus Speed (Default Speed/ High Speed/ UHS). https://www.sdcard.org/developers/overview/bus_speed/index.html, accessed Jun. 30th 2016.

YouTube, 2016. YouTube. https://www.youtube.com/results?search_query=Celery+Framework, accessed Jul. 7th 2016.

ZeroMQ, 2011. ZeroMQ. http://zeromq.org/bindings:python, accessed Mar. 21st 2016.

Appendices

Appendix A

Test Set-Up Configs

This appendix contains commented copies of the configurations files used for the tests reported in this thesis, and the "source scripts" used to source the variables in the configuration files before starting the CMB processes.

First, we give the "source scripts" of the server and the backends, to give the user a perspective of how the configurations are all connected. Thereafter, we list and comment on the different configuration files, starting with the files containing the "secret" environment variables for the server and backends, before finishing with the "machine specific" environment variables for the server and backends.

A.1 Source Scripts Files

A.1.1 Server Source Script File

Listing A.1: Environment variables source-script of test-server.

```
1  #! /bin/bash
2
3  set -a
4
5  # Find out where this file is located:
6  THIS_FILE="${BASH_SOURCE[0]}"
7  while [ -h "$THIS_FILE" ]; do # resolve $THIS_FILE until the file is no longer a
       symlink
8          DIR="$( cd -P "$( dirname "$THIS_FILE" )" && pwd )"
9          THIS_FILE="$(readlink "$THIS_FILE")"
10         [[ $THIS_FILE != /* ]] && THIS_FILE="$DIR/$THIS_FILE" # if
               $THIS_FILE was a relative symlink, we need to resolve it relative to the
               path where the symlink file was located
11 done
12
13 # Always assume that structure is cmb/cmb-flask/scripts, cmb/cmb-flask/configs
14 cur_dir=$(pwd)
15 CMB_DIR=$(realpath "$cur_dir/$(dirname $THIS_FILE)/../..")
16 CMB_FLASK_CONFIGS_DIR="$CMB_DIR/cmb-flask/configs"
17
18 # Source passwords, usernames, secrets...
19 SECRETS="$CMB_FLASK_CONFIGS_DIR/test-secrets.cfg"
20 source $SECRETS
21
22 # Needs CMB_DIR variable set(!), and secrets sourced!!!
23 MACHINE_SETTINGS="$CMB_FLASK_CONFIGS_DIR/test-server.cfg"
24 source "$MACHINE_SETTINGS"
```

A.1.2 Backends Source Script File

Listing A.2: Environment variables source-script of backends.

```
1  #!/bin/bash
2
3  set -a
4
5  # Find out where this file is located:
6  THIS_FILE="${BASH_SOURCE[0]}"
7  while [ -h "$THIS_FILE" ]; do # resolve $THIS_FILE until the file is no longer a
       symlink
8      DIR="$( cd -P "$( dirname "$THIS_FILE" )" && pwd )"
9      THIS_FILE="$(readlink "$THIS_FILE")"
10     [[ $THIS_FILE != /* ]] && THIS_FILE="$DIR/$THIS_FILE" # if
           $THIS_FILE was a relative symlink, we need to resolve it relative to the
           path where the symlink file was located
11 done
12
13 # Always assume that structure is cmb-board/scripts
14 cur_dir=$(pwd)
15 CMB_DIR=$(realpath "$cur_dir/$(dirname $THIS_FILE)/..")
16
17 # Source passwords, usernames, secrets...
18 source "$CMB_DIR/configs/test-secrets.cfg"
19
20 # Needs CMB_DIR variable set(!)
21 source "$CMB_DIR/configs/odroid-xu3.cfg"
```

A.2 Secret Environment Variable(s) Config Files

A.2.1 Server Secrets Config File

Listing A.3: Secret/sensitive environment variables of test-server.

```
1  # /bin/bash
2
3  CMB_USER="<redacted Ubuntu OS user name>"
4  CMB_USER_GROUP="<redacted Ubuntu OS user group>"
5
6  CMB_SERVER_SSH_PW="<redacted pw>"
7  BACKEND_SHARED_PW="<redacted pw>"
8
9  CMB_MAIL_PASSWORD="<redacted pw>"
10 CMB_MAIL_USERNAME="<redacted email>"
11
12 CMB_SECRET_KEY="<redacted pw>"
13 CMB_TOKEN_SECRET="<redacted pw>"
14
15 CMB_MYSQL_USER="<redacted user name>"
16 CMB_MYSQL_PASSWORD="<redacted pw>"
17 CMB_MYSQL_DATABASE="cmb_dev-tests"
```

A.2.2 Backends Secrets Config File

Listing A.4: Secret/sensitive environment variables of backends.

```
1 BACKEND_ID=<redacted id #>
2 BACKEND_SSH_PW="<redacted pw>"
3 BACKEND_SHARED_PW="<redacted pw>"
4
5 CMB_USER="<redacted Ubuntu OS user name>"
6 CMB_SERVER_PORT=<redacted port #>
7 CMB_SERVER_SSH_PORT=<redacted port #>
8 CMB_SERVER="<redacted url>"
```

A.3 Machine-Specific Environment Variable(s) Config Files

A.3.1 Server Specific Config File

Listing A.5: Machine-specific environment variables of test-server.

```
1  VERSION="dev"
2  SERVER_PORT=<redacted port #>
3  GUNICORN_LOG_LEVEL="debug"
4
5  CMB_SERVER="0.0.0.0"
6  BOARD_IPs="\
7  <redacted ip>,\
8  <redacted ip>,\
9  <redacted ip>"
10 CMB_MYSQL_SERVER="mysql.idi.ntnu.no"
11
12 MAIL_PORT=<redacted port #>
13 MAIL_USE_SSL=True
14 MAIL_USE_TLS=False
15 MAIL_SERVER="smtp.gmail.com"
16
17 FLASK_DIR="$CMB_DIR/cmb-flask"
18 FLASK_VENV_DIR="$FLASK_DIR/venv"
19 FRONTEND_DIR="$CMB_DIR/cmb-frontend"
20 MALI_DIR="$CMB_DIR/Mali_OpenCL_SDK_v1.1.0"
21
22 CMB_LOG_FOLDER="$FLASK_DIR/logs"
23 UPLOAD_FOLDER="$FLASK_DIR/problems"
24
25 SQLALCHEMY_DATABASE_URI="mysql://$CMB_MYSQL_USER:
       $CMB_MYSQL_PASSWORD@$CMB_MYSQL_SERVER/
       $CMB_MYSQL_DATABASE"
```

A.3.2 Backends Specific Config File

Listing A.6: Machine-specific environment variables of backends.

```
1 VERSION="dev"
2
3 BACKEND_DIR="/home/climber/cmb-board/"
4
5 timeoutLength=90
6 targetTemperature=60
7 samplingInterval=10000
8 temperatureFile="/sys/devices/10060000.tmu/temp"
```

A.4 Test-Server Gunicorn start-script/config

Listing A.7: Test-server's Gunicorn start-script/config.

```
1  #!/bin/bash
2
3  # Find out where this file is located:
4  THIS_FILE="${BASH_SOURCE[0]}"
5  while [ -h "$THIS_FILE" ]; do # resolve $THIS_FILE until the file is no longer a
       symlink
6    DIR="$( cd -P "$( dirname "$THIS_FILE" )" && pwd )"
7    THIS_FILE="$(readlink "$THIS_FILE")"
8    [[ $THIS_FILE != /* ]] && THIS_FILE="$DIR/$THIS_FILE" # if $THIS_FILE
         was a relative symlink, we need to resolve it relative to the path where the
         symlink file was located
9  done
10
11 set -e
12 source $(dirname $THIS_FILE)/test-source_cmb_envvars.sh
13
14 NUM_WORKERS=3
15 NAME="CMB-Flask"
16 mkdir -p $CMB_LOG_FOLDER
17 GUNICORN_LOG="$CMB_LOG_FOLDER/gunicorn.log"
18
19 # Necessary due to the time it takes for checker.cpp to compile and check big test
20 # results on shortest path timing out the default timeout of gunicorn workers
21 GUNICORN_WORKER_TIMEOUT=120
22
23 echo "Starting $NAME"
24
```

```
25 # Start unicorn
26 GUNICORN_CONFIG=$FLASK_DIR/configs/test-gunicorn.cfg
27 if [[ ! -f $GUNICORN_CONFIG ]]; then
28   $FLASK_VENV_DIR/bin/gunicorn manager:app -b $CMB_SERVER:
         $SERVER_PORT \
29     --preload \
30     --name $NAME \
31     --workers $NUM_WORKERS \
32     --chdir=$FLASK_DIR/source \
33     --timeout $GUNICORN_WORKER_TIMEOUT \
34     --user=$CMB_USER --group=$CMB_USER_GROUP \
35     --log-syslog-prefix="$NAME Gunicorn" \
36     --log-level=$GUNICORN_LOG_LEVEL \
37     --access-logfile $GUNICORN_LOG \
38     --log-file=-
39 else
40   # Not implemented due to file not existing
41   exit 1
42 fi
```

Appendix B

SSH Install and Set-Up Note

The way the proposed system implementation of this thesis's project (Chapter 4) has been implemented, has been with care towards security, stability, and feasibility for future enhancements.

Thus, when having to decide upon how to transfer files between the backends and server of the CMB system, the author of this thesis chose to utilize SSH (Barrett and Silverman, 2001). However, to uphold the care mentioned in the previous paragraph, the author of this project decided to have each backend make its own private/public RSA key pair, and have said key pair require a password to unlock/use them, with the password being unique to each key/pair.

With that solution in mind, each of the backends had their own unique password stored in `cmb-board/configs/*secrets.cfg`, and did not need to share any common password for allowing transfer of files between the backends and server (given that `ssh-copy-id` has been used as instructed in Chapters 5 and 6). Hence, each backend could make use of SSH to connect to the server without having access to anything but their own password unlocking its key pair, and likewise the server could connect to each of the backends, for both the purpose of remote command execution, as well as the purpose of file transfer between machines.

The code installed through the instructions given in Chapter 6, specifies how to generate these private/public RSA key pairs, and the dialog that shows up prompts the user to add a password for the utilization of the new key-par. This password is contained within the `BACKEND_SSH_PW` environment variable, given in line 3 of Listing A.2.2, and is required for the proposed system implementation of Chapter 4 to work.

Hence, after the creation of the key pair has been completed at the server and each

backend, it is recommended to follow the ssh−copy−id steps in Chapters 5 and 6. If the backends are unable to retrieve the files for a submission they have been given to profile from the server after this, it is recommended to first attempt the equivalent SCP (help.ubuntu.com/community/, 2015) commands, and attempt to the decipher the root of the problem through this venue.

If the SCP does not luck out (in other words; it works fine with these commands), what can penultimately be attempted, is to execute the Python code manually, line by line, through the command venv/bin/ipython, which should be installed if the instructions of Chapter 6 have been followed.

The final (though some argue this perhaps should be first, not final) venue to attempt is to perform the SCP from a terminal on the backend, adding −v flag to the command, for more debugging output. The −v flag can be appended up to −vvv, for maximum debug output. Conversely, there should be a log on the server specifying what went wrong in the authorization/connection process, if it happened server-side (or the receiver if the problem is from the server to the backend). The typical location of this log on the recipient machine is `/var/log/auth.log`.

Appendix C

Test-VM Specifications

In this appendix, we list the full commands (and outputs), from which the key stats listed in Chapter 7 to describe the CMB test-server VM, was extracted.

First we list the commands and outputs for the OS and Kernel before we finish with the CPU and Memory.

C.1 OS and Kernel Information

Listing C.1: OS and Kernel information of test-server.

```
1 test-user@test-vm:~$ lsb_release -a
2 No LSB modules are available.
3 Distributor ID: Ubuntu
4 Description: Ubuntu 16.04 LTS
5 Release: 16.04
6 Codename: xenial
7 test-user@test-vm:~$ uname -a
8 Linux test-vm 4.4.0-22-generic #40-Ubuntu SMP Thu May 12 22:03:46 UTC 2016
      x86_64 x86_64 x86_64 GNU/Linux
9 test-user@test-vm:~$
```

C.2 CPU Information

Listing C.2: CPU information of test-server.

```
1  test-user@test-vm:~$ lscpu
2  Architecture:  x86_64
3  CPU op-mode(s):  32-bit, 64-bit
4  Byte Order:  Little Endian
5  CPU(s):  3
6  On-line CPU(s) list: 0-2
7  Thread(s) per core: 1
8  Core(s) per socket: 3
9  Socket(s):  1
10 NUMA node(s):  1
11 Vendor ID:  GenuineIntel
12 CPU family:  6
13 Model:  45
14 Model name:  Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
15 Stepping:  7
16 CPU MHz:  1999.998
17 BogoMIPS:  3999.99
18 Hypervisor vendor: Microsoft
19 Virtualization type: full
20 L1d cache:  32K
21 L1i cache:  32K
22 L2 cache:  256K
23 L3 cache:  20480K
24 NUMA node0 CPU(s): 0-2
25 Flags:  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
       clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl eagerfpu
       pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes xsave avx hypervisor lahf_lm
       xsaveopt
26 test-user@test-vm:~$ cat /proc/cpuinfo
27 processor : 0
28 vendor_id : GenuineIntel
29 cpu family : 6
30 model  : 45
31 model name : Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
32 stepping  : 7
33 microcode : 0xffffffff
34 cpu MHz  : 1999.998
35 cache size : 20480 KB
36 physical id : 0
37 siblings  : 3
38 core id  : 0
39 cpu cores : 3
```

```
40 apicid  : 0
41 initial apicid : 0
42 fpu   : yes
43 fpu_exception : yes
44 cpuid level : 13
45 wp   : yes
46 flags  : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
       clflush mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl eagerfpu
       pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes xsave avx hypervisor lahf_lm
       xsaveopt
47 bugs   :
48 bogomips  : 3999.99
49 clflush size : 64
50 cache_alignment : 64
51 address sizes : 42 bits physical, 48 bits virtual
52 power management:
53
54 <!!! Two identical duplicate processors removed for brevity !!>
55
56 test-user@test-vm:~$
```

C.3 Memory Information

Listing C.3: Memory information of test-server.

```
1  test-user@test-vm:~$ sudo blockdev --getsize64 /dev/sda
2  16106127360
3  test-user@test-vm:~$ cat /proc/meminfo
4  MemTotal:  2338692 kB
5  MemFree:  544796 kB
6  MemAvailable: 1325384 kB
7  Buffers:  302720 kB
8  Cached:  393988 kB
9  SwapCached:  31080 kB
10 Active:  652472 kB
11 Inactive:  250932 kB
12 Active(anon): 74500 kB
13 Inactive(anon): 152740 kB
14 Active(file): 577972 kB
15 Inactive(file): 98192 kB
16 Unevictable:  0 kB
17 Mlocked:  0 kB
18 SwapTotal: 2095100 kB
19 SwapFree:  1950868 kB
```

```
20 Dirty:     80 kB
21 Writeback:    0 kB
22 AnonPages:  204388 kB
23 Mapped:    35804 kB
24 Shmem:    20544 kB
25 Slab:    149600 kB
26 SReclaimable: 127284 kB
27 SUnreclaim:  22316 kB
28 KernelStack:  4208 kB
29 PageTables:  11236 kB
30 NFS_Unstable:  0 kB
31 Bounce:     0 kB
32 WritebackTmp:  0 kB
33 CommitLimit: 3264444 kB
34 Committed_AS: 784248 kB
35 VmallocTotal: 34359738367 kB
36 VmallocUsed:  0 kB
37 VmallocChunk:  0 kB
38 HardwareCorrupted: 0 kB
39 AnonHugePages: 24576 kB
40 CmaTotal:    0 kB
41 CmaFree:    0 kB
42 HugePages_Total: 0
43 HugePages_Free:  0
44 HugePages_Rsvd:  0
45 HugePages_Surp:  0
46 Hugepagesize: 2048 kB
47 DirectMap4k: 114624 kB
48 DirectMap2M: 2375680 kB
49 test@test-vm:~$
```

Appendix D

Backends Specifications

In this appendix, we first list all the commands and the full output of the key stats used to describe OS, Kernel, CPU and Memory specifications of the three backends used in the tests of this project.

Thereafter, we list all installed OS packages on the three backends, *dev1*, *dev2*, and *dev3*. First, we list the ones installed on all three, before we, in turn, list the packages installed on both of a pair, in the order as specified by the headings that follow.

Thus, the results from Chapter 8 can be reproduced, since with the packages installed on all three backends, and the packages installed on each pair-wise combination of the backends, it should be straightforward to find which backend had which packages installed.

D.1 OS and Kernel Specifications of Backends

D.1.1 Backend 1

Listing D.1: OS and Kernel information of backend dev1.

```
1 test-user@<redacted>-odroid-xu3-dev1:~:$ lsb_release -a
2 No LSB modules are available.
3 Distributor ID: Ubuntu
4 Description: Ubuntu 14.04.4 LTS
5 Release: 14.04
6 Codename: trusty
7 test-user@<redacted>-odroid-xu3-dev1:~:$ uname -a
```

```
8 Linux <redacted>-odroid-xu3-dev1 3.10.54+ #1 SMP PREEMPT Wed Sep 10
      14:01:26 UTC 2014 armv7l armv7l armv7l GNU/Linux
9 test-user@<redacted>-odroid-xu3-dev1:~:$
```

D.1.2 Backend 2

Listing D.2: OS and Kernel information of backend dev2.

```
1 test-user@<redacted>-odroid-xu3-dev2:~$ lsb_release -a
2 No LSB modules are available.
3 Distributor ID: Ubuntu
4 Description: Ubuntu 14.04.4 LTS
5 Release: 14.04
6 Codename: trusty
7 test-user@<redacted>-odroid-xu3-dev2:~$ uname -a
8 Linux <redacted>-odroid-xu3-dev2 3.10.69 #1 SMP PREEMPT Thu Feb 12
      15:22:14 BRST 2015 armv7l armv7l armv7l GNU/Linux
9 test-user@<redacted>-odroid-xu3-dev2:~$
```

D.1.3 Backend 3

Listing D.3: OS and Kernel information of backend dev3.

```
1 test-user@<redacted>-odroid-xu3-dev3:~$ lsb_release -a
2 No LSB modules are available.
3 Distributor ID: Ubuntu
4 Description: Ubuntu 15.10
5 Release: 15.10
6 Codename: wily
7 test-user@<redacted>-odroid-xu3-dev3:~$ uname -a
8 Linux <redacted>-odroid-xu3-dev3 3.10.96-78 #1 SMP PREEMPT Fri Feb 12
      05:59:25 BRST 2016 armv7l armv7l armv7l GNU/Linux
9 test-user@<redacted>-odroid-xu3-dev3:~$
```

D.2 CPU Specifications of Backends

D.2.1 Backend 1

Listing D.4: CPU information of backend dev1.

```
1  test-user@<redacted>-odroid-xu3-dev1:~:$ lscpu
2  Architecture:  armv7l
3  Byte Order:    Little Endian
4  CPU(s):    8
5  On-line CPU(s) list: 0-7
6  Thread(s) per core: 1
7  Core(s) per socket: 4
8  Socket(s):    2
9  test-user@<redacted>-odroid-xu3-dev1:~:$ cat /proc/cpuinfo
10 processor : 0
11 model name : ARMv7 Processor rev 3 (v7l)
12 BogoMIPS : 84.00
13 Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
14 CPU implementer : 0x41
15 CPU architecture: 7
16 CPU variant : 0x0
17 CPU part : 0xc07
18 CPU revision : 3
19
20 <!!! Three identical duplicate processors removed for brevity !!>
21
22 processor : 4
23 model name : ARMv7 Processor rev 3 (v7l)
24 BogoMIPS : 120.00
25 Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
26 CPU implementer : 0x41
27 CPU architecture: 7
28 CPU variant : 0x2
29 CPU part : 0xc0f
30 CPU revision : 3
31
32 <!!! Three identical duplicate processors removed for brevity !!>
33
34
35 Hardware : ODROID-XU3
36 Revision : 0000
37 Serial  : 0000000000000000
38 test-user@<redacted>-odroid-xu3-dev1:~:$
```

D.2.2 Backend 2

Listing D.5: CPU information of backend dev2.

```
test−user@<redacted>−odroid−xu3−dev2:~$ lscpu
Architecture:  armv7l
Byte Order:    Little Endian
CPU(s):    8
On−line CPU(s) list: 0−7
Thread(s) per core: 1
Core(s) per socket: 4
Socket(s):    2
test−user@<redacted>−odroid−xu3−dev2:~$ cat /proc/cpuinfo
processor : 0
model name : ARMv7 Processor rev 3 (v7l)
BogoMIPS : 84.00
Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part : 0xc07
CPU revision : 3

<!!! Three identical duplicate processors removed for brevity !!>

processor : 4
model name : ARMv7 Processor rev 3 (v7l)
BogoMIPS : 120.00
Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x2
CPU part : 0xc0f
CPU revision : 3

<!!! Three identical duplicate processors removed for brevity !!>

Hardware : ODROID−XU3
Revision : 0000
Serial  : 0000000000000000
test−user@<redacted>−odroid−xu3−dev2:~$
```

D.2.3 Backend 3

Listing D.6: CPU information of backend dev3.

```
test−user@<redacted>−odroid−xu3−dev3:~$ lscpu
Architecture:  armv7l
```

```
3  Byte Order:   Little Endian
4  CPU(s):   8
5  On-line CPU(s) list: 0-7
6  Thread(s) per core: 1
7  Core(s) per socket: 4
8  Socket(s):   2
9  Model name:   ARMv7 Processor rev 3 (v7l)
10 CPU max MHz:  1400.0000
11 CPU min MHz:  200.0000
12 test-user@<redacted>-odroid-xu3-dev3:~$ cat /proc/cpuinfo
13 processor : 0
14 model name : ARMv7 Processor rev 3 (v7l)
15 BogoMIPS : 84.00
16 Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
17 CPU implementer : 0x41
18 CPU architecture: 7
19 CPU variant : 0x0
20 CPU part : 0xc07
21 CPU revision : 3
22
23 <!!! Three identical duplicate processors removed for brevity !!>
24
25 processor : 4
26 model name : ARMv7 Processor rev 3 (v7l)
27 BogoMIPS : 120.00
28 Features : swp half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
29 CPU implementer : 0x41
30 CPU architecture: 7
31 CPU variant : 0x2
32 CPU part : 0xc0f
33 CPU revision : 3
34
35 <!!! Three identical duplicate processors removed for brevity !!>
36
37 Hardware : ODROID-XU3
38 Revision : 0100
39 Serial  : 0000000000000000
40 test-user@<redacted>-odroid-xu3-dev3:~$
```

D.3 Memory Specifications of Backends

D.3.1 Backend 1

Listing D.7: Memory information of backend dev1.

```
1  test-user@<redacted>-odroid-xu3-dev1:~:$ sudo blockdev --getsize64 /dev/
       mmcblk0
2  31268536320
3  test-user@<redacted>-odroid-xu3-dev1:~:$ cat /proc/meminfo
4  MemTotal:  2043084 kB
5  MemFree:  1288048 kB
6  Buffers:  156436 kB
7  Cached:   85240 kB
8  SwapCached:   0 kB
9  Active:  156880 kB
10 Inactive:  120244 kB
11 Active(anon): 35480 kB
12 Inactive(anon): 3200 kB
13 Active(file): 121400 kB
14 Inactive(file): 117044 kB
15 Unevictable:  0 kB
16 Mlocked:   0 kB
17 HighTotal: 1296384 kB
18 HighFree:  1032004 kB
19 LowTotal:  746700 kB
20 LowFree:  256044 kB
21 SwapTotal:   0 kB
22 SwapFree:   0 kB
23 Dirty:    28 kB
24 Writeback:   0 kB
25 AnonPages:  35528 kB
26 Mapped:   14804 kB
27 Shmem:   3220 kB
28 Slab:   312656 kB
29 SReclaimable: 289120 kB
30 SUnreclaim:  23536 kB
31 KernelStack:  2128 kB
32 PageTables:  1116 kB
33 NFS_Unstable:  0 kB
34 Bounce:    0 kB
35 WritebackTmp:  0 kB
36 CommitLimit: 1021540 kB
37 Committed_AS: 497808 kB
38 VmallocTotal: 245760 kB
39 VmallocUsed: 18152 kB
40 VmallocChunk: 103580 kB
41 test-user@<redacted>-odroid-xu3-dev1:~:$
```

D.3.2 Backend 2

Listing D.8: Memory information of backend dev2.

```
1  test-user@<redacted>-odroid-xu3-dev2:~$ sudo blockdev --getsize64 /dev/
       mmcblk0
2  31268536320
3  test-user@<redacted>-odroid-xu3-dev2:~:$ cat /proc/meminfo
4  MemTotal:  2043108 kB
5  MemFree:  1331516 kB
6  Buffers:  128052 kB
7  Cached:  298596 kB
8  SwapCached:  0 kB
9  Active:  360736 kB
10 Inactive:  128956 kB
11 Active(anon): 65572 kB
12 Inactive(anon): 740 kB
13 Active(file): 295164 kB
14 Inactive(file): 128216 kB
15 Unevictable:  0 kB
16 Mlocked:  0 kB
17 HighTotal: 1296384 kB
18 HighFree:  795480 kB
19 LowTotal:  746724 kB
20 LowFree:  536036 kB
21 SwapTotal:  0 kB
22 SwapFree:  0 kB
23 Dirty:  40 kB
24 Writeback:  0 kB
25 AnonPages:  63072 kB
26 Mapped:  15088 kB
27 Shmem:  3264 kB
28 Slab:  55112 kB
29 SReclaimable: 33924 kB
30 SUnreclaim:  21188 kB
31 KernelStack:  1960 kB
32 PageTables:  1292 kB
33 NFS_Unstable:  0 kB
34 Bounce:  0 kB
35 WritebackTmp:  0 kB
36 CommitLimit: 1021552 kB
37 Committed_AS: 323728 kB
38 VmallocTotal: 245760 kB
39 VmallocUsed: 18168 kB
40 VmallocChunk: 103580 kB
41 test-user@<redacted>-odroid-xu3-dev2:~:$
```

D.3.3 Backend 3

Listing D.9: Memory information of backend dev3.

```
1  test-user@<redacted>-odroid-xu3-dev3:~$ sudo blockdev --getsize64 /dev/
       mmcblk0
2  31322013696
3  test-user@<redacted>-odroid-xu3-dev3:~$ cat /proc/meminfo
4  MemTotal:  2038540 kB
5  MemFree:  1438556 kB
6  Buffers:  25408 kB
7  Cached:  308636 kB
8  SwapCached:  0 kB
9  Active:  276032 kB
10 Inactive:  105268 kB
11 Active(anon): 49936 kB
12 Inactive(anon): 20684 kB
13 Active(file): 226096 kB
14 Inactive(file): 84584 kB
15 Unevictable:  0 kB
16 Mlocked:  0 kB
17 HighTotal: 1296384 kB
18 HighFree:  797032 kB
19 LowTotal:  742156 kB
20 LowFree:  641524 kB
21 SwapTotal:  0 kB
22 SwapFree:  0 kB
23 Dirty:  0 kB
24 Writeback:  0 kB
25 AnonPages:  47292 kB
26 Mapped:  13952 kB
27 Shmem:  23364 kB
28 Slab:  52348 kB
29 SReclaimable: 31820 kB
30 SUnreclaim:  20528 kB
31 KernelStack:  1472 kB
32 PageTables:  1268 kB
33 NFS_Unstable:  0 kB
34 Bounce:  0 kB
35 WritebackTmp:  0 kB
36 CommitLimit: 1019268 kB
37 Committed_AS: 342936 kB
38 VmallocTotal: 245760 kB
39 VmallocUsed: 18740 kB
40 VmallocChunk: 103580 kB
41 test-user@<redacted>-odroid-xu3-dev3:~$
```

D.4 Packages installed on all three backends

1. accountsservice
2. acl
3. adduser
4. alsa-base
5. alsa-utils
6. anacron
7. apport
8. apport-gtk
9. apport-symptoms
10. apt
11. aptdaemon
12. aptdaemon-data
13. apt-utils
14. aria2
15. at-spi2-core
16. autoconf
17. automake
18. autopoint
19. autotools-dev
20. avahi-daemon
21. avahi-utils
22. axel
23. base-files
24. base-passwd
25. bash
26. bash-completion
27. bc
28. bind9-host
29. binutils
30. blueman
31. bluez
32. bluez-cups
33. bsdmainutils
34. bsdutils
35. build-essential
36. busybox-initramfs
37. bzip2
38. ca-certificates
39. colord
40. console-setup
41. coreutils
42. cpio
43. cpp
44. cpp-4.9
45. cpp-5
46. cracklib-runtime
47. cron
48. cups
49. cups-browsed
50. cups-bsd
51. cups-client
52. cups-common
53. cups-core-drivers
54. cups-daemon
55. cups-filters
56. cups-filters-core-drivers
57. cups-ppdc
58. cups-server-common
59. curl
60. dash
61. dbus
62. dbus-x11
63. dc
64. dconf-cli
65. dconf-gsettings-backend:armhf
66. dconf-service
67. debconf
68. debconf-i18n
69. debhelper
70. debianutils
71. desktop-file-utils
72. dh-autoreconf
73. dh-python
74. dialog
75. dictionaries-common
76. diffstat
77. diffutils
78. dmsetup
79. dmz-cursor-theme
80. dnsmasq-base
81. dpkg
82. dpkg-dev
83. e2fslibs:armhf
84. e2fsprogs
85. eject
86. ethtool

87. evolution-data-server-common
88. fail2ban
89. ffmpegthumbnailer
90. file
91. findutils
92. fontconfig
93. fontconfig-config
94. fonts-dejavu-core
95. fonts-freefont-ttf
96. fonts-kacst
97. fonts-kacst-one
98. fonts-khmeros-core
99. fonts-lao
100. fonts-lklug-sinhala
101. fonts-sil-abyssinica
102. fonts-sil-padauk
103. fonts-takao-pgothic
104. fonts-thai-tlwg
105. fonts-tibetan-machine
106. fonts-tlwg-garuda
107. fonts-tlwg-kinnari
108. fonts-tlwg-loma
109. fonts-tlwg-mono
110. fonts-tlwg-norasi
111. fonts-tlwg-purisa
112. fonts-tlwg-sawasdee
113. fonts-tlwg-typewriter
114. fonts-tlwg-typist
115. fonts-tlwg-typo
116. fonts-tlwg-umpush
117. fonts-tlwg-waree
118. foomatic-db-compressed-ppds
119. fuse
120. g++
121. g++-4.9
122. g++-5
123. galculator
124. gawk
125. gcc
126. gcc-4.9
127. gcc-4.9-base:armhf
128. gcc-5
129. gcc-5-base:armhf
130. gconf2
131. gconf2-common
132. gconf-service
133. gconf-service-backend
134. gcr
135. gdb
136. genisoimage
137. gettext
138. gettext-base
139. gfortran
140. ghostscript
141. ghostscript-x
142. gir1.2-atk-1.0
143. gir1.2-gnomekeyring-1.0
144. gir1.2-gst-plugins-base-1.0
145. gir1.2-gstreamer-1.0
146. gir1.2-gtk-2.0
147. gir1.2-notify-0.7
148. gir1.2-packagekitglib-1.0
149. gir1.2-soup-2.4
150. git
151. git-man
152. gksu
153. glib-networking:armhf
154. glib-networking-common
155. glib-networking-services
156. glmark2-data
157. glmark2-es2
158. gnome-accessibility-themes
159. gnome-desktop3-data
160. gnome-icon-theme
161. gnome-icon-theme-symbolic
162. gnome-keyring
163. gnome-menus
164. gnome-themes-standard:armhf
165. gnome-themes-standard-data
166. gnome-user-guide
167. gnupg
168. gpgv
169. grep
170. groff-base
171. gsettings-desktop-schemas
172. gsfonts
173. gstreamer1.0-clutter
174. gstreamer1.0-plugins-base:armhf
175. gstreamer1.0-plugins-good:armhf
176. gstreamer1.0-pulseaudio:armhf

177. gstreamer1.0-x:armhf
178. gtk2-engines:armhf
179. gtk2-engines-murrine:armhf
180. gtk2-engines-pixbuf:armhf
181. guvcview
182. gvfs:armhf
183. gvfs-backends
184. gvfs-common
185. gvfs-daemons
186. gvfs-libs:armhf
187. gzip
188. hardening-includes
189. hdparm
190. hicolor-icon-theme
191. hostname
192. hplip
193. hplip-data
194. htop
195. hunspell-en-ca
196. hunspell-en-us
197. hyphen-en-us
198. ibus
199. ibus-gtk3:armhf
200. ibus-gtk:armhf
201. ifupdown
202. imagemagick-common
203. im-config
204. indicator-application
205. initramfs-tools
206. initramfs-tools-bin
207. initscripts
208. init-system-helpers
209. inputattach
210. insserv
211. intltool-debian
212. iproute2
213. iptables
214. iputils-arping
215. iputils-ping
216. isc-dhcp-client
217. isc-dhcp-common
218. iso-codes
219. joe
220. kbd
221. kerneloops-daemon
222. keyboard-configuration
223. klibc-utils
224. kmod
225. language-pack-en
226. language-pack-en-base
227. language-pack-gnome-en
228. language-pack-gnome-en-base
229. language-selector-common
230. language-selector-gnome
231. laptop-detect
232. less
233. libaa1:armhf
234. libaacs0:armhf
235. libaccountsservice0:armhf
236. libacl1:armhf
237. libandroid-properties1
238. libapparmor1:armhf
239. libappindicator3-1
240. libapt-pkg-perl
241. libarchive13:armhf
242. libarchive-extract-perl
243. libarchive-zip-perl
244. libart-2.0-2:armhf
245. libasan1:armhf
246. libasan2:armhf
247. libasn1-8-heimdal:armhf
248. libasound2:armhf
249. libasound2-data
250. libasound2-plugins:armhf
251. libassuan0:armhf
252. libasyncns0:armhf
253. libatasmart4:armhf
254. libatk1.0-0:armhf
255. libatk1.0-data
256. libatk-bridge2.0-0:armhf
257. libatlas3-base
258. libatlas-base-dev
259. libatlas-dev
260. libatomic1:armhf
261. libatspi2.0-0:armhf
262. libattr1:armhf
263. libaudio2:armhf
264. libaudit1:armhf
265. libaudit-common
266. libauthen-sasl-perl

267. libavahi-client3:armhf
268. libavahi-common3:armhf
269. libavahi-common-data:armhf
270. libavahi-core7:armhf
271. libavahi-glib1:armhf
272. libavc1394-0:armhf
273. libbind9-90
274. libblas3
275. libblas-dev
276. libblkid1:armhf
277. libbluetooth3:armhf
278. libbluray1:armhf
279. libbsd0:armhf
280. libburn4
281. libbz2-1.0:armhf
282. libc6:armhf
283. libc6-dbg:armhf
284. libc6-dev:armhf
285. libcaca0:armhf
286. libcairo2:armhf
287. libcairo-gobject2:armhf
288. libcanberra0:armhf
289. libcanberra-gtk0:armhf
290. libcanberra-gtk3-0:armhf
291. libcap2:armhf
292. libcap2-bin
293. libc-ares2:armhf
294. libc-bin
295. libcc1-0:armhf
296. libcddb2
297. libc-dev-bin
298. libcdio13
299. libcdio-cdda1
300. libcdio-paranoia1
301. libcdparanoia0:armhf
302. libcec
303. libcgmanager0:armhf
304. libchromaprint0:armhf
305. libck-connector0:armhf
306. libclass-accessor-perl
307. libclone-perl
308. libcloog-isl4:armhf
309. libclutter-1.0-0:armhf
310. libclutter-gst-2.0-0:armhf
311. libclutter-gtk-1.0-0:armhf
312. libcolamd2.8.0:armhf
313. libcomerr2:armhf
314. libcrack2:armhf
315. libcroco3:armhf
316. libcups2:armhf
317. libcupscgi1:armhf
318. libcupsfilters1:armhf
319. libcupsimage2:armhf
320. libcupsmime1:armhf
321. libcupsppdc1:armhf
322. libcurl3:armhf
323. libcurl3-gnutls:armhf
324. libdatrie1:armhf
325. libdb5.3:armhf
326. libdbus-1-3:armhf
327. libdbus-glib-1-2:armhf
328. libdbusmenu-glib4:armhf
329. libdbusmenu-gtk3-4:armhf
330. libdbusmenu-gtk4:armhf
331. libdc1394-22:armhf
332. libdca0:armhf
333. libdconf1:armhf
334. libdebconfclient0:armhf
335. libdevmapper1.02.1:armhf
336. libdigest-hmac-perl
337. libdirectfb-1.2-9:armhf
338. libdjvulibre21:armhf
339. libdjvulibre-text
340. libdns100
341. libdpkg-perl
342. libdrm2:armhf
343. libdrm-dev:armhf
344. libdrm-exynos1:armhf
345. libdrm-freedreno1:armhf
346. libdrm-nouveau2:armhf
347. libdrm-omap1:armhf
348. libdrm-radeon1:armhf
349. libdv4:armhf
350. libdvdnav4:armhf
351. libdvdread4:armhf
352. libedit2:armhf
353. libegl1-mesa:armhf
354. libelf1:armhf
355. libemail-valid-perl
356. libenca0:armhf

357. libenchant1c2a:armhf
358. libencode-locale-perl
359. liberror-perl
360. libestr0
361. libevent-2.0-5:armhf
362. libexempi3:armhf
363. libexif12:armhf
364. libexpat1:armhf
365. libexpat1-dev:armhf
366. libfaad2:armhf
367. libffi6:armhf
368. libfftw3-double3:armhf
369. libfftw3-single3:armhf
370. libfile-basedir-perl
371. libfile-copy-recursive-perl
372. libfile-listing-perl
373. libflac8:armhf
374. libfontconfig1:armhf
375. libfontembed1:armhf
376. libfontenc1:armhf
377. libfreetype6:armhf
378. libfribidi0:armhf
379. libfuse2:armhf
380. libgail18:armhf
381. libgbm1:armhf
382. libgcc1:armhf
383. libgcc-4.9-dev:armhf
384. libgcc-5-dev:armhf
385. libgck-1-0:armhf
386. libgconf-2-4:armhf
387. libgcr-3-common
388. libgcr-base-3-1:armhf
389. libgcr-ui-3-1:armhf
390. libgd3:armhf
391. libgdbm3:armhf
392. libgdk-pixbuf2.0-0:armhf
393. libgdk-pixbuf2.0-common
394. libgeoclue0:armhf
395. libgeoip1:armhf
396. libgfortran3:armhf
397. libgif4:armhf
398. libgksu2-0
399. libgl1-mesa-dri:armhf
400. libgl1-mesa-glx:armhf
401. libglapi-mesa:armhf
402. libgles1-mesa:armhf
403. libgles2-mesa:armhf
404. libglib2.0-0:armhf
405. libglib2.0-bin
406. libglib2.0-data
407. libglu1-mesa:armhf
408. libgmp10:armhf
409. libgnome-keyring0:armhf
410. libgnome-keyring-common
411. libgnome-menu-3-0
412. libgnutls-openssl27:armhf
413. libgomp1:armhf
414. libgpg-error0:armhf
415. libgpgme11:armhf
416. libgphoto2-6:armhf
417. libgpm2:armhf
418. libgpod4:armhf
419. libgpod-common
420. libgraphite2-3:armhf
421. libgs9
422. libgs9-common
423. libgsm1:armhf
424. libgssapi3-heimdal:armhf
425. libgssapi-krb5-2:armhf
426. libgssdp-1.0-3
427. libgstreamer1.0-0:armhf
428. libgstreamer-plugins-base1.0-0:armhf
429. libgstreamer-plugins-good1.0-0:armhf
430. libgtk2.0-0:armhf
431. libgtk2.0-common
432. libgtk-3-0:armhf
433. libgtk-3-bin
434. libgtk-3-common
435. libgtop2-common
436. libgudev-1.0-0:armhf
437. libgupnp-1.0-4
438. libgusb2:armhf
439. libgutenprint2
440. libgweather-common
441. libgxps2:armhf
442. libhardware2
443. libharfbuzz0b:armhf
444. libharfbuzz-icu0:armhf
445. libhcrypto4-heimdal:armhf
446. libheimbase1-heimdal:armhf

447. libheimntlm0-heimdal:armhf
448. libhpmud0
449. libhtml-parser-perl
450. libhtml-tagset-perl
451. libhtml-tree-perl
452. libhttp-cookies-perl
453. libhttp-date-perl
454. libhttp-message-perl
455. libhttp-negotiate-perl
456. libhunspell-1.3-0:armhf
457. libhx509-5-heimdal:armhf
458. libhybris-common1
459. libibus-1.0-5:armhf
460. libice6:armhf
461. libidn11:armhf
462. libido3-0.1-0:armhf
463. libiec61883-0:armhf
464. libieee1284-3:armhf
465. libimobiledevice4:armhf
466. libindicator3-7
467. libindicator7
468. libio-html-perl
469. libio-pty-perl
470. libio-socket-inet6-perl
471. libio-socket-ssl-perl
472. libio-string-perl
473. libipc-run-perl
474. libipc-system-simple-perl
475. libisc95
476. libisccc90
477. libisccfg90
478. libiso9660-8
479. libisofs6
480. libiw30:armhf
481. libjack-jackd2-0:armhf
482. libjasper1:armhf
483. libjavascriptcoregtk-3.0-0:armhf
484. libjbig0:armhf
485. libjbig2dec0
486. libjpeg8:armhf
487. libjpeg-turbo8:armhf
488. libjs-jquery
489. libjson-c2:armhf
490. libjson-glib-1.0-0:armhf
491. libjson-glib-1.0-common
492. libjte1
493. libk5crypto3:armhf
494. libkate1
495. libkeyutils1:armhf
496. libklibc
497. libkmod2:armhf
498. libkpathsea6
499. libkrb5-26-heimdal:armhf
500. libkrb5-3:armhf
501. libkrb5support0:armhf
502. liblapack3
503. liblapack-dev
504. liblcms2-2:armhf
505. libldap-2.4-2:armhf
506. libldb1:armhf
507. liblightdm-gobject-1-0
508. liblircclient0
509. liblist-moreutils-perl
510. liblocale-gettext-perl
511. liblog-message-simple-perl
512. liblqr-1-0:armhf
513. libltdl7:armhf
514. liblua5.2-0:armhf
515. liblwp-mediatypes-perl
516. liblwp-protocol-https-perl
517. liblwres90
518. liblzma5:armhf
519. liblzo2-2:armhf
520. libmad0:armhf
521. libmagic1:armhf
522. libmailtools-perl
523. libmhash2:armhf
524. libmicrohttpd10
525. libmirclient-dev:armhf
526. libmm-glib0:armhf
527. libmnl0:armhf
528. libmodule-pluggable-perl
529. libmount1:armhf
530. libmp3lame0:armhf
531. libmpc3:armhf
532. libmpdec2:armhf
533. libmpeg2-4:armhf
534. libmpfr4:armhf
535. libmtdev1:armhf
536. libmtp9:armhf

537. libmtp-common
538. libmtp-runtime
539. libmysqlclient18:armhf
540. libnatpmp1
541. libnautilus-extension1a
542. libncurses5:armhf
543. libncursesw5:armhf
544. libneon27-gnutls
545. libnet-dbus-perl
546. libnet-dns-perl
547. libnet-domain-tld-perl
548. libnetfilter-conntrack3:armhf
549. libnet-http-perl
550. libnet-ip-perl
551. libnetpbm10
552. libnet-smtp-ssl-perl
553. libnet-ssleay-perl
554. libnewt0.52:armhf
555. libnfnetlink0:armhf
556. libnih1:armhf
557. libnih-dbus1:armhf
558. libnl-3-200:armhf
559. libnl-genl-3-200:armhf
560. libnl-route-3-200:armhf
561. libnm-gtk-common
562. libnotify4:armhf
563. libnspr4:armhf
564. libnss3:armhf
565. libnss3-nssdb
566. libnss-mdns:armhf
567. libogg0:armhf
568. libopenobex1
569. liborc-0.4-0:armhf
570. libp11-kit0:armhf
571. libp11-kit-gnome-keyring:armhf
572. libpackagekit-glib2-16:armhf
573. libpam0g:armhf
574. libpam-gnome-keyring:armhf
575. libpam-modules:armhf
576. libpam-modules-bin
577. libpam-runtime
578. libpam-systemd:armhf
579. libpango-1.0-0:armhf
580. libpango1.0-0:armhf
581. libpangocairo-1.0-0:armhf
582. libpangoft2-1.0-0:armhf
583. libpangox-1.0-0:armhf
584. libpangoxft-1.0-0:armhf
585. libpaper1:armhf
586. libpaper-utils
587. libparse-debianchangelog-perl
588. libpcap0.8:armhf
589. libpci3:armhf
590. libpciaccess0:armhf
591. libpciaccess-dev:armhf
592. libpcre3:armhf
593. libpcsclite1:armhf
594. libperlio-gzip-perl
595. libpipeline1:armhf
596. libpixman-1-0:armhf
597. libpixman-1-dev
598. libpng12-0:armhf
599. libpod-latex-perl
600. libpolkit-agent-1-0:armhf
601. libpolkit-backend-1-0:armhf
602. libpolkit-gobject-1-0:armhf
603. libpoppler-glib8:armhf
604. libpopt0:armhf
605. libportaudio2:armhf
606. libprocps3:armhf
607. libprotobuf-dev:armhf
608. libptexenc1
609. libpthread-stubs0-dev:armhf
610. libpulse0:armhf
611. libpulsedsp:armhf
612. libpulse-mainloop-glib0:armhf
613. libpurple-bin
614. libpwquality1:armhf
615. libpwquality-common
616. libpython2.7:armhf
617. libpython2.7-dev:armhf
618. libpython2.7-minimal:armhf
619. libpython2.7-stdlib:armhf
620. libpython3.4:armhf
621. libpython3.4-dev:armhf
622. libpython3.4-minimal:armhf
623. libpython3.4-stdlib:armhf
624. libpython3-dev:armhf
625. libpython3-stdlib:armhf
626. libpython-dev:armhf

627. libpython-stdlib:armhf
628. libqt4-dbus:armhf
629. libqt4-declarative:armhf
630. libqt4-designer:armhf
631. libqt4-dev
632. libqt4-dev-bin
633. libqt4-help:armhf
634. libqt4-network:armhf
635. libqt4-opengl:armhf
636. libqt4-opengl-dev
637. libqt4-qt3support:armhf
638. libqt4-script:armhf
639. libqt4-scripttools:armhf
640. libqt4-sql:armhf
641. libqt4-sql-mysql:armhf
642. libqt4-svg:armhf
643. libqt4-test:armhf
644. libqt4-xml:armhf
645. libqt4-xmlpatterns:armhf
646. libqtcore4:armhf
647. libqtdbus4:armhf
648. libqtgui4:armhf
649. libqtwebkit4:armhf
650. libqtwebkit-dev
651. libquvi7:armhf
652. libquvi-scripts
653. libqwt6
654. libqwt-dev
655. libraptor2-0:armhf
656. librasqal3:armhf
657. libraw1394-11:armhf
658. librdf0:armhf
659. libreadline6:armhf
660. libroken18-heimdal:armhf
661. librsvg2-2:armhf
662. librsvg2-common:armhf
663. libsamplerate0:armhf
664. libsane:armhf
665. libsane-common
666. libsane-hpaio
667. libsasl2-2:armhf
668. libsasl2-modules:armhf
669. libsasl2-modules-db:armhf
670. libschroedinger-1.0-0:armhf
671. libsdl1.2debian:armhf
672. libsdl-image1.2:armhf
673. libsecret-1-0:armhf
674. libsecret-common
675. libselinux1:armhf
676. libsemanage1:armhf
677. libsemanage-common
678. libsensors4:armhf
679. libsepol1:armhf
680. libsgutils2-2
681. libshout3:armhf
682. libsigsegv2:armhf
683. libslang2:armhf
684. libsm6:armhf
685. libsmbclient:armhf
686. libsndfile1:armhf
687. libsnmp30:armhf
688. libsnmp-base
689. libsocket6-perl
690. libsoup2.4-1:armhf
691. libsoup-gnome2.4-1:armhf
692. libspectre1:armhf
693. libspeex1:armhf
694. libspeexdsp1:armhf
695. libsqlite3-0:armhf
696. libss2:armhf
697. libssh2-1:armhf
698. libssh-4:armhf
699. libssl1.0.0:armhf
700. libstartup-notification0:armhf
701. libstdc++-4.9-dev:armhf
702. libstdc++-5-dev:armhf
703. libstdc++6:armhf
704. libsub-identify-perl
705. libsub-name-perl
706. libtalloc2:armhf
707. libtasn1-6:armhf
708. libtcl8.6:armhf
709. libtdb1:armhf
710. libterm-ui-perl
711. libtevent0:armhf
712. libtext-charwidth-perl
713. libtext-iconv-perl
714. libtext-levenshtein-perl
715. libtext-soundex-perl
716. libtext-wrapi18n-perl

717. libthai0:armhf
718. libthai-data
719. libtheora0:armhf
720. libtiff5:armhf
721. libtimedate-perl
722. libtinfo5:armhf
723. libtk8.6:armhf
724. libtool
725. libubsan0:armhf
726. libudev1:armhf
727. libudisks2-0:armhf
728. libunistring0:armhf
729. liburi-perl
730. libusb-0.1-4:armhf
731. libusb-1.0-0:armhf
732. libustr-1.0-1:armhf
733. libutempter0
734. libuuid1:armhf
735. libv4l-0:armhf
736. libv4lconvert0:armhf
737. libva1:armhf
738. libvdpau1:armhf
739. libvisual-0.4-0:armhf
740. libvorbis0a:armhf
741. libvorbisenc2:armhf
742. libvorbisfile3:armhf
743. libvte9
744. libvte-common
745. libwavpack1:armhf
746. libwayland-client0:armhf
747. libwayland-cursor0:armhf
748. libwayland-dev
749. libwayland-egl1-mesa:armhf
750. libwayland-server0:armhf
751. libwbclient0:armhf
752. libwebcam0
753. libwebkitgtk-3.0-0:armhf
754. libwebkitgtk-3.0-common
755. libwebp5:armhf
756. libwebpmux1:armhf
757. libwhoopsie0
758. libwind0-heimdal:armhf
759. libwmf0.2-7:armhf
760. libwnck22
761. libwnck-3-0:armhf
762. libwnck-3-common
763. libwnck-common
764. libwrap0:armhf
765. libwww-perl
766. libwww-robotrules-perl
767. libx11-6:armhf
768. libx11-data
769. libx11-dev:armhf
770. libx11-xcb1:armhf
771. libx11-xcb-dev:armhf
772. libxau6:armhf
773. libxau-dev:armhf
774. libxaw7:armhf
775. libxcb1:armhf
776. libxcb1-dev:armhf
777. libxcb-dri2-0:armhf
778. libxcb-dri2-0-dev:armhf
779. libxcb-dri3-0:armhf
780. libxcb-dri3-dev:armhf
781. libxcb-glx0:armhf
782. libxcb-glx0-dev:armhf
783. libxcb-icccm4:armhf
784. libxcb-image0:armhf
785. libxcb-keysyms1:armhf
786. libxcb-present0:armhf
787. libxcb-present-dev:armhf
788. libxcb-randr0:armhf
789. libxcb-randr0-dev:armhf
790. libxcb-render0:armhf
791. libxcb-render0-dev:armhf
792. libxcb-shape0:armhf
793. libxcb-shape0-dev:armhf
794. libxcb-shm0:armhf
795. libxcb-sync1:armhf
796. libxcb-sync-dev:armhf
797. libxcb-xfixes0:armhf
798. libxcb-xfixes0-dev:armhf
799. libxcb-xv0:armhf
800. libxcomposite1:armhf
801. libxcursor1:armhf
802. libxdamage1:armhf
803. libxdamage-dev:armhf
804. libxdmcp6:armhf
805. libxdmcp-dev:armhf
806. libxext6:armhf

807. libxext-dev:armhf
808. libxfixes3:armhf
809. libxfixes-dev:armhf
810. libxfont1:armhf
811. libxft2:armhf
812. libxi6:armhf
813. libxinerama1:armhf
814. libxkbcommon0:armhf
815. libxkbcommon-dev
816. libxkbfile1:armhf
817. libxkbfile-dev:armhf
818. libxklavier16
819. libxml2:armhf
820. libxml-parser-perl
821. libxml-twig-perl
822. libxmu6:armhf
823. libxmuu1:armhf
824. libxpm4:armhf
825. libxrandr2:armhf
826. libxrender1:armhf
827. libxres1:armhf
828. libxshmfence1:armhf
829. libxshmfence-dev:armhf
830. libxslt1.1:armhf
831. libxss1:armhf
832. libxt6:armhf
833. libxtables10
834. libxtst6:armhf
835. libxv1:armhf
836. libxvidcore4:armhf
837. libxxf86dga1:armhf
838. libxxf86vm1:armhf
839. libxxf86vm-dev:armhf
840. libyajl2:armhf
841. libyelp0
842. libzvbi0:armhf
843. libzvbi-common
844. lightdm-gtk-greeter
845. lintian
846. linux-firmware
847. linux-libc-dev:armhf
848. linux-sound-base
849. lm-sensors
850. locales
851. login
852. logrotate
853. lp-solve
854. lsb-base
855. lsb-release
856. lshw
857. m4
858. make
859. makedev
860. man-db
861. mawk
862. mesa-utils
863. mesa-utils-extra
864. mime-support
865. mobile-broadband-provider-info
866. modemmanager
867. module-init-tools
868. mount
869. mscompress
870. multiarch-support
871. mysql-common
872. nautilus-data
873. ncurses-base
874. ncurses-bin
875. netbase
876. netcat-openbsd
877. netpbm
878. net-tools
879. network-manager
880. network-manager-gnome
881. notification-daemon
882. obex-data-server
883. openprinting-ppds
884. openssh-client
885. openssh-server
886. openssh-sftp-server
887. openssl
888. p11-kit
889. p11-kit-modules:armhf
890. parted
891. passwd
892. patch
893. patchutils
894. pciutils
895. pcmciautils
896. perl

897. perl-base
898. perl-modules
899. pkg-config
900. po-debconf
901. policykit-1
902. policykit-desktop-privileges
903. poppler-data
904. poppler-utils
905. powermgmt-base
906. ppp
907. printer-driver-c2esp
908. printer-driver-foo2zjs
909. printer-driver-foo2zjs-common
910. printer-driver-gutenprint
911. printer-driver-hpcups
912. printer-driver-min12xxw
913. printer-driver-pnm2ppa
914. printer-driver-postscript-hp
915. printer-driver-ptouch
916. printer-driver-pxljr
917. printer-driver-sag-gdi
918. printer-driver-splix
919. procps
920. psmisc
921. pulseaudio
922. pulseaudio-module-x11
923. pulseaudio-utils
924. python
925. python2.7
926. python2.7-dev
927. python2.7-minimal
928. python3
929. python3.4
930. python3.4-dev
931. python3.4-minimal
932. python3-apport
933. python3-apt
934. python3-aptdaemon
935. python3-aptdaemon.gtk3widgets
936. python3-aptdaemon.pkcompat
937. python3-chardet
938. python3-dbus
939. python3-debian
940. python3-defer
941. python3-dev
942. python3-distupgrade
943. python3-gi
944. python3-minimal
945. python3-pkg-resources
946. python3-problem-report
947. python3-pycurl
948. python3-six
949. python3-software-properties
950. python3-update-manager
951. python3-xkit
952. python-apt
953. python-apt-common
954. python-cairo
955. python-chardet
956. python-chardet-whl
957. python-colorama-whl
958. python-crypto
959. python-dbus
960. python-dbus-dev
961. python-decorator
962. python-dev
963. python-distlib-whl
964. python-gi
965. python-gobject
966. python-gobject-2
967. python-gtk2
968. python-html5lib-whl
969. python-imaging
970. python-ldb
971. python-minimal
972. python-numpy
973. python-pip-whl
974. python-pkg-resources
975. python-requests
976. python-requests-whl
977. python-samba
978. python-scipy
979. python-setuptools-whl
980. python-six
981. python-six-whl
982. python-talloc
983. python-tdb
984. python-urllib3
985. python-urllib3-whl
986. python-virtualenv

987. qdbus
988. qpdf
989. qt4-default
990. qt4-linguist-tools
991. qt4-qmake
992. qtchooser
993. qtcore4-l10n
994. quilt
995. readline-common
996. resolvconf
997. rfkill
998. rsync
999. rsyslog
1000. samba-common
1001. samba-common-bin
1002. samba-libs:armhf
1003. sane-utils
1004. screen
1005. sed
1006. sensible-utils
1007. sgml-base
1008. shared-mime-info
1009. software-properties-common
1010. software-properties-gtk
1011. sound-theme-freedesktop
1012. ssl-cert
1013. strace
1014. stress
1015. sudo
1016. system-config-printer-common
1017. system-config-printer-gnome
1018. system-config-printer-udev
1019. systemd-shim
1020. sysvinit-utils
1021. sysv-rc
1022. t1utils
1023. tar
1024. tcl
1025. tcl8.6
1026. tex-common
1027. texlive-binaries
1028. time
1029. tk
1030. tk8.6
1031. transmission-common
1032. transmission-gtk
1033. ttf-indic-fonts-core
1034. ttf-ubuntu-font-family
1035. tzdata
1036. u-boot-tools
1037. ubuntu-drivers-common
1038. ubuntu-keyring
1039. ubuntu-release-upgrader-core
1040. ubuntu-release-upgrader-gtk
1041. ucf
1042. udev
1043. udisks2
1044. ufw
1045. unattended-upgrades
1046. unzip
1047. update-inetd
1048. update-manager
1049. update-manager-core
1050. update-notifier
1051. update-notifier-common
1052. upower
1053. usb-modeswitch
1054. usb-modeswitch-data
1055. usbmuxd
1056. usbutils
1057. util-linux
1058. uvcdynctrl
1059. uvcdynctrl-data
1060. vim
1061. vim-common
1062. vim-runtime
1063. vim-tiny
1064. wamerican
1065. wbritish
1066. wget
1067. whiptail
1068. whois
1069. whoopsie
1070. wireless-tools
1071. wpasupplicant
1072. x11-apps
1073. x11-common
1074. x11proto-core-dev
1075. x11proto-damage-dev
1076. x11proto-dri2-dev

1077. x11proto-dri3-dev
1078. x11proto-fixes-dev
1079. x11proto-fonts-dev
1080. x11proto-gl-dev
1081. x11proto-input-dev
1082. x11proto-kb-dev
1083. x11proto-present-dev
1084. x11proto-randr-dev
1085. x11proto-render-dev
1086. x11proto-resource-dev
1087. x11proto-scrnsaver-dev
1088. x11proto-video-dev
1089. x11proto-xext-dev
1090. x11proto-xf86bigfont-dev
1091. x11proto-xf86dri-dev
1092. x11proto-xf86vidmode-dev
1093. x11proto-xinerama-dev
1094. x11-session-utils
1095. x11-utils
1096. x11-xkb-utils
1097. x11-xserver-utils
1098. xauth
1099. xbitmaps
1100. xdg-user-dirs
1101. xdg-utils
1102. xfonts-base
1103. xfonts-encodings
1104. xfonts-scalable
1105. xfonts-utils
1106. xinit
1107. xinput
1108. xkb-data
1109. xml-core
1110. xorg
1111. xorg-docs-core
1112. xorg-sgml-doctools
1113. xserver-common
1114. xserver-xorg
1115. xserver-xorg-dev
1116. xserver-xorg-input-all
1117. xserver-xorg-input-evdev
1118. xserver-xorg-input-synaptics
1119. xserver-xorg-video-all
1120. xserver-xorg-video-fbdev
1121. xterm
1122. xtrans-dev
1123. xutils-dev
1124. xz-utils
1125. yelp
1126. yelp-xsl
1127. zenity
1128. zenity-common
1129. zip
1130. zlib1g:armhf
1131. zlib1g-dev:armhf

D.5 Packages installed on backend 1 and not 2

1. aglfn
2. aptitude
3. aptitude-common
4. apt-xapian-index
5. chromium-browser
6. chromium-browser-l10n
7. clinfo
8. docbook-utils
9. firefox-locale-af
10. firefox-locale-ar
11. firefox-locale-bg
12. firefox-locale-ca
13. firefox-locale-cy
14. firefox-locale-da
15. firefox-locale-de
16. firefox-locale-es
17. firefox-locale-et
18. firefox-locale-eu
19. firefox-locale-fi
20. firefox-locale-fr
21. firefox-locale-fy
22. firefox-locale-ga
23. firefox-locale-gd
24. firefox-locale-he
25. firefox-locale-hr
26. firefox-locale-id

27. firefox-locale-it
28. firefox-locale-ko
29. firefox-locale-lv
30. firefox-locale-mk
31. firefox-locale-ml
32. firefox-locale-mn
33. firefox-locale-mr
34. firefox-locale-ms
35. firefox-locale-nn
36. firefox-locale-nso
37. firefox-locale-oc
38. firefox-locale-or
39. firefox-locale-pa
40. firefox-locale-pl
41. firefox-locale-ro
42. firefox-locale-si
43. firefox-locale-sk
44. firefox-locale-sl
45. firefox-locale-sq
46. firefox-locale-sr
47. firefox-locale-sv
48. firefox-locale-sw
49. firefox-locale-ta
50. firefox-locale-te
51. firefox-locale-th
52. firefox-locale-uk
53. firefox-locale-vi
54. firefox-locale-xh
55. firefox-locale-zh-hans
56. firefox-locale-zh-hant
57. firefox-locale-zu
58. gimp
59. gimp-data
60. gitstats
61. gnuplot-nox
62. groff
63. heirloom-mailx
64. imagemagick
65. jadetex
66. jq
67. language-pack-af
68. language-pack-af-base
69. language-pack-ar
70. language-pack-ar-base
71. language-pack-ast
72. language-pack-ast-base
73. language-pack-bg
74. language-pack-bg-base
75. language-pack-ca
76. language-pack-ca-base
77. language-pack-crh
78. language-pack-crh-base
79. language-pack-cy
80. language-pack-cy-base
81. language-pack-da
82. language-pack-da-base
83. language-pack-de
84. language-pack-de-base
85. language-pack-dv
86. language-pack-dv-base
87. language-pack-el
88. language-pack-el-base
89. language-pack-et
90. language-pack-et-base
91. language-pack-eu
92. language-pack-eu-base
93. language-pack-fi
94. language-pack-fi-base
95. language-pack-fil
96. language-pack-fil-base
97. language-pack-fy
98. language-pack-fy-base
99. language-pack-ga
100. language-pack-ga-base
101. language-pack-gd
102. language-pack-gd-base
103. language-pack-gnome-af
104. language-pack-gnome-af-base
105. language-pack-gnome-ar
106. language-pack-gnome-ar-base
107. language-pack-gnome-ast
108. language-pack-gnome-ast-base
109. language-pack-gnome-crh
110. language-pack-gnome-crh-base
111. language-pack-gnome-cy
112. language-pack-gnome-cy-base
113. language-pack-gnome-de
114. language-pack-gnome-de-base
115. language-pack-gnome-dv
116. language-pack-gnome-dv-base

117. language-pack-gnome-et
118. language-pack-gnome-et-base
119. language-pack-gnome-fi
120. language-pack-gnome-fi-base
121. language-pack-gnome-fil
122. language-pack-gnome-fil-base
123. language-pack-gnome-fy
124. language-pack-gnome-fy-base
125. language-pack-gnome-ga
126. language-pack-gnome-ga-base
127. language-pack-gnome-gd
128. language-pack-gnome-gd-base
129. language-pack-gnome-he
130. language-pack-gnome-he-base
131. language-pack-gnome-hr
132. language-pack-gnome-hr-base
133. language-pack-gnome-ja
134. language-pack-gnome-ja-base
135. language-pack-gnome-ml
136. language-pack-gnome-ml-base
137. language-pack-gnome-ms
138. language-pack-gnome-ms-base
139. language-pack-gnome-nds
140. language-pack-gnome-nds-base
141. language-pack-gnome-ne
142. language-pack-gnome-ne-base
143. language-pack-gnome-nn
144. language-pack-gnome-nn-base
145. language-pack-gnome-sk
146. language-pack-gnome-sk-base
147. language-pack-gnome-so
148. language-pack-gnome-so-base
149. language-pack-gnome-th
150. language-pack-gnome-th-base
151. language-pack-he
152. language-pack-he-base
153. language-pack-hne
154. language-pack-hne-base
155. language-pack-hr
156. language-pack-hr-base
157. language-pack-hsb
158. language-pack-hsb-base
159. language-pack-hu
160. language-pack-hu-base
161. language-pack-id
162. language-pack-id-base
163. language-pack-it
164. language-pack-it-base
165. language-pack-ja
166. language-pack-ja-base
167. language-pack-ko
168. language-pack-ko-base
169. language-pack-mhr
170. language-pack-mhr-base
171. language-pack-mi
172. language-pack-mi-base
173. language-pack-mk
174. language-pack-mk-base
175. language-pack-ml
176. language-pack-ml-base
177. language-pack-mn
178. language-pack-mn-base
179. language-pack-mr
180. language-pack-mr-base
181. language-pack-ms
182. language-pack-ms-base
183. language-pack-mt
184. language-pack-mt-base
185. language-pack-my
186. language-pack-my-base
187. language-pack-nan
188. language-pack-nan-base
189. language-pack-nds
190. language-pack-nds-base
191. language-pack-ne
192. language-pack-ne-base
193. language-pack-nl
194. language-pack-nl-base
195. language-pack-nn
196. language-pack-nn-base
197. language-pack-nso
198. language-pack-nso-base
199. language-pack-oc
200. language-pack-oc-base
201. language-pack-om
202. language-pack-om-base
203. language-pack-or
204. language-pack-or-base
205. language-pack-os
206. language-pack-os-base

207. language-pack-pa
208. language-pack-pa-base
209. language-pack-pap
210. language-pack-pap-base
211. language-pack-pl
212. language-pack-pl-base
213. language-pack-ps
214. language-pack-ps-base
215. language-pack-ro
216. language-pack-ro-base
217. language-pack-rw
218. language-pack-rw-base
219. language-pack-sc
220. language-pack-sc-base
221. language-pack-sd
222. language-pack-sd-base
223. language-pack-se
224. language-pack-se-base
225. language-pack-shs
226. language-pack-shs-base
227. language-pack-si
228. language-pack-si-base
229. language-pack-sk
230. language-pack-sk-base
231. language-pack-sl
232. language-pack-sl-base
233. language-pack-so
234. language-pack-so-base
235. language-pack-sq
236. language-pack-sq-base
237. language-pack-sr
238. language-pack-sr-base
239. language-pack-ss
240. language-pack-ss-base
241. language-pack-st
242. language-pack-st-base
243. language-pack-sv
244. language-pack-sv-base
245. language-pack-sw
246. language-pack-sw-base
247. language-pack-ta
248. language-pack-ta-base
249. language-pack-te
250. language-pack-te-base
251. language-pack-tg
252. language-pack-tg-base
253. language-pack-th
254. language-pack-th-base
255. language-pack-ti
256. language-pack-ti-base
257. language-pack-tk
258. language-pack-tk-base
259. language-pack-tl
260. language-pack-tl-base
261. language-pack-ts
262. language-pack-ts-base
263. language-pack-tt
264. language-pack-tt-base
265. language-pack-ug
266. language-pack-ug-base
267. language-pack-ur
268. language-pack-ur-base
269. language-pack-uz
270. language-pack-uz-base
271. language-pack-ve
272. language-pack-ve-base
273. language-pack-vi
274. language-pack-vi-base
275. language-pack-wa
276. language-pack-wa-base
277. language-pack-wae
278. language-pack-wae-base
279. language-pack-wo
280. language-pack-wo-base
281. language-pack-xh
282. language-pack-xh-base
283. language-pack-yi
284. language-pack-yi-base
285. language-pack-yo
286. language-pack-yo-base
287. language-pack-zh-hant
288. language-pack-zh-hant-base
289. language-pack-zu
290. language-pack-zu-base
291. libbabl-0.1-0:armhf
292. libboost-iostreams1.54.0:armhf
293. libcwidget3
294. libgegl-0.2-0:armhf
295. libgimp2.0
296. libjavascriptcoregtk-1.0-0:armhf

297. liblua5.1-0:armhf
298. libmng2:armhf
299. libwebkitgtk-1.0-0:armhf
300. libwebkitgtk-1.0-common
301. locate
302. ocl-icd-libopencl1:armhf
303. opencl-headers
304. psutils
305. python-dateutil
306. python-matplotlib
307. python-matplotlib-data
308. python-pyparsing
309. python-tk
310. python-tz
311. texlive-base
312. texlive-fonts-recommended
313. texlive-generic-recommended
314. texlive-latex-base
315. texlive-latex-recommended
316. tipa
317. wdiff
318. wkhtmltopdf
319. xmail

D.6 Packages installed on backend 1 and not 3

1. abiword
2. abiword-common
3. abiword-plugin-grammar
4. abiword-plugin-mathview
5. aglfn
6. anthy
7. anthy-common
8. app-install-data
9. aptitude
10. aptitude-common
11. apturl
12. apturl-common
13. apt-xapian-index
14. arduino
15. arduino-core
16. audacious
17. audacious-plugins:armhf
18. audacious-plugins-data
19. autoconf2.13
20. avahi-dnsconfd
21. avrdude
22. avr-libc
23. binutils-avr
24. bison
25. bluez-alsa:armhf
26. ca-certificates-java
27. ccache
28. cdbs
29. checkinstall
30. chromium-browser
31. chromium-browser-l10n
32. chromium-codecs-ffmpeg-extra
33. clinfo
34. cmake
35. cmake-data
36. comerr-dev
37. command-not-found-data
38. consolekit
39. cpp-4.8
40. culmus
41. cups-driver-gutenprint
42. deadbeef
43. default-jre
44. default-jre-headless
45. dh-apparmor
46. dh-translations
47. docbook
48. docbook-dsssl
49. docbook-to-man
50. docbook-utils
51. docbook-xml
52. docbook-xsl
53. evince
54. evince-common
55. extra-xdg-menus
56. faenza-icon-theme
57. fakeroot
58. file-roller

59. filezilla
60. filezilla-common
61. firefox
62. firefox-locale-af
63. firefox-locale-ar
64. firefox-locale-bg
65. firefox-locale-ca
66. firefox-locale-cy
67. firefox-locale-da
68. firefox-locale-de
69. firefox-locale-en
70. firefox-locale-es
71. firefox-locale-et
72. firefox-locale-eu
73. firefox-locale-fi
74. firefox-locale-fr
75. firefox-locale-fy
76. firefox-locale-ga
77. firefox-locale-gd
78. firefox-locale-he
79. firefox-locale-hr
80. firefox-locale-id
81. firefox-locale-it
82. firefox-locale-ko
83. firefox-locale-lv
84. firefox-locale-mk
85. firefox-locale-ml
86. firefox-locale-mn
87. firefox-locale-mr
88. firefox-locale-ms
89. firefox-locale-nn
90. firefox-locale-nso
91. firefox-locale-oc
92. firefox-locale-or
93. firefox-locale-pa
94. firefox-locale-pl
95. firefox-locale-ro
96. firefox-locale-si
97. firefox-locale-sk
98. firefox-locale-sl
99. firefox-locale-sq
100. firefox-locale-sr
101. firefox-locale-sv
102. firefox-locale-sw
103. firefox-locale-ta
104. firefox-locale-te
105. firefox-locale-th
106. firefox-locale-uk
107. firefox-locale-vi
108. firefox-locale-xh
109. firefox-locale-zh-hans
110. firefox-locale-zh-hant
111. firefox-locale-zu
112. flex
113. flite1-dev:armhf
114. fonts-arabeyes
115. fonts-arphic-ukai
116. fonts-arphic-uming
117. fonts-dejavu
118. fonts-dejavu-extra
119. fonts-droid
120. fonts-farsiweb
121. fonts-khmeros
122. fonts-liberation
123. fonts-lyx
124. fonts-manchufont
125. fonts-mgopen
126. fonts-nafees
127. fonts-nanum
128. fonts-nanum-coding
129. fonts-sil-ezra
130. fonts-sil-scheherazade
131. fonts-takao-gothic
132. fonts-takao-mincho
133. fonts-ukij-uyghur
134. fonts-unfonts-core
135. g++-4.8
136. gcc-4.8
137. gcc-4.8-base:armhf
138. gcc-6-base:armhf
139. gcc-avr
140. gdebi
141. gdebi-core
142. gecko-mediaplayer
143. gfortran-4.8
144. giblib1:armhf
145. gimp
146. gimp-data
147. gir1.2-clutter-1.0
148. gir1.2-clutter-gst-2.0

149. gir1.2-cogl-1.0
150. gir1.2-coglpango-1.0
151. gir1.2-freedesktop
152. gir1.2-gconf-2.0
153. gir1.2-gdkpixbuf-2.0
154. gir1.2-glib-2.0
155. gir1.2-gtk-3.0
156. gir1.2-gtkclutter-1.0
157. gir1.2-gudev-1.0
158. gir1.2-ibus-1.0
159. gir1.2-javascriptcoregtk-3.0
160. gir1.2-json-1.0
161. gir1.2-pango-1.0
162. gir1.2-polkit-1.0
163. gir1.2-rsvg-2.0
164. gir1.2-vte-2.90
165. gir1.2-webkit-3.0
166. gir1.2-wnck-3.0
167. gitstats
168. gnome-common
169. gnome-desktop-data
170. gnome-disk-utility
171. gnome-icon-theme-full
172. gnome-mplayer
173. gnome-panel
174. gnome-panel-data
175. gnome-pkg-tools
176. gnome-system-monitor
177. gnome-system-tools
178. gnumeric
179. gnumeric-common
180. gnumeric-doc
181. gnuplot-nox
182. gobject-introspection
183. gparted
184. gperf
185. gpicview
186. groff
187. gsfonts-x11
188. gstreamer0.10-nice:armhf
189. gstreamer0.10-plugins-base:armhf
190. gstreamer0.10-plugins-good:armhf
191. gstreamer0.10-x:armhf
192. gstreamer1.0-alsa:armhf
193. gstreamer1.0-doc
194. gstreamer1.0-libav:armhf
195. gstreamer1.0-plugins-bad:armhf
196. gstreamer1.0-plugins-bad-doc
197. gstreamer1.0-plugins-base-apps
198. gstreamer1.0-plugins-base-doc
199. gstreamer1.0-plugins-good-doc
200. gstreamer1.0-tools
201. gtk3-engines-unico:armhf
202. gtk-doc-tools
203. gtk-im-libthai:armhf
204. gucharmap
205. gvfs-fuse
206. hardening-wrapper
207. hardinfo
208. heirloom-mailx
209. hunspell-ar
210. hunspell-be
211. hunspell-da
212. hunspell-de-at
213. hunspell-de-ch
214. hunspell-de-de
215. hunspell-eu-es
216. hunspell-fr
217. hunspell-fr-classical
218. hunspell-gl-es
219. hunspell-hu
220. hunspell-ko
221. hunspell-ml
222. hunspell-ne
223. hunspell-ro
224. hunspell-ru
225. hunspell-sr
226. hunspell-sv-se
227. hunspell-uz
228. hunspell-vi
229. hyphen-af
230. hyphen-as
231. hyphen-bn
232. hyphen-ca
233. hyphen-de
234. hyphen-fr
235. hyphen-gu
236. hyphen-hi
237. hyphen-hr
238. hyphen-hu

239. hyphen-it
240. hyphen-kn
241. hyphen-mr
242. hyphen-pa
243. hyphen-pl
244. hyphen-ro
245. hyphen-ru
246. hyphen-sl
247. hyphen-sr
248. hyphen-ta
249. hyphen-te
250. hyphen-zu
251. ibus-anthy
252. ibus-chewing
253. ibus-hangul
254. ibus-m17n
255. ibus-sunpinyin
256. ibus-table
257. ibus-table-cangjie3
258. ibus-table-cangjie5
259. ibus-table-quick-classic
260. ibus-table-wubi
261. ibus-unikey
262. icedtea-7-jre-jamvm:armhf
263. indicator-application-gtk2
264. intltool
265. iotop
266. jade
267. jadetex
268. java-common
269. jq
270. krb5-multidev
271. ladspa-sdk
272. language-pack-af
273. language-pack-af-base
274. language-pack-ar
275. language-pack-ar-base
276. language-pack-ast
277. language-pack-ast-base
278. language-pack-bg
279. language-pack-bg-base
280. language-pack-ca
281. language-pack-ca-base
282. language-pack-crh
283. language-pack-crh-base
284. language-pack-cy
285. language-pack-cy-base
286. language-pack-da
287. language-pack-da-base
288. language-pack-de
289. language-pack-de-base
290. language-pack-dv
291. language-pack-dv-base
292. language-pack-el
293. language-pack-el-base
294. language-pack-et
295. language-pack-et-base
296. language-pack-eu
297. language-pack-eu-base
298. language-pack-fi
299. language-pack-fi-base
300. language-pack-fil
301. language-pack-fil-base
302. language-pack-fy
303. language-pack-fy-base
304. language-pack-ga
305. language-pack-ga-base
306. language-pack-gd
307. language-pack-gd-base
308. language-pack-gnome-af
309. language-pack-gnome-af-base
310. language-pack-gnome-ar
311. language-pack-gnome-ar-base
312. language-pack-gnome-ast
313. language-pack-gnome-ast-base
314. language-pack-gnome-crh
315. language-pack-gnome-crh-base
316. language-pack-gnome-cy
317. language-pack-gnome-cy-base
318. language-pack-gnome-de
319. language-pack-gnome-de-base
320. language-pack-gnome-dv
321. language-pack-gnome-dv-base
322. language-pack-gnome-et
323. language-pack-gnome-et-base
324. language-pack-gnome-fi
325. language-pack-gnome-fi-base
326. language-pack-gnome-fil
327. language-pack-gnome-fil-base
328. language-pack-gnome-fy

329. language-pack-gnome-fy-base
330. language-pack-gnome-ga
331. language-pack-gnome-ga-base
332. language-pack-gnome-gd
333. language-pack-gnome-gd-base
334. language-pack-gnome-he
335. language-pack-gnome-he-base
336. language-pack-gnome-hr
337. language-pack-gnome-hr-base
338. language-pack-gnome-ja
339. language-pack-gnome-ja-base
340. language-pack-gnome-ml
341. language-pack-gnome-ml-base
342. language-pack-gnome-ms
343. language-pack-gnome-ms-base
344. language-pack-gnome-nds
345. language-pack-gnome-nds-base
346. language-pack-gnome-ne
347. language-pack-gnome-ne-base
348. language-pack-gnome-nn
349. language-pack-gnome-nn-base
350. language-pack-gnome-sk
351. language-pack-gnome-sk-base
352. language-pack-gnome-so
353. language-pack-gnome-so-base
354. language-pack-gnome-th
355. language-pack-gnome-th-base
356. language-pack-he
357. language-pack-he-base
358. language-pack-hne
359. language-pack-hne-base
360. language-pack-hr
361. language-pack-hr-base
362. language-pack-hsb
363. language-pack-hsb-base
364. language-pack-hu
365. language-pack-hu-base
366. language-pack-id
367. language-pack-id-base
368. language-pack-it
369. language-pack-it-base
370. language-pack-ja
371. language-pack-ja-base
372. language-pack-ko
373. language-pack-ko-base
374. language-pack-mhr
375. language-pack-mhr-base
376. language-pack-mi
377. language-pack-mi-base
378. language-pack-mk
379. language-pack-mk-base
380. language-pack-ml
381. language-pack-ml-base
382. language-pack-mn
383. language-pack-mn-base
384. language-pack-mr
385. language-pack-mr-base
386. language-pack-ms
387. language-pack-ms-base
388. language-pack-mt
389. language-pack-mt-base
390. language-pack-my
391. language-pack-my-base
392. language-pack-nan
393. language-pack-nan-base
394. language-pack-nb
395. language-pack-nb-base
396. language-pack-nds
397. language-pack-nds-base
398. language-pack-ne
399. language-pack-ne-base
400. language-pack-nl
401. language-pack-nl-base
402. language-pack-nn
403. language-pack-nn-base
404. language-pack-nso
405. language-pack-nso-base
406. language-pack-oc
407. language-pack-oc-base
408. language-pack-om
409. language-pack-om-base
410. language-pack-or
411. language-pack-or-base
412. language-pack-os
413. language-pack-os-base
414. language-pack-pa
415. language-pack-pa-base
416. language-pack-pap
417. language-pack-pap-base
418. language-pack-pl

419. language-pack-pl-base
420. language-pack-ps
421. language-pack-ps-base
422. language-pack-ro
423. language-pack-ro-base
424. language-pack-rw
425. language-pack-rw-base
426. language-pack-sc
427. language-pack-sc-base
428. language-pack-sd
429. language-pack-sd-base
430. language-pack-se
431. language-pack-se-base
432. language-pack-shs
433. language-pack-shs-base
434. language-pack-si
435. language-pack-si-base
436. language-pack-sk
437. language-pack-sk-base
438. language-pack-sl
439. language-pack-sl-base
440. language-pack-so
441. language-pack-so-base
442. language-pack-sq
443. language-pack-sq-base
444. language-pack-sr
445. language-pack-sr-base
446. language-pack-ss
447. language-pack-ss-base
448. language-pack-st
449. language-pack-st-base
450. language-pack-sv
451. language-pack-sv-base
452. language-pack-sw
453. language-pack-sw-base
454. language-pack-ta
455. language-pack-ta-base
456. language-pack-te
457. language-pack-te-base
458. language-pack-tg
459. language-pack-tg-base
460. language-pack-th
461. language-pack-th-base
462. language-pack-ti
463. language-pack-ti-base
464. language-pack-tk
465. language-pack-tk-base
466. language-pack-tl
467. language-pack-tl-base
468. language-pack-ts
469. language-pack-ts-base
470. language-pack-tt
471. language-pack-tt-base
472. language-pack-ug
473. language-pack-ug-base
474. language-pack-ur
475. language-pack-ur-base
476. language-pack-uz
477. language-pack-uz-base
478. language-pack-ve
479. language-pack-ve-base
480. language-pack-vi
481. language-pack-vi-base
482. language-pack-wa
483. language-pack-wa-base
484. language-pack-wae
485. language-pack-wae-base
486. language-pack-wo
487. language-pack-wo-base
488. language-pack-xh
489. language-pack-xh-base
490. language-pack-yi
491. language-pack-yi-base
492. language-pack-yo
493. language-pack-yo-base
494. language-pack-zh-hant
495. language-pack-zh-hant-base
496. language-pack-zu
497. language-pack-zu-base
498. leafpad
499. liba52-0.7.4
500. libaa1-dev
501. libabiword-3.0:armhf
502. libamd2.3.1:armhf
503. libanthy0:armhf
504. libapt-inst1.5:armhf
505. libapt-pkg4.12:armhf
506. libart-2.0-dev
507. libasan0:armhf
508. libasound2-dev:armhf

509. libaspell15
510. libasprintf0c2:armhf
511. libass4:armhf
512. libass-dev:armhf
513. libatk1.0-dev
514. libatk-bridge2.0-dev:armhf
515. libatkmm-1.6-1:armhf
516. libatk-wrapper-java
517. libatk-wrapper-java-jni:armhf
518. libatomic-ops-dev
519. libaudclient2:armhf
520. libaudcore1:armhf
521. libaudit-dev
522. libautodie-perl
523. libavahi-client-dev
524. libavahi-common-dev
525. libavahi-glib-dev
526. libavc1394-dev:armhf
527. libavcodec54:armhf
528. libavcodec-dev
529. libavformat54:armhf
530. libavformat-dev
531. libavresample1:armhf
532. libavutil52:armhf
533. libavutil-dev
534. libbabl-0.1-0:armhf
535. libbinio1ldbl:armhf
536. libbison-dev:armhf
537. libbluetooth-dev
538. libbonobo2-0:armhf
539. libbonobo2-common
540. libbonobo2-dev:armhf
541. libbonoboui2-0:armhf
542. libbonoboui2-common
543. libbonoboui2-dev:armhf
544. libboost1.54-dev
545. libboost-atomic1.54.0:armhf
546. libboost-atomic1.54-dev:armhf
547. libboost-chrono1.54.0:armhf
548. libboost-chrono1.54-dev:armhf
549. libboost-date-time1.54.0:armhf
550. libboost-date-time1.54-dev:armhf
551. libboost-dev
552. libboost-iostreams1.54.0:armhf
553. libboost-serialization1.54.0:armhf
554. libboost-serialization1.54-dev:armhf
555. libboost-system1.54.0:armhf
556. libboost-system1.54-dev:armhf
557. libboost-thread1.54.0:armhf
558. libboost-thread1.54-dev:armhf
559. libboost-thread-dev:armhf
560. libbs2b0
561. libbz2-dev:armhf
562. libcaca-dev
563. libcairo2-dev
564. libcairomm-1.0-1:armhf
565. libcairo-perl
566. libcairo-script-interpreter2:armhf
567. libcamd2.3.1:armhf
568. libcamel-1.2-45
569. libcanberra-dev:armhf
570. libcap-dev:armhf
571. libcap-ng0
572. libccolamd2.8.0:armhf
573. libcdaudio1
574. libcdaudio-dev
575. libcddb2-dev
576. libcdio-dev
577. libcdparanoia-dev:armhf
578. libcdt5
579. libcgraph6
580. libchamplain-0.12-0:armhf
581. libchamplain-gtk-0.12-0:armhf
582. libchewing3:armhf
583. libchewing3-data:armhf
584. libcholmod2.1.2:armhf
585. libchromaprint-dev
586. libclutter-1.0-dev
587. libclutter-gst-2.0-dev
588. libclutter-gtk-1.0-dev
589. libcogl15:armhf
590. libcogl-dev
591. libcogl-pango15:armhf
592. libcogl-pango-dev
593. libcolord1:armhf
594. libcolorhug1:armhf
595. libcompfaceg1
596. libcue1
597. libcurl4-gnutls-dev:armhf
598. libcvaux-dev:armhf

599. libcv-dev:armhf
600. libcwidget3
601. libdaemon0
602. libdbus-1-dev:armhf
603. libdbus-glib-1-dev
604. libdc1394-22-dev:armhf
605. libdca-dev:armhf
606. libdirac-decoder0:armhf
607. libdirac-dev:armhf
608. libdirac-encoder0:armhf
609. libdirectfb-dev
610. libdirectfb-extra:armhf
611. libdiscid0:armhf
612. libdjvulibre-dev:armhf
613. libdmx1:armhf
614. libdmx-dev:armhf
615. libdv4-dev:armhf
616. libdvdnav-dev:armhf
617. libdvdread-dev:armhf
618. libebook-contacts-1.2-0
619. libecal-1.2-16
620. libedataserver-1.2-18
621. libegl1-mesa-dev
622. libegl1-mesa-drivers:armhf
623. libelfg0:armhf
624. libenca-dev
625. libenchant-voikko:armhf
626. libept1.4.12:armhf
627. libevdocument3-4
628. libevview3-3
629. libexempi-dev:armhf
630. libexif-dev
631. libexo-1-0:armhf
632. libexo-common
633. libexo-helpers
634. libfaad-dev:armhf
635. libfakeroot:armhf
636. libfarstream-0.1-0:armhf
637. libffi-dev:armhf
638. libffmpegthumbnailer4
639. libfftw3-bin
640. libfftw3-dev:armhf
641. libflac-dev:armhf
642. libfl-dev:armhf
643. libflite1:armhf
644. libfluidsynth1:armhf
645. libfluidsynth-dev:armhf
646. libfm4
647. libfm-data
648. libfm-extra4
649. libfm-gtk4
650. libfm-gtk-data
651. libfm-modules
652. libfontconfig1-dev
653. libfontenc-dev:armhf
654. libframe6:armhf
655. libfreetype6-dev
656. libfribidi-dev
657. libfs6:armhf
658. libftdi1:armhf
659. libgail-3-0:armhf
660. libgail-common:armhf
661. libgail-dev
662. libgbm-dev
663. libgcc-4.8-dev:armhf
664. libgconf2-4:armhf
665. libgconf2-dev
666. libgcrypt11:armhf
667. libgcrypt11-dev
668. libgda-5.0-4
669. libgda-5.0-common
670. libgdk-pixbuf2.0-dev
671. libgdome2-0
672. libgdome2-cpp-smart0c2a
673. libgegl-0.2-0:armhf
674. libgeis1:armhf
675. libgfortran-4.8-dev:armhf
676. libgif-dev
677. libgimp2.0
678. libgirepository-1.0-1
679. libgirepository1.0-dev
680. libgl1-mesa-dev
681. libglade2-0:armhf
682. libgles1-mesa-dev
683. libgles2-mesa-dev
684. libglib2.0-dev
685. libglib2.0-doc
686. libglibmm-2.4-1c2a:armhf
687. libglib-perl
688. libglu1-mesa-dev

689. libgme0
690. libgme-dev
691. libgmlib1:armhf
692. libgmp3-dev
693. libgmp-dev:armhf
694. libgmpxx4ldbl:armhf
695. libgmtk1:armhf
696. libgmtk1-data
697. libgnome2-0:armhf
698. libgnome2-bin
699. libgnome2-common
700. libgnome2-dev:armhf
701. libgnome-bluetooth11
702. libgnomecanvas2-0:armhf
703. libgnomecanvas2-common
704. libgnomecanvas2-dev:armhf
705. libgnome-desktop-3-7
706. libgnome-keyring-dev
707. libgnomeui-0:armhf
708. libgnomeui-common
709. libgnomeui-dev:armhf
710. libgnomevfs2-0:armhf
711. libgnomevfs2-common
712. libgnomevfs2-dev:armhf
713. libgnutls26:armhf
714. libgnutls-dev
715. libgnutlsxx27:armhf
716. libgoffice-0.10-10
717. libgoffice-0.10-10-common
718. libgpg-error-dev
719. libgphoto2-port10:armhf
720. libgrail6
721. libgraphviz-dev
722. libgrip0
723. libgsf-1-114
724. libgsf-1-common
725. libgsl0-dev
726. libgsl0ldbl
727. libgsm1-dev:armhf
728. libgssrpc4:armhf
729. libgstreamer0.10-0:armhf
730. libgstreamer1.0-dev
731. libgstreamer-plugins-bad1.0-0:armhf
732. libgstreamer-plugins-bad1.0-dev
733. libgstreamer-plugins-base0.10-0:armhf
734. libgstreamer-plugins-base1.0-dev
735. libgstreamer-plugins-good1.0-dev
736. libgtk2.0-dev
737. libgtk2-perl
738. libgtk-3-dev
739. libgtkmathview0c2a
740. libgtkmm-2.4-1c2a:armhf
741. libgtkmm-3.0-1:armhf
742. libgtkspell0
743. libgtop2-7
744. libgucharmap-2-90-7
745. libgudev-1.0-dev
746. libguess1:armhf
747. libgupnp-igd-1.0-4:armhf
748. libgvc6
749. libgvpr2
750. libgweather-3-6
751. libhangul1:armhf
752. libhangul-data
753. libharfbuzz-dev
754. libharfbuzz-gobject0:armhf
755. libhighgui-dev:armhf
756. libhogweed2:armhf
757. libical1
758. libice-dev:armhf
759. libicu52:armhf
760. libid3tag0
761. libid3tag0-dev
762. libidl0:armhf
763. libidl-common
764. libidl-dev:armhf
765. libidn11-dev
766. libiec61883-dev
767. libijs-0.35
768. libilmbase6:armhf
769. libilmbase-dev
770. libimage-exiftool-perl
771. libimlib2
772. libimlib2-dev
773. libiptcdata0
774. libiptcdata0-dev
775. libisl15:armhf
776. libiso9660-dev
777. libiw-dev:armhf

778. libjack-jackd2-dev:armhf
779. libjasper-dev
780. libjbig-dev:armhf
781. libjna-java
782. libjpeg8-dev:armhf
783. libjpeg-dev:armhf
784. libjpeg-progs
785. libjpeg-turbo8-dev:armhf
786. libjpeg-turbo-progs
787. libjson0:armhf
788. libjson-glib-dev
789. libkadm5clnt-mit9:armhf
790. libkadm5srv-mit9:armhf
791. libkate-dev
792. libkdb5-7:armhf
793. libkrb5-dev
794. liblavfile-2.1-0
795. liblavjpeg-2.1-0
796. liblavplay-2.1-0
797. liblcms2-dev:armhf
798. libldap2-dev:armhf
799. liblink-grammar4
800. libllvm3.4:armhf
801. liblockfile1:armhf
802. liblockfile-bin
803. libloudmouth1-0
804. liblqr-1-0-dev
805. libltdl-dev:armhf
806. liblzma-dev:armhf
807. liblzo2-dev:armhf
808. libm17n-0
809. libmad0-dev
810. libmagick++5:armhf
811. libmagickcore5:armhf
812. libmagickcore5-extra:armhf
813. libmagickcore-dev
814. libmagick++-dev
815. libmagickwand5:armhf
816. libmagickwand-dev
817. libmbim-glib0:armhf
818. libmeanwhile1
819. libmenu-cache3
820. libmenu-cache-bin
821. libmessaging-menu0
822. libmetacity-private0a
823. libmicrohttpd-dev
824. libmikmod2:armhf
825. libmikmod2-dev:armhf
826. libmimic0
827. libmimic-dev
828. libminiupnpc8
829. libmirclient7:armhf
830. libmirclientplatform-mesa:armhf
831. libmirprotobuf0:armhf
832. libmirprotobuf-dev:armhf
833. libmjpegtools-dev
834. libmjpegutils-2.1-0
835. libmms0:armhf
836. libmms-dev:armhf
837. libmodplug1
838. libmodplug-dev
839. libmowgli2:armhf
840. libmp3lame-dev:armhf
841. libmpcdec6
842. libmpcdec-dev
843. libmpeg2-4-dev:armhf
844. libmpeg2encpp-2.1-0
845. libmpeg3-1
846. libmpeg3-dev
847. libmpg123-0:armhf
848. libmpg123-dev:armhf
849. libmplex2-2.1-0
850. libmusicbrainz3-6
851. libmysqlclient-dev
852. libncurses5-dev:armhf
853. libnettle4:armhf
854. libnfs1:armhf
855. libnfs-dev:armhf
856. libnice10:armhf
857. libnm-glib4
858. libnm-glib-vpn1
859. libnm-gtk0
860. libnm-util2
861. libnotify-bin
862. libnotify-dev
863. libnss3-1d:armhf
864. libntdb1:armhf
865. libobrender29
866. libobt2
867. libofa0

868. libofa0-dev
869. libogg-dev:armhf
870. libonig2
871. liboobs-1-5
872. libopenal1:armhf
873. libopenal-data
874. libopenal-dev:armhf
875. libopencv2.4-java
876. libopencv2.4-jni
877. libopencv-calib3d2.4:armhf
878. libopencv-calib3d-dev:armhf
879. libopencv-contrib2.4:armhf
880. libopencv-contrib-dev:armhf
881. libopencv-core2.4:armhf
882. libopencv-core-dev:armhf
883. libopencv-dev
884. libopencv-features2d2.4:armhf
885. libopencv-features2d-dev:armhf
886. libopencv-flann2.4:armhf
887. libopencv-flann-dev:armhf
888. libopencv-gpu2.4:armhf
889. libopencv-gpu-dev:armhf
890. libopencv-highgui2.4:armhf
891. libopencv-highgui-dev:armhf
892. libopencv-imgproc2.4:armhf
893. libopencv-imgproc-dev:armhf
894. libopencv-legacy2.4:armhf
895. libopencv-legacy-dev:armhf
896. libopencv-ml2.4:armhf
897. libopencv-ml-dev:armhf
898. libopencv-objdetect2.4:armhf
899. libopencv-objdetect-dev:armhf
900. libopencv-ocl2.4:armhf
901. libopencv-ocl-dev:armhf
902. libopencv-photo2.4:armhf
903. libopencv-photo-dev:armhf
904. libopencv-stitching2.4:armhf
905. libopencv-stitching-dev:armhf
906. libopencv-superres2.4:armhf
907. libopencv-superres-dev:armhf
908. libopencv-ts2.4:armhf
909. libopencv-ts-dev:armhf
910. libopencv-video2.4:armhf
911. libopencv-video-dev:armhf
912. libopencv-videostab2.4:armhf
913. libopencv-videostab-dev:armhf
914. libopenexr6:armhf
915. libopenexr-dev
916. libopenjpeg2:armhf
917. libopenjpeg-dev
918. libopenvg1-mesa:armhf
919. libopts25:armhf
920. libopus0
921. libopus-dev
922. liborbit-2-0:armhf
923. liborbit2:armhf
924. liborbit2-dev
925. liborc-0.4-dev
926. libotf0:armhf
927. libots0
928. libp11-kit-dev
929. libpam-cap:armhf
930. libpanel-applet-4-0
931. libpango1.0-dev
932. libpangomm-1.4-1:armhf
933. libpango-perl
934. libparted0debian1:armhf
935. libpathplan4
936. libpcre3-dev:armhf
937. libpcrecpp0:armhf
938. libperl5.18
939. libpisock9
940. libplist1:armhf
941. libplist-dev
942. libplymouth2:armhf
943. libpng12-dev
944. libpolkit-agent-1-dev
945. libpolkit-gobject-1-dev
946. libpoppler44:armhf
947. libpopt-dev:armhf
948. libpostproc52
949. libprotobuf8:armhf
950. libprotobuf-lite8:armhf
951. libproxy1:armhf
952. libpulse-dev:armhf
953. libpurple0
954. libqmi-glib0:armhf
955. libqpdf13:armhf
956. libquicktime2:armhf
957. librarian0

958. libraw1394-dev:armhf
959. libreadline5:armhf
960. libreadline6-dev:armhf
961. libreadline-dev:armhf
962. librsvg2-dev
963. librtmp0:armhf
964. librtmp-dev
965. librxtx-java
966. libsamplerate0-dev:armhf
967. libsbc1:armhf
968. libsbc-dev:armhf
969. libschroedinger-dev:armhf
970. libsdl1.2-dev
971. libsdl-gfx1.2-4:armhf
972. libsdl-gfx1.2-dev:armhf
973. libsdl-image1.2-dev:armhf
974. libsdl-mixer1.2:armhf
975. libsdl-mixer1.2-dev:armhf
976. libselinux1-dev:armhf
977. libsepol1-dev
978. libsgmls-perl
979. libshout3-dev:armhf
980. libsidplayfp:armhf
981. libsigc++-2.0-0c2a:armhf
982. libslang2-dev:armhf
983. libsmbclient-dev:armhf
984. libsm-dev:armhf
985. libsndfile1-dev
986. libsoundtouch0:armhf
987. libsoundtouch-dev
988. libsoup2.4-dev
989. libsp1c2
990. libspandsp2
991. libspandsp-dev
992. libspeex-dev:armhf
993. libsqlite0
994. libsqlite3-dev:armhf
995. libsrtp0
996. libsrtp0-dev
997. libssh2-1-dev:armhf
998. libssh-dev
999. libssl-dev:armhf
1000. libstartup-notification0-dev:armhf
1001. libstdc++-4.8-dev:armhf
1002. libsunpinyin3:armhf
1003. libswscale2:armhf
1004. libswscale-dev
1005. libsystemd-daemon0:armhf
1006. libsystemd-login0:armhf
1007. libt1-5
1008. libtag1c2a:armhf
1009. libtag1-dev
1010. libtag1-vanilla:armhf
1011. libtagc0:armhf
1012. libtagc0-dev
1013. libtasn1-6-dev
1014. libtelepathy-glib0:armhf
1015. libtheora-dev:armhf
1016. libtidy-0.99-0
1017. libtiff5-dev:armhf
1018. libtiffxx5:armhf
1019. libtinfo-dev:armhf
1020. libtinyxml2.6.2:armhf
1021. libtinyxml-dev:armhf
1022. libts-0.0-0:armhf
1023. libudev-dev
1024. libumfpack5.6.2:armhf
1025. libuniconf4.6
1026. libupower-glib1:armhf
1027. libusb-1.0-0-dev:armhf
1028. libusb-dev
1029. libusbmuxd2
1030. libv4l2rds0:armhf
1031. libv4l-dev:armhf
1032. libvisual-0.4-dev
1033. libvncserver0:armhf
1034. libvo-aacenc0:armhf
1035. libvo-aacenc-dev:armhf
1036. libvo-amrwbenc0:armhf
1037. libvo-amrwbenc-dev:armhf
1038. libvoikko1:armhf
1039. libvorbis-dev:armhf
1040. libvpx1:armhf
1041. libvpx-dev:armhf
1042. libvte-2.90-9
1043. libvte-2.90-common
1044. libwavpack-dev:armhf
1045. libwebpdemux1:armhf
1046. libwebp-dev:armhf
1047. libwildmidi1:armhf

1048. libwildmidi-config
1049. libwildmidi-dev
1050. libwmf-dev
1051. libwpd-0.9-9
1052. libwpg-0.2-2
1053. libwps-0.2-2
1054. libwv-1.2-4:armhf
1055. libwvstreams4.6-base
1056. libwvstreams4.6-extras
1057. libwxbase2.8-0:armhf
1058. libwxgtk2.8-0:armhf
1059. libx264-142:armhf
1060. libxapian22
1061. libxaw7-dev:armhf
1062. libxcb-icccm4-dev:armhf
1063. libxcb-image0-dev:armhf
1064. libxcb-keysyms1-dev:armhf
1065. libxcb-shm0-dev:armhf
1066. libxcb-util0:armhf
1067. libxcb-util0-dev:armhf
1068. libxcb-xf86dri0:armhf
1069. libxcb-xf86dri0-dev:armhf
1070. libxcb-xv0-dev:armhf
1071. libxcomposite-dev
1072. libxcursor-dev:armhf
1073. libxdot4
1074. libxfce4ui-1-0
1075. libxfce4ui-2-0
1076. libxfce4ui-2-dev
1077. libxfce4ui-common
1078. libxfce4util6
1079. libxfce4util-common
1080. libxfce4util-dev
1081. libxfconf-0-2
1082. libxfconf-0-dev
1083. libxfont-dev
1084. libxft-dev
1085. libxi-dev
1086. libxinerama-dev:armhf
1087. libxml2-dev:armhf
1088. libxml2-utils
1089. libxmu-dev:armhf
1090. libxmu-headers
1091. libxmuu-dev:armhf
1092. libxp6:armhf
1093. libxpm-dev:armhf
1094. libxrandr-dev:armhf
1095. libxrender-dev:armhf
1096. libxres-dev
1097. libxslt1-dev:armhf
1098. libxt-dev:armhf
1099. libxtst-dev:armhf
1100. libxv-dev:armhf
1101. libxvidcore-dev:armhf
1102. libyajl-dev
1103. libzbar0
1104. libzbar-dev
1105. libzephyr4:armhf
1106. libzip2
1107. libzip-dev
1108. libzvbi-dev:armhf
1109. light-locker
1110. light-locker-settings
1111. link-grammar-dictionaries-en
1112. localepurge
1113. locate
1114. lockfile-progs
1115. lsof
1116. luatex
1117. lubuntu-artwork
1118. lubuntu-artwork-14-04
1119. lubuntu-icon-theme
1120. lubuntu-lxpanel-icons
1121. lubuntu-software-center
1122. lxappearance
1123. lxappearance-obconf
1124. lxde-common
1125. lxde-core
1126. lxinput
1127. lxlauncher
1128. lxmenu-data
1129. lxpanel
1130. lxpanel-indicator-applet-plugin
1131. lxrandr
1132. lxsession
1133. lxsession-data
1134. lxsession-default-apps
1135. lxsession-edit
1136. lxsession-logout
1137. lxshortcut

1138. lxtask
1139. lxterminal
1140. lynx
1141. lynx-cur
1142. m17n-contrib
1143. m17n-db
1144. mc
1145. mc-data
1146. medit
1147. mesa-common-dev
1148. metacity
1149. metacity-common
1150. minicom
1151. mircommon-dev:armhf
1152. mountall
1153. mplayer2
1154. mtpaint
1155. nano
1156. nettle-dev
1157. ntp
1158. ntpdate
1159. obconf
1160. ocl-icd-libopencl1:armhf
1161. openbox
1162. opencl-headers
1163. openjdk-7-jre:armhf
1164. openjdk-7-jre-headless:armhf
1165. oracle-java8-installer
1166. orbit2
1167. pastebinit
1168. pavucontrol
1169. pcmanfm
1170. perl-doc
1171. pidgin
1172. pidgin-data
1173. pidgin-libnotify
1174. plymouth
1175. plymouth-label
1176. plymouth-theme-lubuntu-logo
1177. plymouth-theme-lubuntu-text
1178. pm-utils
1179. psutils
1180. python-aptdaemon
1181. python-aptdaemon.gtk3widgets
1182. python-colorama
1183. python-commandnotfound
1184. python-cups
1185. python-cupshelpers
1186. python-dateutil
1187. python-debian
1188. python-defer
1189. python-distlib
1190. python-gconf
1191. python-gdbm
1192. python-glade2
1193. python-gnomekeyring
1194. python-gudev
1195. python-html5lib
1196. python-libxml2
1197. python-mako
1198. python-markupsafe
1199. python-matplotlib
1200. python-matplotlib-data
1201. python-notify
1202. python-ntdb
1203. python-pexpect
1204. python-pil
1205. python-pip
1206. python-psutil
1207. python-pycurl
1208. python-pyinotify
1209. python-pyparsing
1210. python-pysqlite2
1211. python-renderpm
1212. python-reportlab
1213. python-reportlab-accel
1214. python-scour
1215. python-setuptools
1216. python-smbc
1217. python-sqlite
1218. python-support
1219. python-tk
1220. python-tz
1221. python-wheel
1222. python-xapian
1223. python-xdg
1224. rarian-compat
1225. realpath
1226. scrot
1227. sessioninstaller

1228. sgml-data
1229. sgmlspl
1230. simple-scan
1231. smbclient
1232. sp
1233. sunpinyin-data
1234. swig
1235. swig2.0
1236. sylpheed
1237. sylpheed-doc
1238. sylpheed-i18n
1239. sylpheed-plugins
1240. synaptic
1241. systemd-services
1242. system-tools-backends
1243. texlive-fonts-recommended
1244. texlive-generic-recommended
1245. texlive-latex-base
1246. texlive-latex-recommended
1247. tipa
1248. transfig
1249. transmission
1250. tsconf
1251. ttf-bengali-fonts
1252. ttf-devanagari-fonts
1253. ttf-gujarati-fonts
1254. ttf-kannada-fonts
1255. ttf-malayalam-fonts
1256. ttf-oriya-fonts
1257. ttf-punjabi-fonts
1258. ttf-tamil-fonts
1259. ttf-telugu-fonts
1260. tzdata-java
1261. ubuntu-extras-keyring
1262. upstart
1263. valgrind
1264. voikko-fi
1265. wbulgarian
1266. wdiff
1267. wfrench
1268. wirish
1269. witalian
1270. wkhtmltopdf
1271. wmanx
1272. wogerman
1273. wspanish
1274. wswedish
1275. wvdial
1276. x11proto-bigreqs-dev
1277. x11proto-composite-dev
1278. x11proto-dmx-dev
1279. x11proto-record-dev
1280. x11proto-xcmisc-dev
1281. x11proto-xf86dga-dev
1282. x11vnc
1283. x11vnc-data
1284. x11-xfs-utils
1285. xarchiver
1286. xdg-user-dirs-gtk
1287. xfburn
1288. xfce4-dev-tools
1289. xfce4-notifyd
1290. xfce4-power-manager
1291. xfce4-power-manager-data
1292. xfconf
1293. xfonts-100dpi
1294. xmail
1295. xmlto
1296. xpad
1297. xscreensaver
1298. xscreensaver-data
1299. xscreensaver-data-extra
1300. xscreensaver-screensaver-bsod
1301. xserver-xorg-input-multitouch
1302. xserver-xorg-video-modesetting
1303. xserver-xorg-video-omap
1304. xserver-xorg-video-vesa
1305. xsltproc
1306. xul-ext-mozvoikko
1307. xul-ext-ubufox
1308. xvfb
1309. yasm

D.7 Packages installed on backend 2 and not 1

1. aspell
2. aspell-en
3. firefox-locale-nb
4. language-pack-gnome-nb
5. language-pack-gnome-nb-base
6. libisl10:armhf
7. wnorwegian

D.8 Packages installed on backend 2 and not 3

1. abiword
2. abiword-common
3. abiword-plugin-grammar
4. abiword-plugin-mathview
5. anthy
6. anthy-common
7. app-install-data
8. apturl
9. apturl-common
10. arduino
11. arduino-core
12. audacious
13. audacious-plugins:armhf
14. audacious-plugins-data
15. autoconf2.13
16. avahi-dnsconfd
17. avrdude
18. avr-libc
19. binutils-avr
20. bison
21. bluez-alsa:armhf
22. ca-certificates-java
23. ccache
24. cdbs
25. checkinstall
26. chromium-codecs-ffmpeg-extra
27. cmake
28. cmake-data
29. comerr-dev
30. command-not-found-data
31. consolekit
32. cpp-4.8
33. culmus
34. cups-driver-gutenprint
35. deadbeef
36. default-jre
37. default-jre-headless
38. dh-apparmor
39. dh-translations
40. docbook
41. docbook-dsssl
42. docbook-to-man
43. docbook-xml
44. docbook-xsl
45. evince
46. evince-common
47. extra-xdg-menus
48. faenza-icon-theme
49. fakeroot
50. file-roller
51. filezilla
52. filezilla-common
53. firefox
54. firefox-locale-en
55. firefox-locale-nb
56. flex
57. flite1-dev:armhf
58. fonts-arabeyes
59. fonts-arphic-ukai
60. fonts-arphic-uming
61. fonts-dejavu
62. fonts-dejavu-extra
63. fonts-droid
64. fonts-farsiweb
65. fonts-khmeros
66. fonts-liberation
67. fonts-lyx
68. fonts-manchufont
69. fonts-mgopen
70. fonts-nafees
71. fonts-nanum
72. fonts-nanum-coding
73. fonts-sil-ezra
74. fonts-sil-scheherazade

75. fonts-takao-gothic
76. fonts-takao-mincho
77. fonts-ukij-uyghur
78. fonts-unfonts-core
79. g++-4.8
80. gcc-4.8
81. gcc-4.8-base:armhf
82. gcc-6-base:armhf
83. gcc-avr
84. gdebi
85. gdebi-core
86. gecko-mediaplayer
87. gfortran-4.8
88. giblib1:armhf
89. gir1.2-clutter-1.0
90. gir1.2-clutter-gst-2.0
91. gir1.2-cogl-1.0
92. gir1.2-coglpango-1.0
93. gir1.2-freedesktop
94. gir1.2-gconf-2.0
95. gir1.2-gdkpixbuf-2.0
96. gir1.2-glib-2.0
97. gir1.2-gtk-3.0
98. gir1.2-gtkclutter-1.0
99. gir1.2-gudev-1.0
100. gir1.2-ibus-1.0
101. gir1.2-javascriptcoregtk-3.0
102. gir1.2-json-1.0
103. gir1.2-pango-1.0
104. gir1.2-polkit-1.0
105. gir1.2-rsvg-2.0
106. gir1.2-vte-2.90
107. gir1.2-webkit-3.0
108. gir1.2-wnck-3.0
109. gnome-common
110. gnome-desktop-data
111. gnome-disk-utility
112. gnome-icon-theme-full
113. gnome-mplayer
114. gnome-panel
115. gnome-panel-data
116. gnome-pkg-tools
117. gnome-system-monitor
118. gnome-system-tools
119. gnumeric
120. gnumeric-common
121. gnumeric-doc
122. gobject-introspection
123. gparted
124. gperf
125. gpicview
126. gsfonts-x11
127. gstreamer0.10-nice:armhf
128. gstreamer0.10-plugins-base:armhf
129. gstreamer0.10-plugins-good:armhf
130. gstreamer0.10-x:armhf
131. gstreamer1.0-alsa:armhf
132. gstreamer1.0-doc
133. gstreamer1.0-libav:armhf
134. gstreamer1.0-plugins-bad:armhf
135. gstreamer1.0-plugins-bad-doc
136. gstreamer1.0-plugins-base-apps
137. gstreamer1.0-plugins-base-doc
138. gstreamer1.0-plugins-good-doc
139. gstreamer1.0-tools
140. gtk3-engines-unico:armhf
141. gtk-doc-tools
142. gtk-im-libthai:armhf
143. gucharmap
144. gvfs-fuse
145. hardening-wrapper
146. hardinfo
147. hunspell-ar
148. hunspell-be
149. hunspell-da
150. hunspell-de-at
151. hunspell-de-ch
152. hunspell-de-de
153. hunspell-eu-es
154. hunspell-fr
155. hunspell-fr-classical
156. hunspell-gl-es
157. hunspell-hu
158. hunspell-ko
159. hunspell-ml
160. hunspell-ne
161. hunspell-ro
162. hunspell-ru
163. hunspell-sr
164. hunspell-sv-se

165. hunspell-uz
166. hunspell-vi
167. hyphen-af
168. hyphen-as
169. hyphen-bn
170. hyphen-ca
171. hyphen-de
172. hyphen-fr
173. hyphen-gu
174. hyphen-hi
175. hyphen-hr
176. hyphen-hu
177. hyphen-it
178. hyphen-kn
179. hyphen-mr
180. hyphen-pa
181. hyphen-pl
182. hyphen-ro
183. hyphen-ru
184. hyphen-sl
185. hyphen-sr
186. hyphen-ta
187. hyphen-te
188. hyphen-zu
189. ibus-anthy
190. ibus-chewing
191. ibus-hangul
192. ibus-m17n
193. ibus-sunpinyin
194. ibus-table
195. ibus-table-cangjie3
196. ibus-table-cangjie5
197. ibus-table-quick-classic
198. ibus-table-wubi
199. ibus-unikey
200. icedtea-7-jre-jamvm:armhf
201. indicator-application-gtk2
202. intltool
203. iotop
204. jade
205. java-common
206. krb5-multidev
207. ladspa-sdk
208. language-pack-gnome-nb
209. language-pack-gnome-nb-base
210. language-pack-nb
211. language-pack-nb-base
212. leafpad
213. liba52-0.7.4
214. libaa1-dev
215. libabiword-3.0:armhf
216. libamd2.3.1:armhf
217. libanthy0:armhf
218. libapt-inst1.5:armhf
219. libapt-pkg4.12:armhf
220. libart-2.0-dev
221. libasan0:armhf
222. libasound2-dev:armhf
223. libaspell15
224. libasprintf0c2:armhf
225. libass4:armhf
226. libass-dev:armhf
227. libatk1.0-dev
228. libatk-bridge2.0-dev:armhf
229. libatkmm-1.6-1:armhf
230. libatk-wrapper-java
231. libatk-wrapper-java-jni:armhf
232. libatomic-ops-dev
233. libaudclient2:armhf
234. libaudcore1:armhf
235. libaudit-dev
236. libautodie-perl
237. libavahi-client-dev
238. libavahi-common-dev
239. libavahi-glib-dev
240. libavc1394-dev:armhf
241. libavcodec54:armhf
242. libavcodec-dev
243. libavformat54:armhf
244. libavformat-dev
245. libavresample1:armhf
246. libavutil52:armhf
247. libavutil-dev
248. libbinio1ldbl:armhf
249. libbison-dev:armhf
250. libbluetooth-dev
251. libbonobo2-0:armhf
252. libbonobo2-common
253. libbonobo2-dev:armhf
254. libbonoboui2-0:armhf

255. libbonoboui2-common
256. libbonoboui2-dev:armhf
257. libboost1.54-dev
258. libboost-atomic1.54.0:armhf
259. libboost-atomic1.54-dev:armhf
260. libboost-chrono1.54.0:armhf
261. libboost-chrono1.54-dev:armhf
262. libboost-date-time1.54.0:armhf
263. libboost-date-time1.54-dev:armhf
264. libboost-dev
265. libboost-serialization1.54.0:armhf
266. libboost-serialization1.54-dev:armhf
267. libboost-system1.54.0:armhf
268. libboost-system1.54-dev:armhf
269. libboost-thread1.54.0:armhf
270. libboost-thread1.54-dev:armhf
271. libboost-thread-dev:armhf
272. libbs2b0
273. libbz2-dev:armhf
274. libcaca-dev
275. libcairo2-dev
276. libcairomm-1.0-1:armhf
277. libcairo-perl
278. libcairo-script-interpreter2:armhf
279. libcamd2.3.1:armhf
280. libcamel-1.2-45
281. libcanberra-dev:armhf
282. libcap-dev:armhf
283. libcap-ng0
284. libccolamd2.8.0:armhf
285. libcdaudio1
286. libcdaudio-dev
287. libcddb2-dev
288. libcdio-dev
289. libcdparanoia-dev:armhf
290. libcdt5
291. libcgraph6
292. libchamplain-0.12-0:armhf
293. libchamplain-gtk-0.12-0:armhf
294. libchewing3:armhf
295. libchewing3-data:armhf
296. libcholmod2.1.2:armhf
297. libchromaprint-dev
298. libclutter-1.0-dev
299. libclutter-gst-2.0-dev
300. libclutter-gtk-1.0-dev
301. libcogl15:armhf
302. libcogl-dev
303. libcogl-pango15:armhf
304. libcogl-pango-dev
305. libcolord1:armhf
306. libcolorhug1:armhf
307. libcompfaceg1
308. libcue1
309. libcurl4-gnutls-dev:armhf
310. libcvaux-dev:armhf
311. libcv-dev:armhf
312. libdaemon0
313. libdbus-1-dev:armhf
314. libdbus-glib-1-dev
315. libdc1394-22-dev:armhf
316. libdca-dev:armhf
317. libdirac-decoder0:armhf
318. libdirac-dev:armhf
319. libdirac-encoder0:armhf
320. libdirectfb-dev
321. libdirectfb-extra:armhf
322. libdiscid0:armhf
323. libdjvulibre-dev:armhf
324. libdmx1:armhf
325. libdmx-dev:armhf
326. libdv4-dev:armhf
327. libdvdnav-dev:armhf
328. libdvdread-dev:armhf
329. libebook-contacts-1.2-0
330. libecal-1.2-16
331. libedataserver-1.2-18
332. libegl1-mesa-dev
333. libegl1-mesa-drivers:armhf
334. libelfg0:armhf
335. libenca-dev
336. libenchant-voikko:armhf
337. libept1.4.12:armhf
338. libevdocument3-4
339. libevview3-3
340. libexempi-dev:armhf
341. libexif-dev
342. libexo-1-0:armhf
343. libexo-common
344. libexo-helpers

345. libfaad-dev:armhf
346. libfakeroot:armhf
347. libfarstream-0.1-0:armhf
348. libffi-dev:armhf
349. libffmpegthumbnailer4
350. libfftw3-bin
351. libfftw3-dev:armhf
352. libflac-dev:armhf
353. libfl-dev:armhf
354. libflite1:armhf
355. libfluidsynth1:armhf
356. libfluidsynth-dev:armhf
357. libfm4
358. libfm-data
359. libfm-extra4
360. libfm-gtk4
361. libfm-gtk-data
362. libfm-modules
363. libfontconfig1-dev
364. libfontenc-dev:armhf
365. libframe6:armhf
366. libfreetype6-dev
367. libfribidi-dev
368. libfs6:armhf
369. libftdi1:armhf
370. libgail-3-0:armhf
371. libgail-common:armhf
372. libgail-dev
373. libgbm-dev
374. libgcc-4.8-dev:armhf
375. libgconf2-4:armhf
376. libgconf2-dev
377. libgcrypt11:armhf
378. libgcrypt11-dev
379. libgda-5.0-4
380. libgda-5.0-common
381. libgdk-pixbuf2.0-dev
382. libgdome2-0
383. libgdome2-cpp-smart0c2a
384. libgeis1:armhf
385. libgfortran-4.8-dev:armhf
386. libgif-dev
387. libgirepository-1.0-1
388. libgirepository1.0-dev
389. libgl1-mesa-dev
390. libglade2-0:armhf
391. libgles1-mesa-dev
392. libgles2-mesa-dev
393. libglib2.0-dev
394. libglib2.0-doc
395. libglibmm-2.4-1c2a:armhf
396. libglib-perl
397. libglu1-mesa-dev
398. libgme0
399. libgme-dev
400. libgmlib1:armhf
401. libgmp3-dev
402. libgmp-dev:armhf
403. libgmpxx4ldbl:armhf
404. libgmtk1:armhf
405. libgmtk1-data
406. libgnome2-0:armhf
407. libgnome2-bin
408. libgnome2-common
409. libgnome2-dev:armhf
410. libgnome-bluetooth11
411. libgnomecanvas2-0:armhf
412. libgnomecanvas2-common
413. libgnomecanvas2-dev:armhf
414. libgnome-desktop-3-7
415. libgnome-keyring-dev
416. libgnomeui-0:armhf
417. libgnomeui-common
418. libgnomeui-dev:armhf
419. libgnomevfs2-0:armhf
420. libgnomevfs2-common
421. libgnomevfs2-dev:armhf
422. libgnutls26:armhf
423. libgnutls-dev
424. libgnutlsxx27:armhf
425. libgoffice-0.10-10
426. libgoffice-0.10-10-common
427. libgpg-error-dev
428. libgphoto2-port10:armhf
429. libgrail6
430. libgraphviz-dev
431. libgrip0
432. libgsf-1-114
433. libgsf-1-common
434. libgsl0-dev

435. libgsl0ldbl
436. libgsm1-dev:armhf
437. libgssrpc4:armhf
438. libgstreamer0.10-0:armhf
439. libgstreamer1.0-dev
440. libgstreamer-plugins-bad1.0-0:armhf
441. libgstreamer-plugins-bad1.0-dev
442. libgstreamer-plugins-base0.10-0:armhf
443. libgstreamer-plugins-base1.0-dev
444. libgstreamer-plugins-good1.0-dev
445. libgtk2.0-dev
446. libgtk2-perl
447. libgtk-3-dev
448. libgtkmathview0c2a
449. libgtkmm-2.4-1c2a:armhf
450. libgtkmm-3.0-1:armhf
451. libgtkspell0
452. libgtop2-7
453. libgucharmap-2-90-7
454. libgudev-1.0-dev
455. libguess1:armhf
456. libgupnp-igd-1.0-4:armhf
457. libgvc6
458. libgvpr2
459. libgweather-3-6
460. libhangul1:armhf
461. libhangul-data
462. libharfbuzz-dev
463. libharfbuzz-gobject0:armhf
464. libhighgui-dev:armhf
465. libhogweed2:armhf
466. libical1
467. libice-dev:armhf
468. libicu52:armhf
469. libid3tag0
470. libid3tag0-dev
471. libidl0:armhf
472. libidl-common
473. libidl-dev:armhf
474. libidn11-dev
475. libiec61883-dev
476. libijs-0.35
477. libilmbase6:armhf
478. libilmbase-dev
479. libimage-exiftool-perl
480. libimlib2
481. libimlib2-dev
482. libiptcdata0
483. libiptcdata0-dev
484. libisl10:armhf
485. libisl15:armhf
486. libiso9660-dev
487. libiw-dev:armhf
488. libjack-jackd2-dev:armhf
489. libjasper-dev
490. libjbig-dev:armhf
491. libjna-java
492. libjpeg8-dev:armhf
493. libjpeg-dev:armhf
494. libjpeg-progs
495. libjpeg-turbo8-dev:armhf
496. libjpeg-turbo-progs
497. libjson0:armhf
498. libjson-glib-dev
499. libkadm5clnt-mit9:armhf
500. libkadm5srv-mit9:armhf
501. libkate-dev
502. libkdb5-7:armhf
503. libkrb5-dev
504. liblavfile-2.1-0
505. liblavjpeg-2.1-0
506. liblavplay-2.1-0
507. liblcms2-dev:armhf
508. libldap2-dev:armhf
509. liblink-grammar4
510. libllvm3.4:armhf
511. liblockfile1:armhf
512. liblockfile-bin
513. libloudmouth1-0
514. liblqr-1-0-dev
515. libltdl-dev:armhf
516. liblzma-dev:armhf
517. liblzo2-dev:armhf
518. libm17n-0
519. libmad0-dev
520. libmagick++5:armhf
521. libmagickcore5:armhf
522. libmagickcore5-extra:armhf
523. libmagickcore-dev

524. libmagick++-dev
525. libmagickwand5:armhf
526. libmagickwand-dev
527. libmbim-glib0:armhf
528. libmeanwhile1
529. libmenu-cache3
530. libmenu-cache-bin
531. libmessaging-menu0
532. libmetacity-private0a
533. libmicrohttpd-dev
534. libmikmod2:armhf
535. libmikmod2-dev:armhf
536. libmimic0
537. libmimic-dev
538. libminiupnpc8
539. libmirclient7:armhf
540. libmirclientplatform-mesa:armhf
541. libmirprotobuf0:armhf
542. libmirprotobuf-dev:armhf
543. libmjpegtools-dev
544. libmjpegutils-2.1-0
545. libmms0:armhf
546. libmms-dev:armhf
547. libmodplug1
548. libmodplug-dev
549. libmowgli2:armhf
550. libmp3lame-dev:armhf
551. libmpcdec6
552. libmpcdec-dev
553. libmpeg2-4-dev:armhf
554. libmpeg2encpp-2.1-0
555. libmpeg3-1
556. libmpeg3-dev
557. libmpg123-0:armhf
558. libmpg123-dev:armhf
559. libmplex2-2.1-0
560. libmusicbrainz3-6
561. libmysqlclient-dev
562. libncurses5-dev:armhf
563. libnettle4:armhf
564. libnfs1:armhf
565. libnfs-dev:armhf
566. libnice10:armhf
567. libnm-glib4
568. libnm-glib-vpn1
569. libnm-gtk0
570. libnm-util2
571. libnotify-bin
572. libnotify-dev
573. libnss3-1d:armhf
574. libntdb1:armhf
575. libobrender29
576. libobt2
577. libofa0
578. libofa0-dev
579. libogg-dev:armhf
580. libonig2
581. liboobs-1-5
582. libopenal1:armhf
583. libopenal-data
584. libopenal-dev:armhf
585. libopencv2.4-java
586. libopencv2.4-jni
587. libopencv-calib3d2.4:armhf
588. libopencv-calib3d-dev:armhf
589. libopencv-contrib2.4:armhf
590. libopencv-contrib-dev:armhf
591. libopencv-core2.4:armhf
592. libopencv-core-dev:armhf
593. libopencv-dev
594. libopencv-features2d2.4:armhf
595. libopencv-features2d-dev:armhf
596. libopencv-flann2.4:armhf
597. libopencv-flann-dev:armhf
598. libopencv-gpu2.4:armhf
599. libopencv-gpu-dev:armhf
600. libopencv-highgui2.4:armhf
601. libopencv-highgui-dev:armhf
602. libopencv-imgproc2.4:armhf
603. libopencv-imgproc-dev:armhf
604. libopencv-legacy2.4:armhf
605. libopencv-legacy-dev:armhf
606. libopencv-ml2.4:armhf
607. libopencv-ml-dev:armhf
608. libopencv-objdetect2.4:armhf
609. libopencv-objdetect-dev:armhf
610. libopencv-ocl2.4:armhf
611. libopencv-ocl-dev:armhf
612. libopencv-photo2.4:armhf
613. libopencv-photo-dev:armhf

614. libopencv-stitching2.4:armhf
615. libopencv-stitching-dev:armhf
616. libopencv-superres2.4:armhf
617. libopencv-superres-dev:armhf
618. libopencv-ts2.4:armhf
619. libopencv-ts-dev:armhf
620. libopencv-video2.4:armhf
621. libopencv-video-dev:armhf
622. libopencv-videostab2.4:armhf
623. libopencv-videostab-dev:armhf
624. libopenexr6:armhf
625. libopenexr-dev
626. libopenjpeg2:armhf
627. libopenjpeg-dev
628. libopenvg1-mesa:armhf
629. libopts25:armhf
630. libopus0
631. libopus-dev
632. liborbit-2-0:armhf
633. liborbit2:armhf
634. liborbit2-dev
635. liborc-0.4-dev
636. libotf0:armhf
637. libots0
638. libp11-kit-dev
639. libpam-cap:armhf
640. libpanel-applet-4-0
641. libpango1.0-dev
642. libpangomm-1.4-1:armhf
643. libpango-perl
644. libparted0debian1:armhf
645. libpathplan4
646. libpcre3-dev:armhf
647. libpcrecpp0:armhf
648. libperl5.18
649. libpisock9
650. libplist1:armhf
651. libplist-dev
652. libplymouth2:armhf
653. libpng12-dev
654. libpolkit-agent-1-dev
655. libpolkit-gobject-1-dev
656. libpoppler44:armhf
657. libpopt-dev:armhf
658. libpostproc52
659. libprotobuf8:armhf
660. libprotobuf-lite8:armhf
661. libproxy1:armhf
662. libpulse-dev:armhf
663. libpurple0
664. libqmi-glib0:armhf
665. libqpdf13:armhf
666. libquicktime2:armhf
667. librarian0
668. libraw1394-dev:armhf
669. libreadline5:armhf
670. libreadline6-dev:armhf
671. libreadline-dev:armhf
672. librsvg2-dev
673. librtmp0:armhf
674. librtmp-dev
675. librxtx-java
676. libsamplerate0-dev:armhf
677. libsbc1:armhf
678. libsbc-dev:armhf
679. libschroedinger-dev:armhf
680. libsdl1.2-dev
681. libsdl-gfx1.2-4:armhf
682. libsdl-gfx1.2-dev:armhf
683. libsdl-image1.2-dev:armhf
684. libsdl-mixer1.2:armhf
685. libsdl-mixer1.2-dev:armhf
686. libselinux1-dev:armhf
687. libsepol1-dev
688. libsgmls-perl
689. libshout3-dev:armhf
690. libsidplayfp:armhf
691. libsigc++-2.0-0c2a:armhf
692. libslang2-dev:armhf
693. libsmbclient-dev:armhf
694. libsm-dev:armhf
695. libsndfile1-dev
696. libsoundtouch0:armhf
697. libsoundtouch-dev
698. libsoup2.4-dev
699. libsp1c2
700. libspandsp2
701. libspandsp-dev
702. libspeex-dev:armhf
703. libsqlite0

704. libsqlite3-dev:armhf
705. libsrtp0
706. libsrtp0-dev
707. libssh2-1-dev:armhf
708. libssh-dev
709. libssl-dev:armhf
710. libstartup-notification0-dev:armhf
711. libstdc++-4.8-dev:armhf
712. libsunpinyin3:armhf
713. libswscale2:armhf
714. libswscale-dev
715. libsystemd-daemon0:armhf
716. libsystemd-login0:armhf
717. libt1-5
718. libtag1c2a:armhf
719. libtag1-dev
720. libtag1-vanilla:armhf
721. libtagc0:armhf
722. libtagc0-dev
723. libtasn1-6-dev
724. libtelepathy-glib0:armhf
725. libtheora-dev:armhf
726. libtidy-0.99-0
727. libtiff5-dev:armhf
728. libtiffxx5:armhf
729. libtinfo-dev:armhf
730. libtinyxml2.6.2:armhf
731. libtinyxml-dev:armhf
732. libts-0.0-0:armhf
733. libudev-dev
734. libumfpack5.6.2:armhf
735. libuniconf4.6
736. libupower-glib1:armhf
737. libusb-1.0-0-dev:armhf
738. libusb-dev
739. libusbmuxd2
740. libv4l2rds0:armhf
741. libv4l-dev:armhf
742. libvisual-0.4-dev
743. libvncserver0:armhf
744. libvo-aacenc0:armhf
745. libvo-aacenc-dev:armhf
746. libvo-amrwbenc0:armhf
747. libvo-amrwbenc-dev:armhf
748. libvoikko1:armhf
749. libvorbis-dev:armhf
750. libvpx1:armhf
751. libvpx-dev:armhf
752. libvte-2.90-9
753. libvte-2.90-common
754. libwavpack-dev:armhf
755. libwebpdemux1:armhf
756. libwebp-dev:armhf
757. libwildmidi1:armhf
758. libwildmidi-config
759. libwildmidi-dev
760. libwmf-dev
761. libwpd-0.9-9
762. libwpg-0.2-2
763. libwps-0.2-2
764. libwv-1.2-4:armhf
765. libwvstreams4.6-base
766. libwvstreams4.6-extras
767. libwxbase2.8-0:armhf
768. libwxgtk2.8-0:armhf
769. libx264-142:armhf
770. libxapian22
771. libxaw7-dev:armhf
772. libxcb-icccm4-dev:armhf
773. libxcb-image0-dev:armhf
774. libxcb-keysyms1-dev:armhf
775. libxcb-shm0-dev:armhf
776. libxcb-util0:armhf
777. libxcb-util0-dev:armhf
778. libxcb-xf86dri0:armhf
779. libxcb-xf86dri0-dev:armhf
780. libxcb-xv0-dev:armhf
781. libxcomposite-dev
782. libxcursor-dev:armhf
783. libxdot4
784. libxfce4ui-1-0
785. libxfce4ui-2-0
786. libxfce4ui-2-dev
787. libxfce4ui-common
788. libxfce4util6
789. libxfce4util-common
790. libxfce4util-dev
791. libxfconf-0-2
792. libxfconf-0-dev
793. libxfont-dev

794. libxft-dev
795. libxi-dev
796. libxinerama-dev:armhf
797. libxml2-dev:armhf
798. libxml2-utils
799. libxmu-dev:armhf
800. libxmu-headers
801. libxmuu-dev:armhf
802. libxp6:armhf
803. libxpm-dev:armhf
804. libxrandr-dev:armhf
805. libxrender-dev:armhf
806. libxres-dev
807. libxslt1-dev:armhf
808. libxt-dev:armhf
809. libxtst-dev:armhf
810. libxv-dev:armhf
811. libxvidcore-dev:armhf
812. libyajl-dev
813. libzbar0
814. libzbar-dev
815. libzephyr4:armhf
816. libzip2
817. libzip-dev
818. libzvbi-dev:armhf
819. light-locker
820. light-locker-settings
821. link-grammar-dictionaries-en
822. localepurge
823. lockfile-progs
824. lsof
825. luatex
826. lubuntu-artwork
827. lubuntu-artwork-14-04
828. lubuntu-icon-theme
829. lubuntu-lxpanel-icons
830. lubuntu-software-center
831. lxappearance
832. lxappearance-obconf
833. lxde-common
834. lxde-core
835. lxinput
836. lxlauncher
837. lxmenu-data
838. lxpanel
839. lxpanel-indicator-applet-plugin
840. lxrandr
841. lxsession
842. lxsession-data
843. lxsession-default-apps
844. lxsession-edit
845. lxsession-logout
846. lxshortcut
847. lxtask
848. lxterminal
849. lynx
850. lynx-cur
851. m17n-contrib
852. m17n-db
853. mc
854. mc-data
855. medit
856. mesa-common-dev
857. metacity
858. metacity-common
859. minicom
860. mircommon-dev:armhf
861. mountall
862. mplayer2
863. mtpaint
864. nano
865. nettle-dev
866. ntp
867. ntpdate
868. obconf
869. openbox
870. openjdk-7-jre:armhf
871. openjdk-7-jre-headless:armhf
872. oracle-java8-installer
873. orbit2
874. pastebinit
875. pavucontrol
876. pcmanfm
877. perl-doc
878. pidgin
879. pidgin-data
880. pidgin-libnotify
881. plymouth
882. plymouth-label
883. plymouth-theme-lubuntu-logo

884. plymouth-theme-lubuntu-text
885. pm-utils
886. python-aptdaemon
887. python-aptdaemon.gtk3widgets
888. python-colorama
889. python-commandnotfound
890. python-cups
891. python-cupshelpers
892. python-debian
893. python-defer
894. python-distlib
895. python-gconf
896. python-gdbm
897. python-glade2
898. python-gnomekeyring
899. python-gudev
900. python-html5lib
901. python-libxml2
902. python-mako
903. python-markupsafe
904. python-notify
905. python-ntdb
906. python-pexpect
907. python-pil
908. python-pip
909. python-psutil
910. python-pycurl
911. python-pyinotify
912. python-pysqlite2
913. python-renderpm
914. python-reportlab
915. python-reportlab-accel
916. python-scour
917. python-setuptools
918. python-smbc
919. python-sqlite
920. python-support
921. python-wheel
922. python-xapian
923. python-xdg
924. rarian-compat
925. realpath
926. scrot
927. sessioninstaller
928. sgml-data
929. sgmlspl
930. simple-scan
931. smbclient
932. sp
933. sunpinyin-data
934. swig
935. swig2.0
936. sylpheed
937. sylpheed-doc
938. sylpheed-i18n
939. sylpheed-plugins
940. synaptic
941. systemd-services
942. system-tools-backends
943. transfig
944. transmission
945. tsconf
946. ttf-bengali-fonts
947. ttf-devanagari-fonts
948. ttf-gujarati-fonts
949. ttf-kannada-fonts
950. ttf-malayalam-fonts
951. ttf-oriya-fonts
952. ttf-punjabi-fonts
953. ttf-tamil-fonts
954. ttf-telugu-fonts
955. tzdata-java
956. ubuntu-extras-keyring
957. upstart
958. valgrind
959. voikko-fi
960. wbulgarian
961. wfrench
962. wirish
963. witalian
964. wmanx
965. wnorwegian
966. wogerman
967. wspanish
968. wswedish
969. wvdial
970. x11proto-bigreqs-dev
971. x11proto-composite-dev
972. x11proto-dmx-dev
973. x11proto-record-dev

974. x11proto-xcmisc-dev
975. x11proto-xf86dga-dev
976. x11vnc
977. x11vnc-data
978. x11-xfs-utils
979. xarchiver
980. xdg-user-dirs-gtk
981. xfburn
982. xfce4-dev-tools
983. xfce4-notifyd
984. xfce4-power-manager
985. xfce4-power-manager-data
986. xfconf
987. xfonts-100dpi
988. xmlto
989. xpad
990. xscreensaver
991. xscreensaver-data
992. xscreensaver-data-extra
993. xscreensaver-screensaver-bsod
994. xserver-xorg-input-multitouch
995. xserver-xorg-video-modesetting
996. xserver-xorg-video-omap
997. xserver-xorg-video-vesa
998. xsltproc
999. xul-ext-mozvoikko
1000. xul-ext-ubufox
1001. xvfb
1002. yasm

D.9 Packages installed on backend 3 and not 1

1. adwaita-icon-theme
2. apache2-bin
3. apg
4. aspell
5. aspell-en
6. atril
7. atril-common
8. bamfdaemon
9. blackbox
10. bluez-obexd
11. bootini
12. brasero
13. brasero-cdrkit
14. brasero-common
15. caja
16. caja-common
17. caja-extensions-common
18. caja-gksu
19. caja-image-converter
20. caja-open-terminal
21. caja-sendto
22. caja-share
23. caja-wallpaper
24. cgmanager
25. cheese-common
26. colord-data
27. console-setup-linux
28. cpufrequtils
29. crda
30. cups-pk-helper
31. dconf-editor
32. deja-dup
33. deja-dup-backend-cloudfiles
34. deja-dup-backend-gvfs
35. deja-dup-backend-s3
36. deja-dup-caja
37. desktop-base
38. dmidecode
39. dns-root-data
40. dosfstools
41. duplicity
42. dvd+rw-tools
43. emacsen-common
44. enchant
45. energymonitor
46. engrampa
47. engrampa-common
48. eom
49. eom-common
50. evolution-data-server
51. evolution-data-server-online-accounts
52. exfat-fuse

53. exfat-utils
54. fbi
55. fonts-guru
56. fonts-guru-extra
57. fonts-lmodern
58. fonts-lohit-guru
59. fonts-mathjax
60. fonts-opensymbol
61. fonts-tlwg-laksaman
62. gdbserver
63. gdisk
64. geoclue
65. geoclue-ubuntu-geoip
66. geoip-database
67. gfortran-5
68. gir1.2-appindicator3-0.1
69. gir1.2-caja
70. gir1.2-freedesktop:armhf
71. gir1.2-gdkpixbuf-2.0:armhf
72. gir1.2-glib-2.0:armhf
73. gir1.2-gtk-3.0:armhf
74. gir1.2-ibus-1.0:armhf
75. gir1.2-javascriptcoregtk-3.0:armhf
76. gir1.2-mate-panel
77. gir1.2-pango-1.0:armhf
78. gir1.2-peas-1.0
79. gir1.2-rb-3.0
80. gir1.2-secret-1:armhf
81. gir1.2-vte-2.91
82. gir1.2-webkit-3.0:armhf
83. gir1.2-wnck-3.0:armhf
84. gkbd-capplet
85. gnome-bluetooth
86. gnome-control-center-shared-data
87. gnome-power-manager
88. gnome-screensaver
89. gnome-session-bin
90. gnome-settings-daemon-schemas
91. gnome-user-share
92. growisofs
93. gsettings-ubuntu-schemas
94. guile-2.0-libs:armhf
95. gvfs-bin
96. hddtemp
97. hexchat
98. hexchat-common
99. hexchat-perl
100. hexchat-plugins
101. hexchat-python
102. humanity-icon-theme
103. hwdata
104. ideviceinstaller
105. ifuse
106. imagemagick-6.q16
107. indicator-applet
108. indicator-bluetooth
109. indicator-datetime
110. indicator-keyboard
111. indicator-messages
112. indicator-network
113. indicator-power
114. indicator-sound
115. init
116. inxi
117. ippusbxd
118. iw
119. kernel-common
120. kodi
121. krb5-locales
122. liba52-0.7.4:armhf
123. libaccount-plugin-1.0-0
124. libaccount-plugin-generic-oauth
125. libaccount-plugin-google
126. libaccounts-glib0:armhf
127. libaccounts-qt5-1:armhf
128. libalgorithm-c3-perl
129. libapache2-mod-dnssd
130. libappindicator1
131. libapr1:armhf
132. libaprutil1:armhf
133. libaprutil1-dbd-sqlite3:armhf
134. libaprutil1-ldap:armhf
135. libapt-inst1.7:armhf
136. libapt-pkg4.16:armhf
137. libaspell15:armhf
138. libasprintf0v5:armhf
139. libasprintf-dev:armhf
140. libass5:armhf
141. libatkmm-1.6-1v5:armhf
142. libatm1:armhf

143. libatrildocument3
144. libatrilview3
145. libavcodec-ffmpeg56:armhf
146. libavformat-ffmpeg56:armhf
147. libavutil-ffmpeg54:armhf
148. libbabeltrace1:armhf
149. libbabeltrace-ctf1:armhf
150. libbamf3-2:armhf
151. libbareword-filehandles-perl
152. libbasicusageenvironment0
153. libbdplus0:armhf
154. libb-hooks-endofscope-perl
155. libb-hooks-op-check-perl
156. libblas-common
157. libboost-date-time1.58.0:armhf
158. libboost-filesystem1.58.0:armhf
159. libboost-system1.58.0:armhf
160. libbrasero-media3-1
161. libbt0v5:armhf
162. libcairomm-1.0-1v5:armhf
163. libcaja-extension1:armhf
164. libcamel-1.2-52
165. libcanberra-gtk3-module:armhf
166. libcanberra-gtk-module:armhf
167. libcanberra-pulse:armhf
168. libcap-ng0:armhf
169. libcec3:armhf
170. libcgi-fast-perl
171. libcgi-pm-perl
172. libcheese7:armhf
173. libcheese-gtk23:armhf
174. libclass-c3-perl
175. libclass-c3-xs-perl
176. libclass-method-modifiers-perl
177. libclass-xsaccessor-perl
178. libclucene-contribs1v5:armhf
179. libclucene-core1v5:armhf
180. libclutter-1.0-common
181. libcmis-0.5-5v5
182. libcogl20:armhf
183. libcogl-common
184. libcogl-pango20:armhf
185. libcogl-path20:armhf
186. libcolord2:armhf
187. libcolorhug2:armhf
188. libcpan-changes-perl
189. libcpan-meta-perl
190. libcpufreq0
191. libcryptsetup4:armhf
192. libdaemon0:armhf
193. libdata-optlist-perl
194. libdata-perl-perl
195. libdata-section-perl
196. libdee-1.0-4:armhf
197. libdevel-caller-perl
198. libdevel-globaldestruction-perl
199. libdevel-lexalias-perl
200. libdmapsharing-3.0-2
201. libdns-export100
202. libdouble-conversion1v5:armhf
203. libdrm-amdgpu1:armhf
204. libdrm-tegra0:armhf
205. libdvbpsi10:armhf
206. libdw1:armhf
207. libebackend-1.2-10
208. libebml4v5:armhf
209. libebook-1.2-16
210. libebook-contacts-1.2-1
211. libecal-1.2-18
212. libedata-book-1.2-25
213. libedata-cal-1.2-27
214. libedataserver-1.2-20
215. libegl1-mesa-dev:armhf
216. libeot0
217. libepoxy0
218. libevdev2:armhf
219. libexiv2-14:armhf
220. libexporter-tiny-perl
221. libexttextcat-2.0-0
222. libexttextcat-data
223. libfcgi-perl
224. libfcitx-config4:armhf
225. libfcitx-gclient0:armhf
226. libfcitx-utils0:armhf
227. libfdisk1:armhf
228. libffmpegthumbnailer4v5
229. libfile-desktopentry-perl
230. libfile-fcntllock-perl
231. libfile-mimeinfo-perl
232. libfile-slurp-perl

233. libfont-afm-perl
234. libfreerdp-cache1.1:armhf
235. libfreerdp-client1.1:armhf
236. libfreerdp-codec1.1:armhf
237. libfreerdp-common1.1.0:armhf
238. libfreerdp-core1.1:armhf
239. libfreerdp-crypto1.1:armhf
240. libfreerdp-gdi1.1:armhf
241. libfreerdp-locale1.1:armhf
242. libfreerdp-primitives1.1:armhf
243. libfreerdp-utils1.1:armhf
244. libgc1c2:armhf
245. libgcrypt20:armhf
246. libgdata22:armhf
247. libgdata-common
248. libgee-0.8-2:armhf
249. libgee2:armhf
250. libgeocode-glib0:armhf
251. libgetopt-long-descriptive-perl
252. libgettextpo0:armhf
253. libgettextpo-dev:armhf
254. libgexiv2-2:armhf
255. libgfortran-5-dev:armhf
256. libgirepository-1.0-1:armhf
257. libgles2-mesa-dev:armhf
258. libglew1.10:armhf
259. libglibmm-2.4-1v5:armhf
260. libgme0:armhf
261. libgmime-2.6-0:armhf
262. libgnome-bluetooth13
263. libgnome-desktop-3-10:armhf
264. libgnomekbd8
265. libgnomekbd-common
266. libgnutls-deb0-28:armhf
267. libgoa-1.0-0b:armhf
268. libgoa-1.0-common
269. libgphoto2-l10n
270. libgphoto2-port12:armhf
271. libgrilo-0.2-1:armhf
272. libgroupsock1
273. libgsasl7
274. libgsl0ldbl:armhf
275. libgtk2.0-bin
276. libgtkmm-2.4-1v5:armhf
277. libgtksourceview2.0-0
278. libgtksourceview2.0-common
279. libgtop2-10
280. libguvcview-1.1-1:armhf
281. libgweather-3-6:armhf
282. libhogweed4:armhf
283. libhtml-format-perl
284. libhtml-form-perl
285. libhttp-daemon-perl
286. libhunspell-1.3-0v5:armhf
287. libhybris
288. libhyphen0
289. libical1a
290. libicu55:armhf
291. libijs-0.35:armhf
292. libilmbase12:armhf
293. libimobiledevice-utils
294. libimport-into-perl
295. libindirect-perl
296. libinput10:armhf
297. libio-stringy-perl
298. libirs-export91
299. libisccfg-export90
300. libisc-export95
301. libisl13:armhf
302. libjs-mathjax
303. libkyotocabinet16v5:armhf
304. liblangtag1
305. liblangtag-common
306. liblcms2-utils
307. liblexical-sealrequirehints-perl
308. liblivemedia23
309. libllvm3.6v5:armhf
310. liblog-message-perl
311. libmagickcore-6.q16-2:armhf
312. libmagickcore-6.q16-2-extra:armhf
313. libmagickwand-6.q16-2:armhf
314. libmailutils4:armhf
315. libmarco-private0:armhf
316. libmate-desktop-2-17:armhf
317. libmatedict6
318. libmatekbd4:armhf
319. libmatekbd-common
320. libmate-menu2:armhf
321. libmatemixer0:armhf
322. libmatemixer-common

323. libmate-panel-applet-4-1
324. libmate-sensors-applet-plugin0
325. libmate-slab0:armhf
326. libmateweather1:armhf
327. libmateweather-common
328. libmate-window-settings1:armhf
329. libmatroska6v5:armhf
330. libmbim-glib4:armhf
331. libmbim-proxy
332. libmedia1
333. libminiupnpc10:armhf
334. libmirclient9:armhf
335. libmircommon5:armhf
336. libmircommon-dev:armhf
337. libmirprotobuf3:armhf
338. libmodplug1:armhf
339. libmodule-build-perl
340. libmodule-implementation-perl
341. libmodule-runtime-perl
342. libmodule-signature-perl
343. libmoo-perl
344. libmoox-handlesvia-perl
345. libmpcdec6:armhf
346. libmro-compat-perl
347. libmultidimensional-perl
348. libmythes-1.2-0:armhf
349. libnamespace-autoclean-perl
350. libnamespace-clean-perl
351. libndp0:armhf
352. libnettle6:armhf
353. libnm0:armhf
354. libnm-glib4:armhf
355. libnm-glib-vpn1:armhf
356. libnm-gtk0:armhf
357. libnm-util2:armhf
358. libntlm0:armhf
359. liboauth0:armhf
360. libopenexr22:armhf
361. libopenjpeg5:armhf
362. libopus0:armhf
363. libotr5
364. libp8-platform2:armhf
365. libpackage-constants-perl
366. libpackage-stash-perl
367. libpackage-stash-xs-perl
368. libpadwalker-perl
369. libpanel-applet0
370. libpangomm-1.4-1v5:armhf
371. libparams-classify-perl
372. libparams-util-perl
373. libparams-validate-perl
374. libparted2:armhf
375. libpath-tiny-perl
376. libpcre16-3:armhf
377. libpcrecpp0v5:armhf
378. libpeas-1.0-0
379. libpeas-common
380. libperl5.20
381. libplank0:armhf
382. libplank-common
383. libplist3:armhf
384. libplist-utils
385. libpod-markdown-perl
386. libpod-readme-perl
387. libpoppler52:armhf
388. libpostproc-ffmpeg53:armhf
389. libpotrace0
390. libprotobuf9v5:armhf
391. libprotobuf-lite9v5:armhf
392. libproxy1v5:armhf
393. libproxy-tools
394. libqmi-glib1:armhf
395. libqmi-proxy
396. libqofono-qt5-0:armhf
397. libqpdf13v5:armhf
398. libqt5core5a:armhf
399. libqt5dbus5:armhf
400. libqt5gui5:armhf
401. libqt5network5:armhf
402. libqt5opengl5:armhf
403. libqt5printsupport5:armhf
404. libqt5qml5:armhf
405. libqt5quick5:armhf
406. libqt5sql5:armhf
407. libqt5sql5-sqlite:armhf
408. libqt5webkit5:armhf
409. libqt5widgets5:armhf
410. libqt5x11extras5:armhf
411. libqt5xml5:armhf
412. libqt5xmlpatterns5:armhf

413. libqwt-headers
414. libraw10:armhf
415. libreoffice-avmedia-backend-gstreamer
416. libreoffice-common
417. libreoffice-core
418. libreoffice-gnome
419. libreoffice-gtk
420. libreoffice-l10n-en-za
421. libreoffice-style-galaxy
422. libreoffice-style-human
423. libresid-builder0c2a
424. librest-0.7-0:armhf
425. librevenge-0.0-0:armhf
426. librhythmbox-core9
427. librole-tiny-perl
428. librsync1:armhf
429. librtmp1:armhf
430. libruby2.1:armhf
431. libsdl2-2.0-0:armhf
432. libseccomp2:armhf
433. libshine3:armhf
434. libsidplay2v5
435. libsigc++-2.0-0v5:armhf
436. libsignon-extension1:armhf
437. libsignon-glib1:armhf
438. libsignon-plugins-common1:armhf
439. libsignon-qt5-1:armhf
440. libsmartcols1:armhf
441. libsoftware-license-perl
442. libsoxr0:armhf
443. libssh-gcrypt-4:armhf
444. libstrictures-perl
445. libsub-exporter-perl
446. libsub-exporter-progressive-perl
447. libsub-install-perl
448. libswresample-ffmpeg1:armhf
449. libswscale-ffmpeg3:armhf
450. libsynctex1
451. libsystemd0:armhf
452. libtag1v5:armhf
453. libtag1v5-vanilla:armhf
454. libtexlua52
455. libtexluajit2
456. libtext-template-perl
457. libtie-ixhash-perl
458. libtimezonemap1:armhf
459. libtimezonemap-data
460. libtinyxml2.6.2v5:armhf
461. libtotem-plparser18:armhf
462. libtotem-plparser-common
463. libtry-tiny-perl
464. libtwolame0:armhf
465. libtxc-dxtn-s2tc0:armhf
466. libtype-tiny-perl
467. libtype-tiny-xs-perl
468. libudev-dev:armhf
469. libunicode-utf8-perl
470. libunique-1.0-0
471. libunity9:armhf
472. libunity-control-center1
473. libunity-protocol-private0:armhf
474. libunity-scopes-json-def-desktop
475. libunity-settings-daemon1
476. libunwind8
477. libupnp6
478. libupower-glib3:armhf
479. liburl-dispatcher1:armhf
480. libusageenvironment1
481. libusbmuxd2:armhf
482. libva-drm1:armhf
483. libvariable-magic-perl
484. libva-x11-1:armhf
485. libvcdinfo0
486. libvisual-0.4-plugins:armhf
487. libvlc5
488. libvlccore8
489. libvncclient1:armhf
490. libvpx2:armhf
491. libvte-2.91-0
492. libvte-2.91-common
493. libwacom2:armhf
494. libwacom-bin
495. libwacom-common
496. libwebrtc-audio-processing-0:armhf
497. libwinpr-crt0.1:armhf
498. libwinpr-dsparse0.1:armhf
499. libwinpr-environment0.1:armhf
500. libwinpr-file0.1:armhf
501. libwinpr-handle0.1:armhf
502. libwinpr-heap0.1:armhf

503. libwinpr-input0.1:armhf
504. libwinpr-interlocked0.1:armhf
505. libwinpr-library0.1:armhf
506. libwinpr-path0.1:armhf
507. libwinpr-pool0.1:armhf
508. libwinpr-registry0.1:armhf
509. libwinpr-rpc0.1:armhf
510. libwinpr-sspi0.1:armhf
511. libwinpr-synch0.1:armhf
512. libwinpr-sysinfo0.1:armhf
513. libwinpr-thread0.1:armhf
514. libwinpr-utils0.1:armhf
515. libx11-protocol-perl
516. libx264-146:armhf
517. libx265-59:armhf
518. libxapian22v5
519. libxcb-composite0:armhf
520. libxcb-render-util0:armhf
521. libxcb-util1:armhf
522. libxcb-xkb1:armhf
523. libxkbcommon-x11-0:armhf
524. libxml-xpathengine-perl
525. libyaml-0-2:armhf
526. libzeitgeist-2.0-0:armhf
527. libzip4:armhf
528. libzzip-0-13:armhf
529. lightdm-gtk-greeter-settings
530. linux-image-3.10.96-78
531. linux-image-xu3
532. linux-tools-4.2.0-23
533. linux-tools-4.2.0-23-generic-lpae
534. linux-tools-common
535. linux-tools-generic-lpae
536. live-boot-initramfs-tools
537. lmodern
538. mailutils
539. mailutils-common
540. mali-x11
541. marco
542. marco-common
543. mate-applets
544. mate-applets-common
545. mate-backgrounds
546. mate-control-center
547. mate-control-center-common
548. mate-desktop
549. mate-desktop-common
550. mate-desktop-environment
551. mate-desktop-environment-core
552. mate-desktop-environment-extra
553. mate-desktop-environment-extras
554. mate-gnome-main-menu-applet
555. mate-icon-theme
556. mate-icon-theme-faenza
557. mate-indicator-applet
558. mate-indicator-applet-common
559. mate-media
560. mate-media-common
561. mate-menus
562. mate-netspeed
563. mate-netspeed-common
564. mate-notification-daemon
565. mate-notification-daemon-common
566. mate-panel
567. mate-panel-common
568. mate-polkit:armhf
569. mate-polkit-common
570. mate-power-manager
571. mate-power-manager-common
572. mate-screensaver
573. mate-screensaver-common
574. mate-sensors-applet
575. mate-sensors-applet-common
576. mate-session-manager
577. mate-settings-daemon
578. mate-settings-daemon-common
579. mate-system-monitor
580. mate-system-monitor-common
581. mate-terminal
582. mate-terminal-common
583. mate-themes
584. mate-user-guide
585. mate-user-share
586. mate-user-share-common
587. mate-utils
588. mate-utils-common
589. media-player-info
590. menu
591. menu-xdg
592. mesa-common-dev:armhf

593. mir-client-platform-mesa-dev:armhf
594. mousetweaks
595. mozo
596. myspell-en-au
597. myspell-en-gb
598. myspell-en-za
599. mythes-en-au
600. mythes-en-us
601. ncurses-term
602. network-manager-pptp
603. network-manager-pptp-gnome
604. ntfs-3g
605. odroid-platform-5422
606. ofono
607. openoffice.org-hyphenation
608. p7zip-full
609. pidgin-otr
610. pinentry-gnome3
611. plank
612. pluma
613. pluma-common
614. postfix
615. pptp-linux
616. printer-driver-brlaser
617. python3-bs4
618. python3-cairo
619. python3-cups
620. python3-cupshelpers
621. python3-html5lib
622. python3-lxml
623. python3-mako
624. python3-markupsafe
625. python3-pexpect
626. python3-pil:armhf
627. python3-pyinotify
628. python3-renderpm:armhf
629. python3-reportlab
630. python3-reportlab-accel:armhf
631. python3-requests
632. python3-smbc
633. python3-systemd
634. python3-uno
635. python3-urllib3
636. python3-virtualenv
637. python3-xdg
638. python-boto
639. python-caja
640. python-caja-common
641. python-cffi
642. python-cffi-backend
643. python-cloudfiles
644. python-cryptography
645. python-enum34
646. python-gtksourceview2
647. python-idna
648. python-ipaddress
649. python-lockfile
650. python-mate-menu
651. python-ndg-httpsclient
652. python-openssl
653. python-pil:armhf
654. python-ply
655. python-pyasn1
656. python-pycparser
657. qttranslations5-l10n
658. rename
659. rhythmbox
660. rhythmbox-data
661. rhythmbox-plugin-cdrecorder
662. rhythmbox-plugins
663. rhythmbox-plugin-zeitgeist
664. rtkit
665. ruby
666. ruby2.1
667. rubygems-integration
668. seahorse
669. session-migration
670. shotwell
671. shotwell-common
672. signond
673. signon-keyring-extension
674. signon-plugin-oauth2
675. signon-ui
676. signon-ui-service
677. signon-ui-x11
678. smartmontools
679. ssh-import-id
680. systemd
681. systemd-sysv
682. tcpd

683. tcptrack
684. telnet
685. thunderbird
686. thunderbird-locale-en
687. thunderbird-locale-en-gb
688. thunderbird-locale-en-us
689. tlp
690. tlp-rdw
691. traceroute
692. ttf-ancient-fonts-symbola
693. ubuntu-mate-icon-themes
694. ubuntu-mate-lightdm-theme
695. ubuntu-mate-themes
696. ubuntu-mate-wallpapers
697. ubuntu-mate-wallpapers-common
698. ubuntu-mate-wallpapers-wily
699. ubuntu-minimal
700. ubuntu-mobile-icons
701. ubuntu-mono
702. ubuntu-system-service
703. ubuntu-touch-sounds
704. unity-control-center
705. unity-control-center-signon
706. unity-settings-daemon
707. uno-libs3
708. ure
709. ureadahead
710. urfkill
711. va-driver-all:armhf
712. vdpau-va-driver:armhf
713. virtualenv
714. vlc
715. vlc-data
716. vlc-nox
717. vlc-plugin-notify
718. vlc-plugin-samba
719. wireless-regdb
720. wodim
721. xserver-xorg-core
722. xserver-xorg-input-wacom
723. xserver-xorg-video-armsoc-5422
724. xul-ext-calendar-timezones
725. xul-ext-gdata-provider
726. xul-ext-lightning
727. zeitgeist-core

D.10 Packages installed on backend 3 and not 2

1. adwaita-icon-theme
2. apache2-bin
3. apg
4. atril
5. atril-common
6. bamfdaemon
7. blackbox
8. bluez-obexd
9. bootini
10. brasero
11. brasero-cdrkit
12. brasero-common
13. caja
14. caja-common
15. caja-extensions-common
16. caja-gksu
17. caja-image-converter
18. caja-open-terminal
19. caja-sendto
20. caja-share
21. caja-wallpaper
22. cgmanager
23. cheese-common
24. colord-data
25. console-setup-linux
26. cpufrequtils
27. crda
28. cups-pk-helper
29. dconf-editor
30. deja-dup
31. deja-dup-backend-cloudfiles
32. deja-dup-backend-gvfs
33. deja-dup-backend-s3
34. deja-dup-caja
35. desktop-base
36. dmidecode

37. dns-root-data
38. dosfstools
39. duplicity
40. dvd+rw-tools
41. emacsen-common
42. enchant
43. energymonitor
44. engrampa
45. engrampa-common
46. eom
47. eom-common
48. evolution-data-server
49. evolution-data-server-online-accounts
50. exfat-fuse
51. exfat-utils
52. fbi
53. fonts-guru
54. fonts-guru-extra
55. fonts-lmodern
56. fonts-lohit-guru
57. fonts-mathjax
58. fonts-opensymbol
59. fonts-tlwg-laksaman
60. gdbserver
61. gdisk
62. geoclue
63. geoclue-ubuntu-geoip
64. geoip-database
65. gfortran-5
66. gir1.2-appindicator3-0.1
67. gir1.2-caja
68. gir1.2-freedesktop:armhf
69. gir1.2-gdkpixbuf-2.0:armhf
70. gir1.2-glib-2.0:armhf
71. gir1.2-gtk-3.0:armhf
72. gir1.2-ibus-1.0:armhf
73. gir1.2-javascriptcoregtk-3.0:armhf
74. gir1.2-mate-panel
75. gir1.2-pango-1.0:armhf
76. gir1.2-peas-1.0
77. gir1.2-rb-3.0
78. gir1.2-secret-1:armhf
79. gir1.2-vte-2.91
80. gir1.2-webkit-3.0:armhf
81. gir1.2-wnck-3.0:armhf
82. gkbd-capplet
83. gnome-bluetooth
84. gnome-control-center-shared-data
85. gnome-power-manager
86. gnome-screensaver
87. gnome-session-bin
88. gnome-settings-daemon-schemas
89. gnome-user-share
90. growisofs
91. gsettings-ubuntu-schemas
92. guile-2.0-libs:armhf
93. gvfs-bin
94. hddtemp
95. hexchat
96. hexchat-common
97. hexchat-perl
98. hexchat-plugins
99. hexchat-python
100. humanity-icon-theme
101. hwdata
102. ideviceinstaller
103. ifuse
104. imagemagick
105. imagemagick-6.q16
106. indicator-applet
107. indicator-bluetooth
108. indicator-datetime
109. indicator-keyboard
110. indicator-messages
111. indicator-network
112. indicator-power
113. indicator-sound
114. init
115. inxi
116. ippusbxd
117. iw
118. kernel-common
119. kodi
120. krb5-locales
121. liba52-0.7.4:armhf
122. libaccount-plugin-1.0-0
123. libaccount-plugin-generic-oauth
124. libaccount-plugin-google
125. libaccounts-glib0:armhf
126. libaccounts-qt5-1:armhf

127. libalgorithm-c3-perl
128. libapache2-mod-dnssd
129. libappindicator1
130. libapr1:armhf
131. libaprutil1:armhf
132. libaprutil1-dbd-sqlite3:armhf
133. libaprutil1-ldap:armhf
134. libapt-inst1.7:armhf
135. libapt-pkg4.16:armhf
136. libaspell15:armhf
137. libasprintf0v5:armhf
138. libasprintf-dev:armhf
139. libass5:armhf
140. libatkmm-1.6-1v5:armhf
141. libatm1:armhf
142. libatrildocument3
143. libatrilview3
144. libavcodec-ffmpeg56:armhf
145. libavformat-ffmpeg56:armhf
146. libavutil-ffmpeg54:armhf
147. libbabeltrace1:armhf
148. libbabeltrace-ctf1:armhf
149. libbamf3-2:armhf
150. libbareword-filehandles-perl
151. libbasicusageenvironment0
152. libbdplus0:armhf
153. libb-hooks-endofscope-perl
154. libb-hooks-op-check-perl
155. libblas-common
156. libboost-date-time1.58.0:armhf
157. libboost-filesystem1.58.0:armhf
158. libboost-system1.58.0:armhf
159. libbrasero-media3-1
160. libbt0v5:armhf
161. libcairomm-1.0-1v5:armhf
162. libcaja-extension1:armhf
163. libcamel-1.2-52
164. libcanberra-gtk3-module:armhf
165. libcanberra-gtk-module:armhf
166. libcanberra-pulse:armhf
167. libcap-ng0:armhf
168. libcec3:armhf
169. libcgi-fast-perl
170. libcgi-pm-perl
171. libcheese7:armhf
172. libcheese-gtk23:armhf
173. libclass-c3-perl
174. libclass-c3-xs-perl
175. libclass-method-modifiers-perl
176. libclass-xsaccessor-perl
177. libclucene-contribs1v5:armhf
178. libclucene-core1v5:armhf
179. libclutter-1.0-common
180. libcmis-0.5-5v5
181. libcogl20:armhf
182. libcogl-common
183. libcogl-pango20:armhf
184. libcogl-path20:armhf
185. libcolord2:armhf
186. libcolorhug2:armhf
187. libcpan-changes-perl
188. libcpan-meta-perl
189. libcpufreq0
190. libcryptsetup4:armhf
191. libdaemon0:armhf
192. libdata-optlist-perl
193. libdata-perl-perl
194. libdata-section-perl
195. libdee-1.0-4:armhf
196. libdevel-caller-perl
197. libdevel-globaldestruction-perl
198. libdevel-lexalias-perl
199. libdmapsharing-3.0-2
200. libdns-export100
201. libdouble-conversion1v5:armhf
202. libdrm-amdgpu1:armhf
203. libdrm-tegra0:armhf
204. libdvbpsi10:armhf
205. libdw1:armhf
206. libebackend-1.2-10
207. libebml4v5:armhf
208. libebook-1.2-16
209. libebook-contacts-1.2-1
210. libecal-1.2-18
211. libedata-book-1.2-25
212. libedata-cal-1.2-27
213. libedataserver-1.2-20
214. libegl1-mesa-dev:armhf
215. libeot0
216. libepoxy0

217. libevdev2:armhf
218. libexiv2-14:armhf
219. libexporter-tiny-perl
220. libexttextcat-2.0-0
221. libexttextcat-data
222. libfcgi-perl
223. libfcitx-config4:armhf
224. libfcitx-gclient0:armhf
225. libfcitx-utils0:armhf
226. libfdisk1:armhf
227. libffmpegthumbnailer4v5
228. libfile-desktopentry-perl
229. libfile-fcntllock-perl
230. libfile-mimeinfo-perl
231. libfile-slurp-perl
232. libfont-afm-perl
233. libfreerdp-cache1.1:armhf
234. libfreerdp-client1.1:armhf
235. libfreerdp-codec1.1:armhf
236. libfreerdp-common1.1.0:armhf
237. libfreerdp-core1.1:armhf
238. libfreerdp-crypto1.1:armhf
239. libfreerdp-gdi1.1:armhf
240. libfreerdp-locale1.1:armhf
241. libfreerdp-primitives1.1:armhf
242. libfreerdp-utils1.1:armhf
243. libgc1c2:armhf
244. libgcrypt20:armhf
245. libgdata22:armhf
246. libgdata-common
247. libgee-0.8-2:armhf
248. libgee2:armhf
249. libgeocode-glib0:armhf
250. libgetopt-long-descriptive-perl
251. libgettextpo0:armhf
252. libgettextpo-dev:armhf
253. libgexiv2-2:armhf
254. libgfortran-5-dev:armhf
255. libgirepository-1.0-1:armhf
256. libgles2-mesa-dev:armhf
257. libglew1.10:armhf
258. libglibmm-2.4-1v5:armhf
259. libgme0:armhf
260. libgmime-2.6-0:armhf
261. libgnome-bluetooth13
262. libgnome-desktop-3-10:armhf
263. libgnomekbd8
264. libgnomekbd-common
265. libgnutls-deb0-28:armhf
266. libgoa-1.0-0b:armhf
267. libgoa-1.0-common
268. libgphoto2-l10n
269. libgphoto2-port12:armhf
270. libgrilo-0.2-1:armhf
271. libgroupsock1
272. libgsasl7
273. libgsl0ldbl:armhf
274. libgtk2.0-bin
275. libgtkmm-2.4-1v5:armhf
276. libgtksourceview2.0-0
277. libgtksourceview2.0-common
278. libgtop2-10
279. libguvcview-1.1-1:armhf
280. libgweather-3-6:armhf
281. libhogweed4:armhf
282. libhtml-format-perl
283. libhtml-form-perl
284. libhttp-daemon-perl
285. libhunspell-1.3-0v5:armhf
286. libhybris
287. libhyphen0
288. libical1a
289. libicu55:armhf
290. libijs-0.35:armhf
291. libilmbase12:armhf
292. libimobiledevice-utils
293. libimport-into-perl
294. libindirect-perl
295. libinput10:armhf
296. libio-stringy-perl
297. libirs-export91
298. libisccfg-export90
299. libisc-export95
300. libisl13:armhf
301. libjavascriptcoregtk-1.0-0:armhf
302. libjs-mathjax
303. libkyotocabinet16v5:armhf
304. liblangtag1
305. liblangtag-common
306. liblcms2-utils

307. liblexical-sealrequirehints-perl
308. liblivemedia23
309. libllvm3.6v5:armhf
310. liblog-message-perl
311. liblua5.1-0:armhf
312. libmagickcore-6.q16-2:armhf
313. libmagickcore-6.q16-2-extra:armhf
314. libmagickwand-6.q16-2:armhf
315. libmailutils4:armhf
316. libmarco-private0:armhf
317. libmate-desktop-2-17:armhf
318. libmatedict6
319. libmatekbd4:armhf
320. libmatekbd-common
321. libmate-menu2:armhf
322. libmatemixer0:armhf
323. libmatemixer-common
324. libmate-panel-applet-4-1
325. libmate-sensors-applet-plugin0
326. libmate-slab0:armhf
327. libmateweather1:armhf
328. libmateweather-common
329. libmate-window-settings1:armhf
330. libmatroska6v5:armhf
331. libmbim-glib4:armhf
332. libmbim-proxy
333. libmedia1
334. libminiupnpc10:armhf
335. libmirclient9:armhf
336. libmircommon5:armhf
337. libmircommon-dev:armhf
338. libmirprotobuf3:armhf
339. libmng2:armhf
340. libmodplug1:armhf
341. libmodule-build-perl
342. libmodule-implementation-perl
343. libmodule-runtime-perl
344. libmodule-signature-perl
345. libmoo-perl
346. libmoox-handlesvia-perl
347. libmpcdec6:armhf
348. libmro-compat-perl
349. libmultidimensional-perl
350. libmythes-1.2-0:armhf
351. libnamespace-autoclean-perl
352. libnamespace-clean-perl
353. libndp0:armhf
354. libnettle6:armhf
355. libnm0:armhf
356. libnm-glib4:armhf
357. libnm-glib-vpn1:armhf
358. libnm-gtk0:armhf
359. libnm-util2:armhf
360. libntlm0:armhf
361. liboauth0:armhf
362. libopenexr22:armhf
363. libopenjpeg5:armhf
364. libopus0:armhf
365. libotr5
366. libp8-platform2:armhf
367. libpackage-constants-perl
368. libpackage-stash-perl
369. libpackage-stash-xs-perl
370. libpadwalker-perl
371. libpanel-applet0
372. libpangomm-1.4-1v5:armhf
373. libparams-classify-perl
374. libparams-util-perl
375. libparams-validate-perl
376. libparted2:armhf
377. libpath-tiny-perl
378. libpcre16-3:armhf
379. libpcrecpp0v5:armhf
380. libpeas-1.0-0
381. libpeas-common
382. libperl5.20
383. libplank0:armhf
384. libplank-common
385. libplist3:armhf
386. libplist-utils
387. libpod-markdown-perl
388. libpod-readme-perl
389. libpoppler52:armhf
390. libpostproc-ffmpeg53:armhf
391. libpotrace0
392. libprotobuf9v5:armhf
393. libprotobuf-lite9v5:armhf
394. libproxy1v5:armhf
395. libproxy-tools
396. libqmi-glib1:armhf

397. libqmi-proxy
398. libqofono-qt5-0:armhf
399. libqpdf13v5:armhf
400. libqt5core5a:armhf
401. libqt5dbus5:armhf
402. libqt5gui5:armhf
403. libqt5network5:armhf
404. libqt5opengl5:armhf
405. libqt5printsupport5:armhf
406. libqt5qml5:armhf
407. libqt5quick5:armhf
408. libqt5sql5:armhf
409. libqt5sql5-sqlite:armhf
410. libqt5webkit5:armhf
411. libqt5widgets5:armhf
412. libqt5x11extras5:armhf
413. libqt5xml5:armhf
414. libqt5xmlpatterns5:armhf
415. libqwt-headers
416. libraw10:armhf
417. libreoffice-avmedia-backend-gstreamer
418. libreoffice-common
419. libreoffice-core
420. libreoffice-gnome
421. libreoffice-gtk
422. libreoffice-l10n-en-za
423. libreoffice-style-galaxy
424. libreoffice-style-human
425. libresid-builder0c2a
426. librest-0.7-0:armhf
427. librevenge-0.0-0:armhf
428. librhythmbox-core9
429. librole-tiny-perl
430. librsync1:armhf
431. librtmp1:armhf
432. libruby2.1:armhf
433. libsdl2-2.0-0:armhf
434. libseccomp2:armhf
435. libshine3:armhf
436. libsidplay2v5
437. libsigc++-2.0-0v5:armhf
438. libsignon-extension1:armhf
439. libsignon-glib1:armhf
440. libsignon-plugins-common1:armhf
441. libsignon-qt5-1:armhf
442. libsmartcols1:armhf
443. libsoftware-license-perl
444. libsoxr0:armhf
445. libssh-gcrypt-4:armhf
446. libstrictures-perl
447. libsub-exporter-perl
448. libsub-exporter-progressive-perl
449. libsub-install-perl
450. libswresample-ffmpeg1:armhf
451. libswscale-ffmpeg3:armhf
452. libsynctex1
453. libsystemd0:armhf
454. libtag1v5:armhf
455. libtag1v5-vanilla:armhf
456. libtexlua52
457. libtexluajit2
458. libtext-template-perl
459. libtie-ixhash-perl
460. libtimezonemap1:armhf
461. libtimezonemap-data
462. libtinyxml2.6.2v5:armhf
463. libtotem-plparser18:armhf
464. libtotem-plparser-common
465. libtry-tiny-perl
466. libtwolame0:armhf
467. libtxc-dxtn-s2tc0:armhf
468. libtype-tiny-perl
469. libtype-tiny-xs-perl
470. libudev-dev:armhf
471. libunicode-utf8-perl
472. libunique-1.0-0
473. libunity9:armhf
474. libunity-control-center1
475. libunity-protocol-private0:armhf
476. libunity-scopes-json-def-desktop
477. libunity-settings-daemon1
478. libunwind8
479. libupnp6
480. libupower-glib3:armhf
481. liburl-dispatcher1:armhf
482. libusageenvironment1
483. libusbmuxd2:armhf
484. libva-drm1:armhf
485. libvariable-magic-perl
486. libva-x11-1:armhf

487. libvcdinfo0
488. libvisual-0.4-plugins:armhf
489. libvlc5
490. libvlccore8
491. libvncclient1:armhf
492. libvpx2:armhf
493. libvte-2.91-0
494. libvte-2.91-common
495. libwacom2:armhf
496. libwacom-bin
497. libwacom-common
498. libwebkitgtk-1.0-0:armhf
499. libwebkitgtk-1.0-common
500. libwebrtc-audio-processing-0:armhf
501. libwinpr-crt0.1:armhf
502. libwinpr-dsparse0.1:armhf
503. libwinpr-environment0.1:armhf
504. libwinpr-file0.1:armhf
505. libwinpr-handle0.1:armhf
506. libwinpr-heap0.1:armhf
507. libwinpr-input0.1:armhf
508. libwinpr-interlocked0.1:armhf
509. libwinpr-library0.1:armhf
510. libwinpr-path0.1:armhf
511. libwinpr-pool0.1:armhf
512. libwinpr-registry0.1:armhf
513. libwinpr-rpc0.1:armhf
514. libwinpr-sspi0.1:armhf
515. libwinpr-synch0.1:armhf
516. libwinpr-sysinfo0.1:armhf
517. libwinpr-thread0.1:armhf
518. libwinpr-utils0.1:armhf
519. libx11-protocol-perl
520. libx264-146:armhf
521. libx265-59:armhf
522. libxapian22v5
523. libxcb-composite0:armhf
524. libxcb-render-util0:armhf
525. libxcb-util1:armhf
526. libxcb-xkb1:armhf
527. libxkbcommon-x11-0:armhf
528. libxml-xpathengine-perl
529. libyaml-0-2:armhf
530. libzeitgeist-2.0-0:armhf
531. libzip4:armhf
532. libzzip-0-13:armhf
533. lightdm-gtk-greeter-settings
534. linux-image-3.10.96-78
535. linux-image-xu3
536. linux-tools-4.2.0-23
537. linux-tools-4.2.0-23-generic-lpae
538. linux-tools-common
539. linux-tools-generic-lpae
540. live-boot-initramfs-tools
541. lmodern
542. mailutils
543. mailutils-common
544. mali-x11
545. marco
546. marco-common
547. mate-applets
548. mate-applets-common
549. mate-backgrounds
550. mate-control-center
551. mate-control-center-common
552. mate-desktop
553. mate-desktop-common
554. mate-desktop-environment
555. mate-desktop-environment-core
556. mate-desktop-environment-extra
557. mate-desktop-environment-extras
558. mate-gnome-main-menu-applet
559. mate-icon-theme
560. mate-icon-theme-faenza
561. mate-indicator-applet
562. mate-indicator-applet-common
563. mate-media
564. mate-media-common
565. mate-menus
566. mate-netspeed
567. mate-netspeed-common
568. mate-notification-daemon
569. mate-notification-daemon-common
570. mate-panel
571. mate-panel-common
572. mate-polkit:armhf
573. mate-polkit-common
574. mate-power-manager
575. mate-power-manager-common
576. mate-screensaver

577. mate-screensaver-common
578. mate-sensors-applet
579. mate-sensors-applet-common
580. mate-session-manager
581. mate-settings-daemon
582. mate-settings-daemon-common
583. mate-system-monitor
584. mate-system-monitor-common
585. mate-terminal
586. mate-terminal-common
587. mate-themes
588. mate-user-guide
589. mate-user-share
590. mate-user-share-common
591. mate-utils
592. mate-utils-common
593. media-player-info
594. menu
595. menu-xdg
596. mesa-common-dev:armhf
597. mir-client-platform-mesa-dev:armhf
598. mousetweaks
599. mozo
600. myspell-en-au
601. myspell-en-gb
602. myspell-en-za
603. mythes-en-au
604. mythes-en-us
605. ncurses-term
606. network-manager-pptp
607. network-manager-pptp-gnome
608. ntfs-3g
609. odroid-platform-5422
610. ofono
611. openoffice.org-hyphenation
612. p7zip-full
613. pidgin-otr
614. pinentry-gnome3
615. plank
616. pluma
617. pluma-common
618. postfix
619. pptp-linux
620. printer-driver-brlaser
621. python3-bs4
622. python3-cairo
623. python3-cups
624. python3-cupshelpers
625. python3-html5lib
626. python3-lxml
627. python3-mako
628. python3-markupsafe
629. python3-pexpect
630. python3-pil:armhf
631. python3-pyinotify
632. python3-renderpm:armhf
633. python3-reportlab
634. python3-reportlab-accel:armhf
635. python3-requests
636. python3-smbc
637. python3-systemd
638. python3-uno
639. python3-urllib3
640. python3-virtualenv
641. python3-xdg
642. python-boto
643. python-caja
644. python-caja-common
645. python-cffi
646. python-cffi-backend
647. python-cloudfiles
648. python-cryptography
649. python-enum34
650. python-gtksourceview2
651. python-idna
652. python-ipaddress
653. python-lockfile
654. python-mate-menu
655. python-ndg-httpsclient
656. python-openssl
657. python-pil:armhf
658. python-ply
659. python-pyasn1
660. python-pycparser
661. qttranslations5-l10n
662. rename
663. rhythmbox
664. rhythmbox-data
665. rhythmbox-plugin-cdrecorder
666. rhythmbox-plugins

667. rhythmbox-plugin-zeitgeist
668. rtkit
669. ruby
670. ruby2.1
671. rubygems-integration
672. seahorse
673. session-migration
674. shotwell
675. shotwell-common
676. signond
677. signon-keyring-extension
678. signon-plugin-oauth2
679. signon-ui
680. signon-ui-service
681. signon-ui-x11
682. smartmontools
683. ssh-import-id
684. systemd
685. systemd-sysv
686. tcpd
687. tcptrack
688. telnet
689. texlive-base
690. thunderbird
691. thunderbird-locale-en
692. thunderbird-locale-en-gb
693. thunderbird-locale-en-us
694. tlp
695. tlp-rdw
696. traceroute
697. ttf-ancient-fonts-symbola
698. ubuntu-mate-icon-themes
699. ubuntu-mate-lightdm-theme
700. ubuntu-mate-themes
701. ubuntu-mate-wallpapers
702. ubuntu-mate-wallpapers-common
703. ubuntu-mate-wallpapers-wily
704. ubuntu-minimal
705. ubuntu-mobile-icons
706. ubuntu-mono
707. ubuntu-system-service
708. ubuntu-touch-sounds
709. unity-control-center
710. unity-control-center-signon
711. unity-settings-daemon
712. uno-libs3
713. ure
714. ureadahead
715. urfkill
716. va-driver-all:armhf
717. vdpau-va-driver:armhf
718. virtualenv
719. vlc
720. vlc-data
721. vlc-nox
722. vlc-plugin-notify
723. vlc-plugin-samba
724. wireless-regdb
725. wodim
726. xserver-xorg-core
727. xserver-xorg-input-wacom
728. xserver-xorg-video-armsoc-5422
729. xul-ext-calendar-timezones
730. xul-ext-gdata-provider
731. xul-ext-lightning
732. zeitgeist-core