TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

# TIE-02500 RINNAKKAISUUS, KEVÄT 2015

# HARJOITUSTYÖN SUUNNITTELUDOKUMENTTI

218605 Eriksson Mikael

218100 Finta Albert

# SISÄLLYS

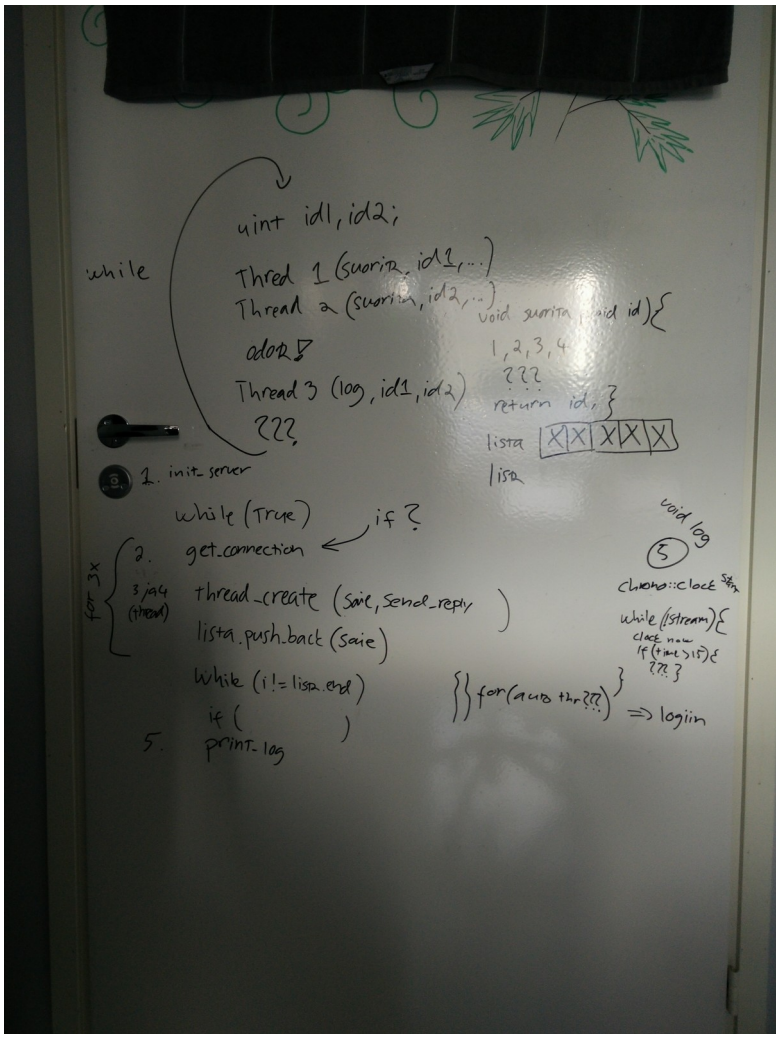
# 1 JOHDANTO

Tässä raportissa on dokumentoitu TIE-02500 Rinnakkaisuus -kurssin harjoitustyönä toteutetun rinnakkaisen http-palvelimen suunnittelu- ja koodausprosessi. Dokumentissa kuvataan ohjelman rakenne ja siinä käytetty rinnakkaisuusratkaisu. Lisäksi tarkastellaan rinnakkaisuuteen liittyvä optimointikysymyksiä, sekä pohditaan tulevia kehityskohteita ohjelman toiminnassa.

# 2 OHJELMAN SUUNNITTELU

Ohjelman suunnittelu aloitettiin pohtimalla sopivaa rinnakkaista toteutusta. Suunnittelu päätettiin aloittaa juuri säikeiden ja rinnakkaisuuden suunnittelulla, koska toteutuksen muuttaminen tulisi olemaan myöhemmin työlästä. Lisäksi ohjelman toiminnalliset osat olivat valmiina saatavilla, joten jo heti alussa saatiin periaatteessa toimiva ohjelma, jonka suorituskykyä voitiin hioa.

Suunnittelun lähtökohdaksi otettiin rakenne, jossa on kolme säiettä. Säikeet 1 ja 2 palvelevat asiakkaita, ja säie 3 kirjoittaa lokia. Kuva 1 esittää suunnitteluprosessin tätä vaihetta.




*Kuva 1: Alustava ajatus ohjelman toiminnasta*

Jo suunnittelun alkuvaiheessa päätimme lisätä säikeiden määrää. Tehtävänannon mukaan vaadittiin vähintään kolme säiettä, mutta katosimme että huomattavasti suurempi säiden lukumäärä palvelee tarkoitusta paremmin. Tarkempia testejä optimaalisesta säikeiden lukumäärästä on esitetty kappaleessa 3.1.

Tässä vaiheessa suunnittelutyötä ajattelimme toteuttaa myös tiedoston kirjoituksen rinnakkaisena operaationa asiakkaiden palvelemisen kanssa. Tämä olisi mahdollista, joskin vain yksi säie kerrallaan voi kirjoittaa tiedostoon. Käytännön haasteiden takia päädyimme kuitenkin lopullisessa ratkaisussa toteuttamaan lokitiedoston kirjoittamisen sekventiaalisesti. Tämän katsottiin olevan riittävä rinnakkaisuuden taso, sillä http-palvelimen päätehtävähän on asiakkaden palveleminen eikä lokin kirjoittaminen. Lopullisen toteutuksen rakenne on kuvattu tarkemmin luvussa 3.

# 3 RINNAKKAINEN TOTEUTUS

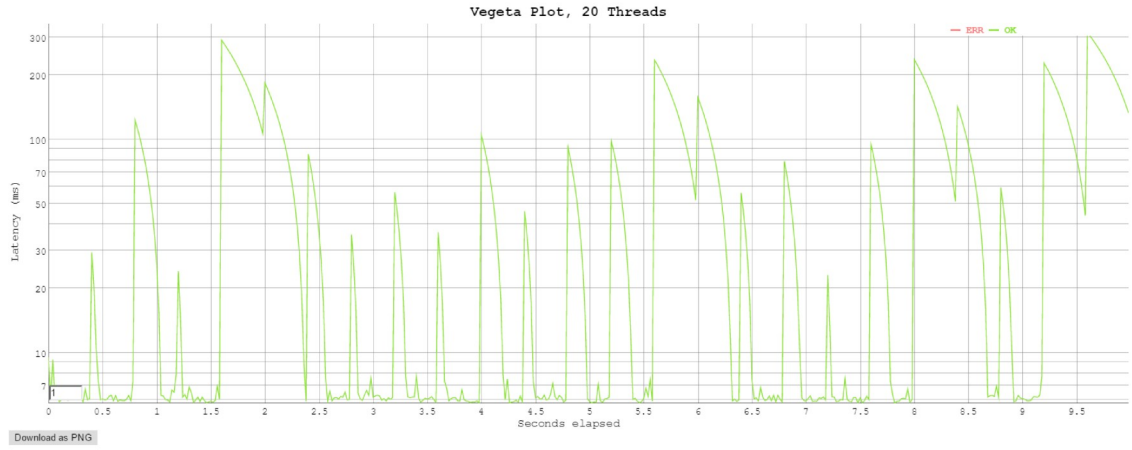
Tässä kappaleessa kuvataan ohjelman kehitystyön etenemistä, rinnakkaisuuden suunnittelua ja siihen liittyvää testaustyötä.

Rinnakkaisuusratkaisuksi valittiin toteutus, jossa 100 säiettä suorittaa rinnakkain palvelinoperaatioita, ja lokitiedoston kirjoittaminen tapahtuu sekventiaalisesti, kun 100 http-kyselyä on suoritettu. Toisin sanoen, asiakkaiden palveleminen keskeytyy pieneksi hetkeksi aina sadan asiakaskyselyn välein. Tämä on kuitenkin hyväksyttävää, koska alkuperäisessä sarjatoteutuksessa lähes vastaava viive tapahtui jokaisen asiakaskyselyn jälkeen. Näin tärkeimmät operaatiot, eli asiakkaiden palveleminen tapahtuu pienellä viiveellä, koska asiakkaita palvelevia säikeitä on 100, ja niitä suoritetaan rinnakkain. Tämä ratkaisu antaa riittään hyvän rinnakkaisuusasteen, joskin tiedostimme, että ratkaisu ei ole paras mahdollinen.

Yksi säie, joka palvelee aseiakkaita, suorittaa sekventiaalisesti seuraavat operaatiot: yhden http-yhteyden haku käsittelyyn, vastauksen rakentaminen ja vastauksen lähettäminen asiakkaalle kirjaston avulla. Nämä operaatiot on toteutettu sekventiaalisena koodina, koska ne tulee suorittaa juuri tässä järjestyksessä. Operaatioiden jakaminen eri säikeisiin olisi tuonut vain ylimääräisiä synkronointiongelmia, eikä se olisi nopeuttanut suoritusta mitenkään.

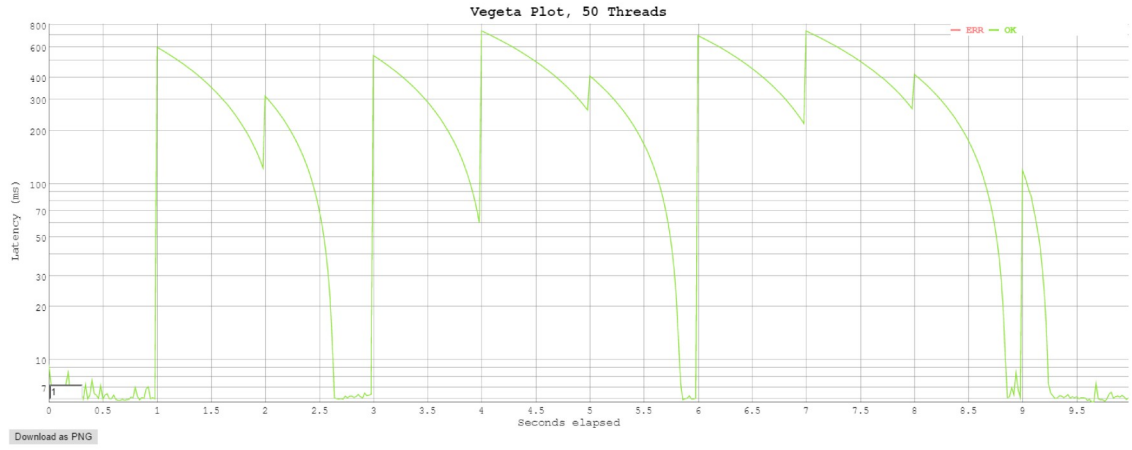
## 3.1 Säikeiden optimaalinen määrä

Ohjelman rinnakkaista suoriutumista testattiin VEGETAn avulla. Testeissä käytettiin 20, 50 ja 100 säiettä. 20 säikeellä (Kuva 2) ohjelman suorituskyky oli keskinkertainen. Pahimmillaan latenssit olivat 200 – 300 ms luokkaa, ja huomattava osa latensseista oli 30 – 100 ms välillä.



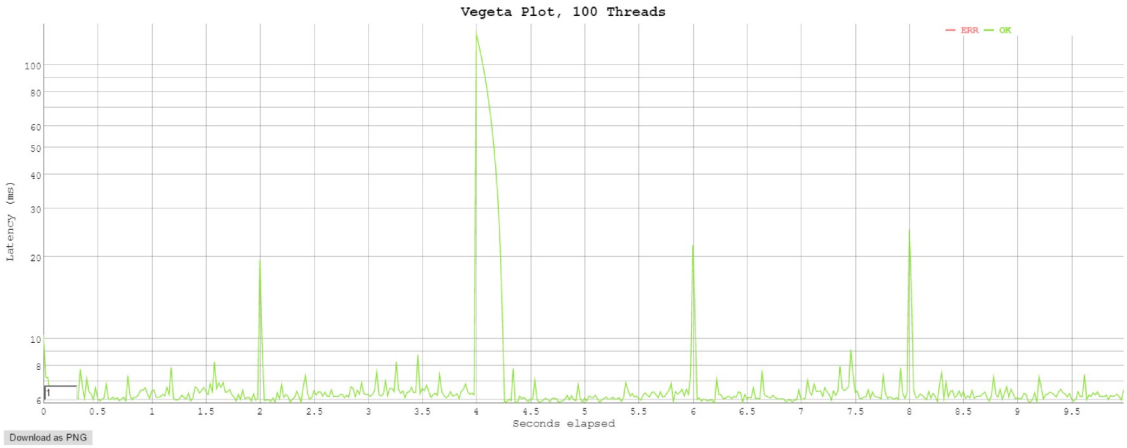

*Kuva 2: Suorituskyky 20 säikeellä*

50 säikeellä ohjelman suorituskyky näytti olevan entistäkin heikompi (Kuva 3). Pahimmillaan latenssit olivat 700 ms luokkaa, ja suuri osa latensseista oli yli 200 ms. Ohjelman suoriutuminen oli siis verrattain heikkoa.



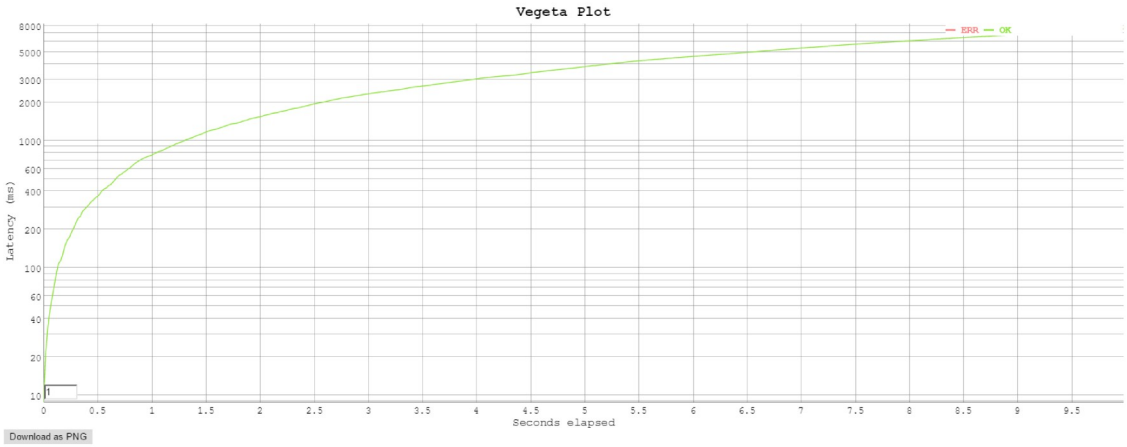

*Kuva 3: Suorituskyky 50 säikeellä*

100 säikeellä ohjelma toimi jouhevasti ja ilman suuria yllätyksiä (Kuva 4). Ainoastaan kerran latenssi nousi yli 100 millisekuntiin, ja 20 ms piikkejä oli vain kolme. Muuten vasteajat olivat pitkälti 6 – 8 ms.




*Kuva 4: Suorituskyky 100 säikeellä*

Vertailun vuoksi myös alkuperäisen sarjallisen ohjelman suorituskyky testattiin VEGETAlla (Kuva 5). Testin perusteella huomattiin, että kaikki rinnakkaiset toteutukset ovat huomattavasti alkuperäistä sarjallista toteutusta nopeampia. Sarjallisella toteutuksella ohjelman toiminta hidastuu koko ajan. Vaikuttaisi, että palvelin ei pysy ollenkaan asiakaspyyntöjen tahdissa.




*Kuva 5: Sarjallisen toteutuksen suorituskyky*

# 4 KEHITYSKOHTEITA

Ohjelman toteutuksessa päädyttiin suorittamaan lokitiedostoon kirjoitus sarjassa palvelinoperaatioiden kanssa. Ratkaisuun päädyttiin täysin rinnakkaisessa toteutuksessa vastaantulleiden haasteiden johdosta. Lokitiedoston kirjoitus olisi kuitnekin mahdollista toteuttaa rinnakkain varsinaisten palvelinoperaatioiden kanssa, jolloin palvelin pystyisi palvelemaan asiakkaita myös lokitiedoston kirjoituksen aikana. Tällä vältyttäisiin pieneltä viiveeltä, joka nyt koetaan aina sadan asiakaspyynnön välein.

Myös säikeiden optimaalista lukumäärää saatetaan joutua muuttamaan, kun lokitiedoston kirjoittaminen muutetaan rinnakkaiseksi operaatioksi. Tässä vaiheesa ei kuitenkaan voida sanoa, pitäisikö säikeiden lukumäärää kasvattaa tai pienentää.