jamk.fi

VIKASIETOISEN, KOMPAKTIN JA KUSTANNUSTEHOKKAAN WWW-PALVELINKLUSTERIN SUUNNITTELU

Pyry Lahti

Opinnäytetyö
Huhtikuu 2014

Tietotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) Lahti, Pyry	Julkaisun laji Opinnäytetyö	Päivämäärä 01.05.2014
	Sivumäärä 79	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)

Työn nimi
VIKASIETOISEN, KOMPAKTIN JA KUSTANNUSTEHOKKAAN WWW-PALVELINKLUSTERIN SUUNNITTELU

Koulutusohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Rantonen, Mika; Häkkinen, Antti

Toimeksiantaja(t)
Jyväskylän yliopisto, Koulutuksen tutkimuslaitos, Peda.net-projekti

Tiivistelmä

Opinnäytetyö tehtiin Jyväskylän yliopiston koulutuksen tutkimuslaitoksen Peda.net-projektille. Työn tarkoituksena oli suunnitella, hankkia ja toteuttaa Peda.net-projektin palvelinjärjestelmän tueksi uudet klusteroidut kustannustehokkaat laskentakoneet edellisten tilalle. Tavoitteena oli saada pieneen tilaan mahtuvat koneet, joissa olisi paljon laskentatehoa ja jotka olisivat hinta/laatu-suhteeltaan edullisia.

Opinnäytetyön teoriaosa sisältää perustietoja palvelinratkaisuista, Linux-palvelinkäyttöjärjestelmästä, HTTP:stä, Apache HTTP-palvelimesta, SMP:stä sekä erilaisista klusterointiratkaisuista.

Suorituskyvyn muutosta mitattiin tekemällä testit ensin vanhan klusterin koneella ja tämän jälkeen uuden klusterin koneella. Testit tehtiin ApacheBenchiä ja cURL:ia käyttämällä. Testeissä mitattiin laskentakoneiden suorituskykyä Peda.net-ympäristössä kuormittamalla niitä rinnakkaisilla ja peräkkäisillä sivuhauilla.

Suorituskykymittauksissa selvisi, että uudet klusterikoneet nopeuttaisivat yksittäisiä sivulatauksia noin kolmanneksella ja paransivat rinnakkaisuutta merkittävästi.

Avainsanat (asiasanat)
Peda.net, Klusterointi, Linux, HTTP, SMP

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
Lahti, Pyry	Bachelor´s Thesis	01052014
	Pages 79	Language Finnish
		Permission for web publication (X)

Title
DESIGN OF FAULT-TOLERANT, COMPACT AND COST-EFFECTIVE WEB SERVER CLUSTER

Degree Programme
Information Technology

Tutor(s)
Rantonen, Mika; Häkkinen, Antti

Assigned by
University of Jyväskylä, Institute for Educational Research, peda.net project

Abstract

The thesis was assigned by the University of Jyväskylä, Educational Research, Peda.net project. The purpose of this project was to design, procure and implement new clustered computing machines for the project Peda.net. The goal was to get as packable equipment as possible and get it to be as cost-effective as possible.

The theoretical part of the thesis consists of basic information about different server solutions, Linux server as an operating system, basics of HTTP and Apache HTTP Server, SMP and basics of clustering and a variety of different clustering solutions.

The change in performance was measured by the same tests, first with an older cluster machine followed by a new one. The tests were conducted with ApacheBench and cURL. The tests measured the performance of computing machines in Peda.net environment by exerting them with parallel and sequential load.

The performance measurements showed that the new cluster machines sped up an individual page load by about a third and improved parallelism significantly.

Keywords
Peda.net, Clustering, Linux, HTTP, SMP

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

LYHENTEET

ASF	Apache Software Foundation
CPU	Central Processing Unit
DSO	Dynamic Shared Object
FAT	File Allocation Table
FSF	Free Software Foundation
GB	Gigabyte
GNU	GNU’s Not Unix
GPL	General Public License
GPU	Graphic Processing Unit
GUI	Graphical User Interface
HPCC	High Performance Computing and Communications
HT	Hyper Threading
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LAN	Local Area Network
LTS	Long Term Support
MIMD	Multiple Instruction Multiple Data stream
MISD	Multiple Instruction Single Data stream
MPM	Multi-Process Modules
NFS	Network File System
NASA	National Aeronautics and Space Administration
PC	Personal Computer
PHP	Hypertext Preprocessor

PU	Processing Unit
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
SHA-1	Salausfunktio
SIMD	Single Instruction Multiple Data stream
SISD	Single Instruction Single Data stream
SMP	Symmetric Multiprocessing
SPOF	Single Point of Failure
SSL	Secure Sockets Layer
SSI	Single System Image
TCP	Transmission Control Protocol
TDP	Thermal Design Power
U tai RU	Rack Unit, 1U = 1,75” (4,445 cm)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VFAT	Virtual File Allocation Table

1 LÄHTÖKOHDAT

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Jyväskylän yliopiston koulutuksen tutkimuslaitos. Koulutuksen tutkimuslaitos on Jyväskylän yliopiston erillislaitos, joka on perustettu vuonna 1968. Tutkimuslaitos toimii valtakunnallisena koulutuksen tutkimuskeskuksena, jossa työntekijöitä on noin 80. Koulutuksen tutkimuslaitoksen tutkimuskohteisiin kuuluu koko koulutusjärjestelmä aina perusopetuksesta korkeakoulutukseen mukaan lukien aikuiskoulutuksen, työelämän suhteet sekä verkkoympäristöt. Koulutuksen tutkimuslaitos on erikoistunut suuriin kansainvälisiin vertailututkimuksiin, joista tunnetuimpana on PISA, mutta mukana on myös kansallisia, alueellisia sekä yksittäisten koulujen toimintaa edistäviä tutkimuksia. Tutkimuslaitoksen tavoitteita ovat mm. opettajien, oppilaitosten sekä päätöksentekijöiden tukeminen, jotta koulutusta saadaan kehitettyä parhaalla mahdollisella tavalla. Tutkimuslaitoksen yhteistyökumppaneita ovat mm. ministeriöt, opetushallitus, eri arviointineuvostot sekä paikalliset koulutuksen järjestäjät. (Esittely 2013)

Tämän työn toimeksiantajana toimi koulutuksen tutkimuslaitoksen Peda.net-projekti. Peda.net on yliopiston rekisteröimä tavaramerkki, jonka suojassa kehitetään sähköisiä oppimisen ratkaisuja. Peda.net työllistää tällä hetkellä 6 työntekijää ja asiakkaina on noin 140 kuntaa. Peda.net tarjoaa kouluille ja kunnille verkkopalveluita, koulutusta, tukea, teknistä ylläpitoa sekä varmuuskopiointia. Peda.net oppimisalusta toimii lähes kaikissa ympäristöissä kaikilla laittella, joille on selain saatavilla. Oppimisalustan mukautuvuuden ja helppokäyttöisyyden myötä opiskelijat voivat rakentaa verkkoympäristöstään yksilöllisiä tarpeitaan vastaavan. Oppimisalustan ideana on myös se, että oppilaat ja opettajat voivat verkostoitua muiden koulujen Peda.net käyttäjien kanssa. (Peda.net 2014)

1.2 Tavoitteet

Opinnäytetyön tavoitteena oli suunnitella, hankkia ja toteuttaa Peda.net-projektin palvelinjärjestelmän tueksi uudet klusteroidut ja kustannustehokkaat laskentakoneet entisten tilalle. Tarve uusille laskentakoneille syntyi, kun käyttäjiä ryhdyttiin ohjaamaan uuden Peda.net-palvelun pariin. Palvelun uudet ja monipuolisemmat toiminnot lisäävät verkkoon sekä palvelimiin kohdistuvaa kuormaa. Tämä johti palvelujen yhteiskäyttäjämäärän kasvuun ja palvelun käytön lisääntymiseen, jolloin nykyiset laskentakoneet kävivät hitaiksi. Laskentakoneiden määrää klusterissa tuli myös kasvattaa palvelun nopeuttamiseksi sekä palvelun saatavuuden parantamiseksi. Rautapäivitysten lisäksi laskentakoneiden käyttöjärjestelmien sijaintia sekä klusterointimenetelmiä tulisi miettiä uudelleen sekä tutkia tulevia mahdollisia ongelmakohtia kokonaisuudessa.

2 PALVELIMET

2.1 Yleistä

Palvelinrautaa valitessa ovat vaihtoehdot jaettavissa neljään eri kategoriaan; torni-palvelimiin, rack-palvelimiin, blade-palvelimiin sekä käyttäjille sunnatuista komponenteista kasattuihin palvelimiin, josta käytetään nimeä commodity computing.

2.2 Torni-palvelimet

Torni-palvelimet näyttävät ulkoisesti tavallisilta käyttäjille suunnatuilta tietokoneilta. Niitä käytetään nykyään vain tapauksissa, joissa palvelimia tarvitaan vain muutamia, sillä ne vievät muita palvelinvaihtoehtoja enemmän tilaa sekä niiden sähkönkulutus on suurempi. Muita niiden huonoja puolia ovat muun muassa niiden fyysinen hallinta, joka useimmiten vaatii monitorin, näppäimistön sekä hiiren. Myös torni-palvelimien kaapelointi voi olla haastavaa, sillä niiden sisältäessä useampia I/O-laitteita sekä verkkoadaptereja, jo yhden palvelimen kaapelien määrä voi kasvaa turhan suureksi. Muutaman palvelimen tapauksessa palvelinratkaisun hyvinä puolina on niiden edullisuus verrattuna rack- ja blade-palvelimiin sekä niiden helpompi jäähdytyksen suunnittelu. (Lowe 2011)

2.3 Rack-palvelimet

Rack-palvelimet ovat rack-hyllyihin asennettavia standardin kokoisia palvelimia, yleensä 19” leveitä ja korkeus vaihtelee 1U - 8U (U tai RU) välillä tarpeen mukaan. Tyypillinen täysikokoinen rack-hylly on 42U korkea. Yhdessä rack-hyllyssä on useita asennusaukkoja/telakoita, joihin voidaan pinota päällekkäin useita rack-palvelimia, varavirta-akkuja, kytkimiä tai reitittimiä. Rack-palvelimien hyvinä puolina on muun muassa se, että sijaitessaan rack-hyllyissä ne saadaan pysymään siististi ja kaapelit

voidaan vetää johdonmukaisesti hyllyn reunoja pitkin niille tarkoitetuista paikoista. Tilaa ne vievät huomattavasti vähemmän kuin sama määrä torni-palvelimia. Huonoa on mm. se, että rack-palvelimille tulisi varata käyttöön niille suunniteltu tila/huone, jotta sen ilmanvaihto saadaan pitämään palvelimet ja muut oheislaitteet viileinä. Koska rack-palvelimet ovat niin sanottuja ”custom-tuotteita”, ovat ne kalliimpia kuin mitä yhtä tehokas rauta muuten olisi. (Lowe 2011)

2.4 Blade-palvelimet

Blade-palvelimet ovat niin sanottuja korttipalvelimia, joissa yhden palvelimen kaikki komponentit on laitettu samalle kortille. Blade-palvelimet asennetaan niille tarkoitettuun kehikkoon (yleensä blade-korttien koko on valmistajakohtainen eli mahtuu vain saman valmistajan kehikkoon), käytännössä vain laittamalla kortti sisään. Kehikkoon voidaan asentaa muun muassa useita palvelimia, verkko-moduuleita, tuulettimia, virtalähteitä jne. Moduuleja asentaessa ei tarvita erillisiä virtajohtoja tai verkkokaapeleita, sillä ne ovat kiinteästi blade-kehikossa integroituina. Blade-palvelinjärjestelmän etuina muihin palvelinjärjestelmiin verrattuna on sen pienempi virran kulutus, koska kehikko käyttää sähköä vain sen verran mitä kortit tarvitsevat. Blade-palvelimet vievät myös huomattavasti vähemmän tilaa kuin muut vastaavanlaiset järjestelmät ja kaapelointia on helpotettu huomattavasti, sillä yksittäinen kortti ei siis tarvitse erikseen kaapeleita. Vikatilanteiden sattuessa blade-kehikosta voidaan irrottaa yksittäisiä kortteja muita häiritsemättä ja korvata ne uusilla (hot swap). Blade-palvelimien hankinta ei kuitenkaan ole hinta/laatu-suhteeltaan järkevää, jos palvelimia tarvitaan vain muutamia. (Lowe 2011)

2.5 Commodity computing

Commodity computing on saanut alkunsa siitä, kun yrityksille on tullut tarve saada käyttöönsä tehokkaampia palvelinkokonaisuuksia. Sen sijaan, että oltaisiin hankittu kalliita supertietokoneita on ostettu useampia tavallisia kotitietokoneita, jotka on

klusteroitu suuremman laskentatehon saamiseksi. Klusteriin voitiin myös lisätä yrityksen henkilökunnan omasta käytöstä poistuneita tietokoneita. Commodity computingin yleistyttyä on siihen kehitelty runsaasti erilaisia vapaan lähdekoodin ratkaisuja. Näin saadaan helposti tehtyä laskentatehokas sekä luotettava klusteri ympäristö. Commodity computingin perusperiaatteita käyttävät nykyään useat suuret palveluntarjoajat, kuten Google sekä Facebook. Facebookilla sekä Microsoftilla on myös aiheeseen perehtynyt erillinen OpenCompute-projekti, jonka tarkoituksena on suunnitella sekä kehittää parempia ja edullisempia avoimia palvelinympäristöjen ratkaisuja. (Commodity computing 2014)

3 APACHE

3.1 Yleistä

Apache Software Foundation (ASF) on voittoa tavoittelematon 501(c)(3) yritys, joka tuottaa avoimen lähdekoodin ohjelmistoprojekteja ilmaiskäyttöön. 501(c)(3) yritys on voittoa tavoittelematon ja vapautettu verotuksesta. ASF:n toiminta perustuu vapaaehtoistyöhön. Yrityksen tunnetuimmat projektit ovat Apache Hypertext Transfer Protocol (HTTP) Server, Apache Ant, Apache SpamAssassin sekä Apache Tomcat. (Apache 2014)

3.2 Apache HTTP-palvelin

Apache HTTP-palvelin on tunnettu internetin suosituimpana web-palvelinohjelmistona. Apachen menestyksen perustana on se, että se on avoimen lähdekoodin ohjelmisto, mikä tarkoittaa, että sen lähdekoodi on vapaasti saatavilla ja muokattavissa kaikille käyttäjille. Tämä mahdollistaa sen, että Apachea pystyy kuka tahansa kehittämään, luomaan ja jakamaan siihen uusia ominaisuuksia ja moduuleja. Avoin lähdekoodi nopeuttaa myös Apachessa löytyneiden virheiden korjausta, sillä virheen voi joko korjata itse tai ilmoittaa viasta eteenpäin, jolloin Apachen internetyhteisöstä mitä todennäköisimmin joku sen korjaa ja jakaa korjauksesta tiedot muille käyttäjille. Kuten suurin osa avoimen lähdekoodin ohjelmistoista Apache on suojattu lisenssillä, joka rajoittaa sen jakelua. Apache on lisensoitu sen oman lisenssin alle, joka on huomattavasti sallivampi kuin esimerkiksi General Public License (GPL). Apachen käyttöä ei ole suoraan tuettu ASF:n toimesta, sen laajan internetyhteisön takia, joka hoitaa suurimman osan ongelman ratkaisusta. (Wainwright 2002)

Apache ei ole käyttäjän silmissä sama kuin normaali sovellus, jota voidaan ajaa, esimerkiksi MS Word tai Excel. Sen sijaan se toimii käyttäjältä ”piilossa” tarjoten palveluja sovelluksille, jotka siihen ottavat yhteyttä, kuten internetselain. UNIX-

terminologiassa sovelluksia, jotka tarjoavat palveluita sen sijaan, että suoraan keskustelisivat käyttäjän kanssa, kutsutaan daemoneiksi. (Wainwright 2002)

Apache on suunniteltu toimimaan verkon yli niin, että sovellukset, jotka keskustelevat sen kanssa, voivat sijaita eri tietokoneilla. Näitä sovelluksia kutsutaan yleisesti asiakkaiksi. Yleisimmät asiakkaat Apache-palvelimelle ovat web-selaimet, pois lukien web-robotit, jotka indeksoivat web-sivustoja. Apachen päätehtävä on vastaanottaa asiakkaan pyyntö (request), joka sisältää yksityiskohdat pyynnöstä, ja lähettää takaisin oikeanlainen vastaus tai vastaus (response), miksi pyyntöön ei pystytä vastaamaan halutulla tavalla. Useissa tapauksissa yksinkertaistettuna pyyntö koskee tietokoneen levyllä sijaitsevia HTML- sivuja, tiedostoja tai scriptiä, joka palauttaa vastauksena HTML-sivun jne. Apache käyttää HTTP-protokollaa keskustellakseen asiakkaiden kanssa. HTTP on pyyntö-vastaus-protokolla, joka määrittelee sen, millaisilla pyynnöillä asiakkaat keskustelevat ja miten Apache niihin vastaa. (Wainwright 2002)

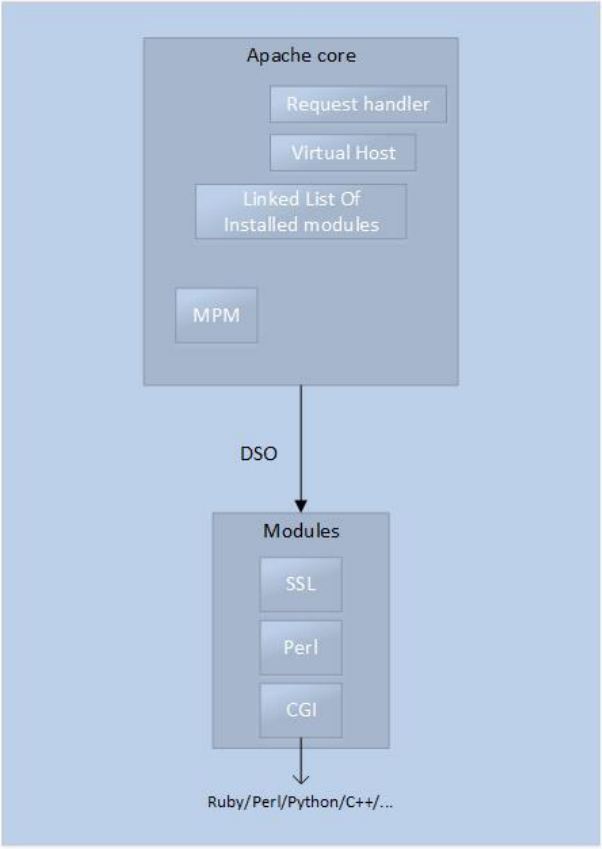
Apachea konfiguroidaan käsin tekemällä muutoksia sen konfuguraatiotiedostoihin, jotka vaikuttavat sen toiminnallisuuksiin. Apache sisältää suuren määrän ohjaustiedostoja, ja jokaisen asennetun moduulin mukana tulee niitä lisää. Kyseinen lähestymistapa tekee Apachen konfiguroimisesta monipuolista ja antaa ylläpitäjälle hyvät mahdollisuudet ylläpitää palvelimen turvallisuutta sekä muita ominaisuuksia. Miinuspuolena voidaan Apachessa pitää sitä, ettei siinä ole virallista graafista käyttöliittymää (GUI). (Wainwright 2002)

Yksi Apachen suurimmista vahvuuksista on sen modulaarinen rakenne. Apache pakettia asentaessa tulee siinä mukana vain ydinjoukko ominaisuuksia ja kaikki muut ominaisuudet ovat erillisinä moduuleina. Moduulit voidaan ladata valmiiksi ja asentaa samalla Apachen yhteydessä tai lisätä Apacheen sen ollessa käynnissä yksi kerrallaan. Modulaarisuuden hyötynä on muun muassa se, että ylimääräisiä moduuleja ei tarvitse erikseen poistaa, jolloin ne eivät jää ns. roikkumaan palvelimelle ja kuluttamaan ylimääräistä prosessoritehoa sekä muistia. Apache sisältää myös moni-prosessimoduulin (Multi-Processing Module, MPM), joka kontrolloi sitä, miten Apache käsittelee asiakkaiden pyyntöjä. Apache sisältää ainakin

kaksi erillistä MPM-moduulia, Prefork ja Worker, joista vain yksi voi olla kerrallaan toiminnassa. Prefork-moduuli ei säikeistä töitään ja näin ollen toimii samankaltaisesti kuin Apache 1.3. Tässä huonona puolena on se, että kun moduulille joudutaan määrittelemään ”*MaxClients*”-arvo, joka asettaa suurimman mahdollisen asiakas määrän, jota voidaan palvella samanaikaisesti. Tämän takia Apache joutuu varaamaan kyseiselle määrälle tarvittavan määrän muistia, vaikkei asiakkaita olisikaan. Jos kyseistä arvoa joudutaan nostamaan liian korkealle, kärsii siitä palvelimen suorituskyky. (Wainwright 2002)

Pientä asiakasmäärää taas pystytään palvelemaan nopeammin kuin esimerkiksi worker-moduulia käytettäessä. Worker-moduuli taas jakaa tulevat työnsä useammalle säikeelle, jolloin se pystyy hoitamaan enemmän töitä nopeammin ja vähemmällä muistin kulutuksella. Kyseisessä moduulissa luodaan tuleville töille oma lapsi-prosessi, joka luo tarvittavan määrän säikeitä, jotka hoitavat työt. Säikeiden etuna on mm. se, että ne jakavat yhteisen muistin, jolloin ne pystyvät vaihtamaan tietoa keskenään. (Wainwright 2002)

Työn valmistuttua Apache jättää aikaisempaan työhön tarvitut säikeet odottamaan uusia töitä, jolloin niitä ei työn tullessa tarvitse uudelleen käynnistää. Kuitenkin hetken kuluttua ne sulkeutuvat, jonka jälkeen Apache ei käytä niihin enää muistia. Apache-palvelin saattaa itse vaihtaa MPM-moduulia sen mukaan, käytetäänkö PHP:ta vai ei. Esimerkiksi worker-moduulin kanssa ei ole suositeltavaa käyttää PHP:tä, sillä se on toistaiseksi epävakaa yhdistelmä. Apache tukee myös kolmannen osapuolen moduuleja. Esimerkiksi kolmannen osapuolen moduuli ”*mod_fastcgi”*, joka nopeuttaa Apachen toimintaa nopeuttamalla asiakkaan ja palvelimen välistä keskustelua sekä vähentämällä palvelimen resurssien käyttöä kyseiseen operaatioon. On myös olemassa paljon muita kolmannen osapuolen moduuleja, joista suosituimpina mainittakoon SSL ja perl. Esimerkki Apachen rakenteesta kuviossa 1. (Wainwright 2002)



Kuvio 1. Apachen rakenne

Kuviossa 1 Apachen ytimessä (core), on käytössä ”request handler”, joka ottaa vastaan tulevat pyynnöt. Virtual Host sisältää Apachen asetustietoja. Ytimessä on myös listaus käytössä olevista moduuleista sekä käytössä oleva MPM-moduuli, eli prefork tai worker. Dynamically Shared Object (DSO), mahdollistaa moduulien käytön jälkeenpäin asennettuina. (Apache 2014)

4 HTTP

4.1 Yleistä

HTTP on taustalla toimiva protokolla, jota kaikki web-palvelimet ja niiden asiakkaat käyttävät. Hypertext Markup Languagen (HTML) määrittäessä miten web-sivut on kuvattu, HTTP huolehtii siitä, miten asiakkaat tekevät tietopyyntöjä palvelimelle ja miten palvelin niihin vastaa. HTTP pyyntö/vastaus on tilaton protokolla (stateless protocol), mikä tarkoittaa sitä, etteivät web-palvelimen ja asiakkaan välisen vuoropuhelun pyyntöjen tulokset ole riippuvaisia edellisten pyyntöjen tuloksista ja kaikki asiakkaat saavat samalla pyynnöllä saman tuloksen. HTTP-pyyntö alkaa siitä, kun asiakas ilmoittaa, mitä haluaa tehdä (esim. *GET)* sekä mihin haluaa kyseisen pyynnön tehdä liittämällä sen komennon *GET* perään muodossa Uniform Resource Locator (URI). Rivin loppuun tulee vielä lisätä käytettävät HTTP-protokolla versio esim. 1.1. (Wainwright 2002)

GET /index.html HTTP/1.1
Host: www.URL.com

Onnistuneen pyynnön seurauksena saadaan takaisin tila-koodi 200. Tyypillinen vastaus Apache palvelimelta näyttää tältä.

HTTP/1.x 200 OK
Date: Tue, 06 Jul 2024 13:09:12 GMT
Server: Apache/2.0.34 (Unix)
Content-Type: text/html; charset=utf-8
Content-Length: 104

Pyynnön vastaukseen sisältyvät tila-koodi, sivun ikä, palvelimen tiedot, sisällön tyyppi sekä pituus. Tässä erityisen tärkeänä pidetään kohtaa *Content-Type*, sillä se kertoo asiakkaalle, mitä vastauksella tehdään sekä *Content-Length*, joka kertoo vastauksen kokonaispituuden. (Wainwright 2002)

Virheen sattuessa vastauksena saadaan

HTTP/1.1 404 Not Found

HTTP metodilla kerrotaan palvelimelle, minkälainen pyyntö sille halutaan tehdä. HTTP metodeita on 8 erilaista.

- GET – Haetaan palvelimen otsikkotiedot sekä resurssit.
- HEAD – Haetaan vain otsikkotiedot.
- POST – Lähetetään tietoa palvelimelle.
- OPTIONS – Kysytään palvelimelta sallitut metodit.
- TRACE – Jäljittää pyynnön, jotta nähdään mitä palvelin näkee.
- DELETE – Poistaa resurssin palvelimelta.
- PUT – Luodaan tai vaihdetaan tiedosto palvelimella.
- CONNECT – Yhteyspyyntö, jotta voidaan hyödyntää esim. SSL:ää.

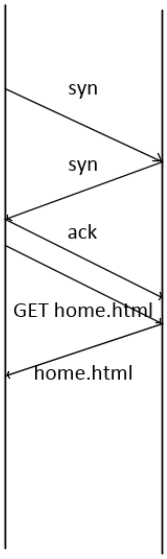
URI on tekstimuotoinen merkkijono, jolla identifioidaan, joko nimen, sijainnin tai muun palvelimen tunnistaman muodon mukaan haluttu resurssi. URI:t, joiden avulla resursseja voidaan internetistä hakea, ovat myös URL:iä (Uniform Resource Locator). URI yksinkertaisimmillaan voi olla /. Esimerkki URI:t. (Wainwright 2002)

/
/index.html
/esimerkki/erkki.htm:80
http://www.URL.com/kuvat/kuva.jpg

HTTP-viestit sisältävät HTTP-otsikkokenttiä, joiden tarkoituksena on lisätä tiedon määrää yksittäisissä viesteissä. Tiedon tarkoituksena on saada sekä palvelin että asiakas ymmärtämään toistensa viestit oikein. HTTP-otsikkotiedot on jaettu kolmeen eri ryhmään – pyyntö-otsikko (request header), vastaus-otsikko (response header) sekä kokonaisuus-otsikko (Entity header). (Wainwright 2002)

Pyyntö-otsikot ovat asiakkaan lähettämiä viestejä palvelimelle, joko lisätäkseen tietoja tai muokkaamaan pyyntöä. Esimerkkinä Accept-Language-otsikko, joka tiedottaa palvelinta asiakkaan käyttämästä kielestä. Vastaus-otsikot ovat palvelimen lähettämiä vastausviestejä asiakkaan pyyntöön. Apachen lähettämiä vakio-otsikoita ovat muun muassa päivämäärä- (*Date*) ja yhteys-otsikot (*Connection*). Kokonaisuusotsikot viestissä lisäävät HTTP-viestiin sen kokonaisuutta kuvailevaa tietoa, niin sanottua ”meta-tietoa”. HTTP-pyynnöt voivat lisätä viestiin meta-tietoja vain käyttäen POST ja PUT menetelmiä. Hyödyllisiä meta-tietoja viesteissä ovat mm. *Expires* sekä *Last-Modified. Expires* kertoo kuinka kauan kyseinen tieto on voimassa olevaa ja *Last-Modified* kertoo selaimelle onko kyseinen avonainen sivu päivitetty vai vanhentunut. (Wainwright 2002)

HTTP-yhteyden muodostamisen sekvenssikaavio on esitetty kuviossa 2. TCP-yhteys muodostetaan siten, että kone lähettää SYN-paketin, palvelin vastaa SYN-ACK-paketilla ja kone vahvistaa yhteyden muodostuksen ACK-paketilla. Muodostetun TCP-yhteyden yli pyydetään HTML-sivua palvelimelta.



Kuvio 2. HTTP-sekvenssikaavio

4.2 SPDY

SPDY on Googlen kehittämä verkkoprotokolla, jonka tarkoituksena on nopeuttaa web-sivujen latautumista. SPDY on HTTP:n kaltainen protokolla, joka toimii HTTP:n rinnalla. SPDY toimii vain, jos asiakas ja palvelin molemmat sen käyttöä tukevat. SPDY ei kuitenkaan korvaa HTTP:tä vaan muokkaa sen toimintaa nopeammaksi. SPDY:n merkittäviä etuja ovat muun muassa se, että siinä sallitaan useampi yhtäaikainen HTTP-pyyntö yhdessä TCP istunnossa. Myös kaistankäyttöä vähennetään pakkaamalla sekä poistamalla turhia ja ylimääräisiä otsikoita. SPDY lisää web-käyttäjien tuvallisuutta, koska sen taustalla toimii SSL-protokolla. SPDY tiedustelee automaattisesti HTTP-protokollan ensiviestin yhteydessä tukeeko molemmat, asiakas sekä palvelin kyseistä protokollaa ja aloittaa sen käyttämisen molempien tukiessa. Nykyään SPDY on käytössä suurista palveluntuottajista mm. Googlella, Facebookissa sekä Youtubessa. SPDY:ä käyttää automaattisesti tällä hetkellä selaimet Chrome sekä Firefox. (SPDY 2014)

5 SYMMETRIC MULTIPROCESSING (SMP)

5.1 Yleistä

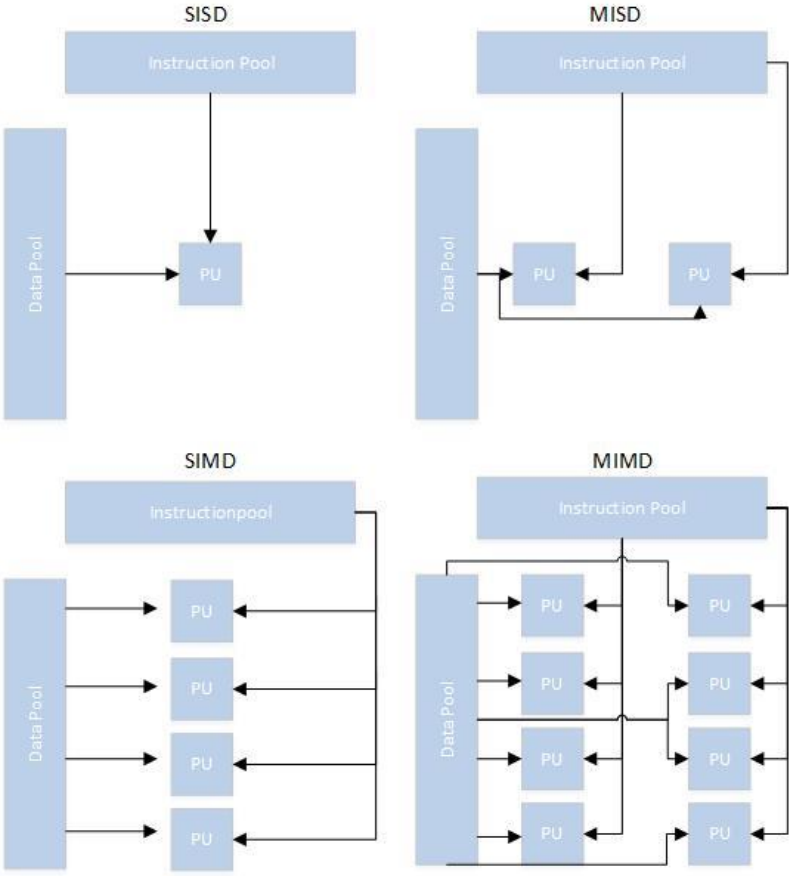
Perinteisesti tietokoneen toiminta on ymmärretty peräkkäisinä operaatioina. On yleisesti ajateltu, että useimmat ohjelmointikielet vaatisivat ohjelmoijan tekevän algoritmit niin, että ne toteutetaan käskysarjoina prosessorin lukiessa annettua käskysarjaa järjestyksessä käsky käskyltä. Tämä käsitys ei kuitenkaan ole koskaan pitänyt varsinaisesti paikkaansa. Kuten mikrotoiminnan tasolla useat ohjaussignaalit generoidaan yhtäaikaisesti, jolloin tapahtuu päällekkäisiä operaatioita, ei siis pelkästään peräkkäisiä. Nykypäivänä komponenttien tehon kasvun ja hinnan laskun seurauksena ovat valmistajat keskittyneet panostamaan enemmän rinnakkaisuuteen parantaakseen kokoonpanojen tehokkuutta ja luotettavuutta. (Stallings 2009)

5.2 Rinnakkaislaskenta-arkkitehtuurit

On hyödyllistä nähdä, mihin rinnakkaislaskentakategoriaan SMP sijoittuu ja miten rinnakkaislaskenta on kehittynyt ajan kuluessa. Rinnakkaislaskennasta ensimmäisen luokittelun on esittänyt (ks. Kuva 3) Michael J. Flynn vuonna 1966. Luokkien määrittely pohjautuu kyseisessä arkkitehtuurissa olevien rinnakkaisten prosessien mahdollisen maksimi määrän mukaan. (Stallings 2009)

Single instruction single data stream (SISD) tarkoittaa sitä, että yksittäinen prosessori hakee ja suorittaa yhden käskyn muistista, minkä jälkeen CPU suorittaa yhden ”data-virran”. Kyseisessä arkkitehtuurissa ei siis ole rinnakkaisuutta. Single instruction multiple data streamissa (SIMD) käytetään useampaa datavirtaa yhtä käskyä kohden, esimerkkinä GPU tai vektoriprosessori. Multiple instruction single data stream (MISD) arkkitehtuurissa suoritetaan useita käskyjä yhdestä datavirrasta. Kyseistä arkkitehtuuria ei käytetä käytännössä missään. Multiple instruction multiple data stream (MIMD) taas tarkoittaa sitä, kun useat itsenäiset prosessorit suorittavat eri

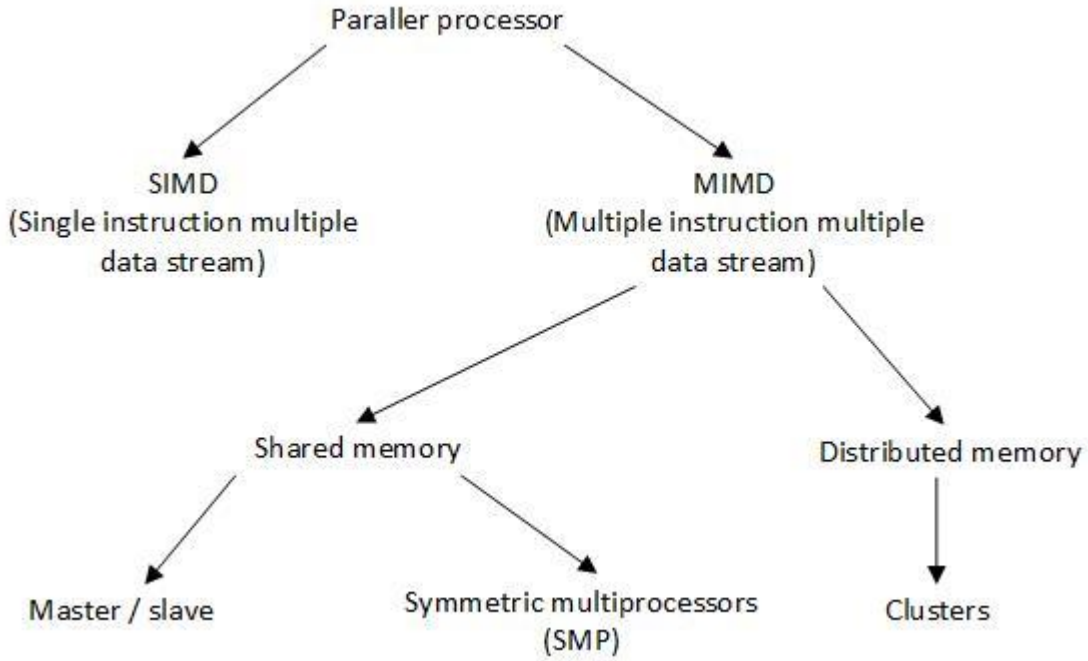
käskyjä eri datavirrasta. MIMD:iä käyttävät yleisesti SMP sekä klusterit. (Stallings 2009)



Kuvio 3. Flynnin taksonomia

MIMD-arkkitehtuurissa pääosassa ovat siis prosessorin ominaisuudet, koska niiden täytyy pystyä hakemaan ja suorittamaan rinnakkain kaikki tarvittavat käskyt, jotta saadaan kaikki data käsiteltyä. MIMD-arkkitehtuuria käyttävät kokoonpanot voidaan vielä jakaa ala-luokkiin sen perusteella miten ne kommunikoivat keskenään, kuten kuviossa 4 on kuvattu. Jos jokaisella prosessorilla on käytössään vain sille osoitettua muistia, on käytännössä jokainen prosessori kuin yksittäinen tietokone. Kommunikointi tietokoneiden välillä voi tapahtua joko niille tarkoitettua linjaa pitkin tai ethernetin välityksellä, jolloin kyseinen kokoonpano olisi nimeltään klusteri. Jos prosessorit jakavat käytettävän muistin keskenään, jolloin jokainen prosessori käyttäisi yhteisessä muistissa olevaa dataa, ne myös pystyisivät kommunikoimaan keskenään kyseisen datan välityksellä, jolloin kyseisen kokoonpanon nimi olisi

”shared-memory multiprocessor”. Shared-memory multiprocessor voidaan jakaa vielä kahteen ala-luokkaan riippuen siitä, jaetaanko työt prosessorien kesken master/slave-periaatteella vai symmetrisesti. Master/slave arkkitehtuurissa käyttöjärjestelmää ajaa aina master-prosessori, kun taas toinen prosessori pystyy ajamaan ohjelmia, jotka master on sille määritellyt. Master-prosessori hoitaa myös prosessien ja säikeiden (thread) aikataulutukset. Jos prosessi tai säie on aktiivinen ja slave-prosessori haluaa sitä palvella, täytyy sen ensin lähettää pyyntö master-prosessorille. Kyseisessä arkkitehtuurissa virheiden määrä prosessorien välillä on olematon, sillä toinen prosessoreista siis päättää kaikista tapahtumista. Haittapuolina voidaan pitää sitä, että jos master-prosessori ”kaatuu”, niin kaatuu samalla koko järjestelmä tai, koska master-prosessorin pitää yksin hoitaa kaikki aikataulutus ja prosessien hoito voi siitä kehkeytyä pullonkaula koko systeemille. (Stallings 2009)

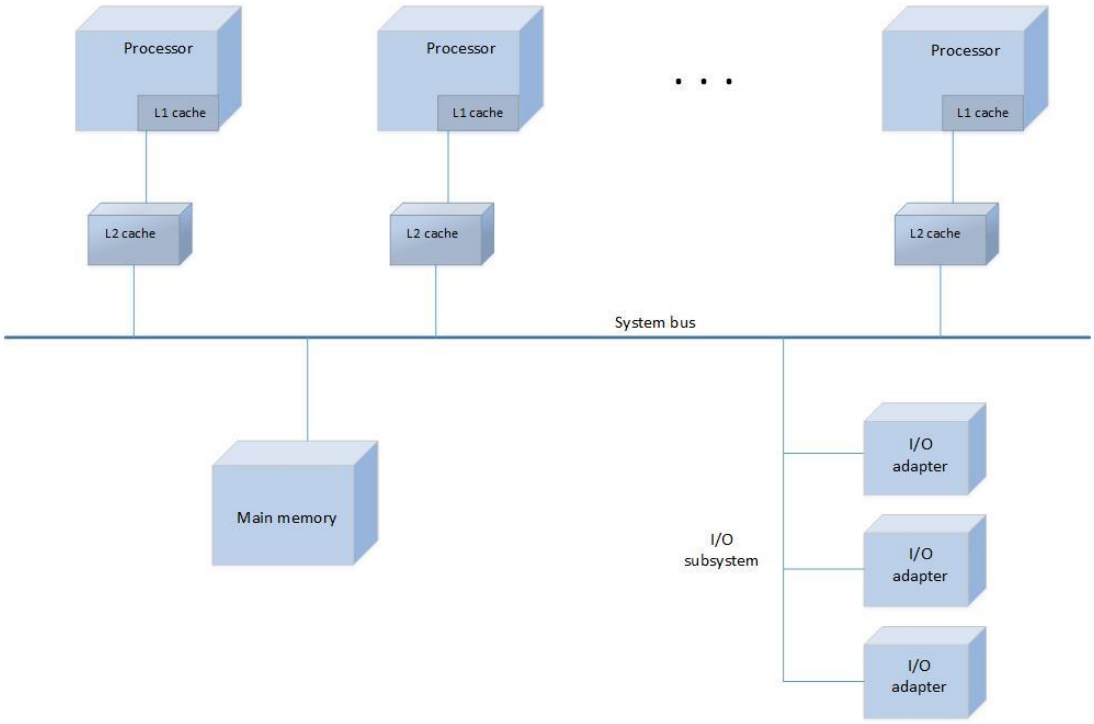


Kuvio 4. Rinnakkaisprosessointiarkkitehtuurit

SMP-arkkitehtuurissa voidaan kerneliä ajaa millä tahansa prosessorilla ja tyypillisesti jokainen prosessori tekee omat ajastuksensa prosessien ja säikeiden ajoon. Yleensä kernel on koostunut useammasta prosessista ja säikeestä, jolloin kernelin osia voidaan käyttää rinnakkain. SMP vaatii kerneliltä hiukan enemmän monimuotoisuutta, jotta sitä pystytään rinnakkaisesti suorittamaan. Esimerkiksi pitää

varmistaa ettei useampi prosessori ota samaa prosessia suorittaakseen tai ettei yksittäinen tärkeä prosessi häviä suoritettavien prosessien jonossa. (Stallings 2009)

Kuvio 5 havainnollistaa SMP:n toimintaa. Kuvasta näkyy, että käytössä on useampia prosessoreja, joista jokaisella on oma ohjausyksikkö, laskentayksikkö sekä rekisterit. Jokaisella prosessorilla on pääsy yhteiseen jaettuun muistiin sekä I/O laitteisiin. Kuviossa näkyvän väylän kautta prosessorit pystyvät keskustelemaan keskenään jaetun muistin kautta, mutta niiden on myös joissakin tapauksissa mahdollista keskustella suoraan toisilleen. (Stallings 2009)



Kuvio 5. SMP-arkkitehtuuri

5.3 SMP-käyttöjärjestelmässä huomioitavaa

SMP:tä käytettäessä käyttöjärjestelmän tulee pystyä toimimaan niin, ettei siinä näy eroa käyttäjälle verrattuna yhden suorittimen kokoonpanoon. Suoritettaessa

sovelluksia käyttöjärjestelmän tulisi reagoida samanlailla, oli vapaana yksi tai useampia prosessoreja. SMP käyttöjärjestelmässä huomioitavia asioita ovat seuraavat. (Stallings 2009)

- Yhtäaikaiset prosessit ja säikeet saattavat aiheuttaa ongelmia kernelissä, mikäli se ei tue useamman prosessin tai säikeen ajoa samanaikaisesti. Kernelin toimintojen tulee olla myös keskeytettävissä ja myöhemmin uudelleen ajettavissa eri prosessorilla kohdasta mihin kyseinen operaatio jäi, tästä käytetään englannin kielessä termiä reentrant. Kun käyttöjärjestelmässä useampi prosessori ajaa yhtä tai useampia sovelluksia, tulee kernelin taulujen sekä hallintorakenteen olla hyvin hoidettu, jotta vältytään virheellisiltä operaatioilta tai umpikujilta (deadlock) . (Stallings 2009)

- Aikataulutuksen tulee pystyä suorittamaan jokainen prosessori. Jos kernel-tasolla käytetään säikeistystekniikkaa, on mahdollista ajastaa useampia säikeitä yhdestä sovelluksesta useammalle prosessorille. (Stallings 2009)

- Muistin hallinnassa käyttöjärjestelmän tulee osata hyödyntää raudan rinnakkaisuutta, kuten moniporttista muistia parhaimman suorituskyvyn saavuttamiseksi. Näennäismuistin käyttö usean prosessorin kokoonpanossa täytyy myös olla koordinoitu johdonmukaisesti parhaan suorituskyvyn saavuttamiseksi. (Stallings 2009)

- Luotettavuuden ja vikasietoisuuden parantamiseksi käyttöjärjestelmän tulee pystyä kaatumaan vain siltä osin, mistä ko. prosessori kaatuu. Eli käyttöjärjestelmän ajastimen ja muiden osien tulee tunnistaa käytöstä poistunut prosessori ja rakentaa hallintataulut uudelleen. (Stallings 2009)

6 KLUSTEROINTI

6.1 Yleistä

Klusterointi on erityisen tärkeä atk-järjestelmien kehitysaste. Klusterointi tarjoaa vaihtoehdon symmetriselle multiprosessoinnille (SMP), kun halutaan suurempaa suoritustehoa ja parempaa saatavuutta palvelinympäristöihin. Klusterointi tarkoittaa yleensä useampaa tietokonetta (node), jotka toimivat yhdessä kuin yksi tietokone, hyötynä joko korkeampi saatavuus (High Availability), kuorman tasaus (Load Balancing) tai suurteholaskentaa varten oleva Beowulf klusterointi. Termillä node tarkoitetaan klusterissa tietokonetta, joka voi siis toimia yksittäisenä yksikkönä tai yhdessä muiden nodejen kanssa. (Stallings 2009)

6.2 Klusteroinnista saatavat hyödyt

Absoluuttisen skaalautumisen ansiosta on mahdollista luoda suuria multiprosessori klustereita, joidenka teho ylittää suurimpien yksittäisten koneiden tehon.
Inkrementaalinen skaalautuvuus, joka tarkoittaa sitä, että klusteri konfiguroidaan siten, että voidaan siihen myöhemmin lisätä nodeja. Klusterin rakentaminen voidaan siis aloittaa pienestä ja tarpeen kasvaessa lisätä siihen osia, jotta vältytään koko järjestelmän uusimiselta ja säästetään sitä myöten laitekustannuksissa.
Korkeaa saatavuutta parantaa se, että klusterin jokainen node pystyy toimimaan erillään muista, eikä muut klusteri ole riippuvainen yhden noden toiminnasta. Tämä tarkoittaa se sitä, että yhden noden kaatuessa palvelun laatu ei juuri heikkene eikä katkoksia synny, jolloin vikasietoisuus paranee.
Klusterissa hinta/teho-suhde on huomattavasti parempi, koska siinä voidaan käyttää ns. normaali käyttöön tarkoitettuja tietokoneen osia. Klusterissa on mahdollista saada aikaan huomattavasti tehokkaampi kokonaisuus verrattuna siihen, jos käytettäisiin yksittäistä supertietokonetta ja halvemmalla. (Stallings 2009)

6.3 Saatavuus

Käytettävyyttä mitataan yleisesti ajassa, jona palvelu saa olla tavoittamattomissa yhteensä yhden kalenterivuoden aikana. Käytettävyysasteen mittauksen ymmärtämisen helpottamiseksi siinä käytetään ”uptime prosenttia vuodessa”, taulukon 1. mukaisesti.

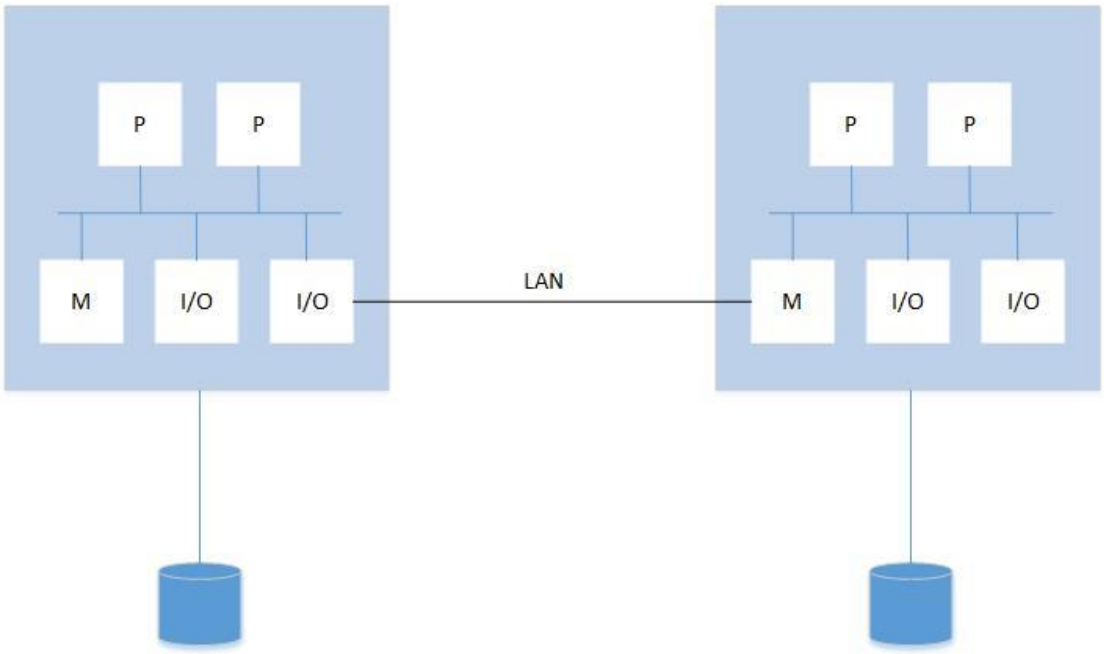
Taulukko 1. Saatavuuden tasot

Saatavuuden tasot			
Saatavuuden taso	Uptime prosentti	Downtime per vuosi	Downtime per päivä
Yhden 9	90,0000 %	36,5 päivää	2,4 tuntia
Kahden 9	99,0000 %	3,56 päivää	14 minuuttia
Kolmen 9	99,9000 %	8,76 tuntia	86 sekuntia
Neljän 9	99,9900 %	52,6 minuuttia	8,6 sekuntia
Viiden 9	99,9990 %	5,25 minuuttia	0.86 sekuntia
Kuuden 9	99,9999 %	31,5 sekuntia	8,6 millisekuntia

Tietotekniikan näkökannalta parhaan saatavuuden takaamiseksi tulee huomioon ottaa sekä verkko että laitteistot. Palvelinkeskuksissa on yleensä määritelty erikseen parhaat käytännöt, joita tulee noudattaa. Käytännöt on määritelty erikseen tiedon varastoinnille sekä tietoverkoille. Tietojen varastoinnissa tulee yleisesti ottaa huomioon kovalevyjen kahdentaminen (RAID), varmuuskopiointi sekä useampi yhteys tallennuskapasiteettien luo. Laitteiden ja verkotuksen kannalta tulisi suunnittelu tehdä niin, että yksittäisen palvelimen, verkkolaitteen tai verkkojohdon rikkoutuessa tai mennessä epäkuntoon, yhteydet palveluihin ja tietoihin säilyvät. Näin vältetään ns. ”Single Point of Failure” (SPOF). (Oracle 2010)

6.4 Klusterointikonfiguraatiot

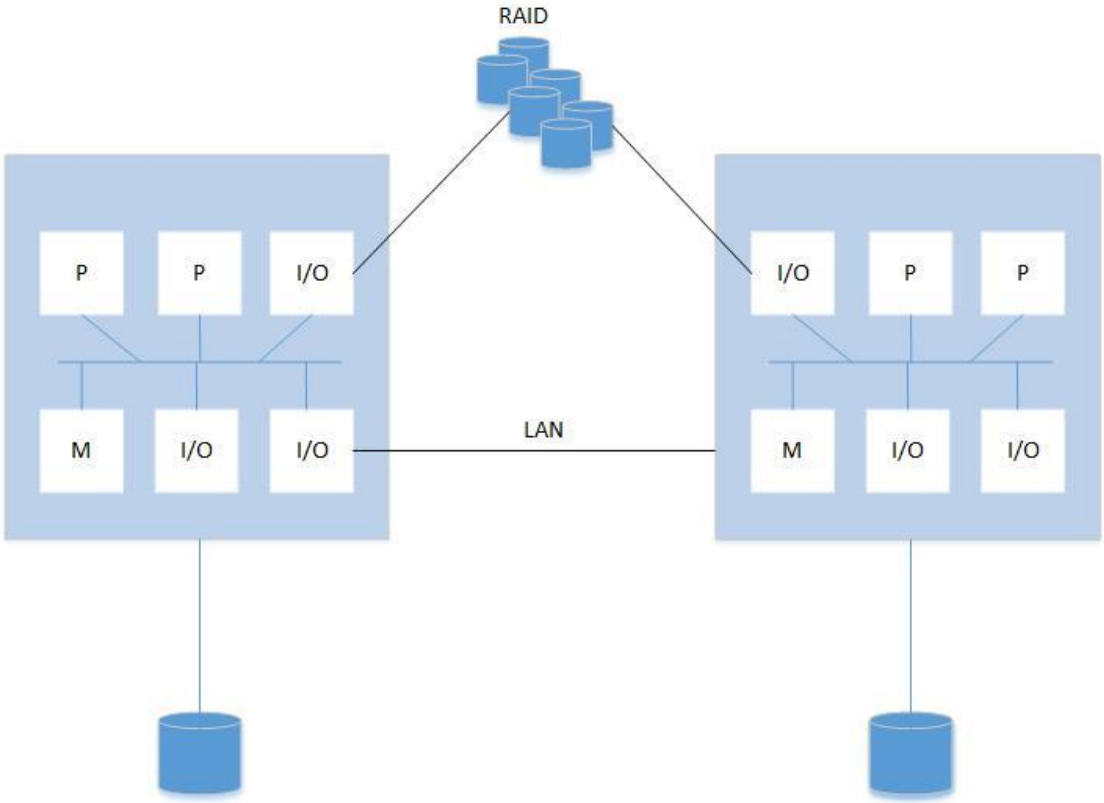
Klusterit voidaan määritellä useisiin eri kategorioihin ja useilla eri tavoilla. Näistä yleisin ja yksinkertaisin määrittely on klusterin sisäisen levyjaon perusteella. Kuviossa 6 voidaan nähdä kahden noden muodostama klusteri, molemmat klusterit sisältävät omat levytilansa. Nodet ovat yhteydessä toisiinsa linkillä, jonka kautta ne koordinoivat klusterin toimintaa. Linkki voi olla Local Area Network-yhteys (LAN), joka on jaettu myös muiden koneiden kanssa, jotka eivät kuulu klusteriin tai linkki voi olla pelkästään nodejen väliseen yhteydenpitoon tarkoitettu. (Stallings 2009)



Kuvio 6. Standy-server, ei jaettua levyä

Kuviossa 7 on klusteri, joka sisältää jaetun levyn. Kyseisessä tapauksessa nodejen välillä on edelleen linkki, joka koordinoi nodejen muuta toimintaa. Levyjaoille on oma esim. Redundant Array of Independent Disks –linkki (RAID), kuten kuviossa 7, joka on suoraan yhdistetty tietokoneisiin klusterin sisällä. RAID:in tai muun samantapaisen levyjaon käyttö klustereissa on yleistä, jos klusterin tarkoituksena on parantaa palvelun saatavuutta usean noden klusterissa. Tällä vältetään järjestelmässä SPOF.

Esimerkiksi dataa ei jää saavuttamattomiin yksittäisen noden kiintolevylle sen kaatuessa. (Stallings 2009)



Kuvio 7. Standby-server jaetulla levyllä

6.5 Klusterointimenetelmät

Klusterointimenetelmää valittaessa tulisi tietää, minkälaiseen käyttöön kyseinen klusteri tulee, jotta voidaan valita sille oikea klusterointimalli sekä menetelmä. Yleinen vanhempi menetelmä on ”passive standby”, jossa yksinkertaisesti ensisijaiseksi määritelty node hoitaa kaiken kuorman toisen noden ollessa valmiustilassa tekemättä mitään, odottaen ensisijaisen noden menemistä toimintakyvyttömäksi ja ottaen sitten sen tehtävät. Siksi ensisijainen node lähettää ns. ”heartbeat”-viestejä säännöllisesti toissijaiselle nodelle, jotta se tietää ensisijaisen noden olevan toimintakunnossa. Jos viesti ei lähde tai ei jostain syystä pääse perille, olettaa toissijainen node, että ensisijainen node on kaatunut ja alkaa se hoitaa tämän tehtäviä. Kyseinen menetelmä siis parantaa saatavuutta, muttei suorituskykyä. Jos

”heartbeat”-viesti on ainut data, joka kulkee kahden noden välillä ja joilla ei ole yhteistä jaettua levyä, on nykystandardien mukaan häilyvää voidaanko kyseistä menetelmää määritellä klusteriksi. Klusterointi termillä tulisi tarkoittaa kahta tai useampaa nodea, jotka pystyvät yhdessä prosessoimaan niille annettuja tehtäviä ja näyttämään ulospäin siltä, että prosesseja suorittaisi vain yksi tietokone. (Stallings 2009)

Passive-standby-konfiguraatiotia hieman muokkaamalla saadaan ”active secondary”-menetelmä käyttöön. Menetelmällä tarkoitetaan esimerkiksi samanlaista kokoonpanoa, kuin edellisessä ”passive standby”-menetelemässä. Kuitenkin myös toissijainen palvelin auttaa jatkuvasti datan prosessoimisessa, joka parantaa koko klusterin suorituskykyä sekä saatavuutta. Kyseisessä menetelmässä tulee olla nodejen välillä käytössä jaettu levy. (Stallings 2009)

Seuraava lähestymistapa klusterin luomiseen on nimeltään ”separate server”, jossa jokaisella nodella tulee olla oma levytilansa, johon ensisijainen node kopioi omat tietonsa, jolloin jokainen node pysyy ajan tasalla. Menetelmä vaatii runsaasti kaistaa nodejen välillä, koska tietoa liikkuu paljon, jotta saadaan jokainen node pysymään ajan tasalla. Tässä merkittävänä hyötynä on esimerkiksi se, että jos käyttäjä on suorittamassa ohjelmaa ja sitä hoitava node kaatuu, voi seuraava node ottaa ohjelman suoritettavakseen ja jatkaa siitä mihin edellinen jäi, eikä käyttäjälle tule siitä katkosta. Menetelmän vaarana voi kuitenkin olla verkosta aiheutuva pullonkaula (Rantonen 2014). (Stallings 2009)

Kun halutaan vähentää viivettä datansiirrossa levyjen välillä, käytetään klusterointimenetelmää nimeltä ”servers connected to disks”. Kyseisessä klusterissa useimmat nodet ovat yhteydessä jaettuun levyyn, josta jokaiselle nodelle on lohkaistu alue, jonka se ”omistaa”. Noden kaatuessa voi seuraava node ottaa kaatuneen noden levyosion haltuun ja jatkaa sen töitä. (Stallings 2009)

On myös mahdollista, että useammat nodet käyttävät yhtä suurta jaettua levyä yhdessä, jolloin käytetään menetelmää ”Server share discs”, jolloin jokaisella nodella on yhtäläinen käyttöoikeus levyyn. Toimiakseen kunnolla kyseinen menetelmä

tarvitsee kuitenkin vierelleen ohjelmiston joka valvoo, ettei useampi node käytä samaan aikaan samaa dataa, jolloin vältytään ylimääräisiltä virheiltä. (Stallings 2009)

6.6 Klusterointimallit

Varsinaiset klusteroinnissa käytettävät mallit ovat kahden yksikön malli sekä useamman yksikön järjestelmä. Näissä käytettävissä malleissa sovellettavat järjestelmät ovat vektoriprosessointi, rinnakkaisprosessointi, load balance sekä Beowulf-klusterointi.

Vektoriprosessoinnissa tehtävät jaetaan useampaan osaan ja suoritetaan jonona. Vektoriprosessoinnissa siis yksittäinen tehtävä kulkee jonossa yksiköltä toiselle ja jokainen yksikkö hoitaa eri tehtävää kyseisestä prosessista. (Stallings 2009)

Rinnakkaisprosessointia voidaan käyttää kahden yksikön mallina tai kuorman jakona tärkeälle palvelulle. Kahden yksikön mallia voidaan käyttää suojaamaan tärkeää palvelua, jolloin toisen yksikön kaatuessa toinen ottaa sen palvelut ylläpidettäväkseen. Kuorman jaossa palvelulle tuleva kuorma jaetaan kahden yksikön kesken. (Stallings 2009)

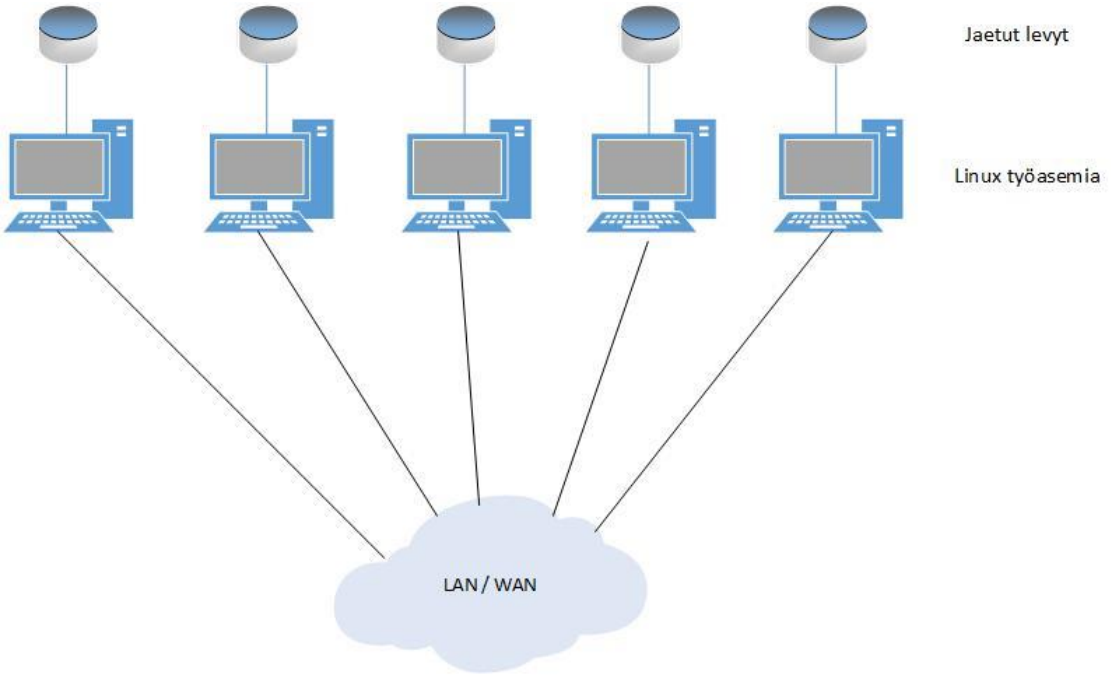
Kuormantasaus klusterointi (Load balancing) toimii siten, että kun palvelin klusterille tulee palvelupyyntö, palvelupyyntö kulkee niin sanotun edustapalvelimen kautta, joka jakaa tasaisesti kaikki tulevat pyynnöt klusterissa olevien palvelimien kesken. Yksittäisen palvelimen kaatuessa tai saadessa liikaa kuormaa, ei sille enää ohjata lisää tehtäviä. Tämä siksi, ettei yhdelle palvelimelle ohjata enempää tehtäviä kuin se pystyisi hoitamaan. Load balance klusterien tulisi yleisesti olla inkrementaalisesti skaalautuvia, jotta niistä saataisiin aina maksimaalinen hyöty. (Stallings 2009)

6.7 Beowulf & linux -klusteri

Beowulf-projekti, toiselta nimeltään High Performance Computing and Communications (HPCC) käynnistettiin vuonna 1994 National Aeronautics and Space Administra-

tionin (NASA) toimesta. Projektin tavoitteena oli tutkia ja kehittää normaaleista tietokoneista tehtävää klusteria, joka pystyisi laskemaan niin massiivisia laskutoimituksia, että silloisen tekniikan yksittäinen tietokone ei siitä suoriutuisi. Projektin yhtenä tavoitteena oli myös pitää klusterin hinta mahdollisimman alhaisena. Nykyään Beowulf-klusterointi on erittäin laajalti käytetty sekä yksi tärkeimmistä klusterointimenetelmistä. (Stallings 2009)

Beowulf-klusteroinnin (ks. Kuvio 8) ideana on yhdistää kaksi tai useampi tietokone-yksikköjä toimimaan yhdessä koordinoidusti, jolloin ne voivat yhdessä laskea suurempia tehtäviä, kuin mitä yksittäinen tietokone pystyisi. (Stallings 2009)



Kuvio 8. Beowulf Linux -klusteri

Vaikka Beowulf-klusterin käytännössä saa asennettua useille alustoille, on se yleisimmin käytetty Linuxin kanssa. Klusteri koostuu useammasta Linux-työasemasta, jotka voivat sisältää eriäviä komponentteja toisiinsa nähden. Jokainen työasema sisältää erillisen jaetun levytilan, jolla voidaan esimerkiksi jakaa tiedostoja tai virtuaalista muistia klusterissa olevien nodejen kesken. Linux-työasemat (nodet) ovat yhteydessä toisiinsa yleisimmin ethernetin välityksellä, joko kaikki yhden kytkimen takana tai reititettynä useampien kytkimien/reitittimien välillä. Käytetyt tiedonsiirtonopeudet ovat 10 Mbps, 100 Mbps sekä 1 Gbps. (Stallings 2009)

Beowulf-ohjelmisto asennetaan liitännäisenä Linux-pohjaiseen käyttöjärjestelmä ympäristöön. Sekä Beowulf että Linux-käyttöjärjestelmä ovat avoimen lähdekoodin ilmaisia ja rojalti vapaita tuotteita. Klusterin jokainen node ajaa omaa kopiotaan Linux-käyttöjärjestelmästä. Jokainen käyttöjärjestelmä sisältää tarvittavat Beowulf ohjelmistot ja voi toimia tarvittaessa täysin itsenäisesti. (Stallings 2009)

6.8 Klusterikoneiden arkkitehtuuri

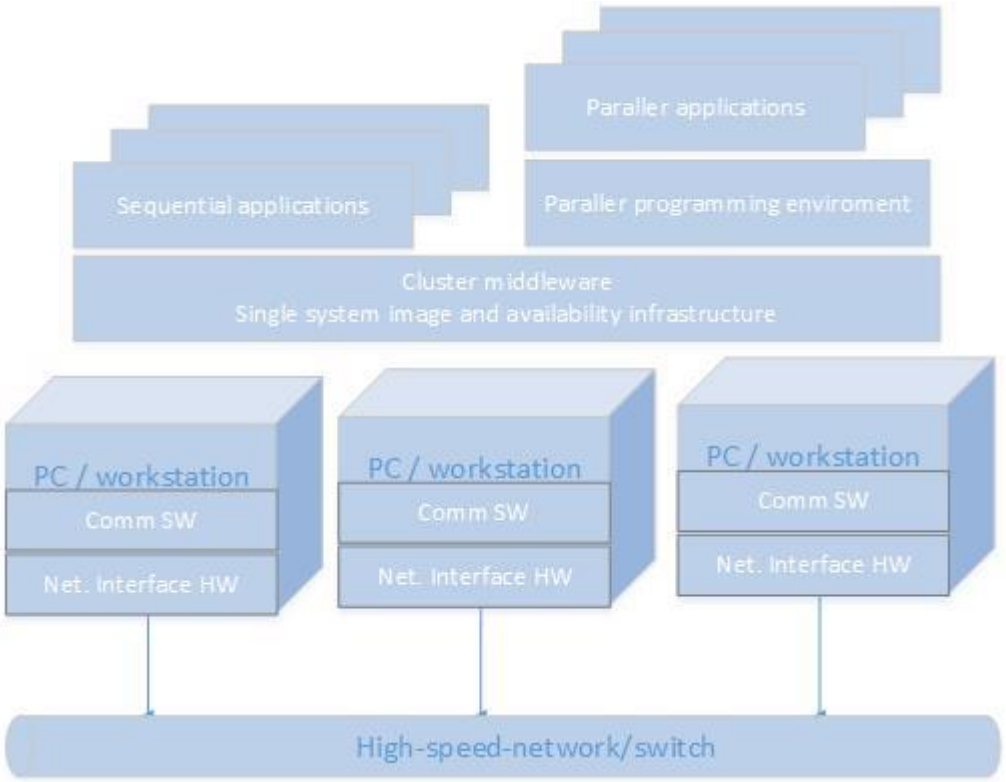
Kuviossa 9 nähdään tyypillinen malli klusterissa olevien koneiden arkkitehtuurista. Yksittäiset nodet ovat yhteydessä toisiinsa ethernetin välityksellä ja jokainen node pystyy siis toimimaan itsenäisenä tietokoneena. Lisäksi jokaiseen nodeen on asennettu sovellustasolle ”cluster middleware”-ohjelmisto, jonka ansiosta nodet pystyvät toimimaan niin, että ulospäin klusterin toiminta näyttää siltä, että se toimisi kuin yksittäinen tietokone, josta käytetään nimeä single-system image (SSI). Middlewaren vastuulla on klusterissa myös High Availability eli korkea saatavuus sekä yksittäisissä nodeissa sattuneiden virheiden kontrollointi. Jotta klusterista voidaan käyttää nimitystä SSI klusteri, täytyy sen täyttää seuraavat ehdot:

- Single entry point: Käyttäjä kirjautuu klusteriin, ei yksittäiseen tietokoneeseen.
- Single User Interface: Käyttäjä käyttää yhtä samaa käyttöliittymää, riippumatta miltä koneelta klusteriin kirjaudutaan.
- Single control point: Kun klusterin asetuksia halutaan muuttaa, ne tehdään oletusasetuksena olevan noden kautta.
- Single job management: Yksi node hoitaa työnjaon klusterissa.
- Single file hierarchy: Klusterissa käytetään vain yhtä tiedostojärjestelmää.
- Single virtual networking: Käyttäjä näkee itsensä vai yhdessä verkossa, vaikka saattaa olla yhteydessä useisiin.
- Single memory space: Ohjelmat näkevät vain yhden muistiavaruuden.
- Single I/O space: Node voi käyttää minkä tahansa toisen noden liitäntöjä riippumatta sen fyysisestä paikasta. *

- Single process space: Kaikki prosessit suoritetaan samoilla prosessi tunnisteilla. *
- Checkpointing: Jos suorituksessa ollut prosessi kaatuu, voidaan sitä lähteä suorittamaan edellisen checkpointin kohdalta. *
- Process migration: Kuorman tasaus vaatii tämän. *

Tähdellä merkityt eivät ole pakollisia, mutta ne on koettu tärkeiksi.

Klusterissa voi olla SSI-ominaisuuksia kolmella eri tasolla: rautatasolla, käyttöjärjestelmätasolla sekä sovellustasolla. Rautatason SSI:n tarkoituksena on saada klusteriin kirjautuneet käyttäjät näkemään klusteri yhtenä koneena. Käyttöjärjestelmätason SSI:n ominaisuudet luovat SSI imagen. Sovellustasolla SSI:n ominaisuudet luovat sinne nipun ohjelmistoja joista käytetään nimeä ”cluster middleware” (Klusterointiteknologiat 2014).



Kuvio 9. Klusterikoneen arkkitehtuuri

6.9 Klusterit vs SMP

Klusteri sekä symmetrinen multiprosessointi (SMP) tarjoavat konfiguraation, jossa paljon prosessitehoa vaativille sovelluksille annetaan käyttöön useamman prosessorin laskentatehot. Molempia ratkaisuja on kaupallisesti saatavilla. SMP:n vahvuuksina verrattuna klusteriin on sen helpompi hallinta sekä konfigurointi. SMP on huomattavasti lähempänä yhden prosessorin mallia, jolle useat sovellukset ovat alkuperäisesti konfiguroitu. Muutoksena yhden prosessorin mallista SMP:hen on scheduler-funktion käyttö. Toinen huomattava etu SMP:llä on useimmiten sen pienempi fyysisen tilan tarve sekä pienempi virran kulutus kuin vastaavanlaisen klusterin olisi. SMP on toistaiseksi myös merkittävästi vakaampi järjestelmä, kuin mitä klusteri tällä hetkellä on. Kuitenkin pitkällä aikavälillä klusterin edut, kuten inkrementaalinen sekä absoluuttinen skaalautuvuus, teho/hinta-suhde ja korkean saatavuuden mahdollisuudet tekevät klusterista huomattavasti varteenotettavamman järjestelmän kuin mihin SMP pystyisi. Usein käytetty

vaihtoehto saadaan käyttämällä molempia ratkaisuja samassa konfiguraatiossa. Tällöin ne eivät sulje toisiaan pois. (Stallings 2009)

7 LINUX

7.1 Yleistä

Linux on GPLv2-lisenssin alla oleva ilmainen avoimen lähdekoodin käyttöjärjestelmäydin. Linuxin on kehittänyt vuonna 1991 suomenruotsalainen Linus Torvalds Unix-järjestelmän pohjalta. Linux-ydin eli kernel on käyttöjärjestelmän rakenteen, luokituksen sekä ominaisuuksien määrittelevä osa, joka mahdollistaa muiden ohjelmistojen toimivuuden. Yleisesti Linuxilla kuitenkin tarkoitetaan Linux-distribuutiota, jakelupakettia eli kokonaista käyttöjärjestelmää, jossa ytimen päälle on kasattu mm. apuohjelmat, käyttöliittymät sekä sovellukset valmiiksi paketiksi loppukäyttäjää varten (Linux 2014).

GNU's Not Unix (GNU) on vuonna 1984 aloitettu projekti, jonka tavoitteena oli kehittää ilmainen käyttöjärjestelmä. GNU/Linux on yleisesti käytössä oleva käyttöjärjestelmä, jossa on Linuxin ydin sekä GNU:n ohjelmistot. GPL on General Public License, joka on GNU-projektin yleinen lisenssi. Lisenssi on tarkoitettu vapaiden ohjelmistojen julkaisemiseen, mikä antaa kaikille oikeudet käyttää, kopioida, muuttaa ja jakaa kyseisiä ohjelmistoja sekä lähdekoodia (GNU 2014).

Koska Linux on ilmainen ja lähdekoodi on kaikkien käytettävissä, tuli siitä jo aikaisessa vaiheessa varteenotettava vaihtoehto jo käytössä oleville UNIX-työasemille, jotka tuolloin olivat IBM:n tai Sun Microsystems:in tarjoamia. Myös Linuxin vahvuutena on ollut mahdollisuus ladata sille Free Software Foundationin (FSF) alaisuudessa olevia sovelluksia. FSF:n tavoitteena on luoda ilmaisia sovelluksia, jotka ovat laadukkaita, vakaita sekä alustastaan riippumattomia. FSF:n GNU-projekti taas tarjoaa työkaluja ohjelmistokehittäjille ja GNU:n yleinen lisenssi GPL on myös FSF:n hyväksymä. (Stallings 2009)

Myös Torvalds käytti GNU-projektin työkaluja aloittaessaan Linux-kernelin kehittämisen ja sittemmin julkaisi sen GPL-lisenssin alaisena. Nykyään Linuxia kehittää joukko ohjelmoijia ympäri maailmaa ja sen myötä Linuxin huima kehitys on

saanut sen osaksi yritysmaailmaa. Ei pelkästään sen takia, että se on ilmainen vaan sen kernelin korkean laadun, modulaarisuuden sekä konfiguroimisen helppouden myötä. Koska lähdekoodi on avoin, on ohjelmistovalmistajien helppo optimoida tuotteitaan Linux-alustalle. (Stallings 2009)

7.2 Modulaarinen rakenne

Linux-kernel on UNIX kernelien tapaan ”yhtenäinen” (monolithic). Yhtenäinen kernel tarkoittaa yksinkertaisesti sitä, että se sisältää virtualisoituna kaikki käyttöjärjestelmän toiminnallisuudet yhdessä isossa koodi-palikassa, joka toimii kuin yksittäinen prosessi yhdessä osoiteavaruudessa. Kaikilla kernelin toiminnallisilla komponenteilla täytyy olla pääsy kaikkiin sen datarakenteisiin. Jos kernelissä tapahtuu muutoksia joihinkin sen moduuleihin, täytyisi tavanomaisen ”yhtenäisen” käyttöjärjestelmän tapaan sen uudelleen käynnistää itsensä, jotta muutos saataisiin voimaan. Esimerkkinä tästä asennettaessa uusi ajuri tai lisättäessä uusi tiedostojärjestelmä toiminto. Kyseistä toiminnallisuutta Linuxin osalta voitaisiin pitää akuuttina, sillä Linuxia kehitetään maailmanlaajuisesti yksittäisten ohjelmoijien toimesta heikon organisoinnin alaisena. Tämän vuoksi Linux-kernel ei käytä suoraan mikrokernel-tyyppistä lähestymistapaa, vaan se pyrkii ottamaan monista eri kernel-tyypeistä parhaita ominaisuuksia. Linux käyttäytyy tässä suhteessa kuin modulaarinen kernel, joka on kuin kokoelma moduuleja, joita voidaan lisätä ja vähentää tarpeen mukaan. Käytännössä siis moduulissa on dataa ja yhteys kerneliin. Moduulit voidaan yhdistää tai irroittaa kernelistä näin haluttaessa, jolloin koko kerneliä ei tarvitse uudelleen käynnistää. (Stallings 2009)

Tyypillisesti yksi moduuli pystyy toteuttamaan toimintoja kuten tiedostojärjestelmän, laiteajurit tai muita kernelin ylemmän kerroksen toimintoja. Moduuli ei kuitenkaan suorita prosesseja tai säikeitä, vaan se luo kernel säikeitä, jotka suorittavat tarvittavat prosessit. Eli vaikka Linux ajatellaan yhtenäiseksi kerneliksi, sen modulaarinen rakenne auttaa sitä pääsemään yli yhtenäisen kernelin heikkouksista. Seuraavassa moduulien tärkeät ominaispiirteet.

Dynamic linking tarkoittaa sitä, että moduuli voidaan ladata ja linkittää kerneliin sen ollessa jo käynnissä/toiminnassa. Moduuli voidaan myös poistaa halutessa milloin tahansa. Dynaaminen linkitys helpottaa konfigurointia sekä säästää kernelin muistia. Linuxissa käyttäjä tai sovellus voi itse lisätä tai poistaa moduuleja käyttämällä insmod- ja rmmod-komentoja. Kun taas kernel itsessään vahtii moduulien sen hetkistä tarpeellisuutta olla yhdistettynä kerneliin ja liittää/poistaa yhdeyden tarpeen mukaan. (Stallings 2009)

Stackable modules tarkoittaa sitä, että moduulit on järjestetty hierarkisesti. Yksittäiset moduulit toimivat ”kirjastoina” niiden ollessa viitattuina korkeammalta hierarkiasta ”client-moduulien” toimesta ja niin edelleen. Kyseisessä mallissa yhteyksiä moduulien väleillä voidaan määritellä erikseen, joista saatavat merkittävät hyödyt ovat muun muassa:

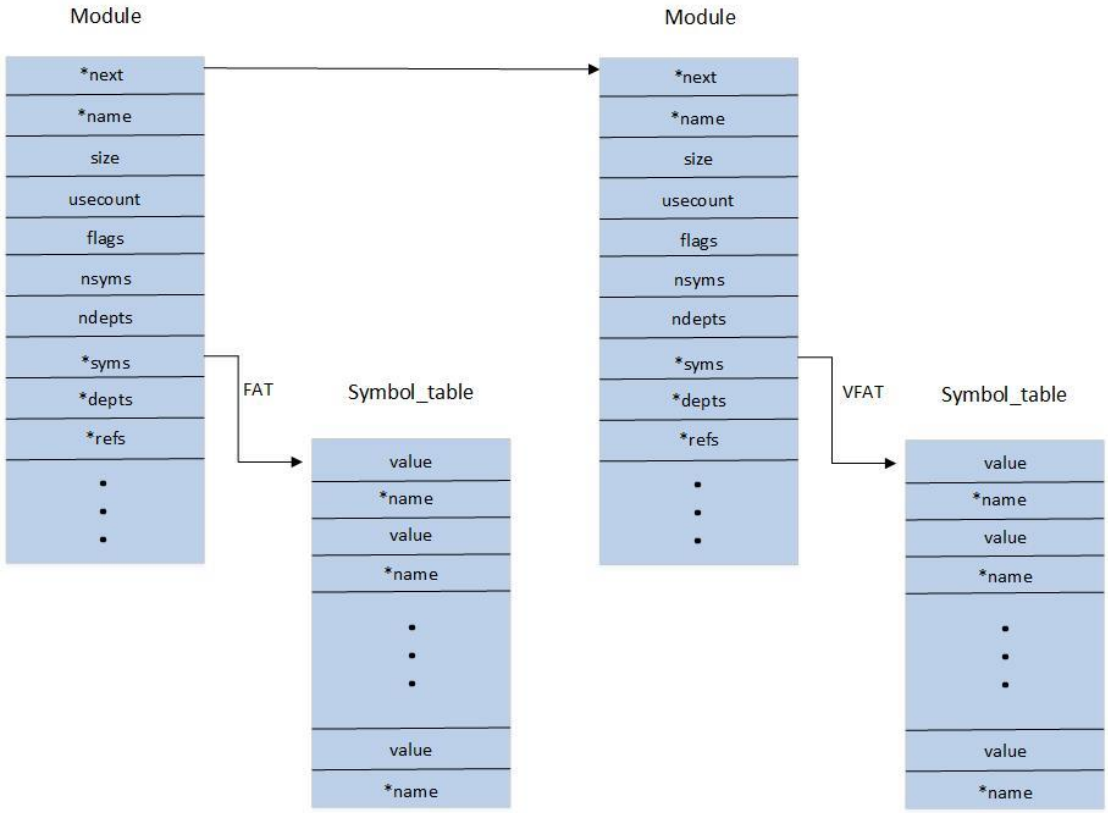
1. Ryhmälle samankaltaisia moduuleja voidaan (esim. laiteajurit samantyyppisille laitteille) määritellä yksi moduuli, jotta vältetään samankaltaisten moduulien kopiointi.
2. Kernel tietää milloin tarpeelliset moduulit ovat jo liitettyinä sekä osaa olla poistamatta moduulin yhteyttä kerneliin, kun sitä joku vielä moduuli tarvitsee.

Kuviossa 10 nähdään esimerkki kernelistä, jolloin sinne on luotu 2 moduulia File Allocation Table (FAT) ja Virtual File Allocation Table (VFAT). Kuviosta nähdään myös, että FAT on yhdistetty kerneliin ennen VFAT:iä. Molempia moduuleja määrittelevät niiden taulut. Moduulin taulu sisältää seuraavat elementit:

- *next: Osoittaa seuraavan moduulin
- *name: Moduulin nimi
- size: Moduulin koko
- usecount: Moduulin käyttölaskuri
- flags: Moduuulin liput
- nsyms: Vietyjen symbolien lukumäärä
- ndepts: Viitattujen moduulien lukumäärä

- *syms: Moduulin symboli-taulun osoitin
- *depts: Osoitin moduuli-listaan, joihin ko. moduuli viittaa
- *refs: Osoitin moduuli-listaan, jotka käyttävät ko. moduulia

Symboli-taulu määrittelee ne symbolit, jotka ovat ko. moduulin alaisia, mutta käytössä muualla. (Stallings 2009)



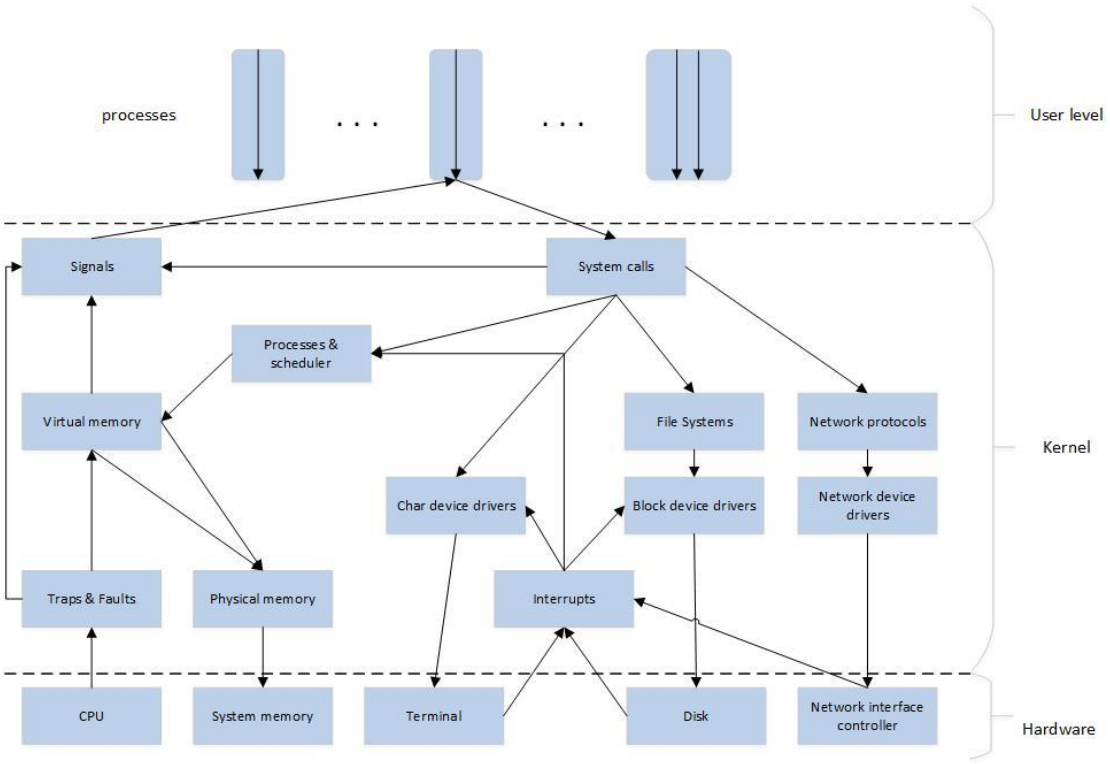
Kuvio 10. Kernel-moduulit ja niiden sisältö

7.3 Kernelkomponentit

Kuviosta 11 voidaan nähdä Linux kernelin pääkomponentit, kuten IA-64 arkkitehtuurissa on toteutettuna (esim. Intel Itanium). Kuviossa nähdään muutamia prosesseja kernelin päällä, joissa jokainen nuoli kuvastaa yhtä säiettä. Kernel itsessään sisältää kokoelman komponentteja, jotka ovat vuorovaikutuksessa toistensa kanssa. Kuvion nuolet osoittavat vuorovaikutuksien päätapahtumien

suuntia. Alimmassa kerroksessa oleva ”rautakerros” on myös kuvattu kokoelmana erillisiä komponentteja, joissa nuolet kuvaavat kernelin ja rautakerroksen komponenttien vuorovaikutuksia. Tietysti kaikki kernelin komponentit ovat yhteydessä prosessoriin, mutta kuvion selkeyttämiseksi sitä ei ole kuvaan piirretty. Seuraavassa kernelin komponenttien merkitykset lyhyesti:

- Signals: Kernel käyttää signaaleja kutsumaan prosesseja. Esimerkkinä signaaleja voidaan käyttää ilmoittamaan prosesseille tapahtuneista virheistä, kuten nollalla jaosta.
- System calls: System calls pyytää kerneliltä tiettyjä palveluja. Palveluja on noin muutama sata erilaista, jotka voidaan jakaa kuuteen eri kategoriaan: tiedostojärjestelmä, prosessit, aikataulutus, sisäinen kommunikaatio, socket (verkot) ja sekalaiset.
- Processes and scheduler: Luo, hoitaa ja ajastaa prosesseja.
- Virtual memory: Jakaa ja hoitaa virtuaalista muistia prosesseille.
- File systems: Tarjoaa globaalin hierarkisen nimiavaruuden tiedostoille, hakemistoille ja muille tiedosto-tyyppisille objekteille sekä tarjoaa tiedostojärjestelmän toimintoja.
- Network protocols: Socket tuki TCP/IP protokollaa varten.
- Character device drivers: Hallinnoi laitteita, jotka tarvitsevat kernelin lähettämään tai vastaanottamaan dataa bitti kerrallaan. Esimerkkinä terminaalit, modeemit ja printterit.
- Block device drivers: Hallinnoi laitteita, jotka lukevat datan blockeina, kuten CD-ROMit ja magneettilevyt (levykkeet ja HDD)
- Network device drivers: Hallinnoi verkkorajapinta kortteja ja portteja jotka ovat yhteydessä verkkolaitteisiin, kuten reitittimiin.
- Traps and faults: Käsittelee prosessorin tekemiä virheitä kuten viat muistissa.
- Physical memory: Käsittelee muistin ja virtuaalimuistin käyttöä.
- Interrupts: Käsittelee ulkoisten laitteiden virheitä kuten hiiren ja näppäimistön. (Stallings 2009)



Kuvio 11. Kernel-komponentit

7.4 Linux Ubuntu-server-jakelu

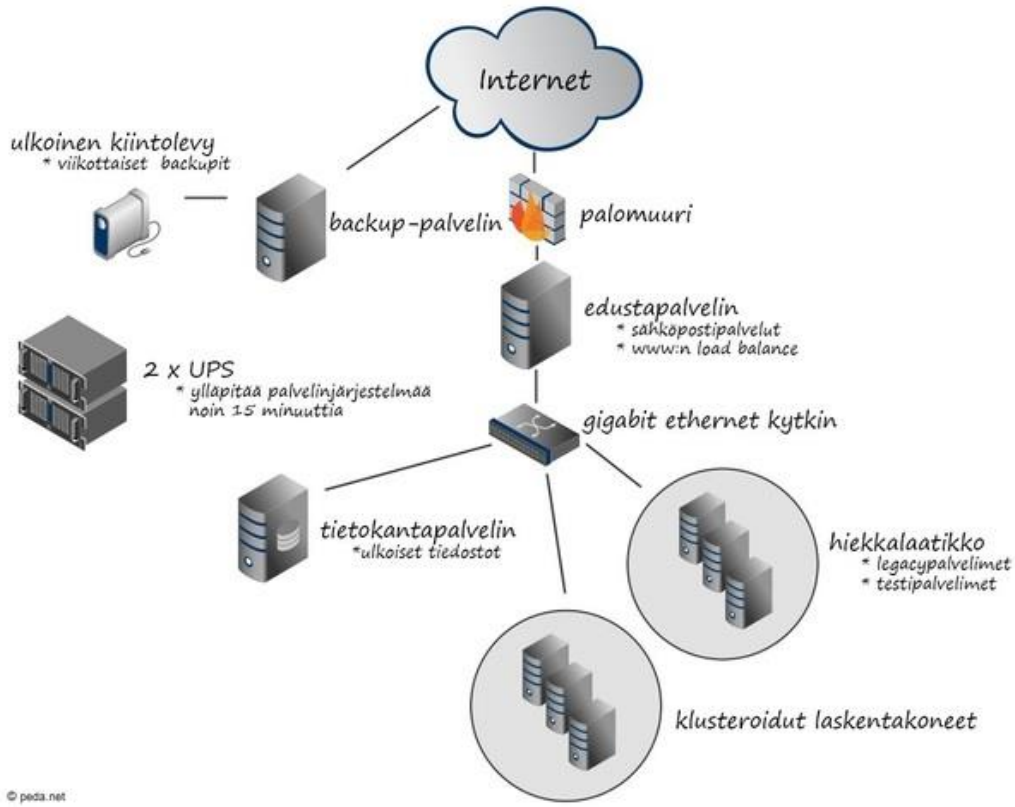
Ubuntu on tällä hetkellä yleisimmin käytössä oleva Debian-Linux-jakelu. Debian pohjaiset jakelut käyttävät deb-paketointi formaattia ja työkaluja hallitakseen sovelluksia. Ubuntu on saanut nimensä eteläafrikkalaisesta ideologiasta, joka tarkoittaa uskoa yleismaalliseen jakamisen siteeseen, joka yhdistää koko ihmisyyden ideaa. Ubuntu käyttää vakiona Unity-ikkunointijärjestelmää (entinen Gnome). Ubuntu jakelupaketti oli myös ensimmäisiä luomassa uusia menetelmiä ajaa Linuxia, esimerkiksi live-CD, jonka käyttöönotto mahdollisti saada Ubuntu toimintavalmiiksi käyttöön erittäin lyhyessä ajassa. Alun perin Ubuntu-projektin tarkoituksena oli tehdä käyttäjille käyttöjärjestelmä, joka olisi täysin ilmainen, mahdollisimman helppo asentaa ja helppo käyttää. Helpottaakseen siirtymistä Windows-käyttöjärjestelmästä Ubuntuun siinä tulee myös mukana Windowsista tutut ikkunointijärjestelmään implementoidut internet selaimet, tekstityökalut, taulukkolaskentaohjelmistot sekä pikaviestimet. (Ubuntu 2014) (Negus 2012)

Ubuntussa tulee mukana myös graafinen ohjelmisto, jonka avulla voidaan siihen ladata eri ohjelmistoja, pelejä jne. Tämä helpottaa ja nopeuttaa huomattavasti erilaisten sovelluksien etsimistä ja asentamista. Ubuntu-serverin uutena ominaisuutena on OpenStack-ominaisuus, joka mahdollistaa pilvessä olevien palvelimien ”Hot Swappaamisen” keskenään, jolloin esimerkiksi palvelinraudan päivittäminen tai palvelintoimintojen siirtäminen eri raudalle on nopeaa ja helppoa. Ubuntusta on valittavissa joko palvelinversio tai normaalikäyttöön suunnattu versio. Ubuntua on saatavilla 32- ja 64-bittisenä versiona. Ubuntu lupaa käyttäjilleen uuden version 6 kuukauden välein johon 9 kuukauden tuen sekä kahden vuoden välein Long Term Support (LTS) –version jossa viiden vuoden tuki. (Ubuntu 2014) (Negus 2012)

8 TOTEUTUS

8.1 Peda.net

Kuukausittain Peda.netissä on noin 1,150,100 vierailua ja 37,705,578 hakua. Toistaiseksi vierailut ja haut jakaantuvat uuden ja vanhan palvelun välillä seuraavasti. Uudessa palvelussa kuukautiset haut ovat noin 9,059,513 ja vierailut 122,636. Vanhassa palvelussa hakua tulee noin 9,059,513 ja vierailuja 122,636. Peda.netin topologia esitetty kuviossa 12.



Kuvio 12. Peda.netin topologia

Järjestelmässä edustapalvelin jakaa saapuvat pyynnöt klusterissa oleville laskentakoneille, joka toimii ensimmäisenä kuormantasaus pisteenä. Klusterissa oleva laskentakone hakee tarvittavan datan tietokantapalvelimelta ja palauttaa sitten

datan asiakkaalle. Toinen kuormantasaus kohta on laskentakoneiden käyttöjärjestelmään sijoitettu Pgbouncer, joka rajaa laskentakoneilta lähteviä yhtäaikaisia yhteyksien määriä tietokantapalvelimelle.

8.2 Klusterikoneiden komponentit

Klusterikoneiden tavoitteena oli olla mahdollisimman vähän tilaa vieviä, tehokkaita sekä hinta/laatu-suhteeltaan mahdollisimman hyviä eli ”best bang for the buck”. Koneiden vaatimuksena oli edellisten lisäksi 16 GB RAM ja 1 GB-verkkoadapteri. Alkuperäisenä ajatuksena klusterikoneiksi oli Intelin NUC-koneet, mutta hinta/laatu-suhteeltaan mini-ITX- koteloon komponenteista yhdistelty tietokone tuli huomattavasti edullisemmaksi sekä tehokkaammaksi. Klusterikoneita klusteriin tuli viisi kappaletta ja ne näyttivät seuraavanlaisilta.

- Intel Core i5 4670K 3.4 GHz LGA1150 -suoritin
- Asrock H81M-ITX LGA1150 - mini-ITX
- Thermaltake Element Q Mini ITX -kotelo 220w
- Kingston KVR16N11 16 GB 1600 MHz CL11
- Kingston SSD 60 GB
- Kozuti SCKZT-1000 40mm CPU Cooler

Prosessoriksi kokoonpanoon valittiin Hashwell-arkkitehtuuria käyttävä Intelin 4-ytiminen prosessori sen teho/hinta-suhteen ollessa hyvä sekä se, että virtuaaliytimistä ei olisi PHP-laskennassa (suurin osa klusterin laskennasta on PHP:tä) rahan edestä hyötyä. Olennaista on tarkastella niitä rajoitteita, joiden puitteissa hyperthreading (virtuaaliytimet) toimivat hyvin ja todeta että PHP-koodi ei yleisesti täytä vaatimuksia. Hyperthreading (HT, Intel trademark) aiheuttaa siis aina saman prosessoriytimen toiselle HT-yksiköille haittaa sotkemalla L1-muistia ja häiritsemällä jossain määrin branch prediction -ominaisuutta. Molemmat näistä ovat olennaisen tärkeitä PHP-koodin suorittamiselle. HT toimii parhaiten, jos HT-yksiköt ajavat samassa muistiavaruudessa olevaa koodia ja dataa. Jos jokin koodi on kirjoitettu multithreading -tekniikkaa hyvin tukevaksi niin se toimii hyvin HT-yksiköiden kanssa

olettaen, että käyttöjärjestelmä ei laita samalle oikealle ytimelle muita prosesseja samaan aikaan. Rinnakkaiset PHP-prosessit eivät toimi multithreading-mallin mukaisesti vaan ovat aidosti erillisiä prosesseja. Emolevyn valintaan vaikuttivat lähinnä riittävät liitännät eli 1 GB verkkoyhteys sekä hinta. Kotelossa taas vaikuttavina tekijöinä olivat koko sekä hinta. Kyseinen kotelo sisälsi 220 W virtalähteen. Keskusmuistit ja SSD-levy valittiin hinnan mukaan. SSD-levy nopeuttaa käyttöjärjestelmän toimintaa paikallisella koneella verrattuna edelliseen malliin jossa käyttöjärjestelmät sijaitsivat muistitikuilla. SSD myös nopeuttaa swapissa olevan tiedon hakemista. Keskusmuistissa katsottiin, että 1600 MHz-taajuudella sekä CL11-latenssilla olisi riittävä tarpeisiin nähden. Kozutin prosessorin jäähdytin valittiin sen max 95 W Thermal Design Power (TDP) -arvon sekä matalan korkeuden vuoksi. Prosessorin TDP- ollessa 84 W olisi suotavaa, että prosessorin jäähdytin ylittäisi lukeman, koska kotelon ollessa pieni ja ilmanvaihtoa sinne ei lisätty erillisillä kotelotuulettimilla. Näin ollen pitäisi sen pystyä jäähdyttämään prosessori luotettavasti tai sen kellotaajuudet laskisivat lämpöjen noustessa. Kotelo antoi prosessorin jäähdyttimelle 45 mm tilaa ja vakiojäähdytin oli korkeudeltaan 50 mm, Kozutin ollessa 40 mm. Vaikka prosessorin vakiojäähdytin olisi koteloon mahtunut, olisi ollut arveluttavaa olisiko sen jäähdyttämisteho ollut riittävä.

8.3 Laskentakoneiden toiminta

Jokaiseen laskentakoneeseen asennettiin Linux Ubuntu käyttöjärjestelmästä minimaalinen asennus, jotta vältyttiin koneiden tarkoitukseen nähden turhilta komponenteilta. Ubuntun lisäksi olennaisimpina niihin asennettiin Apache sekä mod-php.

Apachen sites/enabled/000-default lisättiin seuraavat konfiguraatiot.

```
<VirtualHost *:80>
...
 RewriteRule ^/(.*)$ /nfs/code/peda.net/peda.net/index.php
</VirtualHost>
```

Konfiguraatiolla on polun mukaisesti tehty liitos (Mount) Network File Systemin (NFS) yli "Storage"-tietokantapalvelimelta. PHP:n apc-välimuisti on konfiguroitu *apc.stat=0*

-määrityksellä, jolloin kukin PHP-tiedosto ladataan vain yhden kerran NFS:n yli Apache child:n käynnistämisen jälkeen. Laskentakone tallentaa tarvittavat tiedostot hakemistoon /nfs/data/peda.net/files, joka on myös NFS:n yli mountattu. Täällä olevia tiedostoja ei koskaan muokata vaan ne tallennetaan alun perin tiedoston sisällöstä lasketun SHA-1-tunnisteen mukaisilla tiedoston nimillä. Tiedoston nimien laskennassa on tarkistus estämässä olemassa olevan tiedoston korvaamista väärällä, vaikka SHA-1-tunnisteet olisivat samat, mutta tiedoston sisältö eri. Tiedoston vastaanottamisessa tiedostoa verrataan aina olemassa olevaan tiedostoon tavutasolla vaikka SHA-1-tunniste näyttää samalta.

8.4 Vanha vs uusi klusteri

Vanhassa klusterissa laskentatehoa toivat 4 kpl laskentakoneita, jotka oli jaettu niin, että Peda.netin vanhalla palvelulla oli käytössään 2 kpl laskentakoneita sekä uudella palvelulla 2 kpl. Uuden palvelun laskentakoneiden kokoonpanossa on Core 2 Duo 3,00 GHz prosessorit sekä 8 GB keskusmuistia ja vanhan puolen koneissa oli Intel Core 2 CPU 6300 1.86GHz ja 2 GB keskusmuistia. Klusteri toimi load balance klusterina, jossa edustapalvelin toimi kuorman jakajana ja tällöin koneiden ei tarvinnut jakaa muistia keskenään.

Vanhassa klusterissa prosessoritehoa oli yhteensä 9,72 GHz (2*3 GHz + 2*1,86 GHz) ja muistia 20 GB (2*8 GB + 2*2GB). Uudessa klusterissa prosessoritehoa oli 68 GHz (5*3,5 GHz) ja muistia 80 GB (16*5 GB), eli prosessori tehoa oli enemmän noin 7-kertainen määrä ja muistia 4-kertainen määrä.

9 TESTIT JA TULOKSET

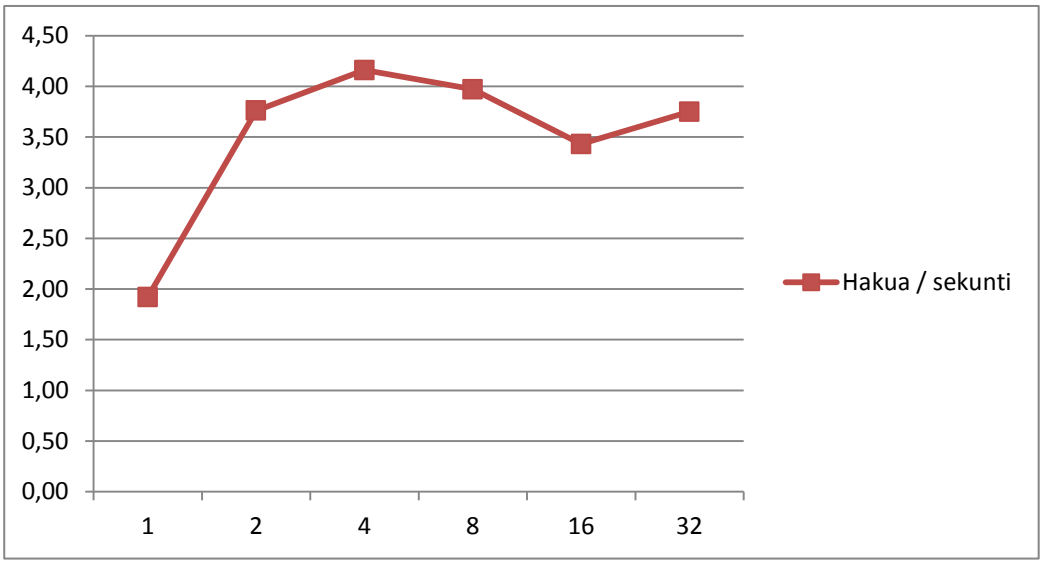
Jotta saatiin vertailua koneiden välillä tehtyä, siirrettiin uudemman palvelun puolelta laskentakone hiekkalaatikkoon ja sillä suoritettiin muutamia testejä. Testejä ei siis tehty load balancerin läpi, sillä se itsessään rajoittaisi syötettävän kuorman määrää laskentakoneille, jolloin ei niitä saataisi riittävästi kuormitettua.

Testi 1 piti sisällään 100 sivuhakua Peda.net etusivusta, ensin 1:llä käyttäjällä, sitten 2:lla yhtäaikaisella käyttäjällä, 4:llä, 8:lla, 16:lla ja lopuksi 32:lla. Testi tehtiin ApacheBenchiin tehdyllä scriptillä, joka suoritti kyseiset toiminnot. Testissä ApacheBench aloitti seuraavan sivun lataamisen edellisen valmistuessa. Tehokkaampi järjestelmä parantaa arvoa ”haettua sivua per sekunti”. Testi 1 suoritettiin ensin vanhalla laskentakoneella ja sen jälkeen uudella.

Taulukko 2. Tulokset vanhalla laskentakoneella

Yhtäaikaiset yhteydet	1	2	4	8	16	32
Hakua / sekunti	1,92	3,76	4,16	3,97	3,43	3,75

Kuvion 13 tuloksista voidaan nähdä, että yksi prosessori (Core 2 Duo 3,00 GHz, 2 ytiminen) hakee sekunnissa 1,92 sivua, josta suurin osa ajasta kuluu PHP:n laskemiseen. Kun taas sivun yhtäaikaisten lataajien määrä nostetaan kahteen, yhdessä sekunnissa pystytään lataamaan 3,76 sivua, sillä kyseisen koneen prosessorissa on 2 ydintä, jotka voivat yhtäaikaisesti hakea sivuja. Vaikka yhtäaikaisien lataajien määrää nostetaan ekspotentiaalisesti, niin 1 sivun hakuaika pysyy samana. Sivujen haun nopeuteen voidaan arvioida vaikuttavan osaltaan prosessorin nopeus sekä suuremmalta osalta prosessorin ytimien määrä.

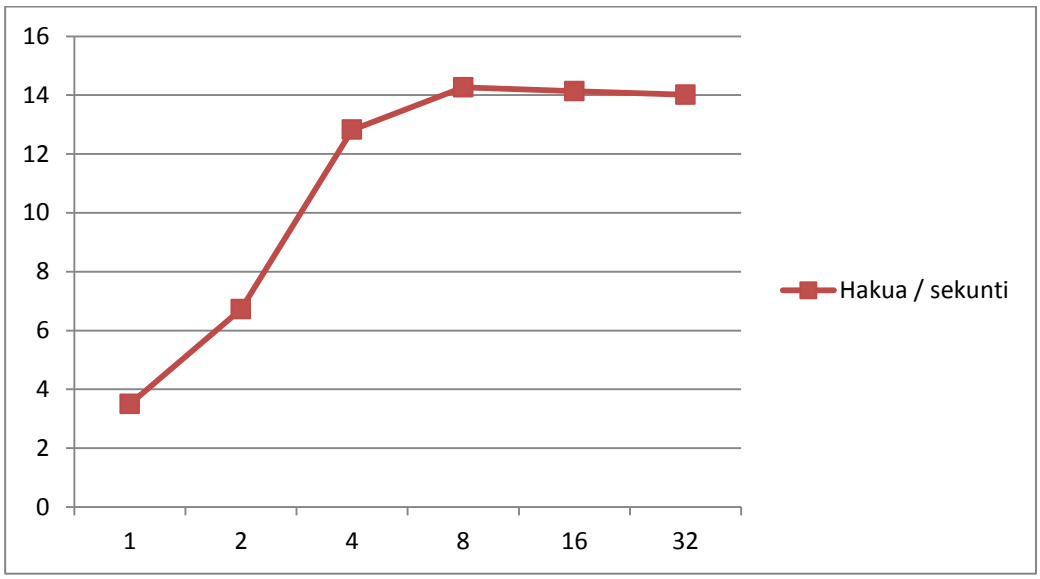


Kuvio 13. Testi 1 vanhalla laskentakoneella

Taulukko 3. Tulokset uudella laskentakoneella

Yhtäaikaiset yhteydet	1	2	4	8	16	32
Hakua / sekunti	3,5	6,48	12,29	13,58	11,54	9

Kuvion 14 tuloksista voidaan suoraan nähdä kuinka prosessorin ytimien määrä vaikuttaa hakujen määrän kasvuun suhteessa enemmän kuin yksittäisen prosessorin tehon ero. Yhden yksittäisen haun nopeus kasvaa noin kolmanneksen.



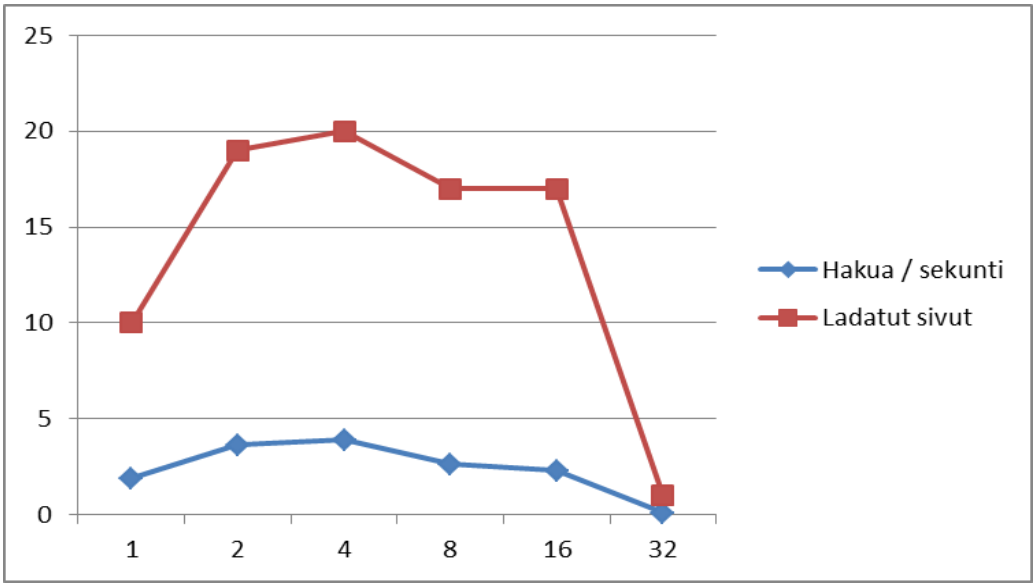
Kuvio 14. Testi 1 uudella laskentakoneella

Testissä 2 ApacheBench suoritti 5 sekunnin aikana Peda.net pääsivun latauksia. Ensin 1:llä käyttäjällä, 2:lla, 4:llä, 8:lla, 16:lla sekä 32:lla. Kyseisessä testissä ApacheBench ei odota yksittäisen haun valmistumista vaan tekee uusia hakuja odottamatta edellisten valmistumista.

Taulukko 4. Testin 2 tulokset vanhalla laskentakoneella

Yhtäaikaiset yhteydet	1	2	4	8	16	32
Hakua / sekunti	1,92	3,63	3,9	2,63	2,32	0,1
Ladatut sivut	10	19	20	17	17	1

Kuvion 15 tuloksista voidaan huomata, että kun prosessoria kuormitetaan 1 tai 2 ”asiakkaan” toimesta, pysyy se normaalin suorituskyvyn tasolla. Vaihdettaessa 8 tai useamman asiakkaan kuormittamiseen, prosessorin kuormitus nousi huomattavasti joka huononsi suorituskykyä huomattavasti ja lopulta lähes kaatoi sen.

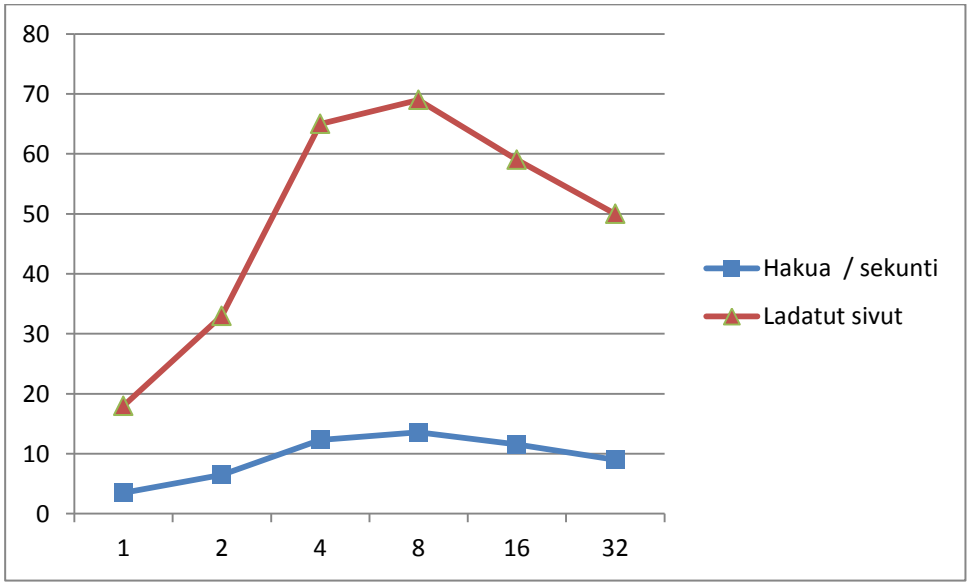


Kuvio 15. Testi 2 vanhalla laskentakoneella

Taulukko 5. Tulokset uudella laskentakoneella

Yhtäaikaiset yhteydet	1	2	4	8	16	32
Hakua / sekunti	3,5	6,48	12,29	13,58	11,54	9
Ladatut sivut	18	33	65	69	59	50

Kuvion 16 tuloksista uudella laskentakoneella voidaan selkeästi nähdä, että yhden ytimen suorituskyky nousee kolmanneksella. Ytimien määrän lisääntyminen kahdesta neljään voidaan havaita 2 ja 4 rinnakkaisen yhteyden testissä, jossa hakua/sekunti –arvo tuplaantuu. Uuden prosessorin kehittyneemmän teknologian ja paremman suorituskyvyn eron huomaa kun siirrytään useampiin rinnakkaisiin yhteyksiin. Esimerkiksi 32 yhtäaikaisen yhteyden testistä suoriudutaan vielä, vaikkakin suorituskyky tässä laskee huomattavasti.



Kuvio 16. Testi 2 uudella laskentakoneella

Testin 3 tarkoituksena oli mitata yksittäisen sivun latautumisnopeutta. Mitattava sivu sisälsi lähes kaikki mahdolliset komponentit, jotka Peda.netin uusi palvelu sisältää, tehden sivusta näin ollen mahdollisimman raskaan. Testi tehtiin cURL:illa. Testin

tulokset löytyvät tarkemmin liitteestä 6. Testin tulos vahvisti yksittäisen sivun latautumisnopeuden kasvua noin kolmanneksella.

Worker 1 (vanha laskentakone) 3.074s

Worker 5 (uusi laskentakone) 1.993s

10 YHTEENVETO

Testien tarkoituksena oli mitata käyttäjille näkyviä viiveen muutoksia palvelinraudan päivittämisen myötä. Palvelun nopeuden voidaan huomata kasvaneen noin kolmanneksella. Rinnakkaisuus lisääntyi huomattavasti, jonka vaikutus ilmenee kun palvelua käyttävät yhtäaikaisesti useat käyttäjät.

Koko järjestelmän suorituskykytestiä eli pullonkaulaa ei päästy testaamaan, koska identtistä testiympäristöä ei ole sekä koko palvelua ei voi ”kyykäyttää” sen ollessa tuotannossa. Laskennallisiksi faktoiksi ja arvailuiksi voitaisiin arvioida, että kun tietokantapalvelimessa on 16 GB RAM ja PostgreSQL:n tietokannan koko on 12 GB. Tietokanta on siis lukemisen osalta CPU ja RAM (nopeus) -rajoitteinen, ei IO-rajoitteinen. Tietokannan IO-rajoitukset tulevat vastaan vasta tietoa muutettaessa, mutta nykyisessä palvelussa muutoksia tulee niin vähän suhteessa lukemiseen, että tästä ei tule ongelmaa. Palvelin taas tallentaa tietokantansa RAID1-levyjärjestelmään, joka on rakennettu Intel SSD 320 -sarjan levyistä. RAID1-järjestelmää ajaa Linux, ei rautakortti.

Klusterin tyyppi on load balance, ei laskentatehoa lisäävä. Apache ei tue klusteroitua laskemista ko. prosesseissa, jolloin laskentatehoa lisäävä klusteri olisi tarpeeton.

Työn tavoitteissa mielestäni onnistuttiin, sillä koko klusteri on vain hiukan yli yhden tornipalvelimen kokoinen ja laskentatehoa edelliseen verrattuna saatiin lisää palvelujen tarvitsema määrä, joka jättää reilusti varaa kasvulle tulevaisuudessakin. Seuraavat käyttäjille näkyvät päivitykset järjestelmään tehdään kooditasolla, ei rautaa rajalle tyylisesti.

LÄHTEET

Apache. 2014. Apache verkkosivut. Viitattu 26.3.2014 https://www.apache.org/foundation/faq.html#what

Commodity computing. 2014. Techopedia verkkosivut. Viitattu 7.4.2014. http://www.techopedia.com/definition/29128/commodity-computing

Esittely. 2013. Jyväskylän koulutuksen tutkimuslaitoksen verkkosivut. Viitattu 12.2.2014. https://ktl.jyu.fi/esittely

GNU. 2014. GNU's Not Unix verkkosivut. Viitattu 26.3.2014. https://www.gnu.org

Klusterointiteknologiat. 2013. Cluster technologies, verkkosivut. Viitattu 1.3.2014. http://www.cro-ngi.hr/en/technologies/cluster-technologies/

Linux. 2014. Linux verkkosivut. Viitattu 23.3.2014. http://www.linux.fi/wiki

Lowe, S. 2011. The pros and cons of tower, rack, and blade servers. Viitattu 4.4.2014 http://www.techrepublic.com/blog/the-enterprise-cloud/the-pros-and-cons-of-tower-rack-and-blade-servers/

Negus, C. 2012. Linux Bible (The comprehensive, tutorial resource, eight edition). John Wiley & Sons.

Oracle. 2010. Best Practices for data reliability with oracle VM server for sparc. Viitattu 25.4.2014 http://www.oracle.com/technetwork/articles/systems-hardware-architecture/vmsrvrsparc-reliability-163931.pdf

Peda.net. 2014. Peda.net verkkosivut. Viitattu 12.2.2014. http://peda.net

Rantonen, M. 2014. Lehtori yrityksessä JAMK University of Applied Sciences. Tapaaminen 25.3.2014.

SPDY. 2014. Chrome-projektien verkkosivut. Viitattu 7.4.2014. http://www.chromium.org/spdy/spdy-whitepaper

Stallings, W. 2009. Operating Systems (Internals and Design Principles, Sixth Edition). Pearson Education.

Ubuntu. 2014. Linux Ubuntu verkkosivut. Viitattu 26.3.2014. http://www.ubuntu.com/

Wainwright, P. 2002. Professional Apache 2.0. Wrox Press.

LIITTEET

Liite 1. ApacheBench scripti

```
#!/bin/sh

echo "---------------------------------"
echo "Palvelinten suorituskykytestit"
echo "---------------------------------"
echo "Paina Enter suorittaaksesi tai CTRL+C peruuttaaksesi testit"
read key

for cc in 1 2 4 8 16 32; do
        echo "$cc rinnakkaisen yhteyden testi 100 sivuhakua"
        echo "---------------------------------"
        ab -n 100 -c $cc $1
        echo "---------------------------------"
done

for cc in 1 2 4 8 16 32; do
        echo "$cc rinnakkaisen yhteyden testi 5 sek"
        echo "---------------------------------"
        ab -t 5 -c $cc $1
        echo "---------------------------------"
done
```

Liite 2. ApacheBench 1

```
ab -n 100 -c 1 http://worker2/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>

Benchmarking worker2 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      1
Time taken for tests:   52.098 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1203400 bytes
HTML transferred:       1154600 bytes
Requests per second:    1.92 [#/sec] (mean)
Time per request:       520.981 [ms] (mean)
Time per request:       520.981 [ms] (mean, across all concurrent requests)
Transfer rate:          22.56 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   498  521   6.0    522     533
Waiting:      495  517   5.7    518     528
Total:        498  521   6.0    522     533

Percentage of the requests served within a certain time (ms)
  50%    522
  66%    524
  75%    524
  80%    524
  90%    526
  95%    529
  98%    531
  99%    533
 100%    533 (longest request)

 ab -n 100 -c 2 http://worker2/
```

```
Benchmarking worker2 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      2
Time taken for tests:   26.580 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1203400 bytes
HTML transferred:       1154600 bytes
Requests per second:    3.76 [#/sec] (mean)
Time per request:       531.600 [ms] (mean)
Time per request:       265.800 [ms] (mean, across all concurrent requests)
Transfer rate:          44.21 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   510  531  62.6    522     967
Waiting:      507  528  62.6    519     963
Total:        510  532  62.6    522     967

Percentage of the requests served within a certain time (ms)
  50%    522
  66%    524
  75%    526
  80%    527
  90%    530
  95%    533
  98%    966
  99%    967
 100%    967 (longest request)

ab -n 100 -c 4 http://worker2/

Benchmarking worker2 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80
```

```
Document Path:        /
Document Length:      11546 bytes

Concurrency Level:    4
Time taken for tests:  24.050 seconds
Complete requests:    100
Failed requests:      0
Write errors:         0
Total transferred:    1203400 bytes
HTML transferred:     1154600 bytes
Requests per second:   4.16 [#/sec] (mean)
Time per request:     962.016 [ms] (mean)
Time per request:     240.504 [ms] (mean, across all concurrent requests)
Transfer rate:        48.86 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   536  949  84.4    945    1101
Waiting:      533  943  84.0    937    1094
Total:        537  949  84.4    945    1102

Percentage of the requests served within a certain time (ms)
  50%    945
  66%    992
  75%   1019
  80%   1025
  90%   1047
  95%   1069
  98%   1091
  99%   1102
 100%   1102 (longest request)

ab -n 100 -c 8 http://worker2/

Benchmarking worker2 (be patient).....done

Server Software:      Apache/2.2.22
Server Hostname:      worker2
Server Port:          80

Document Path:        /
Document Length:      11546 bytes

Concurrency Level:    8
Time taken for tests:  25.164 seconds
Complete requests:    100
```

```
Failed requests:        0
Write errors:           0
Total transferred:      1203400 bytes
HTML transferred:       1154600 bytes
Requests per second:    3.97 [#/sec] (mean)
Time per request:       2013.121 [ms] (mean)
Time per request:       251.640 [ms] (mean, across all concurrent requests)
Transfer rate:          46.70 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   856 1978 250.9   1987    2341
Waiting:      851 1968 249.6   1975    2321
Total:        856 1978 250.9   1987    2341

Percentage of the requests served within a certain time (ms)
  50%   1987
  66%   2046
  75%   2119
  80%   2146
  90%   2261
  95%   2301
  98%   2337
  99%   2341
 100%   2341 (longest request)

ab -n 100 -c 16 http://worker2/

Benchmarking worker2 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      16
Time taken for tests:   29.197 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1203400 bytes
HTML transferred:       1154600 bytes
Requests per second:    3.43 [#/sec] (mean)
Time per request:       4671.499 [ms] (mean)
```

```
Time per request:       291.969 [ms] (mean, across all concurrent requests)
Transfer rate:          40.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:  2225 4601 1311.6   4252    8080
Waiting:     2222 4576 1306.7   4234    8042
Total:       2226 4601 1311.6   4252    8080

Percentage of the requests served within a certain time (ms)
  50%   4252
  66%   4446
  75%   4549
  80%   4698
  90%   7437
  95%   7659
  98%   7909
  99%   8080
 100%   8080 (longest request)

ab -n 100 -c 32 http://worker2/

Benchmarking worker2 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      32
Time taken for tests:   26.634 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1203400 bytes
HTML transferred:       1154600 bytes
Requests per second:    3.75 [#/sec] (mean)
Time per request:       8522.814 [ms] (mean)
Time per request:       266.338 [ms] (mean, across all concurrent requests)
Transfer rate:          44.12 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
```

Processing: 2647 7863 1421.2   8167   9497
Waiting: 2642 7824 1412.2   8133   9435
Total: 2647 7864 1421.2   8167   9497

Percentage of the requests served within a certain time (ms)
 50%   8167
 66%   8382
 75%   8655
 80%   8730
 90%   9070
 95%   9187
 98%   9401
 99%   9497
100%   9497 (longest request)

Liite 3. ApacheBench 2

```
ab -t 5 -c 1 http://worker2/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker2 (be patient)
Finished 10 requests

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      1
Time taken for tests:   5.218 seconds
Complete requests:      10
Failed requests:        0
Write errors:           0
Total transferred:      120340 bytes
HTML transferred:       115460 bytes
Requests per second:    1.92 [#/sec] (mean)
Time per request:       521.759 [ms] (mean)
Time per request:       521.759 [ms] (mean, across all concurrent requests)
Transfer rate:          22.52 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   515  522   4.3    524     527
Waiting:      512  518   4.1    519     524
Total:        515  522   4.3    524     528

Percentage of the requests served within a certain time (ms)
  50%    524
  66%    524
  75%    524
  80%    525
  90%    528
  95%    528
  98%    528
  99%    528
 100%    528 (longest request)
```

```
ab -t 5 -c 2 http://worker2/

Benchmarking worker2 (be patient)
Finished 19 requests

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      2
Time taken for tests:   5.235 seconds
Complete requests:      19
Failed requests:        0
Write errors:           0
Total transferred:      228646 bytes
HTML transferred:       219374 bytes
Requests per second:    3.63 [#/sec] (mean)
Time per request:       551.073 [ms] (mean)
Time per request:       275.536 [ms] (mean, across all concurrent requests)
Transfer rate:          42.65 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   513  523   4.8    524     531
Waiting:      510  519   4.6    521     526
Total:        513  523   4.8    524     531

Percentage of the requests served within a certain time (ms)
  50%    524
  66%    526
  75%    527
  80%    527
  90%    530
  95%    531
  98%    531
  99%    531
 100%    531 (longest request)

ab -t 5 -c 4 http://worker2/

Benchmarking worker2 (be patient)
Finished 20 requests
```

```
Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      4
Time taken for tests:   5.132 seconds
Complete requests:      20
Failed requests:        0
Write errors:           0
Total transferred:      240680 bytes
HTML transferred:       230920 bytes
Requests per second:    3.90 [#/sec] (mean)
Time per request:       1026.387 [ms] (mean)
Time per request:       256.597 [ms] (mean, across all concurrent requests)
Transfer rate:          45.80 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   868  972  62.4    980    1068
Waiting:      864  966  62.2    969    1063
Total:        868  972  62.4    981    1068

Percentage of the requests served within a certain time (ms)
  50%    981
  66%   1009
  75%   1031
  80%   1035
  90%   1058
  95%   1068
  98%   1068
  99%   1068
 100%   1068 (longest request)

ab -t 5 -c 8 http://worker2/

Benchmarking worker2 (be patient)
Finished 17 requests


Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80
```

```
Document Path:        /
Document Length:      11546 bytes

Concurrency Level:    8
Time taken for tests:  6.469 seconds
Complete requests:    17
Failed requests:      0
Write errors:         0
Total transferred:    204578 bytes
HTML transferred:     196282 bytes
Requests per second:   2.63 [#/sec] (mean)
Time per request:     3044.101 [ms] (mean)
Time per request:     380.513 [ms] (mean, across all concurrent requests)
Transfer rate:        30.88 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:  1975 2300 237.8   2310    2671
Waiting:     1940 2285 244.5   2306    2667
Total:       1975 2300 237.8   2310    2671

Percentage of the requests served within a certain time (ms)
  50%   2302
  66%   2480
  75%   2534
  80%   2536
  90%   2610
  95%   2671
  98%   2671
  99%   2671
 100%   2671 (longest request)

ab -t 5 -c 16 http://worker2/

Benchmarking worker2 (be patient)
Finished 17 requests

Server Software:      Apache/2.2.22
Server Hostname:      worker2
Server Port:          80

Document Path:        /
Document Length:      11546 bytes

Concurrency Level:    16
Time taken for tests:  7.324 seconds
```

```
Complete requests:      17
Failed requests:        0
Write errors:           0
Total transferred:      204578 bytes
HTML transferred:       196282 bytes
Requests per second:    2.32 [#/sec] (mean)
Time per request:       6893.310 [ms] (mean)
Time per request:       430.832 [ms] (mean, across all concurrent requests)
Transfer rate:          27.28 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:  3293 4232 325.9   4298    4742
Waiting:     3250 4198 328.5   4235    4718
Total:       3293 4232 325.8   4298    4742

Percentage of the requests served within a certain time (ms)
  50%   4284
  66%   4341
  75%   4365
  80%   4415
  90%   4607
  95%   4742
  98%   4742
  99%   4742
 100%   4742 (longest request)

 ab -t 5 -c 32 http://worker2/

Benchmarking worker2 (be patient)
Finished 1 requests

Server Software:        Apache/2.2.22
Server Hostname:        worker2
Server Port:            80

Document Path:          /
Document Length:        11546 bytes

Concurrency Level:      32
Time taken for tests:   9.959 seconds
Complete requests:      1
Failed requests:        0
Write errors:           0
Total transferred:      12034 bytes
HTML transferred:       11546 bytes
```

```
Requests per second:    0.10 [#/sec] (mean)
Time per request:       318689.440 [ms] (mean)
Time per request:       9959.045 [ms] (mean, across all concurrent requests)
Transfer rate:          1.18 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:  9959 9959   0.0   9959    9959
Waiting:     9931 9931   0.0   9931    9931

Total:       9959 9959   0.0   9959    9959
```

Liite 4. Apachebench 3

```
---------------------------------
Palvelinten suorituskykytestit
---------------------------------
Paina Enter suorittaaksesi tai CTRL+C peruuttaaksesi testit
1 rinnakkaisen yhteyden testi 100 sivuhakua
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      1
Time taken for tests:   28.565 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1223400 bytes
HTML transferred:       1174700 bytes
Requests per second:    3.50 [#/sec] (mean)
Time per request:       285.645 [ms] (mean)
Time per request:       285.645 [ms] (mean, across all concurrent requests)
Transfer rate:          41.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   269  285   3.7    286     293
Waiting:      267  283   3.7    284     291
Total:        270  286   3.7    286     293

Percentage of the requests served within a certain time (ms)
  50%    286
  66%    287
  75%    288
  80%    288
  90%    290
```

```
 95%    291
 98%    292
 99%    293
100%    293 (longest request)
---------------------------------
2 rinnakkaisen yhteyden testi 100 sivuhakua
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      2
Time taken for tests:   14.870 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1223400 bytes
HTML transferred:       1174700 bytes
Requests per second:    6.72 [#/sec] (mean)
Time per request:       297.401 [ms] (mean)
Time per request:       148.701 [ms] (mean, across all concurrent requests)
Transfer rate:          80.34 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   282  297   9.1    299     336
Waiting:      279  294   8.8    296     330
Total:        283  297   9.1    299     337

Percentage of the requests served within a certain time (ms)
  50%    299
  66%    301
  75%    302
  80%    303
  90%    304
  95%    305
  98%    335
```

```
 99%    337
 100%    337 (longest request)
---------------------------------
4 rinnakkaisen yhteyden testi 100 sivuhakua
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      4
Time taken for tests:   7.799 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1223400 bytes
HTML transferred:       1174700 bytes
Requests per second:    12.82 [#/sec] (mean)
Time per request:       311.941 [ms] (mean)
Time per request:       77.985 [ms] (mean, across all concurrent requests)
Transfer rate:          153.20 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   297  311   4.4    311     321
Waiting:      294  308   4.3    309     318
Total:        297  311   4.4    312     321

Percentage of the requests served within a certain time (ms)
  50%    312
  66%    313
  75%    314
  80%    314
  90%    316
  95%    320
  98%    321
  99%    321
 100%    321 (longest request)
```

```
---------------------------------
8 rinnakkaisen yhteyden testi 100 sivuhakua
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      8
Time taken for tests:   7.014 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1223400 bytes
HTML transferred:       1174700 bytes
Requests per second:    14.26 [#/sec] (mean)
Time per request:       561.091 [ms] (mean)
Time per request:       70.136 [ms] (mean, across all concurrent requests)
Transfer rate:          170.34 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:   345  552  52.9    559     681
Waiting:      343  548  52.4    555     678
Total:        346  553  52.9    560     681

Percentage of the requests served within a certain time (ms)
  50%    560
  66%    570
  75%    579
  80%    584
  90%    606
  95%    635
  98%    643
  99%    681
 100%    681 (longest request)
---------------------------------
16 rinnakkaisen yhteyden testi 100 sivuhakua
```

```
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient).....done

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      16
Time taken for tests:   7.076 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      1223400 bytes
HTML transferred:       1174700 bytes
Requests per second:    14.13 [#/sec] (mean)
Time per request:       1132.169 [ms] (mean)
Time per request:       70.761 [ms] (mean, across all concurrent requests)
Transfer rate:          168.84 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:   378 1103 168.7   1122    1405
Waiting:      375 1093 166.6   1112    1402
Total:        378 1103 168.7   1122    1405

Percentage of the requests served within a certain time (ms)
  50%   1122
  66%   1156
  75%   1176
  80%   1193
  90%   1255
  95%   1324
  98%   1383
  99%   1405
 100%   1405 (longest request)
---------------------------------
32 rinnakkaisen yhteyden testi 100 sivuhakua
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
```

Benchmarking worker5 (be patient).....done

Server Software: Apache/2.2.22
Server Hostname: worker5
Server Port: 80

Document Path: /
Document Length: 11747 bytes

Concurrency Level: 32
Time taken for tests: 7.139 seconds
Complete requests: 100
Failed requests: 0
Write errors: 0
Total transferred: 1223400 bytes
HTML transferred: 1174700 bytes
Requests per second: 14.01 [#/sec] (mean)
Time per request: 2284.616 [ms] (mean)
Time per request: 71.394 [ms] (mean, across all concurrent requests)
Transfer rate: 167.34 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.1	0	0
Processing:	864	2156 556.1	2144	4050
Waiting:	861	2142 554.1	2131	4046
Total:	865	2157 556.1	2144	4050

Percentage of the requests served within a certain time (ms)
50% 2144
66% 2249
75% 2320
80% 2363
90% 2722
95% 3412
98% 3717
99% 4050
100% 4050 (longest request)

Liite 5. ApacheBench 4

```
---------------------------------
1 rinnakkaisen yhteyden testi 5 sek
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient)

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      1
Time taken for tests:   5.143 seconds
Complete requests:      18
Failed requests:        0
Write errors:           0
Total transferred:      220212 bytes
HTML transferred:       211446 bytes
Requests per second:    3.50 [#/sec] (mean)
Time per request:       285.703 [ms] (mean)
Time per request:       285.703 [ms] (mean, across all concurrent requests)
Transfer rate:          41.82 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   280  285   3.1    286     291
Waiting:      278  283   2.9    283     288
Total:        280  286   3.1    286     291

Percentage of the requests served within a certain time (ms)
  50%    286
  66%    287
  75%    288
  80%    289
  90%    289
  95%    291
  98%    291
  99%    291
```

```
 100%    291 (longest request)
---------------------------------
2 rinnakkaisen yhteyden testi 5 sek
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient)

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      2
Time taken for tests:   5.094 seconds
Complete requests:      33
Failed requests:        0
Write errors:           0
Total transferred:      415951 bytes
HTML transferred:       399393 bytes
Requests per second:    6.48 [#/sec] (mean)
Time per request:       308.701 [ms] (mean)
Time per request:       154.351 [ms] (mean, across all concurrent requests)
Transfer rate:          79.75 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   289  300   4.8    300     308
Waiting:      286  297   4.9    298     305
Total:        289  300   4.8    300     308

Percentage of the requests served within a certain time (ms)
  50%    300
  66%    303
  75%    303
  80%    304
  90%    306
  95%    308
  98%    308
  99%    308
 100%    308 (longest request)
---------------------------------
```

4 rinnakkaisen yhteyden testi 5 sek

This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient)

Server Software: Apache/2.2.22
Server Hostname: worker5
Server Port: 80

Document Path: /
Document Length: 11747 bytes

Concurrency Level: 4
Time taken for tests: 5.288 seconds
Complete requests: 65
Failed requests: 0
Write errors: 0
Total transferred: 824815 bytes
HTML transferred: 791699 bytes
Requests per second: 12.29 [#/sec] (mean)
Time per request: 325.433 [ms] (mean)
Time per request: 81.358 [ms] (mean, across all concurrent requests)
Transfer rate: 152.31 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.1	0	0
Processing:	300	311 4.0	311	320
Waiting:	297	308 3.9	308	317
Total:	301	311 4.0	311	320

Percentage of the requests served within a certain time (ms)
50% 311
66% 313
75% 313
80% 314
90% 315
95% 318
98% 319
99% 320
100% 320 (longest request)

---

8 rinnakkaisen yhteyden testi 5 sek

```
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient)

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      8
Time taken for tests:   5.079 seconds
Complete requests:      69
Failed requests:        0
Write errors:           0
Total transferred:      844146 bytes
HTML transferred:       810543 bytes
Requests per second:    13.58 [#/sec] (mean)
Time per request:       588.897 [ms] (mean)
Time per request:       73.612 [ms] (mean, across all concurrent requests)
Transfer rate:          162.30 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:   492  563  37.8    560     732
Waiting:      487  558  37.7    555     725
Total:        493  563  37.8    560     732

Percentage of the requests served within a certain time (ms)
  50%    558
  66%    575
  75%    583
  80%    590
  90%    603
  95%    620
  98%    641
  99%    732
 100%    732 (longest request)
---------------------------------
16 rinnakkaisen yhteyden testi 5 sek
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
```

```
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking worker5 (be patient)

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      16
Time taken for tests:   5.111 seconds
Complete requests:      59
Failed requests:        0
Write errors:           0
Total transferred:      734035 bytes
HTML transferred:       704815 bytes
Requests per second:    11.54 [#/sec] (mean)
Time per request:       1386.039 [ms] (mean)
Time per request:       86.627 [ms] (mean, across all concurrent requests)
Transfer rate:          140.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:  1029 1252 215.3   1187    2108
Waiting:     1025 1243 214.5   1176    2100
Total:       1029 1253 215.3   1187    2108

Percentage of the requests served within a certain time (ms)
  50%   1185
  66%   1225
  75%   1272
  80%   1341
  90%   1585
  95%   1684
  98%   1931
  99%   2108
 100%   2108 (longest request)
---------------------------------
32 rinnakkaisen yhteyden testi 5 sek
---------------------------------
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking worker5 (be patient)

Server Software:        Apache/2.2.22
Server Hostname:        worker5
Server Port:            80

Document Path:          /
Document Length:        11747 bytes

Concurrency Level:      32
Time taken for tests:   5.558 seconds
Complete requests:      50
Failed requests:        0
Write errors:           0
Total transferred:      611700 bytes
HTML transferred:       587350 bytes
Requests per second:    9.00 [#/sec] (mean)
Time per request:       3557.226 [ms] (mean)
Time per request:       111.163 [ms] (mean, across all concurrent requests)
Transfer rate:          107.47 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:  1876 2664 678.6   2362    4201
Waiting:     1873 2649 680.5   2335    4188
Total:       1876 2664 678.6   2362    4201

Percentage of the requests served within a certain time (ms)
  50%   2362
  66%   2547
  75%   3359
  80%   3613
  90%   3849
  95%   3922
  98%   4201
  99%   4201
 100%   4201 (longest request)
---------------------------------
```

Liite 6. cURL

```
xxxx@peda:/etc/apache2$ time curl -I --cookie "session-id=417ca91c-a713-46d4-
a7ca-f4d0a6a81e1f" "http://worker1/p/xxxx/kopioitu-sis%C3%A4lt%C3%B6:move"
HTTP/1.1 200 OK
Date: Thu, 17 Apr 2014 11:28:21 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.10
Cache-Control: private, max-age=0, s-maxage=0
Expires: Thu, 17 Apr 2014 11:28:24 GMT
Frame-Options: DENY
Vary: Accept,User-Agent
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Connection: close
Content-Type: text/html; charset=UTF-8

real    0m3.074s
user    0m0.004s
sys     0m0.000s

xxxx@peda:/etc/apache2$ time curl -I --cookie "session-id=417ca91c-a713-46d4-
a7ca-f4d0a6a81e1f" "http://worker5/p/xxxx/kopioitu-sis%C3%A4lt%C3%B6:move"
HTTP/1.1 200 OK
Date: Thu, 17 Apr 2014 11:28:36 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.8
Cache-Control: private, max-age=0, s-maxage=0
Expires: Thu, 17 Apr 2014 11:28:38 GMT
Frame-Options: DENY
Vary: Accept,User-Agent
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Connection: close
Content-Type: text/html; charset=UTF-8


real    0m1.993s
user    0m0.000s
sys     0m0.004s
```