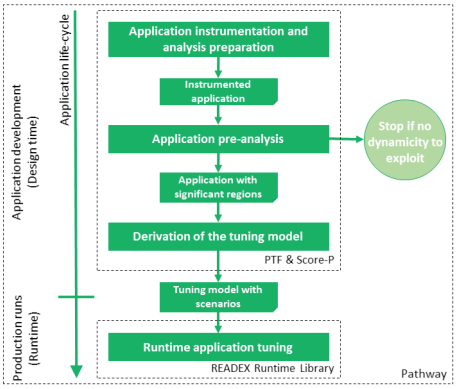
READEX gjør dynamiske HPC-applikasjoner energieffektive

EU-prosjektet READEX (Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing) kombinerer teknikker fra superdatamaskindomenet (high performance computing, HPC) og innvevde systemer (embedded systems). Slik blir HPC programmer med dynamisk oppførsel energieffektive.



Figur 1: Oversikt over READEX metodikk.

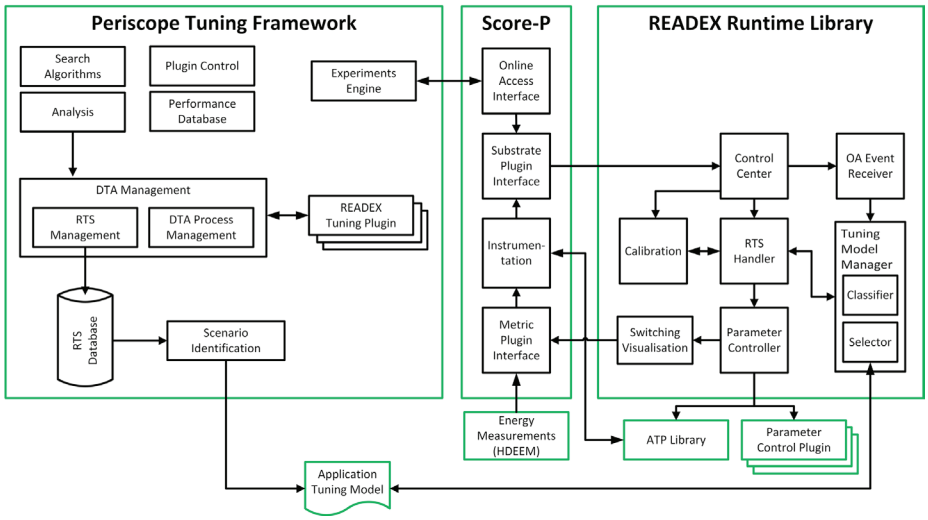
Av Per Gunnar Kjeldsberg, NTNU

Ved design av innvevde systemer har man lenge måttet fokusere på energieffektivitet. Det er motivasjon nok å tenke på en mobiltelefon som man helst vil lade så sjelden som mulig eller en sensornode som lever så lenge batteriet varer. De senere år har energieffektivitet også blitt viktig for datasentre med superdatamaskiner. Strømregningen vil i mange tilfeller langt overskride kostnaden for innkjøp og annen drift av maskinene. Hvilken ytelse du får ut av systemet er dessuten gjerne styrt av hvor varme maskinene blir. For høy temperatur krever at klokkefrekvensen skrues ned eller at komponenter skrus helt av.

System scenario-basert design

En av teknikkene som har vært benyttet for å spare energi innen innvevde systemer er å utnytte det faktum at mange applikasjoner oppfører seg dynamisk. Dette reflekteres da ofte i at det stilles varierende krav til hvilke ressurser det er behov for. Noen ganger skal for eksempel systemet prosessere store videobilder med mye bevegelse, noe som krever mye minne og høy ytelse. Andre ganger er det små bilder med lite bevegelse, med tilhørende reduserte krav til lagerplass og prosessering.

System scenario basert design utnytter slik dynamikk gjennom en delt designtid- og kjøretidmetodikk. I forbindelse med design av systemet studeres og profileres applikasjonen nøye slik at ulike kjøretidssituasjoner kan avdekkes, hver med sine spesifikke krav til ressurser som for eksempel lagerplass (RAM) og ytelse (klokkefrekvens). Disse kjøretidssituasjonene er gjerne knyttet opp mot interne egenskaper i programkoden slik som løkkestrukturer og kontrollsteg. Det gjør dem vanskelige å oppdage ved bare å se på systemets overordnede oppførsel. Man må istedenfor studere og profilere koden, også for å avdekke hvilke interne variable som er bestemmende for dynamikken i oppførselen. For å redusere systemkompleksiteten samles kjøretidssituasjoner med liknende ressurskrav i et begrenset antall scenarier. Deretter optimaliseres systemet for kjøring av hvert enkelt scenario, det utvikles mekanismer for å detektere det kommende scenariet, og for å svitsje mellom dem. Hvis vi igjen tenker på videobehandling, så vil liknende bildestørrelser samles i et scenario, og prosesseringsplattformen skrur av eventuelt overflødig dataminne og tilpasser klokkefrekvens og forsyningsspenning til det gjeldende kravet. Dermed spares energi når kravene ikke er maksimale. Når applikasjonen brukes i kjøretid overvåkes variable som bestemmer bildestørrelse, og plattformen svitsjer automatisk til den konfigurasjonen som trengs for det kommende scenariet.



HPC autotuning

Også for HPC-systemer kan man spare energi ved tilpasse applikasjon

og plattform til varierende ressursbehov. Noen ganger er det overføring av store datamengder fra minne som er flaskehalsen, og da kan klokkefrekvensen på prosessoren settes ned. Andre ganger er det behov for å foreta beregninger så raskt som mulig, og da med høyest mulig klokkefrekvens. Et annet eksempel er at det kan variere dynamisk hva som er det optimale antall tråder som bør kjøres i parallell på slike systemer.

Verktøyet PTF (Periscope Tuning Framework) er et automatisert rammeverk for søk etter optimale plattformkonfigurasjoner for å redusere energiforbruket i HPC applikasjoner. PTF sitt hovedprinsipp er bruk av formalisert ekspertkunnskap og strategier kodet inn i et antall «tuning plugins». Disse benyttes ved automatisk kjøring av eksperimenter som evaluerer konsekvensen av ulike plattformkonfigurasjoner. Det er for eksempel en egen plugin for å evaluere avveiningen mellom eksekveringstid og energiforbruk ved dynamisk spennings- og frekvensskalering (DVFS). Et viktig element her er også effektive og intelligente søkealgoritmer siden løsningsrommet for alternative konfigurasjoner er svært stort, og hvert eksperiment tar tid og må kjøres på en virkelig plattform. Den opprinnelige versjonen av PTF utførte såkalt statisk autotuning. Det vil si at målet er å finne den konfigurasjonen som i gjennomsnitt er best for applikasjonen. I kjøretid settes denne konfigurasjonen i starten og beholdes gjennom hele eksekveringen.

HW config.	Region	Core freq.	Uncore freq.	#Threads	Energy saving	Time saving
Deafault	All	2.4 GHz	3.0 GHz	24		
Static	All	2.5 GHz	2.2 GHz	16	15.7%	-6.2%
Dynamic	Vh	2.5 GHz	1.4 GHz	24		
	Vk	2.1 GHz	1.4 GHz	24		
	GMRES	1.7 GHz	2.2 GHz	8		
	PRINT	2.5 GHz	3.0 GHz	8		
	TOTAL				34.0%	10.9%

Tabell 1: Resultater ved dynamisk rekonfigurering av frekvenser og antall tråder i BEM4I.

Kombinasjon av teknikker

I READEX-prosjektet har vi kombinert teknikker fra system scenarios og autotuning. Figur 1 viser den overordnete flyten, som er delt mellom designtid og kjøretid. I designtid foretas først en instrumentering av applikasjonen ved å sette inn såkalte probefunksjoner rundt ulike regioner i koden. Regioner kan for eksempel være funksjoner eller løkkestrukturer og probefunksjonene samler data og informasjon om regionen under eksekvering. Instrumenteringen kan skje både automatisk ved hjelp av verktøyet Score-P og ved å la brukeren utnytte sin domeneekspertise til å indikere deler av koden som har dynamisk potensiale. Deretter kjøres applikasjonen på den aktuelle HPC plattformen for å avdekke potensielt dynamisk oppførsel. Et eksempel på slik kan være at systemer skifter mellom regioner som er minnebundet og prosesseringsbundet. Her avgjøres det også om de ulike regionene i applikasjonen er signifikante eller ikke, basert på om kjøretiden er lang nok til å være av relevans. Gitt at dynamisk oppførsel detekteres, brukes PTF til å søke etter optimale konfigurasjoner for hver enkelt av de signifikante regionene. Ja, fordi en region kan ha ulik optimal konfigurasjon avhengig, for eksempel, av hvilken funksjon som har kalt regionen, så søkes det etter optimale konfigurasjoner for hver av disse ulike kjøretidssituasjonene. Kjøretidssituasjoner med identisk eller liknende konfigurasjon grupperes i et begrenset antall scenarier. Informasjon om disse, samt hvordan de skal detekteres i kjøretid, samles så i en Application Tuning Model (ATM) i form av en serialisert tekstfil.

Når applikasjonen eksekveres i kjøretid, for eksempel for å generere en ny værmelding for norske-

READEX-partnere på møte i Trondheim i september 2017. Konsortiet består av tyske TU Dresden, TU München og GNS mbH, tsjekkiske IT4Innovation, National University of Ireland Galway, Intel France og Norges teknisk-naturvitenskapelige universitet. Fra NTNU deltar Institutt for elektroniske systemer og Institutt for datateknologi og informatikk.

kysten, så starter prosessen med å lese ATM inn i et READEX kjøretidsbibliotek. Herfra detekteres så de ulike kjøretidssituasjonene som oppstår, og plattformen og applikasjonen konfigureres i henhold til det tilhørende scenariet. Figur 2 viser et relativt detaljert bilde av hvordan de ulike verktøyene spiller sammen. Uten å gå i ytterligere detaljer her så kan nevnes at det eksisterer ulike teknikker for evaluering av energiforbruk, både basert på egne FPGA-kort koblet til nodene i superdatamaskinen og ulike register for energiovervåking inne i selve prosessorkjernene. Videre samspiller kjøretidsbiblioteket med et antall parameterkontroll plugins, som tar seg av den praktiske konfigureringen av plattformen, for eksempel ved å endre klokkefrekvens og forsyningsspenning. Kjøretidsbiblioteket har dessuten en egen kalibreringsmekanisme, bygd på maskinlæringsprinsipper, som håndterer uventede situasjoner i kjøretid.

Energisparingsresultater

Det er selvsagt varierende hvor mye dynamikk det er å finne i ulike applikasjoner. Gjennom prosjektet har vi imidlertid funnet at dette er til stede i svært mange tilfeller, men samtidig også sett at det er vanskelig og meget tidkrevende å detektere og optimalisere dette manuelt. Det trengs verktøy av den typen som er utviklet i READEX. Et eksempel på resultater som er oppnådd er vist i tabell 1. BEM4I er et bibliotek for løsning av partielle differensiallikninger som for eksempel brukes innen klimamodellering og simulering av væskeflyt. BEM4I har fire signifikante regioner. To er prosesseringsbundet, Vh og Vk, en er minnebundet, GMRES, og en er IO-bundet, PRINT. Vi ser at statisk optimalisering gir en energibesparelse på 15,7% sammenliknet med maskinens standardinnstilling. Ved hjelp av READEX-verktøyene er det funnet at ved dynamisk konfigurering av klokkefrekvens og forsyningsspenning for henholdsvis prosessorkjernene (core) og de delene av noden som ikke er del av en kjerne (uncore), samt hvor mange tråder som kjøres i parallell, så kan det spares 34% energi. Verktøyene genererer dessuten det som skal til for eksekvering i kjøretid, og i besparelsen er overhead fra kjøretidsbiblioteket også medregnet. I dette tilfellet gir dynamisk konfigurering av antall tråder dessuten forbedring av såkalte NUMA-effekter, noe som også resulterer i tidsbesparelser i kjøretid.

I samarbeid med Aker BP har vi i READEX eksperimentert med applikasjonen OptEWE. Den inneholder beregningstunge deler av en algoritme som genererer 3D-bilder av øvre deler av jordoverflaten basert på seismiske bølger. Her oppnådde vi 9,7% redusert energiforbruk på bekostning av 7,0% økning i kjøretid.

Konsortium

READEX er nylig avsluttet etter tre års samarbeid mellom akademiske og industrielle partnere i fem europeiske land. Koordinator har vært Technische Universität Dresden i Tyskland. Figur 3 viser et bilde av konsortiet samlet til møte i Trondheim i september 2017. READEX er et prosjekt av typen «Future and Emerging Technologies» og har fått støtte fra EUs Horizon 2020 forskningsprogram gjennom avtale 671657. Mere detaljer om prosjektet finner du på https://www.readex.eu.