TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta

Arttu Klemettilä

SUORITUSAIKOJEN JA ENERGIANKULUTUKSEN OPTIMOIMINEN TIETOKONEKLUSTEREIDEN JONONHALLINTAJÄRJESTELMISSÄ

Mat-2.4108 Sovelletun matematiikan erikoistyöt

Espoo 1.9.2009

Työn ohjaajat:

FT Tapio Niemi, DI Jukka Kommeri

Tekijä: Arttu Klemettilä

Työn nimi: Suoritusaikojen ja energiankulutuksen optimoiminen tietokoneklustereiden jononhallintajärjestelmissä

Päivämäärä: 1.9.2009 Kieli: Suomi Sivumäärä: 3+16

Tutkinto-ohjelma: Teknillinen fysiikka ja matematiikka

Valvoja: Prof. Harri Ehtamo

Ohjaajat: FT Tapio Niemi, DI Jukka Kommeri

Tietokoneverkkojen kasvaminen ja yleistyminen ovat herättäneet keskustelua myös ympäristöasioiden tiimoilta. Valtavien laskentaverkkojen sähkönkulutus muodostaa valtaosan käyttökuluista, ja pienetkin vähennykset saavat aikaan merkittäviä säästöjä kustannuksissa ja pienentävät ympäristöön kohdistuvaa kuormaa. Erilaisia lähestymistapoja energiankulutuksen vähentämiseksi on useita, mutta tässä työssä keskitytään jononhallintajärjestelmään ja töiden ajastamiseen.

Koska suurin osa tietokoneiden sähkönkulutuksesta on peräisin kulutuksen vakioosasta, saavutetaan vähiten energiaa kuluttava ratkaisu lähes poikkeuksetta minimoimalla töiden kokonaisläpimenoaika. Työssä perehdytään kokonaissuoritusajan minimoimiseen jononhallintajärjestelmän avulla. Erilaisten algoritmien toiminnan kannalta on keskeistä tunnistaa laskentakoneiden pullonkaularesurssit ja varmistaa näiden resurssien mahdollisimman tehokas hyödyntäminen.

Työssä tutkitaan yhdellä ytimellä useamman työn ajamista yhtäaikaisesti. Tätä tietoa voidaan käyttää pohjana monimutkaisemmille ajastusalgoritmeille ja toisaalta sen avulla voidaan päätellä millaisia töitä laskentakoneilla kannattaa ajaa yhtäaikaisesti. Käy ilmi, että tavallisesti parhaan suorituskyvyn saavuttaa ajamalla noin 3-5 työtä suoritinydintä kohden.

Myös käytettävillä laskentakoneilla on suuri merkitys energiankulutuksessa. Testiohjelmat ajettiin kolmessa erilaisessa testiympäristössä. Tutkimuksen perusteella voidaan arvioida, että uusissa laskentakoneissa on selvästi vanhoja koneita parempi hyötysuhde. Myös uudet, lähinnä kannettaviin tietokoneisiin tarkoitetut energiansäästöprosessorit osoittautuivat erittäin ympäristöystävällisiksi, vaikkakin huomattavasti perinteisempiä laskentakoneita hitaammiksi.

Avainsanat: Laskentaverkot, klusterit, jononhallintajärjestelmät, ympäristöystävällinen laskenta

Sisältö

1 Johdanto

Erityisesti informaatioteknologioiden ja internetin kehittyminen koko maapallonlaajuisiksi palveluiksi ovat kiihdyttäneet tietokoneiden maailmanvalloitusta valtaviin mittasuhteisiin. Useat web-yritykset sijoittavat suuria määriä rahaa palvelinkeskuksiin ja erilaisiin tietoverkkojärjestelmiin ympäri maailman. Kuluttajapalveluiden ohella myös useiden yritysten tuotekehitysosastojen ja tieteellisen yhteisön laskentakapasiteettivaatimus on kasvanut suunnattomiin lukemiin. Lähes jokaisella informaatioteknologian yrityksellä tai tutkimuslaitoksella on nykyään valtava laskentaan tarkoitettu tietokoneverkko.

Teknologian yleistymisen myötä energiankulutus ja ympäristöarvot ovat olleet yksi viime vuosikymmenten tärkeimmistä puheenaiheista. Tietokoneiden määrän nopea kasvu on kasvattanut huolta myös ympäristöasioiden tiimoilta, sillä laskentakeskusten tehonkulutukset ovat usein valtavia. Keskimääräinen tietokone kuluttaa sähköä 100W - 300W ja kun näitä tietokoneita yhdistetään satoja tai tuhansia, kokonaiskulutus kasvaa erittäin suureksi. Näissä mittasuhteissa kymmenenkin prosentin säästö ei vain kevennä ympäristökuormaa, vaan saa myös aikaan suuria säästöjä kustannuksissa [1]. Valtaosa suurten tietokonekeskusten kustannuksista syntyykin juuri käyttöenergiasta.

Tämä työ on tehty Euroopan hiukkasfysiikan tutkimuskeskuksen (CERN) yhteydessä toimivassa Helsinki Institute of Physicsin teknologiaryhmässä [2]. Todennäköisesti vuonna 2009 käynnistyvä Large Hadron Collider (LHC) -hiukkaskiihdytin tuottaa valtaisat määrät analysoitavaa dataa, jonka analysointia varten on rakennettu World Wide LHC Computing Grid (WLCG). WLCG on valtava tietokoneverkko, joka sisältää yli 50 000 prosessoriydintä 170 laskentakeskuksessa ympäri maailman, eli säästöjen saavuttaminen on paitsi ympäristön kannalta tärkeää, niin myös taloudellisesti merkittävää. [3, 4]

Työssä perehdytään tarkemmin suuren työjoukon kokonaisläpäisyajan tutkimiseen, sekä yksittäisen työn vaatimaan energian tarkasteluun. Keskeinen kysymys on, miten työt kannattaa allokoida lasketaverkkoon siten, että kokonaisläpäisyaika minimoituu ja toisaalta töiden kokonaissähkönkulutus saadaan mahdollisimman pieneksi. Koska noin 60 % tietokoneiden sähkönkulutuksesta muodostuu tyhjäkäyntikulutuksesta, nämä kaksi tavoitetta saavutetaan yleensä samanaikaisesti [1]. Perinteisesti tutkimukset ovat kuitenkin keskittyneet erityisesti yksittäisen työn läpäisyajan minimoimiseen, mutta nyt tavoitteena on minimoida koko työjoukon kokonaisläpäisyaika ja täten myös energiankulutus.

Tapoja hillitä energiankulutusta on useita, mutta tässä työssä keskitytään jononhallintajärjestelmän optimointiin. Erityisesti kiinnitetään huomiota koneiden pullonkaulojen tunnistamiseen, ja siihen, kuinka monta työtä on tehokasta ajaa yhtäaikaisesti. Perinteisesti laskentatöitä allokoidaan yksi jokaista laskentaydintä kohden, mutta tällä tavalla ei välttämättä saavuteta maksimaalista hyötysuhdetta. Kuitenkin ajamalla useita töitä yhtäaikaisesti, voidaan saavuttaa merkittäviäkin suorityskyvyn parannuksia. [5]

Aluksi esitellään lasketaverkkojen perusominaisuuksia sekä allokointijärjestelmän yksinkertaiset ominaisuudet. Tämän jälkeen käydään läpi testiohjelmina käytetyt ohjelmistot ja laitteet, sekä mittaukset. Lopuksi esitellään mittauksista saadut tulokset, joista päätellään laskentaverkon pullonkaulat.

2 Laskentaverkot

2.1 Laskentaverkkojen rakenne

Aluksi on syytä käydä läpi muutamia käsitteitä liittyen laskentaverkkoihin. Klusteri (*Cluster*) on joukko tietokoneita, jotka toimivat läheisessä yhteistyössä muodostaen yhden suuren tietokoneen. Klusterit sisältävät tyypillisesti kymmenistä satoihin tietokonetta, joita kutsutaan tavallisesti laskentakoneiksi (*Computing Node*). Lasketakoneissa on yksi tai useampi prosessori, joissa kussakin yksi tai useampi ydin (*Core*). Lasketakoneita hallitaan hallintakoneella (*Frontend*), joka allokoi laskettavat työt laskentakoneille jonojenhallintakoneessa ajettavan jononhallintajärjestelmän (*Queue management system*) avulla.

Työt (*Job*) ovat laskennallisia kokonaisuuksia, jotka annetaan koko verkon laskettavaksi jononhallintajärjestelmän kautta. Työt koostuvat useasta tehtävästä (*Task*), jotka ovat jononhallintajärjestelmän kannalta pienin yksittäinen osatyö. Tehtäviä lähetetään lasketakoneiden jonoihin (*Queue*), joista ne siirtyvät suorittimelle prosessoitavaksi. Jonossa voi olla myös useampi laskentakone tai kokonainen klusteri. Kun kaikki työn tehtävät on saatu suoritettua, voidaan mitata kokonaisläpäisyaika. Jakamalla tehtävien määrä läpäisyajalla saadaan laskettua työvuo (*Throughput*), joka kertoo kuinka monta työtä järjestelmä kykenee suorittamaan aikayksikössä.

Laskentaverkko (*Grid*) voidaan käsittää yhtenä suurena klusterina. Laskentaverkko koostuu tyypillisesti useista klustereista, jotka voivat olla hyvin kaukana toisistaan, jopa eripuolilla maapalloa. Laskentaverkko lähettää töitä klustereille jotka allokoivat ne edelleen omille laskentakoneilleen. Tällä tavoin saadaan valtavan suuri järjestelmä, jossa laskenta on hajautettu laajalle alalle. Laskentaverkon yhteenlaskettu suoritinytimien määrä voi helposti olla kymmenissä tuhansissa, jolloin laajojenkin laskentatöiden läpivienti onnistuu kohtuullisessa ajassa.

Viime vuosikymmenten tietokonearkkitehtuurien kehitys on johtanut entistä enemmän tilanteeseen, jossa yksittäisten suorittimien tehot eivät enää kasva voimakkaasti, vaan sen sijaan niiden määrää kasvatetaan. On tavallista, että jokaisessa prosessorissa on vähintään kaksi ydintä, mutta erityisesti laskentaan tarkoitetuissa prosessoreissa yritimiä voi olla neljä tai kahdeksan. Samalla kun yksittäiset tietokoneet laskevat useita töitä yhtäaikaisesti, tietokoneita on alettu yhdistää klustereiksi ja siitä edelleen valtaviksi laskentaverkoiksi. Tämä asettaa luonnollisesti uusia haasteita erityisesti ohjelmistoarkkitehtuurille, sillä ohjelmia on kyettävä ajaamaan usealla ytimellä yhtäaikaisesti. Useat yritykset ja tutkimuslaitokset ovatkin tehneet periaatepäätöksiä laskentaohjelmien muuttamisesta paremmin rinnakkaislaskentaan

sopiviksi. [4]

2.2 Jononhallintajärjestelmät

Klustereita ja laskentaverkkoja hallitaan erityisillä ohjelmistoilla, jononhallintajärjestelmillä. Järjestelmän tärkein tehtävä on allokoida tai ajastaa annetut tehtävät käytössäoleville laskentakoneille.

Tehtävien jakaminen lasketakoneille voidaan hoitaa usealla eri tavalla. Valittava menetelmä riippuu laskettavien töiden luonteesta ja resurssivaatimuksista. Töiden ominaisuuksia, kuten oletettua lasketa-aikaa, muisti- tai I/O- vaatimuksia ei välttämättä tiedetä etukäteen ja työt voivat myös olla keskenään hyvin erilaisia, jolloin hallintajärjestelmän on otettava huomioon eri töiden vaatimat resurssit. Töillä voi olla myös järjestysvaatimuksia, eli jokin työ on laskettava ennen toista työtä. Nämä rajoitukset voivat johtaa tilanteeseen, jossa hallintajärjestelmän on käytettävä laskentakoneita puoliteholla samalla kun odotellaan toisen työn valmistumista tai resurssien vapautumista.

Yksinkertainen esimerkkitilanne töiden hallinnasta voisi olla seuraavanlainen. Työhön kuuluu 5 tehtävää, joista tehtävä 1 on suoritettava ennen tehtävää 2. Kukin tehtävä kestää noin 3 minuuttia lukuunottamatta tehtävää yksi, joka kestää keskimäärin 6 minuuttia. Laskentakoneessa on käytettävissä 1 gigatavu muistia, ja jokainen työ vaatii 500 megatavua muistia. Muuten töitä voidaan suorittaa mielivaltaisessa järjestyksessä ja rinnakkain. Laskentakoneessa on 2 prosessoriydintä, ja suoritusajat ovat likimain vakiot, riippumatta siitä, ajetaanko yhtäaikaisesti yhtä vai kahta työtä. Miten hallintajärjestelmän on ajastettava työt, jotta työt saadaan järjestelmän läpi mahdollisimman nopeasti?

Ajastus voidaan suorittaa esimerkiksi siten, että aluksi laskentakoneelle annetaan työt 1 ja 3. Kun tällöin muisti on täynnä eikä koneille voi antaa enempää työtä ilman että laskenta hidastuu merkittävästi. Kun noin kolmen minuutin kuluttua työ 3 valmistuu, niin lasketakoneelle voidaan lähettää uusi työ, työ 4. Työt 4 ja 1 valmistuvat likimain samaan aikaan, ja koneen muistiin mahtuu tällöin 2 työtä. Lähetetään työt 2 ja 5, jotka valmistuvat noin kolmen minuutin kuluttua. Näin saatiin kaikki tehtävät suoritettua ja aikaa kului noin 9 minuuttia.

Todellisuudessa tilanne ei ole näin yksinkertainen, sillä suoritusaikoja ei tunneta tarkkaan eikä resurssivaatimukset ole yksikäsitteisiä. Lisäksi useamman työn yhtäaikainen prosessointi saattaa vaikuttaa muun muassa prosessointiaikaan tai muistinkulutukseen.

Erilaisia algoritmeja jonojen hallintaan on olemassa useita, ja yksinkertaisimmat niistä voidaan jakaa neljään ryhmään:

- *Yksi työ jokaista prosessoriydintä kohden.* Tämä on perinteinen ja yksinkertaisin tapa hallita jonojärjestelmää. Jokaiselle ytimelle annetaan yksi työ prosessoitavaksi ja kun se saadaan valmiiksi, otetaan jonosta uusi työ prosessoitavaksi.

- *Useampi työ jokaista prosessoriydintä kohden.* Algoritmi on käytännössä täysin sama kuin ensimmäinen, mutta nyt samalle prosessorille annetaan useampi työ prosessoitavaksi yhtäaikaisesti. Yksittäisen työn suoritusaika todennäköisesti hidastuu, mutta kokonaissuoritusaika voi lyhentyä.

- *Kuormaan perustuvat menetelmät.* Kuten edellä, laskentaytimelle voidaan antaa useita tehtäviä yhtäaikaisesti prosessoitavaksi, mutta niiden lukumäärä määräytyy laskentakoneen resurssien, kuten muistin ja I/O-liikenteen käyttöasteen perusteella. [2]

- *Yksi työ useammalle ytimelle.* Jotta tämä menetelmä olisi tehokas, työt on voitava rinnakkaistaa useammalle prosessoriytimelle. Tällöin yksittäiset työt saadaan ajettua systeemin läpi nopeammin.

Lukuunottamatta viimeistä menetelmää, kukin algoritmi on edellisen algoritmin yleistys, mutta vaatii aina lisää informaatiota suoritettavista prosesseista. Mikäli laskentakoneelle antaa liian monta työtä prosessoitavaksi kerralla, kokonaissuoritusaika alkaa hidastumaan. Tämän työn tarkoituksena on selvittää, mikä on optimaalinen määrä yhtäaikaisesti suoritettavia ohjelmia, ja toisaalta mistä tämän optimin arvo johtuu.

2.3 Jononhallintaohjelmistot

Erilaisia jononhallintajärjestelmiä on saatavilla useita. Eräitä tunnetuimpia ovat Torque [1], OpenPBS [2], LSF [3], Condor [4] ja Sun Grid Engine [5]. [2]

Tässä työssä käytetään Sun Microsystemssin kehittämää Sun Grid Engineä (SGE), joka on hyvin monipuolinen ja ilmainen jononhallintajärstelmä. SGE:tä käytetään laajasti hallinnoimaan myös suuria tietokoneverkoja. Tehtävien allokointi tapahtuu kiintein aikavälein ja se perustuu töiden resurssivaatimuksille sekä laskentakoneiden käytettävissä olevien resurssien monitorointiin. [6]

SGE:n ajastusalgoritmin voi valita vapaasti, mutta oletuksena töiden ajastaminen tapahtuu lähettämällä töitä saapumisjärjestyksessä sille laskentakoneelle, jolla resurssien käyttö on vähäisintä. Laskentakoneille voidaan asettaa resurssikohtaisia ylärajoja, jonka jälkeen koneelle ei enää lähetä uusia töitä ennen kuin resurssin käyttötaso on tippunut rajan alle. Koneille asetetaan myös aina jokin kiinteä yläraja, jota useampaa työtä jonoon ei voi lähettää. Tätä rajaa kutsutaan tässä työssä ydinkohtaiseksi suorituspaikkamääräksi (*Number of slots*). Resurssitasoja voidaan myös skaalata eri laskentakoneilla, jolloin saadaan tarkempi tieto laskentakoneen todellisesta kuormitusasteesta. Töille voidaan myös asettaa resurssivaatimuksia tai

[1] www.clusterresources.com/pages/products/torque-resource-manager.php
[2] http://www.openpbs.org
[3] www.platform.com/Products/Workload-Management
[4] www.cs.wisc.edu/condor
[5] gridengine.sunsource.net

prioriteettejä, joiden avulla voidaan taata töille esimerkiksi tarpeeksi käytettävissä olevaa muistia tai määrittää eri töiden keskinäisiä suoritusjärjestyksiä. [2]

SGE sisältää myös ARCo raportointikonsolin, jonka avulla laskentakoneiden resurssien seuraaminen myös jälkikäteen on helppoa. SGE monitoroi automaattisesti laskentakoneiden suoritin-, muisti- ja I/O-käyttöä, joita tarkastelemalla voidaan paikallistaa jonon pullonkaulat. Tämän tutkimuksen yhteydessä töiden suoritinajat ja muut ominaisuudet luetaan ARCon kautta. ARCoon voidaan myös lisätä omia mittareita, jolloin esimerkiksi sähkönkulutuksen työkohtaisen kulutuksen mittaaminen voidaan automatisoida.

2.4 Energiankulutus

Tietokoneen energiankulutus koostuu kahdesta osasta: kulutuksen vakio- ja kuormaosasta. Kun kone on päällä, ilman ylimääräistä kuormaa, se kuluttaa vakio-osan verran sähköä. Kuormaosaan taas lasketaan kaikki koneen hyötykäytöstä johtuva ylimääräinen teho. Prosessorit, muistit, kovalevyt ja kaikki muut tietokoneen osat kuluttavat sitä enemmän sähköä mitä enemmän niitä käytetään. Käyttöjärjestelmän ja muiden taustaprosessien aiheuttama kuorma lasketaan kuitenkin kulutuksen vakio-osaan.

Tyypillisesti tavallisten laskentakäyttöön tarkoitettujen tietokoneiden tehot ovat suuruusluokassa 100 W - 300 W. Vakio-osuuden suuruudet ovat tyypillisesti noin kaksi kolmasosaa kokonaistehonkulutuksesta, ja kuorman suhteen tehonkulutus kasvaa prosessoitavan kuorman mukaisesti maksimitehoon. [1]

Koska vakio-osan osuus on huomattavan suuri, on työn suorittamiseen käytetystä energiasta suurin osa peräisin juuri kulutuksen vakio-osasta. Tästä syystä usein energiataloudellisin tapa viedä työt systeemin läpi on minimoida työjoukon kokonaisläpäisyaika. Laskentakoneiden kapasiteetin vain osittainen hyödyntäminen luonnollisesti vähentää kuormatehoa, mutta tällöin kulutuksen vakio-osasta aiheutuva energiankulutus kasvattaa kokonaiskulutusta suhteellisesti enemmän. Tämä ilmiö voidaan havaita mittauksista varsin helposti, ja onkin erittäin vaikea löytää tilannetta jossa nopeudesta luopumisella saavutettaisiin hyötyä energiankulutuksessa.

Energiankulutuksenhallintaan on useita lähestymistapoja. Koneiden automaattista sammuttamista ja käynnistämistä ovat tutkineet muun muassa Pinheiro et al. [7]. Tässä menetelmässä toimettomat koneet sammutetaan jos niitä ei tarvita, jolloin tyhjäkäyntikulutus vähenee huomattavasti. Muun muassa Ge et al. [8] sekä While, Kappiah et al. [9] ovat tutkineet klustereiden suoritinydinten tehon dynaamista muuntamista, jolloin toimettomat laskentakoneet kuluttavat vähemmän energiaa. Osa tutkimuksesta on keskittynyt tietokonekeskusten ilmastoinnin ja jäähdytyksen energiakulutuksen optimointiin, muun muassa Moore et al. sekä Heath et al. [10, 11].

Tässä työssä keskitytään jononhallintajärjestelmän avulla tapahtuvaan suorituskyvyn optimointiin. Tällä tasolla tapahtuva optimointi on usein hyvin helppo toteuttaa, sillä muutokset eivät vaadi vanhojen laitteiden päivittämistä tai esimerkiksi uu-

den jäähdytysjärjestelmän hankintaa. Kaikki yleisimmät jononhallintajärjestelmät tukevat tämänkaltaisten asetusten vaihtamista, ja ainakin yksinkertaiset algoritmit ovat hyvin helppoja implementoida.

3 Mittaukset

3.1 Mittauksissa käytettävät laitteistot

Mittaukset pyritään tekemään usealla erilaisella laitteistolla, jotta saatavista tuloksista voidaan johtaa mahdollisimman yleisiä johtopäätöksiä. Ensimmäinen käytetty verkko koostui kolmesta laskentakoneesta, joissa oli kaksi Intel Xeon 2.8 GHz prosessoria, (Supermicro X6DVL-EG2 emolevy, 2048 kB L2 välimuistia, 800 MHz fsb), 2 gigatavua muistia ja 160 gigatavun kovalevy. Toinen testiympäristö oli yksi laskentakone joka sisälsi kaksi AMD Quad-Core Opteron 2376 2.3 Ghz prosessoria (4 MB välimuistia), 32 gigatavua muistia ja 160 gigatavun kovalevy. Kolmantena ympäristönä käytettiin kolmea vähävirtaista Asus EeeBox ThinClient konetta. EeeBoxit sisälsivät yhden Intel Atom N270 prosessorin, 2 gigatavua muistia sekä 160 gigatavun kovalevyn.

Kaikissa järjestelmissä käytettiin Linux-käyttöjärstelmää kernelin versiolla 2.6.18 ja Rocks Linux 5.0 klusterinhallintajärjestelmää. Rocks Linux käyttää Sun Grid Engine 6.0 (*SGE*) työnhallintajärjestelmää. EeeBox koneiden verkkoajureiden kanssa ilmeni ongelmia Rocks Linux 5.0:n kernelin kanssa, joten niiden kanssa päätettiin käyttää uudempaa 5.2 versiota Rocks Linuxista, joka sisälsi myös uudemman 6.2 version SGE:stä.

Testiympäristö	Kon. lkm.	Ydinten lkm./kone	Kellotaajuus (GHz)	Muisti (Gb)
Xeon klust.	3	2	2.8	4
Opteron	1	8	2.3	32
EeeBox klust.	3	1	1.6	2

Taulukko 1: Tiivistelmä testiympäristöistä

3.2 Mittauksissa käytettävät ohjelmat

Klustereiden suoriutumisen testaamista varten kirjoitettiin kolme erillistä ohjelmaa.

- *Suoritinrasitustesti* laskee 10^8 liukulaskutoimitusta. Ohjelman suorittamiseen tarvitaan laskentakoneesta riippuen kymmenistä sekunneista muutamiin minuutteihin prosessoriaikaa, mutta muistia tai I/O:ta ei rasiteta käytännössä lainkaan.

- *Muistirasitustesti* varaa noin 300 megatavua koneen muistista ja kirjoittaa sen täyteen dataa. Tämän jälkeen osaa muistista luetaan ja tieto kopioidaan toiseen kohtaan muistia useita kertoja peräkkäin. Muistissa olevalle datalle tehdään pieni laskutoimitus kopioinnin yhteydessä, jotta vältyttäisiin käännöksenaikaselta optimoinnilta.

- *I/O test*issä koneen I/O väylää pyritään rasittamaan kirjoittamalla laskentakoneen kovalevylle satunnaista dataa noin 300 megatavun verran. Tämän jälkeen data luetaan uudestaan, muokataan hieman ja tallennetaan uudestaan yhteensä 20 kertaa. Koko tiedostoa ei pidetä muistissa koko ohjelman suorituksen ajan, jotta vältytään ylimääräiseltä muistinkulutukselta. Satunnaisuudessa ja välimuokkauksessa käytetään siemenlukuna tehtävän prosessin tunnistenumeroa (PID), jotta vältyttäisiin useamman samanaikaisen prosessin tietojen välimuistittamiselta.

Kukin ohjelma rasittaa vain yhtä koneen resurssia ja jättää muut resurssit suhteellisen vähälle käytölle. Poikkeuksen tekee luonnollisesti suorittimen kuorma, sillä muisti- ja kovalevyoperaatiot vaativat luonnollisesti myös runsaasti suoritinaikaa.

Ylläkuvatun kolmen ohjelman avulla voidaan tutkia kunkin kolmen tärkeän resurssin yksittäisvaikutusta, mutta käytännön sovelluksissa käytetään kaikkia kolmea resurssia. Tämän vuoksi testiohjelmia ajetaan myös *sekaisin* siten, että joka toinen työ on suoritintyö ja neljäsosa töistä on muisti- ja neljäsosa kovalevytöitä.

Sekoituksen lisäksi yhteisvaikutusta tutkitaan ajamalla CERNin CMS projektin testidataa. Käytetty ohjelma laskee *CRAFT-analyysiä* (CMS Running At Four Tesla), joka analysoi LHC:n kosmisen säteilyn testiajodataa vuodelta 2008 [12, 13]. Tämä ohjelma tulee olemaan hyvin lähellä sitä laskentaa, jota tullaan tekemään LHC:n käynnistyttyä ja kun varsinaista törmäysdataa on saatavilla [4]. Ohjelman suoritus kestää koneesta riippuen muutamia minuutteja, ja käyttää noin 250 megatavua muistia. Syötetiedostona käytetään noin 300 megatavun datatiedostoa ja saadut tulokset tallennetaan analyysin jälkeen tekstitietostoon.

3.3 Mittausten suorittaminen

Mittaukset tehdään lähettämällä suuri joukko samanlaisia töitä verkkoon ja mittaamalla läpäisyaika. Energiankulutusta seurataan mittaamalla laskentakoneiden sähkönkulutusta Wattspro -sähkömittarilla, jota hallinnoidaan USB-liittymän kautta jononhallintakoneella. Erityisesti kiinnostavia mitattavia suureita ovat käytetty kokonaisenergia ja keskimääräinen teho. Keskimääräinen työkohtainen energian kulutus saadaan luonnollisesti jakamalla käytetty energia töiden määrällä. Aikamittauksiin ja muuhun monitorointiin käytetään SGE:n omaa kirjanpitojärjestelmää ja ARCo raportointikonsolia.

Koska kaikki ajettavat työt ovat käytännössä keskenään samanlaisia eikä niiden välillä ole riippuvuussuhteita, ainoa muuttuva parametri on kuinka monta työtä koneella ajetaan yhtä aikaa. Kutakin testiohjelmaa lähetetään jonojärjestelmään sellainen

määrä, että niiden kokonaislaskenta-ajaksi tulee noin 30 - 60 minuuttia. Osaa töistä aletaan prosessoida heti työn lähettämisen jälkeen ja muut odottavat, kunnes laskentakone saa aikaisemmat työt prosessoitua. Työn valmistuttua sen suorituspaikka vapautuu. Jokaisella jononhallintakierroksella kaikki tyhjät suorituspaikkat täytetään uusilla töillä, kunnes kaikki paikat ovat jälleen täysiä. Mittauksissa käytetään hallintakierrosten välisen ajanjakson pituudelle SGE:n vakioarvoa, 15 sekunttia.

Johtuen useista, yleensä hallitsemattomista syistä ohjelmien suoritusajat eivät ole vakioita. Tämän vuoksi ajojen on oltava tarpeeksi pitkiä ja niissä on oltava tarpeeksi monta yksittäistä tehtävää. Mittausten hajontaa testattiin ajamalla kullekin testiohjelmalle jollakin suorituspaikkamääräasetuksella kymmenen identtistä kontrolliajoa, jotka kukin vastasivat varsinaisia mittausajoja. Tällä tavalla saatujen mittausten työvuot ja työkohtaiset energiakulutukset osoittautuivat hyvin tarkasti saman suuruisiksi, ja vaihtelu oli suurimmillaankin ainoastaan parin prosentin luokkaa. Tästä poikkeuksena esiin nousi kuitenkin 8-ytiminen Opteron kone, jonka tulosten suuruusluokka vaihtui jostain tuntemattomasta syystä sen jälkeen, kun kone oli jonkin aikaa poissa käytöstä. Peräkkäin ajetut työt antoivat melko tarkasti samat suoritusajat, mutta tauon jälkeen ohjelmien suoritusajat muuttuivat. Tämä ei kuitenkaan muuttanut suoritusaikojen keskinäisiä suhteita ja uudelleen ajetut kontrolliajot antoivat jälleen vain muutaman prosentin heiluntaa. Vaikka tulosten suuruusluokka vaihteli kummallisesti, tuloksista tehtävät päätelmät ovat edelleen vertailukelpoisia, sillä niissä verrataan ainoastaan peräkkäin tehtyjen ajojen suhteellisia muutoksia.

4 Tulokset

Mittauksissa mitattiin suuren työjoukon työvuo, eli kuinka monta työtä systeemi pystyy prosessoimaan tunnissa, keskimääräinen energiankulutus per työ ja keskimääräinen tehon kulutus. Ajot ajettiin kaikilla kolmella testiympäristöllä ja viidellä testiohjelmalla. Kukin ajo toistettiin suorituspaikkojen määrillä 1-5 työtä suoritinydintä kohden, paitsi tapauksissa, joissa laskentakoneen muisti loppui kesken ja sivutusmuistin käyttöönoton vuoksi suoritus hidastui huomattavasti.

Seuraavassa on koottu kunkin testiohjelman tulokset. Taulukoissa on kuvattu tulokset jokaisella testiympäristöllä sekä vakioasetuksilla, että optimaalisella asetuksella. Vakioasetukset ovat tavallinen tapa konfiguroida jononhallintajärjestelmä, eli jokaista prosessoriydintä kohden laskentakoneelle lähetetään yksi työ. Energiansäästömielessä tämä ei kuitenkaan ollut optimaalinen, vaan on yleensä edullisempaa ajaa useampaa työtä yhtäaikaisesti.

Kaikista mittauksista kävi ilmi hyvin samankaltainen trendi. Yhdellä työllä suoritinydintä kohden suorituskyky oli melko heikko. Lisäämällä suoritinkohtaisia paikkoja työvuo parani, kunnes saavutti huippuarvonsa, joka saatiin koneesta riippuen ajamalla joko 3 tai 5 työtä ydintä kohden. Tämän jälkeen vuo kääntyi jälleen laskuun kunnes käyttömuistin loppuessa romahti täysin.

4.1 Suoritinintensiivinen testiohjelma

Testiympäristö	Asetus	Työt/ydin	Työtä/tunti	Wh/työ	keskim. kW/ydin
Xeon klusteri	Vakio	1	176	3.79	111
	Optimaalinen	1	176	3.79	111
Opteron 2x4	Vakio	1	387	0.65	31
	Optimaalinen	5	467	0.56	32
EeeBox klusteri	Vakio	1	24	1.89	15
	Optimaalinen	2	30	1.52	15

Taulukko 2: Suoritinintensiivisen testiohjelman tulokset

Suoritinintensiivisessä testiohjelmassa eri ajojen tulokset olivat yhden suorituspaikan ajoa lukuunottamatta hyvin samanlaiset. Nostamalla suorituspaikkojen määrän yhdestä kahteen tulokset paranivat muutamia kymmeniä prosentteja, mutta tämän jälkeen suorituspaikkojen lisääminen ei muuttanut suoritusnopeutta tai energiankulutusta.

Eri testiympäristöjen nopeuserot kävivät tästä testistä erittäin selkeästi esiin. 8-ytiminen Opteron oli noin 15 kertaa nopeampi kuin EeeBox klusterin koneet, vaikka klusteriin kuului yhteensä kolme konetta. Ytimien määrään suhteutettuna ero on hieman pienempi, vain 6 kertainen, mutta tästä huolimatta huomattavan suuri. EeeBoxien virrankulutus oli ytimien määrään suhteutettuna kuitenkin vain puolet Opteronin kulutuksesta.

Tasaiset tulokset johtuvat todennäköisesti siitä, että ohjelman suoritus riippuu ainoastaan suorittimesta, ja kullakin prosessilla on tietyn vakioarvon verran prosessoitavia liukulukulaskutoimituksia. Näiden laskutoimitusten vieminen suorittimen läpi kestää vakioajan, eikä käyttöjärjestelmä joudu odottamaan esimerkiksi muistiväylän vapautumista.

4.2 Muisti-intensiivinen testiohjelma

Testiympäristö	Asetus	Työt/ydin	Työtä/tunti	Wh/työ	keskim. kW/ydin
Xeon klusteri	Vakio	1	60	11.88	119
	Optimaalinen	3	91	8.44	128
Opteron 2x4	Vakio	1	76	3.47	33
	Optimaalinen	5	87	3.05	33
EeeBox klusteri	Vakio	1	22	2.05	15
	Optimaalinen	4	32	1.48	15

Taulukko 3: Muisti-intensiivisen testiohjelman tulokset

Muisti-intensiivisissä testiajoissa tunnissa suoritettujen töiden määrä saavutti huippuarvonsa 3-5 ydinkohtaisen suorituspaikan kohdalla, mutta saadut tulokset olivat hyvin erilaisia eri testiympäristöjen välillä. Xeon -testiympäristössä tulokset paranivat suorituspaikkoja lisäämällä ja alkoivat huonontumaan 4 ydinkohtaisen suorituspaikan jälkeen. Tämän jälkeen käyttömuisti loppui ja suorituskyky romahti. EeeBox klusterilla tulokset olivat melko tasaisia, vaikkakin arvolla kolme työtä per ydin saavutettiin hieman paremmat tulokset kuin muilla arvoilla. Opteron -koneen valtavan muistimäärän takia muisti ei loppunut kesken, ja tulokset paranivat tasaisesti 5 ydinkohtaiseen suorituspaikkaan asti, kunnes paikkojen lisääminen ei enää muuttanut tulosta merkittävästi.

Tulosten perusteella voidaan päätellä, että muistin ylikuormittaminen nopeuttaa töiden suorittamista. Testiohjelmassa selkeä pullonkaula on muistin nopeus, minkä vuoksi muistiväylä tulisi hyödyntää niin tehokkaasti kuin vain mahdollista. Tämä tarkoittaa sitä, että tietokoneella on aina oltava tarpeeksi prosessoitavaa tarjolla.

Eri testiympäristöjen keskinäisistä tuloksista huomataan myös toinen mielenkiintoinen seikka. Järjestelmien väliset erot ovat hyvin pienet, toisin kuin esimerkiksi suoritinintensiivisissä testeissä. Vaikka järjestelmät ovat eri ikäisiä ja sisältävät erilaista tekniikkaa, muistin nopeus ei ole kehittynyt läheskään samassa suhteessa kuin muu tekniikka. Tämä tulee ottaa huomioon järjestelmiä suunniteltaessa, sillä kalliita komponenttejä ei kannata ostaa, mikäli niitä ei voida hyödyntää yhdessä muun järjestelmän kanssa.

4.3 I/O -intensiivinen testiohjelma

Testiympäristö	Asetus	Työt/ydin	Työtä/tunti	Wh/työ	keskim. kW/ydin
Xeon klusteri	Vakio	1	119	6.06	120
	Optimaalinen	1	119	6.06	120
Opteron 2x4	Vakio	1	122	2.07	31
	Optimaalinen	5	280	1.08	38
EeeBox klusteri	Vakio	1	38	1.27	16
	Optimaalinen	3	65	0.76	16

Taulukko 4: I/O-intensiivisen testiohjelman tulokset

Kolmen perustestin joukosta I/O-testin tulokset olivat selkeimmät, ja kullekin koneelle muodostui selkeä suorituskykymaksimi joko 3 tai 5 ydinkohtaisen suorituspaikan kohdalle. Xeon klusterin tapauksessa tosin perustilan asetus oli selkeästi muita asetuksia parempi. Suuremmilla ydinkohtaisilla suorituspaikkojen määrillä tulokset kuitenkin muodostivat selkeän huipun kolmen suorituspaikan kohdalle.

I/O-testeissä mitattiin lähinnä kovalevyn I/O liikenteen vaikutusta. Vaikka testiohjelma käytti levyä varsin intensiivisesti, sen nopeus ei riittänyt käyttämään kovalevyn koko nopeuskapasiteettiä. Tilanne on tällöin sama kuin muistitestien yhtey-

dessä, eli levyn ollessa pullonkaula, on pidettävä huoli, että levyllä on jatkuvasti tarpeeksi töitä odottamassa.

Tilanne kuitenkin muuttuu työmäärän kasvaessa, sillä tällöin levylle tulee huomattavat määrät yhtäaikaisia luku- ja kirjoituspyyntöjä. Tällöin levyn lukupää joutuu vaihtamaan sijaintiaan levyllä jatkuvasti, ja suuri osa ajasta kuluu lukupään siirtelyyn. Tämä luonnollisesti hidastaa levyoperaatioita merkittävästi. Myös levyvälimuistin määrä ja rakenne voi vaikuttaa kirjotusnopeuteen. Välimuistin loppuessa voi prosessointi hidastua merkittävästi.

Levytekniikka on ilmeisesti kehittynyt paremmin kuin muistitekniikka, sillä uuden ja vanhan koneen välillä suoritusnopeuksissa oli huomattava ero. Opteronin kovalevy kykeni suoriutumaan 40 yhtäaikaisesta prosessista 5 kertaa nopeammin kuin 3 EeeBox-konetta, vaikka kullakin EeeBox-koneella oli oma kovalevynsä. Nopeuteen vaikuttavat muutkin seikat, mutta toisin kuin muistitestien yhteydessä, kovalevytestien nopeudet vaihtelivat eri konetyyppien välillä suuresti.

4.4 Sekoitetut testiohjelmat

Testiympäristö	Asetus	Työt/ydin	Työtä/tunti	Wh/työ	keskim. kW/ydin
Xeon klusteri	Vakio	1	81	8.33	113
	Optimaalinen	4	95	7.18	114
Opteron 2x4	Vakio	1	165	1.53	31
	Optimaalinen	5	233	1.15	33
EeeBox klusteri	Vakio	1	21	2.18	15
	Optimaalinen	3	43	1.09	15

Taulukko 5: Sekoitettujen testiohjelmien tulokset

Sekoitettujen testiohjelmien tapauksessa tulokset muistuttivat hyvin pitkälle jo saatuja tuloksia, mutta tällä kertaa yhdelläkään ympäristöllä saadut tulokset eivät poikenneet merkittävästi muista ympäristöistä. Jollakin yksittäisellä testillä ilmenevät ongelmat keskiarvoistuvat pois ja niiden vaikutus on pieni. Suoritusnopeus kasvaa tiettyyn pisteeseen asti, kunnes optimipisteen jälkeen kääntyy laskuun.

Syyt tälle ovat pitkälti samat kuin yksittäisissä testeissä, joissa oli havaittavissa samanlaista käyttäytymistä tai vain minimaalista vaikutusta eri asetusten välillä. Aikaisempiin testeihin verrattuna optimaalinen ydinkohtaisten suorituspaikkojen lukumäärä oli yleisesti hieman korkeampi kuin yksittäisillä testeillä. Tämä johtuu siitä, että muisti- ja kovalevyväylät eivät täyty vähillä töillä, sillä osa töistä ei käytä väyliä lainkaan. Täyden käyttöasteen saavuttamiseksi tarvitaan siis useampi samantyyppinen työ.

Testiympäristö	Asetus	Työt/ydin	Työtä/tunti	Wh/työ	keskim. kW/ydin
Xeon klusteri	Vakio	1	127	5.65	125
	Optimaalinen	1	127	5.65	125
Opteron 2x4	Vakio	1	188	1.34	32
	Optimaalinen	3	376	0.77	36
EeeBox klusteri	Vakio	1	33	1.53	16
	Optimaalinen	3	45	1.24	18

Taulukko 6: Fysiikkatöiden testitulokset

4.5 Fysiikkatyö

Fysiikkatyön tarkoituksena oli simuloida todellista tietokoneverkossa tapahtuvaa laskentaa. Työt käyttävät sekä tietokoneen muistia, kovalevyä että runsaasti suoritintehoja. Xeon klusterilla tulokset olivat tavallisuudesta poikkeavia, sillä suoritusnopeus ainoastaan laski kun koneille lähetettiin useita töitä yhtäaikaa laskettavaksi. Testejä ei voitu myöskään tehdä kovin suurilla suorituspaikkamäärillä, sillä koneissa oli muistia varsin vähän ja kun yhdelle ytimelle antoi neljän tai useamman työn laskettavaksi kerralla, muisti loppui ja kone joutui käyttämään sivutusmuistia.

Opteron koneen suoritusnopeus tuplaantui kun suorituspaikkojen määrä nostettiin yhdestä kahteen ja se saavutti optimin aikaisempia testejä aikaisemmin, jo kolmen suorituspaikan kohdalla. Työ vaatii erittäin paljon levyoperaatioita suorituksen alkuvaiheilla, jolloin kovalevyväylän tukkeutuminen laskee optimiarvon alemmaksi.

EeeBox klusterilla oli havaittavissa samanlaisia tuloksia. Vaikka optimiarvo suorituspaikkojen lukumääräksi saatiin 3, arvolla 2 päästiin lähes samoihin lukemiin. Lisäksi EeeBoxien kovalevy on suhteellisesti lievemmässä rasituksessa, minkä vuoksi optimi ei laskenut niin paljon kuin Opteronilla.

4.6 Tyhjäkäyntitehot

Jokaisesta ympäristöstä mitattiin vielä tyhjäkäyntitehot, jotta voitaisiin tutkia kuorman vaikutusta koneiden kokonaistehonkulutukseen. Xeon klusterille mitattiin keskimääräiseksi tyhjäkäyntitehoksi 466W, eli 77W ydintä kohden. Opteron-kone kulutti sähköä 195W, eli 24W ydintä kohden, ja EeeBox klusteri taas 40W, eli 13W per ydin.

Tämä tarkoittaa, että noin 70 % koneen sähkönkulutuksesta on peräisin tyhjäkäynnistä. Energiankulutuksen kannalta tämä tarkoittaa, että mikäli halutaan minimoida töiden käyttämä kokonaisenergia, on tärkein tavoite minimoida koneiden päälläoloaika, eli työjoukko on saatava kokonaisuudessaan läpi niin nopeasti kuin mahdollista. Tulos näkyy myös testiohjelmien tuloksissa: mitä enemmän töitä saadaan tunnissa systeemin läpi, sitä vähemmän energiaa töihin kuluu. Tulokset ovat vastaavia kuin Pinheito et al. saivat omissa tutkimuksissaan [7]

Myös eri konetyyppien välillä on selkeät erot. Vanhan Xeon klusterin sähkönkulutus on muihin testijärjestelmiin verrattuna erittäin korkea. Tärkeätä olisi siis hankkia tarpeeksi uusia koneita, jotka laskevat annetut tehtävät mahdollisimman nopeasti. Tietokoneiden tehonkulutus ei ole vuosien mittaa muuttunut kovin paljoa, mutta sen sijaan laskentateho on moninkertaistunut.

Energiatehokkaat EeeBoxit pärjäävät uudelle Opteron koneelle sähkökulutuksen suhteen varsin hyvin, ja joillain testiohjelmalla pääsevät jopa matalampiin kulutuslukemiin. Tyhjäkäyntikulutus on murto-osa Opteron-koneen kulutuksesta. Erityisesti erot kääntyvät EeeBoxien eduksi, jos tietokoneverkkoa ei käytetä jatkuvasti täydellä teholla.

5 Yhteenveto

Testien perusteella voidaan sanoa, että useamman työn ajaminen yhtäaikaisesti on kannattavaa sekä taloudellisesti että kokonaissuoritusajan lyhentämiseksi. Tällä tavoin tietokoneen resurssit saadaan tehokkaammin käyttöön, kun pullonkaularesurssit, kuten muisti tai I/O-väylä hyödynnetään mahdollisimman tehokkaasti.

Suoritinajan suhteen useampien töiden ajaminen yhtäaikaa samalla suoritinytimellä ei tuo niin suurta etua, sillä suoritettavat laskutoimitukset ajetaan prosessorissa peräkkäin ja kukin vie vakioajan. Sen sijaan muistiväylän tehokas hyödyntäminen vaatii, että se on jatkuvasti käytössä. Liian alhainen käyttöaste ei pidä väylää jatkuvasti varattuna, jolloin pullonkaularesurssi aiheuttaa entistä suurempia haittoja. Samankaltainen ilmiö on havaittavissa myös runsaasti kovalevyä ja siten I/O-väylää kuormittavissa testeissä. Kovalevyn ylikuormittaminen voi kuitenkin johtaa suoritusnopeuden laskuun, sillä mikäli kovalevyltä yritetään lukea liian monesta kohdasta yhtäaikaisesti, suuri osa prosessointiajasta kuluu lukupään siirtymien odotteluun.

Testit osoittavat, että käytännön laskentatöille optimaalinen yhtäaikaa yhdelle suoritinytimelle allokoitavien töiden määrä on noin kolmesta viiteen suuruusluokkaa. Työt käyttävät yleensä kaikkia tietokoneen resursseja, jolloin kokonaisuuden voi ajatella olevan edelläkuvattujen töiden yhdistelmä.

Optimaalinen yhtäaikaisesti ytimellä ajettavien töiden määrä riippuu luonnollisesti laskentakoneesta ja sen kapasiteetistä eri resurssien suhteen. Suorittimen suhteen minkäänlaista rajaa ei näyttäisi olevan, ja muistin suhteen töitä kannattaa ajaa niin monta yhtäaikaisesti, kun laskentakoneen muistiin vain mahtuu ilman että joudutaan käyttämään sivutusmuistia. Kovalevyn suhteen raja ei ole yhtä selkeä, mutta sen vaikutukset ovat sitäkin voimakkaammat. Seuraavaksi olisikin selvitettävä, mikä kovalevyn ominaisuus aiheuttaa pullonkaulan, miten sen käyttöastetta voidaan mitata, ja miten pullonkaula voidaan välttää.

Ongelmalliseksi tilanteen tekee se, että eri laskentakoneilla voi olla hyvin erilaisia resursseja käytössään, ja laskettavien ohjelmien vaatimukset eivät ole yksikäsitteisesti määritettävissä. Mikäli samantyyppistä työtä ajetaan useita kertoja peräkkäin, profilointi ja tämän myötä optimaalisten suorituspaikkojen lukumäärän selvittäminen

voivat tuottaa merkittäviä etuja prosessointiajoissa ja täten myös sähkönkulutuksessa. Jos työt taas ovat keskenään hyvin erilaisia ja tai useaa samantyyppistä työtä ajetaan yhtäaikaisesti, profilointi voi muuttua vaikeaksi tai kannattamattomaksi.

Erilaisten jononhallinta-algoritmien lisäksi, erittäin suuri vaikutus laskennan energiavaatimuksiin näyttää olevan konetyypin valinnalla. Laitteiston tulisi olla uutta ja tehokasta, sillä tietokoneiden sähkönkulutus on pysynyt likimain samalla tasolla vuosien myötä, mutta suorituskyky kehittyy voimakkaasti. Uusien laskentakoneiden hankintaan liittyvät investoinnit ovat helposti matalammat kuin niiden käyttöönotosta koituvat säästöt energiakustannuksissa. Uusien energiatehokkaiden, alunperin lähinnä kannettaviin tietokoneisiin tarkoitettujen prosessoreiden käyttäminen laskentakeskuskäyttöön on myös varteenotettava vaihtoehto, mikäli halutaan nimenomaan säästää energiakustannuksissa.

Ajamalla useita töitä samalla suoritinytimellä voidaan siis saavuttaa parhaimmillaan 2-3 kertaisia etuja suoritusaikoihin. Tämä tarkoittaa vastaavan suuruista säästöä myös energiakustannuksissa. Luonnollisesti yhtä merkittäviin tuloksiin ei päästä kaikissa tapaukissa, mutta lähes poikkeuketta yhtäaikaisesti ajettavien töiden määrän nostaminen yhdestä kahteen paransi työkohtaisia energiamääriä noin 10-20% vakioasetuksiin verrattuna, mikä voi tarkoittaa WLCG:n kokoisessa maailmanlaajuisessa laskentaverkossa valtavia säästöjä [1].

Viitteet

[1] Fan, X., Weber, W.-D. ja Barroso L. A. *Power Provisioning for a Warehouse-sized Computer* Proceedings of the 34th annual international symposium on Computer architecture, San Diego, California, USA p.13-23 (2007)

[2] Niemi, T., Kommeri J. Hameri A.-P. *Energy-efficient Scheduling of Grid Computing Clusters* Presented in the 17th international Adcom conference on Advanced Computing and Communication, 14.-17. Dec. 2009.

[3] CERN *Worldwide LHC Computing Grid Technical Site* `http://lcg.web.cern.ch/LCG`, 19.8.2009

[4] Niemi, T., Kommeri J. Hameri A.-P., Happonen K. ja Klem J. *Improving Energy-Efficiency of Grid Computing Clusters* Advances in Grid and Pervasive computing, 4th intl. conference GPC 2009, Geneva CH proceedings. p.110-118

[5] Edmonds, J. *Scheduling in the dark* Theor. Comput. Sci. vol. 235(1) p.109-141 (2000), Elsevier Science Publishers Ltd., Essex UK

[6] Sun Microsystems *Sun Grid Engine Users Manual* `http://gridengine.sunsource.net/nonav/source/browse/~checkout~/gridengine/doc/htmlman/manuals.html`, 19.8.2009

[7] Pinheiro E. et al. *Dynamic cluster reconfiguration for power and performance* Compilers and operating systems for low power, books.google.com 2001

[8] Ge R., Feng X., Cameron K.W. *Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters* SC 2005: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, Washington DC, USA. p34. IEEE Computing Society, Los Alamitos (2005)

[9] Kappiah N., Freeh V.W., Lowenthal D.K. *Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi program.* SC 2005: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, Washington DC, USA. p33. IEEE Computing Society, Los Alamitos (2005)

[10] Moore J., Chase J. Ranganathan P., Sharma R. *Making scheduling "cool": Temperature-aware workload placement in data centers* USENIX Annual Technical Conference p.61-75, huhtikuu 2005

[11] Heath T., Centeno A. P., George P., Ramos L., Jaluria Y. ja Bianchini R. *Mercury and freon: Termperature emulation and management for server systems* International Conference on Architechtural Support for Proramming Languages and Operating Systems (ASPLOS), Lokakuu 2006

[12] CMS Collaboration, Adolphi, R. et al. *The CMS experiment at the CERN LHC* Journal of Intrumentation 3 (2008)

[13] CMS experiment *CMSSW application framework* `http://twiki.cern.ch/twiki/bin/view/CMS/WorkBookCMSSWFramework` 1.8.2009