jamk.fi

Graphite

Reaaliajassa skaalautuva piirtäminen

Samuli Heinovirta

Opinnäytetyö

Tammikuu 2015

Tietoliikennetekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Heinovirta, Samuli	Opinnäytetyö	21.01.2015
	Sivumäärä 93 + 27	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: (x)

Työn nimi

Graphite

Reaaliajassa skaalautuva piirtäminen

Koulutusohjelma

Tietotekniikan (Tietoverkkotekniikan) koulutusohjelma

Työn ohjaaja(t)

Piispanen, Juha

Rantonen, Mika

Toimeksiantaja(t)

Music.Info Finland Oy

Jokipii, Antti

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Music.Info Oy (Musicinfo), joka on perustettu 2012 ja sen tavoitteena on olla musiikin kattavin hakukone. Opinnäytetyön tavoitteena oli toteuttaa toimeksiantajan pyytämä Graphite - valvontajärjestelmän toimeenpano ja tehdä siitä kattava selostus. Työssä luotiin kattava valvontajärjestelmä toimeksiantajan tarpeiden mukaisesti.

Opinnäytetyö toteutettiin toimeksiantajan tuotantojärjestelmään ja kaikki työssä käytetyt ohjelmistot olivat avoimen lähdekoodin ohjelmistoja. Järjestelmä koostui kolmesta palvelimesta ympäri Eurooppaa. Palvelimet jaoteltiin yhdeksi isäntä palvelimeksi ja kahdeksi asiakaspalvelimeksi. Työssä keskityttiin toteuttamaan isäntä palvelimelle Graphite - valvontajärjestelmän perustoiminnot ja sen vaatimat riippuvuudet. Kahdelle asiakaspalvelimelle toteutettiin asiakassovellukset keräämään tietoa ja lähettämään kerätty tieto isäntä palvelimelle.

Työssä toteutettiin valvontajärjestelmän rinnalle hälytysjärjestelmä. Graphite - valvontajärjestelmästä poimittiin arvoja hälytysjärjestelmään. Hälytysjärjestelmän tehtävänä oli lähettää hälytyksiä määriteltyjen arvojen perusteella.

Lopputuloksena oli kattava valvontajärjestelmä toimeksiantajan tarpeiden mukaisesti. Toimeksiannosta luotiin kattava dokumentaatio. Järjestelmällä voidaan ennakoida tilanteita, jotka ovat vaaraksi tuotannolle, ja sitä voidaan käyttää parantamaan tuotantojärjestelmän heikkoja kohtia. Tulevaisuuden tarpeet otettiin huomioon, joten valvontajärjestelmä on toimeksiantajan ympäristöön skaalattavissa.

Avainsanat (asiasanat)

Graphite, valvonta, monitorointi, Carbon, varastointijärjestelmä, Cabot, hälytysjärjestelmä

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Heinovirta, Samuli	Bachelor's thesis	21.01.2015
		Language of publication: Finnish
	Number of pages 93 + 27	Permission for web publication: (x)

Title of publication

Graphite

Scalable Real-time Graphing

Degree programme

Information Technology

Tutor(s)

Piispanen, Juha

Rantonen, Mika

Assigned by

Music.Info Finland Oy

Jokipii, Antti

Abstract

This bachelor's thesis was assigned by Music.Info Oy (Musicinfo). Musicinfo was established in 2012 and its main target is to be the most comprehensive music search engine. The aim of this thesis was to implement the Graphite comprehensive monitoring system to a client's system depending on their needs.

The thesis was implemented directly into the client's production system and all software used in this thesis were open source. The system consisted of three server around Europe. The servers were divided into one host server and two client server. The Graphite's basic functions and its dependencies were implemented into the host server. The two client servers were implemented with client applications to collect information and to send the collected information to the host server. In this study alongside the Graphite monitoring system an alarm system called Cabot was implemented, the task of which was to monitor the defined values and make alarms from them.

The end result was a comprehensive monitoring system for the client's need and extensive documentation was made of the system, which can be used to predict a situation dangerous for production. It can also be used to improve the weak points in the system. Future needs were taken into account; thus, the monitoring system is scalable in the client's environment.

Keywords/tags (subjects)

Graphite, surveillance, monitoring, Carbon, storage system, Cabot, alarm system

Miscellaneous

SISÄLTÖ

KUVIOT

LYHENTEET JA SELITYKSET

AES-256	*Advanced Encryption Standard*, on verkkotyökalu tekstin salaamiseen ja purkamiseen käyttämällä AES – salausalgoritmia. voidaan valita 128-, 192-, tai 256 - bittinen avain salaukseen ja purkamiseen.
Arachnys	Perustettu nykyisten sijoittajien toimesta, ja siellä työskentelee kielitieteilijöitä, riskianalyytikkoja ja teknologiaspesialisteja. Arachnys käyttää vahvaa ihmisten osaamista ja uusinta teknologiaa monitoroidakseen ja vastaanottaakseen kriittisiä yritystietoja.
C	Ohjelmointikieli tietotekniikassa.
CDP	*Cisco Discovery Protocol*, on Cisco Systems:in kehittämä siirtoyhteyskerros protokolla
CLI	*Command Line Interface*, komentorivi käyttöliittymä.
DNS	Domain Name Service, Internet palvelu, joka kääntää domainien nimiä IP - osoitteiksi. Domainien nimet ovat luettavia verkkosivujen osoitteita.
Ethernet	Lähiverkon (LAN) arkkitehtuuri. *Ethernet* on määritetty palvelemaan IEEE 802.3 standardia, joka määrittää fyysisen ja alemman ohjelmistotason.
HMAC-SHA-256	Laskee *Hash-based Message Authentication Code*, sotkentaan perustuvan viestin autentikoinnin koodin käyttämällä SHA256 sotkenta funktiota.
HTTP	*Hypertext Transfer Protocol*, on sovellusprotokolla jaettuun, yhteistyöhypermediainformaatiojärjelmään. HTTP on WWW-tietoliikenteen perusta.
ICMP	*Internet Control Message Protocol* viestejä lähetetään useissa tilanteissa, esimerkiksi silloin, kun tietosähke ei tavoita sen kohdetta, portilla ei ole

	puskurointikapasiteettia lähettämään tietosähkettä ja portti voi ohjata isäntää lähettämään liikennettä lyhempää reittiä
IDE	*Integrated Drive Electronics*, on standardi elektroniselle rajapinnalle, emolevyn ja tietopolun tai ajurin välillä ja tietokoneen tallennuslevyn välillä.
IP kehys	Tietoliikenteessä kehys on tieto, joka lähetetään verkkopisteiden välillä täydellisenä yksikkönä osoitteella ja vaadittavalla hallintaprotokollatiedolla.
IP	*Internet Protocol*, IP – osoite on uniikki osoite tietokoneille tunnistaakseen itsensä ja kommunikoidakseen muiden Internet Protokollan laitteiden kanssa.
IPv4	*Internet Protocol version 4*, Internet Protokolla versio 4 on neljäs IP tarkistus, jota käytetään laitteiden tunnistamisessa verkossa osoitejärjestelmän kautta.
IPv6	*Internet Protocol version 6*, Internet Protokolla versio 6 on uusin versio IP:sta.
IT	*Information Technology*, Informaatiotegnologia
Jenkins	Avoimen lähdekoodin jatkuvan integraation työkalu. Se on kirjoitettu Java - ohjelmointikielellä testaamaan ja raportoimaan irrallisia muutoksia suuremmassa koodipohjassa reaaliaikaisesti.
JSON	*JavaScript Object Notation*, antaa luettavan kokoelman tiedosta, johon voidaan päästä käsiksi erittäin loogisella tavalla.
Kernel	*Kernel* on ohjelma, joka muodostaa keskeisen ytimen tietokoneen käyttöjärjestelmässä. Sillä on täydellinen hallinta kaikesta, mitä tapahtuu järjestelmässä.

LAN	*Local Area Network*, Lähiverkko, joka yhdistää keskenään tietokoneita rajatussa alueessa, kuten kotona, tietokonelaboratorioissa, tai toimistotiloissa.
Linux	Linux on yksinkertaisesti sanottuna käyttöjärjestelmä. Se on ohjelmisto, joka mahdollistaa sovelluksien ja tietokoneen käyttäjän suorittamaan laitteella haluttuja funktioita. Linux on erittäin samankaltainen käyttöjärjestelmä kuin Windows ja OS X
Loopback	Tietoliikenteessä *loopback* on tapa, jolla suoritetaan linjan siirron testejä kytkentäkeskuksessa. *Loopback* on tietoliikenne kanava ainoastaan yhdellä päätepisteellä.
Macintosh	*Macintosh*, useimmin kutsutaan Maciksi, joka oli ensimmäinen tietokone graafisella käyttöliittymällä ja hiirellä.
MySQL	Avoimen lähdekoodin omainen tietokannan hallinnointijärjestelmä, joka perustuu strukturoituun tiedustelukieleen, *Structured Query Language* (SQL)
Net-SNMP	Tarjoaa työkalut ja kirjastot liittyen SNMP:hen sisältäen: laajennettavan agentin, SNMP - kirjaston, työkalun pyytää tai asettaa tietoa SNMP agenteista, työkalut luomaan tai käsittelemään SNMP trapseja jne.
Oracle	*Oracle* tietokanta kerätylle tiedolle, jota käsitellään yksikkönä. Tietokannan tarkoitus on varastoida ja noutaa yhteenkuuluvaan tietoa.
Orbitz	Matkustus teollisuuden johtavia nimiä verkossa. Käyttää innovatiivista teknologiaa ja tekemällä siitä merkityksellistä asiakkailleen.
OS X	Tunnetaan paremmin *Mac OS X*, se on Unix pohjainen graafinen käyttöliittymä käyttöjärjestelmä, jonka on suunnitellut ja markkinoinut Apple.

PDP *Packet Data Protocol*, verkkoprotokolla, jota käytetään pakettikytkentäisten verkkojen keskustelemaan GPRS verkkojen kanssa.

Perl *Perl* on ohjelmointikieli, jota voidaan käyttää monenlaisissa tehtävissä.

PHP *PHP: Hypertext Preprocessor*, avoimen lähdekoodin, palvelinpään, HTML sulautettujen koodauskieli, jota käytetään dynaamisien verkkosivujen tekemiseen.

PID *Process identification number*, automaattisesti määrätään jokaiselle prosessille sitä luotaessa, Unix käyttöjärjestelmissä.

Ping *Ping* on käytännössä perus Internet ohjelma, joka mahdollistaa käyttäjän varmistaa, että tieytty IP -osoite on olemassa ja hyväksyy pyyntöjä. *Pingiä* käytetään varmistamaan, että isäntä tietokone, jota käyttäjä yrittää tavoittaa, todella on toiminnassa.

Pip *Pip* komento on työkalu Pythonin pakettien asentamiseen ja hallitsemiseen.

POSIX *Portable Operating System Interface*, on perhe standardeja, jotka on määritelty IEEE:n toimesta, ylläpitämään yhteensopivuutta käyttöjärjestelmien välillä.

PostgreSQL *PostgreSQL* on yleis- ja objektirelaatiotietokannan tietokanta hallintajärjestelmä.

Python GIL CPythonissa *Global Interpreter Lock*, on muteksi, joka estää Pythonin useiden tavukoodi lankojen suorittamisen samanaikaisesti

RAD *Rapid Application Development*, on käsite, että tuotteen voidaan kehittää nopeasti ja korkeamman laadun kautta.

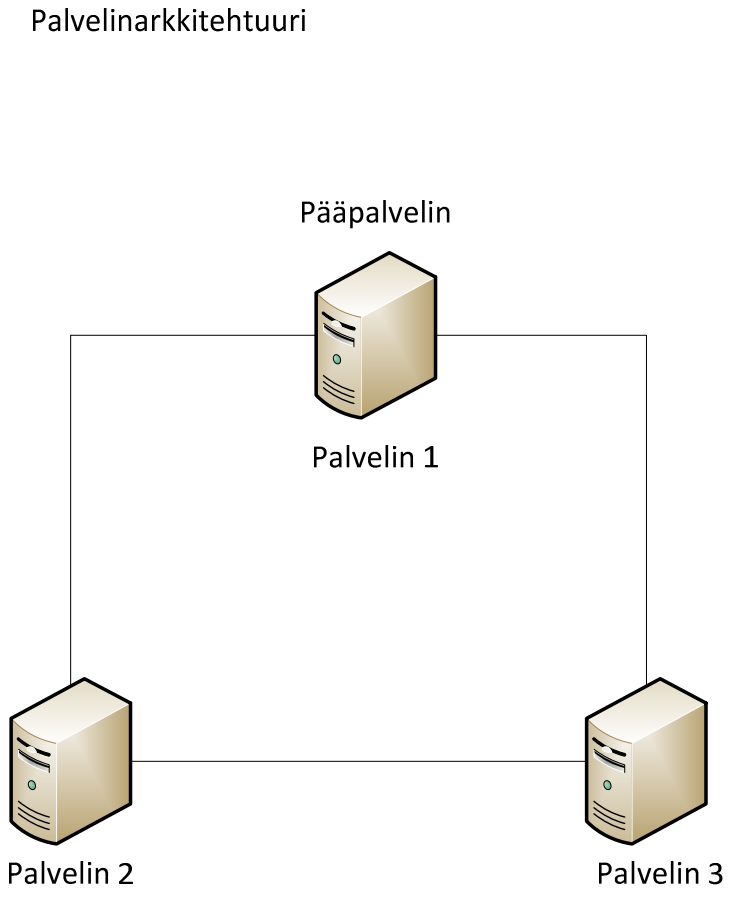
RAM	*Random Access Memory*, tietokoneen muisti, johon päästään käsiksi satunnaisesti. RAM on kaikista yleisin muistin tyyppi, joka löytyy tietokoneista ja muista laitteista.
RRA	*Round-Robin Archive*, sisältää rajoitetun määrän konsolidoitua tietoa.
SCSI	*Small Computer System Interface*, pienen tietokoneen järjestelmän rajapinta, sarja yhdensuuntaisia käyttöliittymä standardeja
SHA-1	*Secure Hash Algorithm 1*, on sulautettu viesti algoritmi, joka ottaa minkä tahansa pituisen viestin <2^64 bittiä ja tuottaa 160 bittisen ulostulon.
stderr	Standard error, standardivirhe viittaa tietovirroissa virhe viestien luontiin prosessissa.
stdout	Standard out, viittaa tietovirtaan, joka tulee käynnissä olevasta prosessista.
Sähköposti	Sähköposti (E-mail, maili) on tietokoneelle tallennettujen viestien vaihtamista tietoliikenteessä. Sähköpostiviestit ovat yleensä koodattu ASCII tekstillä.
TCP	*Transmission Control Protocol*, on standardi, joka määrittää kuinka muodostetaan ja ylläpidetään verkkokeskustelua, jonka kautta sovellusohjelmat voivat vaihtaa tietoa
Tcpdump	Pakettien analysaattori ohjelmisto toteutus, joka monitoroi ja lokittaa TCP/IP liikennettä verkon ja tietokoneen välillä, jota toteutetaan.
TTL	*Time To Live*, on mekanismi, joka rajoittaa tiedon elinikää tietokoneessa tai verkossa.
Twisted	Tapahtumapohjainen moottori, joka on kirjoitettu Pythonilla ja lisensoitu avoimen lähdekoodin alle.

UDP	*User Datagram Protocol*, on viestintäprotokolla, joka tarjoaa rajallisen määrän palvelua, silloin kun viestejä vaihdetaan tietokoneiden välillä verkossa, joka käyttää *Internet Protokollaa*.
UNIX	Unix on käyttöjärjestelmä, joka sai alkunsa Bell laboratoriossa 1969, interaktiivisena osa-aika järjestelmänä. Ritchie Dennistä ja Thompson Keniä pidetään Unixin luojina. Nimi perustuu aikaisempaan järjestelmään nimeltä Multics
URL	*Uniform Resource Locator*, ainutlaatuinen osoite tiedostoon, joka on saatavilla *Internetissä*.
WAN	*Wide Area Network*, tietokoneverkko, joka ulottuu suhteellisen laajalle maantieteelliselle alueelle. Tyypillisesti koostuu kahdesta tai useammasta lähiverkosta.
Windows	Henkilökohtainen tietokoneen käyttöjärjestelmä
Wireshark	Verkon pakettien analysaattori. Yrittää ottaa kiinni verkon paketteja ja yrittää esittää näiden pakettien tiedot niin yksinkertaisesti kuin mahdollista.
XML	*Extensible Markup Language*, on joustava tapa luoda tavallisen informaation formaatteja, ja jakaa molemmat muoto ja tieto *Internetissä*, sisäverkoissa ja muualla.

1 TYÖN LÄHTÖKOHDAT

1.1 Valvonnan tarve

Musicinfolla oli kolme palvelinta (ks. kuvio 1). Palvelimista kaksi sijaitsee ulkomailla ja yksi kotimaassa. Kesällä 2014 syntyi kriittinen tarve palvelimien valvonnalle. Kehityksen aikana huomattiin vakava virhe toiminnollisuudessa. Sen vaikutus ilmeni prosesseissa, jotka alkoivat tipahdella virheen aikana. Jälkikäteen selvisikin, että eräs prosessi oli nostanut muistin käyttöä ja lopulta ajanut palvelimen tilanteeseen, jossa prosesseja alkoi tipahdella pois. Tästä syntyi tarve toteuttaa valvonta valvomaan Musicinfon palvelimien tiloja ja toiminnollisuuksia.



Kuvio 1. Palvelinarkkitehtuuri

1.2 Music.Info Finland OY

Yritys on perustettu 2012, ja sen tavoitteena on olla musiikin kattavin hakukone. Musicinfon ainoa toimipiste sijaitsee Jyväskylässä. Musicinfon logo on esitettynä kuviossa 2. Toiminta on keskittynyt suurimalta osin Start-up - leipomolle, joka sijaitsee Jyväskylän Sepän aukiolla. (What is Musicinfo. n, d)

Kuvio 2. Musicinfon logo

Yritysidean ”isänä” ja yrityksen perustajana toimii Antti Jokipii, joka toimiessaan DJ:nä sai idean musiikinhakukoneesta. Yrityksen toimitusjohtajana toimii Kari Halttunen. Yrityksellä on myös toimihenkilöitä useassa työtehtävässä Musicinfon tarpeiden mukaisesti. (What is Musicinfo. n, d)

Musicinfon henkilöstöllä on kansainvälistä yhteistyötä ja kokemusta Venäjälle, Ranskaan ja Kiinaan. Tämä mahdollistaa hyvän kansainvälisen asiakaskunnan kehittymisen, mikä onkin Musicinfon yksi suurimmista haasteista. (What is Musicinfo. n, d)

1.3 Toimeksianto ja tavoitteet

Tavoitteena oli saada kattava selostus Graphiten, avoimen lähdekoodin valvontatyökalun toiminnasta ja sen konfiguraatiosta. Graphite on monen ohjelman yhdistetty järjestelmä. Jokainen Graphitin sisällä oleva ohjelma on avointa lähdekoodia.

Graphite on järjestelmä, jolla seurataan palvelimien suorituskykyä, muistin käyttöä, kovalevyjen tilan käyttöä jne. Sen avulla voidaan seurata useita palvelimia. Musicinfon palvelimilta seurattaisiin levyn käyttöä, muistin käyttöä, suorituskykyä ja järjestelmän kuormitusta.

Järjestelmä oli Musicinfon ympäristössä toteutettavissa. Työharjoittelun aikana oli musicinfon järjestelmän rakenne tullut tutuksi. Mahdollisuutena oli saada tietoa ja haasteita oikean tuotannon ympäristöstä. Mahdollisuus oli myös järjestelmän kehittämiseen opinnäytetyön aikana, sillä Graphite - järjestelmään voidaan integroida monia hyödyllisiä työkaluja Musicinfon tarpeiden mukaisesti.

2 TIETOPERUSTA

2.1 Palvelinvalvonta

2.1.1 Mitä on palvelinvalvonta?

Palvelimien valvonta määritellään proseksiksi, joka valvoo palvelimien järjestelmän resursseja, kuten CPU:n käyttöä, muistin kulutusta, I/O:ta, verkkoa, levyn käyttöä, prosesseja jne. Palvelimienvalvonta auttaa ymmärtämään järjestelmäresurssien käyttöä, mikä voi auttaa parempaan kapasiteettisuunnitteluun ja tarjoaa parempaa loppukäyttäjän kokemusta. Sovelluksien terveys riippuu suurelta osin niiden alla olevan palvelimen terveydestä. Juurikin tätä palvelimienvalvonta varmistaa, että palvelinkone on kykeneväinen isännöimään sovelluksia. Palvelimienvalvonta tarjoaa käyttöjärjestelmästä liittyviä tietoja, ja kun sitä käytetään muiden seurantatietojen yhteydessä, jota saadaan sovelluksilta, saadaan todellisuuden kuvaa siitä, mitä tapahtuu työskentelevässä järjestelmässä. (Server monitoring. N, d)

2.1.2 Miksi on hyvä valvoa?

Palvelimen suorituskyvyn vianmääritys on tärkein syy sille, miksi valvoa. Esimerkiksi sanotaan, että käyttäjillä on ongelmia yhdistää palvelimelle, ja tällöin on hyvä monitoroida palvelinta vianmäärityksen kannalta. Tällaisessa tapauksessa on hyvä käyttää saatavilla olevia monitorointiresursseja ja niitä käyttämällä ratkaista ongelma. (Stanek, W. 1999)

Toisena yleisenä syynä voidaan palvelimen monitoroinnilla pitää palvelimien suorituskyvyn parantamista. Parantamalla levyn I/O toimintoja, vähentämällä CPU:n käyttöä ja vähentämällä verkkoliikenteen kuormaa palvelimella. Resurssien käytössä on tehtävä useasti kuitenkin kompromisseja. Käyttäjien määrä palvelimella voi kasvaa huomattavan suureksi. Tällöin ei voida vaikuttaa verkon kuormaan, mutta voidaan kuormituksen tasapainotuksella parantaa palvelimen suorituskykyä. Tai jaetaan avain tiedot erillisille levyille. (Stanek, W. 1999)

2.2 I/O toiminnot

Lyhyesti sanottuna sisääntulo (*input*) / ulostulo (*output*). Termiä I/O käytetään kuvaamaan minkä tahansa ohjelman, operaation tai laitteen kuvaamiseen, joka lähettää tietoa tietokoneeseen tai tietokoneesta, ja oheislaitteeseen tai oheislaitteesta. Jokainen lähetys on ulostuloa yhdestä laitteesta ja sisääntuloa toiselle. Laitteet kuten näppäimistöt ja hiiret on vain sisääntulo laitteita, kun taas tulostimet ovat vain ulostulo laitteita. Kirjoitettava CD-levyke on kumpaankin suuntaan toimiva laite. (Beal, V. n, d)

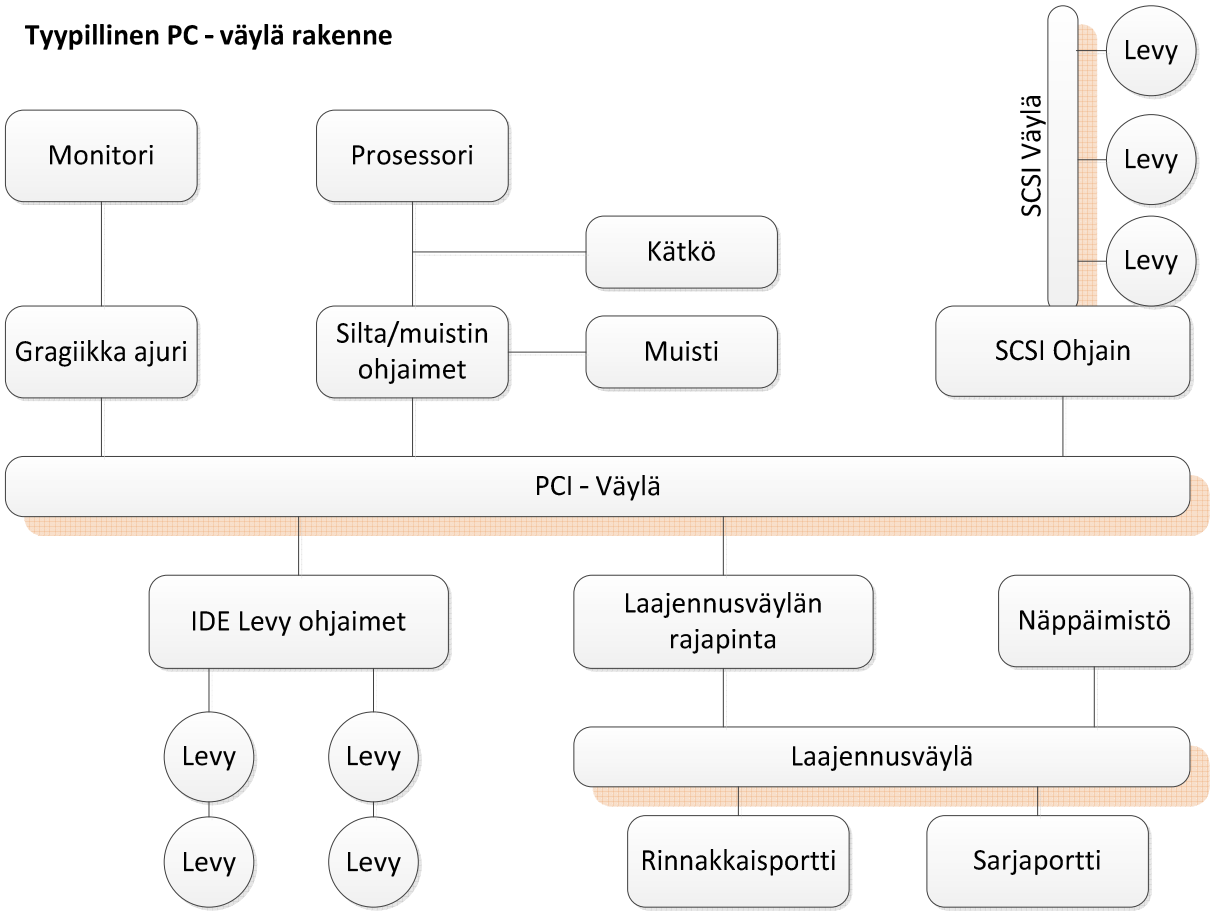
On olemassa sisään- ja ulostulo toimintoja joka puolella tietokoneen kontekstia (ks. kuvio 3). Tavallinen, yksinkertainen I/O laite sisältää hiiren, näppäimistön ja monitorin. (Janssen, C. n, d) (I/O Systems. n, d)

Kuvio 3 havainnollistaa kolme neljästä väylätyypistä, jotka löytyvät modernista PC:stä:

- PCI – väylä: yhdistää korkeannopeuksiset suuren kaistanleveyden laitteet muistin osajärjestelmään ja CPU:n.

- Laajennusväylä: yhdistää hitaamman pienemmän kaistanleveyden laitteet, jotka tyypillisesti kuljettavat puskuroinnilla tietoa yksi merkki kerrallaan.
- SCSI väylä: yhdistää SCSI laitteet yhteiseen SCSI ohjaimeen.
- Ketjutusväylä: ei näy kuviossa, käytetään silloin, kun laitteiden merkkijonot on yhdistetty toisiin aivan kuten ketjun holkit ja ainoastaan yksi laitteista on suoraan yhdistetty isäntään.

(Janssen, C. n, d) (I/O Systems. n, d)



Kuvio 3. PC – väylä

Seuraavat ovat merkittäviä I/O - termejä:

- I/O rajapinnat: nämä tarjoavat vuorovaikutuksen tietokoneen raudan kanssa. Jokaisen laitteen rajapinta on kykeneväinen koodaamaan ja purkamaan I/O - signaaleja ymmärrettävään muotoon sisään- ja tulolaitteille.

- Ohjelmoitavat I/O - sovellukset: monet sovellukset ovat integroituina käyttöjärjestelmiin tarjoten ajoaikaa sisään- ja ulostuloille samanaikaisesti. Parhaana esimerkkinä voidaan pitää C, C++ ja Java -ohjelmoituja sovelluksia, joilla on sisäänrakennettuja kirjastoja käytetyille I/O - operaatioille. Ohjelmat on kirjoitettu niin, että käytetään ainoastaan yhtä kirjastotiedostoa sisääntulona samanaikaisesti, kun näytetään ulostuloa käyttäjälle. Tätä ohjelmointikäsitettä kutsutaan tiedoston käsittelyksi.

- Muistinosoitus - I/O: tietokoneen muisti sisältää blokkeja, joihin varastoidaan sovellukset/prosessit käsittelyyn. Monia osoitusmekanismeja käytetään tähän tarkoitukseen; jokainen käyttää I/O - operaatioita jossakin yhteydessä. Muistinosoituksen käyttäen muistin I/O - operaatioita, ovat indeksoituja osoituksia ja välitöntä osoitteistusta.

(Janssen, C. n, d) (I/O Systems. n, d)

2.3 CPU

CPU (central processing unit) on tietokoneen raudallinen keskusyksikkö, joka kuljettaa tietokoneohjelman ohjeita. Se suorittaa perinteisiä laskennollisia, loogisia ja ulos-/sisääntulo-operaatioita tietokonejärjestelmässä. CPU:ta voitaisiin pitää tietokoneen aivoina: jokaisen ohjeen, oli se miten yksinkertainen tahansa, on kuljettava CPU:n lävitse. CPU:ta kutsutaan myös keskusprosessoriyksiköksi tai lyhyesti prosessoriksi, joten tietokeen teknisiä tietoja luettaessa tyypillisesti CPU viittaa prosessoriin. (CPU. 2014) (Zandbergen, P. n, d)

Mikäli tutkitaan CPU:n eri osia ja niiden toiminnollisuuksia, on muistettava, että kaikessa kyseessä on nopeus. Tietokonetta käytettäessä kaikki ohjeet kuljetetaan erittäin nopeasti. Näiden ohjeiden vaatimusten kasvaessa, kuten esimerkiksi 3D animointi ja videon editointi, vaaditaan CPU:lta enemmän. Kuitenkin teknologisia lähestymisiä prosessoriteknologiassa on ajanut nopeuden tarve. (Zandbergen, P. n, d)

Komponentit

Tyypillisellä CPU:lla on useita komponentteja. Ensimmäisenä on ALU (the arithmetic logic unit) aritmeettislooginen yksikkö, joka suorittaa yksinkertaisia aritmeettisia ja loogisia operaatioita. Toisena on CU (the control unit) ohjausyksikkö, joka hallinnoi useita tietokoneen komponentteja. Se lukee ja tulkitsee ohjeita muistista ja muuntaa ne joukoksi signaaleja aktivoidakseen tietokeen muita osia. Ohjausyksikkö kehottaa aritmeettisloogista yksikköä suorittamaan vaadittavat laskennat. Kolmas on kätkö, joka palvelee korkea nopeuksisena muistina, minne ohjeet voidaan kopioida ja hakea. (CPU. 2014) (Zandbergen, P. n, d)

2.4 RRD

RRD (Round Robin Database) on erikoistunut varastointijärjestelmä. Se mahdollistaa suurien ajanjaksollista tietoa sisältävien tietojen varastoinnin, kuten lämpötilojen ja verkon kaistankäytön seurantaa, jatkuvalla levyn kirjoittamisella. Se tekee tämän käyttämällä hyväkseen tarkkuuden jatkuvia tarpeita. Lyhyesti sanottuna jokainen tietopiste on merkittävä. Halutaan tarkkaa kuvaa jokaisesta tapahtumasta, joka on esimerkiksi tapahtunut viimeisen 24 tunnin aikana sisältäen pienimmätkin tilapäiset piikit verkonkaistassa tai levyn käytössä. Kuitenkin on muistettava, että pitkällä tähtäimellä yleisemmät trendit ovat välttämättömiä. (Yu, J. 2010)

2.5 Daemon

Daemon (Disk and Execution Monitor) on eräänlainen ohjelma Unix käyttöjärjestelmille, joka suoriutuu taustalla, pikemmin kuin suoraan käyttäjän ohjattavana. Se odottaa aktivointia tietyn tapahtuman tai ehdon esiintymisestä. Unix järjestelmissä tyypillisesti ajetaan useita Daemoneja, jotka vastailevat muihin ohjelmiin ja laitteistojen toimintaan. Esimerkkinä voidaan tapahtumasta tai ehdosta, joka laukaisee Damoneja aktiiviseksi, pitää:

- Määritelty aika tai päivä
- Tietyn aikavälin saavuttaminen
- Tiedoston saapuminen tiettyyn hakemistoon
- Sähköpostin saaminen
- Verkkopyyntö, tietyn kommunikaatio linjan kautta.

Ei ole välttämätöntä, että suoritettava tapahtuma tai ehto on tietoinen siitä, että Daemon kuuntelee. Vaikka ohjelmat usein suorittavat tapahtuman vain siksi, että ne ovat epäsuorasti tietoisia siitä, että ne herättävät Daemonin. (Daemon Definition. 2005)

Daemonit ilmennetään yleensä prosesseiksi. Prosessi on esimerkiksi ohjelman suorittaminen. Prosesseja hallinnoidaan kernelillä, joka määrää jokaiselle prosessille prosessin tunnistenumeron (PID). Linuxissa on kolme perustyyppistä prosessia; interaktiivinen, erä ja Daemon. Interaktiiviset prosessit suoritetaan vuorovaikutteisesti käyttäjän toimesta komentoriviltä. Erä prosessit suoritetaan prosessi jonosta ja eivät liity komentoriviin; ne ovat hyviä suorittamaan toistettavia tehtäviä silloin, kun järjestelmän käyttö on alhaista. (Daemon Definition. 2005)

Daemonit tunnistetaan järjestelmän toimesta aivan kuten mikä tahansa prosessi, jonka äitiprosessilla on PID arvo yksi, joka aina edustaa prosessin inittia. Init on aina ensimmäi-

nen prosessi, joka käynnistetään aina, kun Linux tietokone käynnistetään ja pysyy järjestelmässä niin kauan kunnes järjestelmä sammutetaan. Init adoptoi minkä tahansa prosessin, jonka äitiprosessi kuolee ilman lapsiprosessin tilan odottamista. Näin, yleinen menetelmä Daemonin käynnistämiseksi sisältää forkkaamisen yhdesti tai kahdesti, ja äitiprosessin kuolettaminen samalla, kun lapsiprosessi alkaa suorittamaan sen normaalia toimintaa. (Daemon Definition. 2005)

Monissa Unix käyttöjärjestelmissä, mukaan lukien Linux, jokaisella Daemonilla on yksi kirjoitus, *scripti,* lyhyt ohjelma. Kirjoitus voidaan lopettaa, uudelleen käynnistää tai tarkastella sen tilaa. Näiden kirjoitusten hallinnointi perustuu suoritustasoihin. Suoritustaso on konfiguraatio tai järjestelmän operaatio tila, joka sallii vain valittujen prosessien olemassaolon. Käynnistäessä eri suoritustasoille, voi auttaa ratkaisemaan tiettyjä ongelmia, mukaan lukien järjestelmän virheet. (Daemon Definition. 2005)

2.6 Avoin lähdekoodi

2.6.1 Yleistä

Useimmat ohjelmistot, jotka ladataan tai ostetaan, tulevat koottuna valmiina pakettina. Koottuna tarkoitetaan sitä, että itse luojan tekemä koodi, tunnetaan myös lähdekoodina, käytetään ohjelmiston lävitse, jota voidaan kutsua kääntäjäksi. Kääntäjä kääntää lähdekoodin sellaiseen muotoon, että tietokone sen ymmärtää. Pidetään erityisen vaikeana muokata valmista ohjelmistoa, joka on käännetty kääntäjällä. Tästä ohjelmistoversioista ei mahdollista nähdä aivan alkuperäistä lähdekoodia. Useimmat valmistajat pitävätkin tätä etuutena luodessaan ohjelmistoja. Niitä on vaikea kopioida juuri tästä syystä ja tuovat tietynlaista korkeaa laatua heidän tuottamiinsa ohjelmistoihin. (What does open source mean? 2000)

2.6.2 Mitä on avoin lähdekoodi

Avoimen lähdekoodin ohjelmisto on aivan eri puolella edellä mainittua näkökulmaa. Yleisesti avoin lähdekoodi viittaa ohjelmistoon, jonka lähdekoodi on vapaasti jaettuna kaikille käytettäväksi. Sekä on vapaasti muokattavissa sen alkuperäisestä muodosta, ilman minkäänlaisen maksun perimistä sen käytöstä. Se onkin tyypillisesti yhteisöllisesti tuotettu, jossa ohjelmoijat parantavat sen koodia ja jakavat muutokset yhteisön sisällä. Näihin muunnoksiin kannustetaankin useasti yhteisöjen sisällä. Avointa lähdekoodia tukevat ohjelmistojen kehittäjät uskovat siihen, että antamalla kenen tahansa kiinnostuneen muokata lähdekoodia johtavan pitkällä tähtäimellä lopulta virheettömään ja enemmän käytännöllisempään koodiin. (Open Source. n, d) (What does open source mean? 2000)

Jotta ohjelmistoa pidettäisiin ohjelmiston tuotannon kannalta avoimena lähdekoodina, sen tulee täyttää määritellyt kriteerit:

- Ohjelmiston täytyy olla vapaasti jaettavissa
- Lähdekoodi pitää olla sisällytettynä
- Kenen tahansa on lupa muokata lähdekoodia
- Muokatut versiot on vapaasti jaettavissa
- Lisenssin ei tule vaatia mitään ulkopuolista ohjelmistoa tai kohdata minkään muun ohjelmiston kanssa.

(What does open source mean? 2000)

2.6.3 Miksi suositaan enemmän avointa lähdekoodia

Avointa lähdekoodia käytetään siksi, että sitä käyttämällä on suurempi hallinnallinen merkitys. Kaikki on käytännössä täysin vapaasti hallittavissa. Koodia voidaan tarkkailla ja poistaa tai jättää käyttämättä osioita joita ei varmasti haluta käyttää. Lisäksi voidaan vapaasti muokata joitakin osioita mielivaltaisesti, mikäli siihen osioon ei olla tyytyväisiä.

Myös käyttäjät, jotka eivät ole ohjelmoijia, voivat hyötyä vapaan lähdekoodin ohjelmistoista, koska voivat vapaasti käyttää ohjelmistoja mielensä mukaan. Jotkin käyttäjät suosivat avointa lähdekoodia, koska ne auttavat heitä tulemaan paremmiksi koodaajiksi. Esimerkiksi opiskelijat voivat hyödyntää koodia heidän tarpeisiinsa, sekä lukea koodia kehittääkseen itseään paremmaksi. (What is open source? n, d)

Joidenkin mielestä avoin lähdekoodi on paljon turvallisempi ja vakaampi käyttää, kuin mitä jonkin yhtiön omistava ohjelmisto. Suurempana syynä sitä, että kuka tahansa voi katsella ja muokata avointa lähdekoodia ja huomata virheet, jotka alkuperäiseltä suunnittelijalta on mahdollisesti jäänyt huomaamatta. Sekä, kun ohjelmoijat voivat vapaasti työskennellä jonkin osan kanssa ilman luojan lupaa, on ohjelmisto yleensä korjattua ja päivitettyä nopeasti. (What is open source? n, d)

3 GRAPHITE

3.1 Yleistä

Graphite on yritystason seurantatyökalu ja sen ideana on sen pyöriminen hyvin halvemmallakin laitteistolla. Sen on suunnitellut Chris Davis vuonna 2006. Se oli kehittäjällä sivuprojektina, josta lopulta kasvoikin perustettava seurantatyökalu. Graphiten kaksi päätoimintoa ovat numeerisen aikasarja tiedon varastointi ja tämän tiedon muuttaminen kuvaajiksi vaatimusten mukaisesti. Orbitz salli Graphiten julkaistavaksi vapaanlähdekoodin alle, Apache 2.0 lisenssillä vuonna 2008. Chris jatkoi työtä Graphiten parissa ja käyttänyt sitä monien yritysten järjestelmien valvomiseen. Tästä syystä monet suuret yritykset käyttävätkin sitä. (Davis, C. 2011.)

Graphite itsessään on siis reaaliajassa skaalautuva piirtävä järjestelmä, kuitenkin se on hieman toiminta-alueeltaan poissa oleva sovellus. Erityisesti kun se on suunniteltu käsittelemään suuria määriä aikajaksollista tietoa. Esimerkkinä voitaisiin pitää sitä, että Graphite – ohjelmisto olisi hyvä piirtämään osakemarkkinoita, koska ne ovat numeroita jotka vaihtelevat jatkuvasti. (FAQ. 2009.)

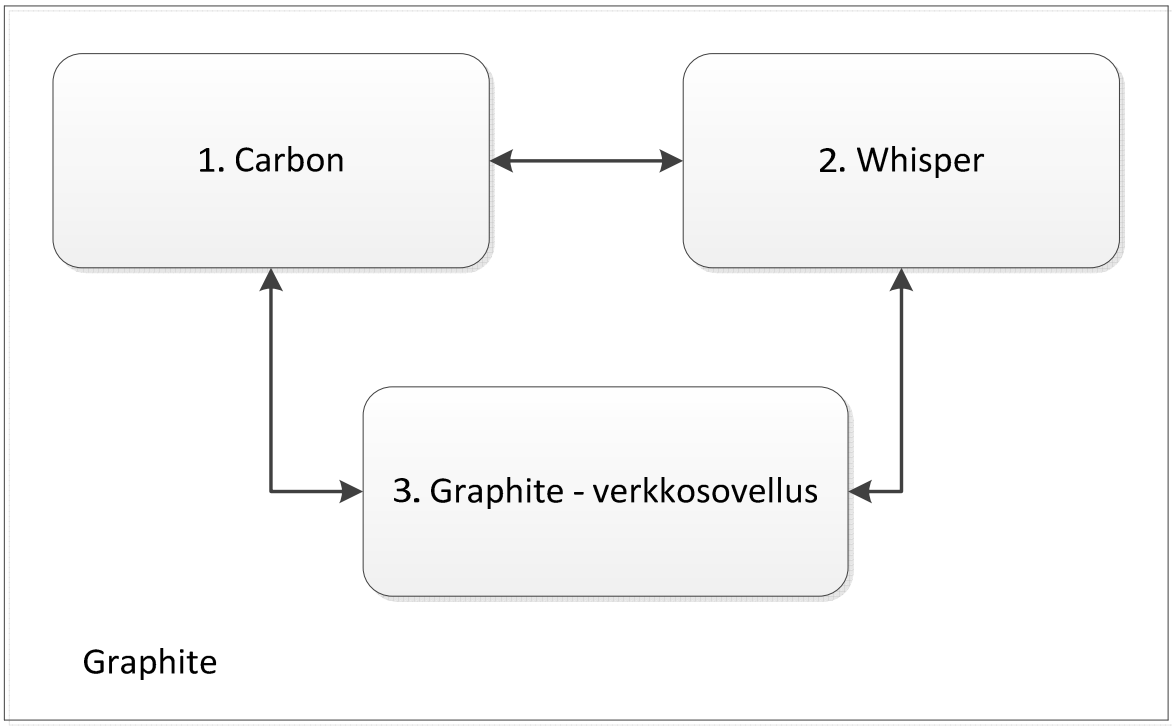
Kuitenkin Graphite – ohjelmisto on monimutkainen järjestelmä, ja jos halutaan vain seurata muutamaa sataa erillistä asiaa, tällöin Graphite – ohjelmisto pienemmissä järjestelmissä saattaa olla liian monimutkainen toteutus. Kuitenkin jos on tarvetta seurata ja luoda kaavioita piirtämällä useista eri asioista, joiden nimiä ei välttämättä ennestään tiedetä, on Graphite – ohjelmisto sopiva vaihtoehto. Yksi yleisempiä käytänteitä Graphite ohjelmistolla on juurikin verkkopohjaisten valvontapaneelien luonti seurannalle ja analysoinnille. (FAQ. 2009.) (Davis, C. n,d)

3.2 Arkkitehtuuri

Yksinkertaisesti sanottuna Graphitin arkkitehtuuri (ks. kuvio 4) koostuu kolmesta ohjelmallisesta pääkomponentista.

- Ensimmäisenä komponenttina pidetään Carbon – ohjelmistoa. Se on Twisted Daemon, joka kuuntelee aikasarjallista tietoa.
- Toisena komponenttina on Whisper - ohjelmisto. Se on yksinkertainen tietokanta kirjasto aikasarjallisen tiedon varastointiin.
- Kolmantena komponenttina tulee Graphite – verkkosovellus. Se on Djangon verkkosovellus, joka tekee vaatimusten mukaisia kuvaajia käyttäen Cairo – ohjelmistoa.

(Davis, C. 2011.)



Kuvio 4. Arkkitehtuuri

3.3 Toiminta

Graphite suorittaa kaksi erittäin yksinkertaista toimintoa. Varastoimalla statistiikkaa, jotka vaihtelevat ajan mittaan ja piirtää niistä kaavioita. On luotu useita ohjelmistoja aikojen saatossa, jotka tekevät samat toiminnot. Kuitenkin se mikä Graphitissa on ainutlaatuista, on se, että tarjoaa toiminnollisuuden verkkopalveluna, joka on helppo käyttää ja erittäin skaalattavissa. (Davis, C. n,d)

Protokolla, joka syöttää tietoa Graphiteen, on niin yksinkertainen, että sen oppisi tekemään käsin muutamassa minuutissa. Kaavioiden mallinnus ja tietopisteiden hakeminen olisi niin helppoa kuin mitä URL:n hakeminen. Tämä tekee Graphiten integroimisen muihin ohjelmiin ja mahdollistaa käyttäjien rakentaa vahvoja sovelluksia sen päälle. Ohjelmiston avain on skaalattavuus ja reaaliaikainen pääsy tietoihin. (Davis, C. n,d)

Graphite on kokonaan kirjoitettu Python – koodauskielellä. Komponentit, jotka mahdollistavat Graphite - järjestelmän saavuttamaan nämä yksinkertaiset tavoitteet. (Davis, C. n,d)

3.4 Whisper

Graphitilla on oma erikoistunut tietokantakirjasto nimeltään Whisper. Sitä voidaan verrata malliltaan round-robin-tietokantaa käyttävään RRD:hen. Mutta Whisperillä on pohjimmiltaan tärkeä hiuksenhieno eroavaisuus, jonka Graphite vaatii. Whisper on tietokantakirjasto, jota ohjelmistot käyttävät manipuloidakseen ja hakeakseen tietoa, joka on varastoitu tiedostoihin ainutlaatuiseen muotoon. Whisperin perustoimintoja ovat uuden Whisper - tiedoston luominen, tiedon hakeminen, uusien tietojen päivittäminen tiedostoihin. (Davis, C. n,d) (FAQ. 2009.)

Whisperin luomisessa on kaksi syytä, jonka mukaan se luotiin. Ensimmäisen syynä pidetään sitä, että RRD:tä suunniteltaessa oletettiin tiedon sijoittamisen olevan aina säännöllinen. Tämä oletus aiheutti RRD:n käyttäytymään ei toivotulla tavalla syöttäessä epäsäännöllisesti esiintyvää tietoa. (FAQ. 2009.)

Graphite suunniteltiin helpottamaan niiden ohjelmistojen visualisointia, joiden arvot eivät tapahdu säännöllisesti. Esimerkkinä voidaan selventää tapahtumalla, jossa epätavallinen pyyntö käsitellään ja latenssi lasketaan ja lähetetään Graphiteen. Käyttämällä RRD:tä tällaisissa tilanteessa johtaa siihen, että tieto asetetaan väliaikaiseen alueeseen tietokannan sisällä. Sieltä sitä ei hyväksytä ennen kuin aikaväli on kulunut. Sekä ennen kuin toinen arvo on lisätty tietokantaan seuraavan aikavälin aikana. Jos tätä ei tapahdu varatun aikavälin aikana, alkuperäinen tieto ylikirjoitetaan, jolloin se häviää. (FAQ. 2009.)

Toisena Whisperin tarkoituksena on suorituskyky. RRD-työkalu on erittäin nopea, todellisuudessa nopeampi, kuin mitä Whisper. Ongelmaksi muodostui kuitenkin RRD:n kanssa, kun Whisperiä oltiin kirjoittamassa, että se mahdollistaa vain yhden arvon asettamisen tietokantaan kerrallaan. Toisin kuin mitä Whisperin kanssa, joka mahdollistaa useamman arvon asettamisen tietokantaa yhdellä kertaa, tiivistämällä ne yhdeksi kirjoitus tapahtumaksi. (FAQ. 2009.)

Tämä on juuri se syy miksi Whisper parantaa suorituskykyä dramaattisesti kovan kuorman alla, koska Graphite operoi monen tiedoston kanssa tehden pieniä operaatioita siellä täällä. Pullonkaulana tässä operaatiossa voidaan pitää ”I/O operaatioita”. Tämä liittyy siihen kuinka monta operaatiota voidaan suorittaa määritellyssä ajassa, kuitenkin Whisper voi esimerkiksi tiivistää 10 operaatiota yhdeksi. Nämä olivat ongelmia silloin kun Whisperiä oltiin kirjoittamassa. (FAQ. 2009.)

3.5 Carbon

3.5.1 Mikä on Carbon?

Graphite muodostetaan kahdesta komponentista, etualan verkkosovelluksesta, ja takaalan Carbon varastointisovelluksesta. Tiedonkeruu agentit luovat yhteyden Carboniin ja lähettävät tietonsa. Carbonin tehtävänä on asettaa tieto saataville reaaliaikaiselle piirturille välittömästi ja yrittää varastoida sen levylle niin nopeasti kuin mahdollista. (Carbon. 2008.)

3.5.2 Carbonin Daemonit

Puhuttaessa Carbonista tarkoitetaan joko yhtä tai useampaa Daemonia, jotka luovat taka-alan varaston Graphiten asennuksessa. Yksinkertaisessa asennuksessa tyypillisesti on vain yksi Daemon ”carbon-cache.py”. Carbon on kuitenkin luotu kolmesta pääprosessista, ”carbon-cache.py”, ” carbon-relay.py” ja ”carbon-aggeregator.py”. Kaikki Carbonin Daemonit kuuntelevat aikajaksoista tietoa ja hyväksyvät sen perinteisien protokollien mukaan. Kuitenkin, eroavaisuus astuu voimaan siinä mitä ne tekevät tiedolle, kun ne ovat vastaanottaneet sen. (Carbon. 2008.) (Davis, C. 2011)

Carbon-cache.py

Carbon-cache.py hyväksyy monien protokollien ylitse metriikoita ja kirjoittaa ne levylle niin tehokkaasti kuin mahdollista. Tämä vaatii metriikoiden kiinniottamisen niiden saapuessa RAM välimuistiin, ja huudella ne levylle Whispers kirjastoa apuna käyttäen, tietyn aikavälin aikana. (Davis, C. 2011)

Tämä Daemon vaatii muutaman konfiguraatio tiedoston, ”Carbon.conf” ja ”Storage-Schemas.conf”, toimiakseen. ”Carbon.conf” – tiedostossa kerrottaan mitä portteja, protokollia, ja liikennettä kuunnellaan. ”Storage-Schema.conf” – tiedosto määrittää politiikat sisääntuleville metriikoille, perustuen regex - kuvioihin. Tämä politiikka siirretään Whisperille, silloin kun ”.wsp” – tiedosto esi-varataan. Se määrää myös, kuinka kauan tietoa säilytetään. (Davis, C. 2011)

Sisään tulevien metriikkojen lisääntyessä, yksittäinen ”carbon.cache.py” tapaus ei välttämättä riitä käsittelemään I/O kuormaa. Skaalattavuutta voidaan laajentaa, yksinkertaisesti monistamalla carbon-cache.py tapauksia ”Carbon-aggreator.py” tai ”Carbon-relay-py” takana. (Davis, C. 2011)

Carbon-relay.py

Carbon-relay.py palvelee kahta selkeää tarkoitusta: replikaatiota ja sirpalointia. Kun käytetään *RELAY_METHOD* = sääntö metodia, *carbon-relay.py* tapaus voidaan ajaa *carbon-*

cahce.py palvelimen tilalla ja välittää kaikki sisääntulevat metriikat useille taka-alalla, eri porteissa tai isännillä, toimiville *carbon-cache.py*:lle. (Davis, C. 2011)

RELAY_METHOD = consistent-hashing tilassa*, DESTINATIONS* asetus määrittää sirpalemaisen jakamisen strategian useiden *carbon-cache.py* taka-alojen välille. Sama johdonmukainen hajautus lista voidaan tarjota Graphiten verkkosovellukseen *CRBON-LINK_HOSTS* asetuksen kautta, levittämään lukemia useiden taka-alojen kesken. (Davis, C. 2011)

Carbon-relay.py konfiguroidaan seuraavanlaisesti:

- Carbon.conf. [relay] osio määrittää kuuntelevan isännän/portit ja *RELAY_METHOD* asetuksen
- Relay-rules.conf. *RELAY_METHOD = sääntö* asetuksella, mallilla/tupla palvelimilla, määritetään tämän tiedoston sisällä siten, että mitkä metriikat kohtaavat tietyt säännöllisen lausekkeen säännöt ohjataan millekin isännälle.

(Davis, C. 2011)

Carbon-aggregator.py

Carbon-aggregator.py voidaan ajaa *carbon-cache.py* edessä puskuroimaan metriikoita ajan kanssa, ennen kuin ne raportoidaan Whisperille. Tämä on käytännöllistä silloin, kun rakeista raportointia ei vaadita, se auttaa vähentämään I/O kuormaa ja Whisperin tiedostojen kokoa, matalampien säilyttämisen poliitikkojen ansiosta. (Davis, C. 2011)

Carbon-aggregator.py konfiguroidaan seuraavanlaisesti:

- Carbon.conf. [aggregator] osio määrittää kuuntelijan ja kohteen isännän/portit
- Aggregation-rules.conf. Määrittää ajanjakson (sekunneissa) ja kertymäfunktion (summan tai keskiarvon) sisääntuleville metriikoille, jotka vastaavat tiettyä mallia. Jokaisen aikavälin lopussa, vastaanotetut arvot kerrytetään ja julkaistaan *carbon-cache.py*:lle yhtenä metriikkana.

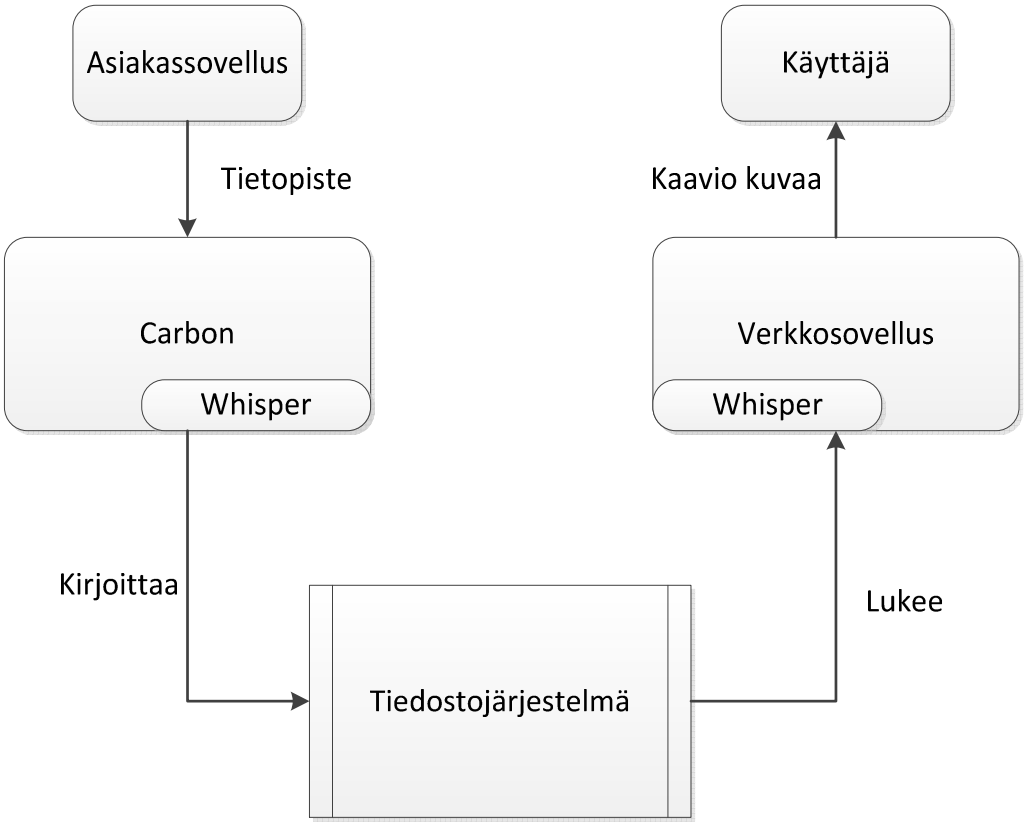
(Davis, C. 2011)

Syy sille miksi Carbon on jaettu kolmeen prosessiin, johtuu Pythonin päänvientivaikeuksista. Alun perin Carbon oli yksittäinen ohjelmisto, jossa nämä erilliset toiminnot suoritettiin ketjuina, mutta valitettavasti Pythonin GIL estää useiden ketjujen käynnissä olemisen samanaikaisesti. Graphiten alkuvaiheesta lähtien toteutettiin laitteistolla, jossa oli hitaat CPU:t. Tarvittiin todellista samanaikaisuutta suorituskyvyn tarpeisiin, joten se jaoteltiin kolmeen putkimaiseen prosessiin. (Carbon. 2008.)

3.5.3 Carbonin toiminnallisuus Graphitissa

Graphiten taka-alalla toimii Daemon prosessi nimeltä ”Carbon-Cache”, paremmin voidaan yleisesti viitata Carbonina. Se on rakennettu Twisted – ohjelman päälle, joka on korkeasti skaalautuva I/O - tapahtumien rakenne Pythonille. Twisted mahdollistaa Carbonin keskustelemaan tehokkaasti suurelle määrälle asiakkaita ja käsittelemään suuren määrän liikennettä pienellä kustannuksella. (Davis, C. n,d)

Carbonin päätehtävänä on varastoida tietopisteitä metriikkoihin, jotka asiakkaat toimittavat. Kuviossa 5 on tämä tietovirta esitettynä. Graphiten terminologiassa, metriikka on mikä tahansa mitattavissa oleva arvo, joka vaihtelee eri ajanjaksoilla. Tietopiste on yksinkertaisesti ”aikaleima” ja ”arvo” pari, joka vastaa mitattua mittaria määritellyn ajanjakson sisällä. Metriikat ovat uniikisti tunnistettavissa niiden nimen perusteella. Ja jokaisen metriikan ja kuten myös tietopisteen nimen tarjoavat asiakkaan sovellukset. (Davis, C. n,d)



Kuvio 5. Tietovirta

Asiakasohjelmiston yleisimpänä tyyppinä on seuranta agentti (ks. kuvio 5). Se kerää järjestelmän tai ohjelmiston mitattavia arvoja ja lähettää nämä kerätyt tiedot Carbonille helposti varastoitavaksi ja visualisoitavaksi. Mitattavilla arvoilla on Graphitessa hierarkkisesti yksinkertaiset nimet, samaan tyyliin kuin mitä tiedostopoluissa, mutta käytetään kuitenkin pistettä rajaamaan hierarkiaa kautta- ja kenoviivojen tilalla. Carbon kunnioittaa mitä tahansa virallista nimeä ja luo sen tietopisteisiin Whisper – tiedoston jokaiselle varastoitavalle arvoille. Whisperin tiedostot varastoidaan Carbonin tiedostopolku hierarkiaan, joka heijastaa piste-eroteltua hierarkiaa jokaisen mittarin nimessä. Esimerkkinä voidaan esittää, että *palvelin.www01.cpuUsage* osoittaa samankaltaiseen hakemistoon */palvelin/www01/cpuUsage.wsp.* (Davis, C. n,d)

Kun asiakassovellukset haluavat lähettää tietopisteitä Graphiteen, sen täytyy muodostaa TCP – yhteys Carboniin, yleensä oletuksena auki porttiin 2003. Asiakassovellus suorittaa kaiken keskustelun, Carbon ei lähetä mitään yhteyden aikana. Asiakassovellus lähettää tietopisteet yksinkertaisena teksti-muodossa, yhteys voidaan jättää auki ja käyttää uudelleen mikäli on tarvetta. Muotona on, että yksi rivi per tietopiste, kuten kuviossa 6 on esitetty, missä jokainen rivi sisältää pilkullisen mittarin nimen, arvon ja Unix aikaleiman erotettuna välillä. Korkeammalla tasolla Carbon ei tee muuta, kuin kuuntelee tätä tiedostomuotoa ja yrittää varastoida sen levylle niin nopeasti kuin mahdollista käyttämällä Whisperiä. (Davis, C. n,d)

```
servers.www01.cpuUsage 42 1286269200
products.snake-oil.salesPerMinute 123 1286269200
[one minute passes]
servers.www01.cpuUsageUser 44 1286269260
products.snake-oil.salesPerMinute 119 1286269260
```

Kuvio 6. Esimerkki tietopisteen muoto. Graphite.Davis, C. N, d

3.6 Skaalautuvuus

CPU näkökulmasta

Graphite skaalautuu horisontaalisesti kummassakin, etualalla ja taka-alalla. Tarkoittaen, että voidaan lisätä laitteita joukkoon, jotta saadaan enemmän läpituontia. Se on myös vikasietoisuutta siinä mielessä, että taka-alan koneen hukkuessa, aiheuttaa se minimaallisen tiedon hävikin. Sekä ei häiritse järjestelmää, jos on olemassa riittävää kapasiteettia jäljellä käsittelemään kuormaa. (FAQ. 2009.)

I/O näkökulmasta

Tästä näkökulmasta kuormituksessa Graphite suorittaa erittäin nopeasti monia pieniä I/O operaatioita moniin eri tiedostoihin. Tämä johtuu siitä, että jokaiselle arvolle, joka lähetetään Graphiteen, varastoidaan omiin tietokanta tiedostoihin. (FAQ. 2009.)

Korkeassa suorituksessa, kun muutamia tuhansia erillisiä arvoja päivitetään samanaikaisesti, vaatii se hyvän RAID järjestelyn. Graphiten taka-ala vastaanottaa tulevaa tietoa, jos levykkeet eivät pysy määrältään suurien kirjoitus tapahtumien mukana joita tapahtuu. Jokainen tietopiste on muutamia tavuja, mutta useammat levykkeet eivät voi tehdä kuin muutamia tuhansia I/O – operaatioita sekunnissa, vaikka ne olisivat kuinka pieniä tahansa. Silloin kun tämä tapahtuu, Graphiten tietokanta moottori Whisper mahdollistaa Carbonin kirjoittaa useita tietopisteitä kerralla. Näin nostamalla kokonaissisääntuloa, sillä hinnalla, että tietoa kerätään muistiin ennen kuin se voidaan kirjoittaa. (FAQ. 2009.)

3.7 Riippuvuudet

3.7.1 Mitä ovat Graphiten riippuvuudet?

Graphite tekee kuvaajia käyttämällä Cairo kaaviokirjastoa. Tämä lisää riippuvuuksia usealle kaavio pohjaisille kirjastoille, jotka eivät tyypillisesti löydy palvelimelta. Useimmissa nykyisissä Linux – jakeluissa on vaadittavat vaatimukset saatavilla peruspaketissa. Pythonin moduuliriippuvuudet voidaan asentaa ”pip”:lla mikäli halutaan, järjestelmän peruspaketin sijaan. Tai käyttämällä Pythonin versiota, joka eroaa järjestelmän oletuksesta. Jotkin moduulit, kuten Cairo, voivat vaatia kirjaston ohjelmisto kehitystunnisteet olevan saatavilla. (Installing Graphite. Davis, C)

3.7.2 Python

Python on tulkinnallinen, olio, ja korkean tason ohjelmointikieli, dynaamisella semantiikalla. Tietokantarakenteissa se on korkeasti rakennettu, yhdistettynä dynaamista kirjoitusta ja sidontaa. Tämä tekee siitä erittäin houkuttelevan RAD:ia varten, sekä myös käytettäväksi skriptaukseen tai liima kielenä yhdistämään olemassa olevat komponentit keskenään. Python on yksinkertainen ja helposti opeteltava, syntaksisesti luotettava ja siitä syystä vähentää ohjelmiston ylläpitokustannuksia. Python tukee moduuleja ja paketteja, jotka suosivat ohjelman modulaarisuutta ja koodin uudelleen käyttöä. Sen tulkinta ja laaja standardien kirjasto on saatavilla lähdekoodissa tai binaarisessa muodossa ilman

veloitusta kaikille suurehkoille alustoille, ja se on vapaasti levitettävissä. (What is Python? Executive Summary. 2014)

Yleensä ohjelmoijat rakastuvat Pythoniin sen lisääntyvän tuottavuuden myötä, jonka se tarjoaa. Koska siinä ei ole kokoamisvaihetta, muokkaus-testaus-virheiden korjauskierros on uskomattoman nopea. Virheiden korjaus on Pythonissa helppoa, virhe tai huono syöttö ei koskaan aiheuta segmentointivirhettä. Sen sijaan, kun tulkitsija havaitsee virheen, se herättää poikkeuksen. Kun ohjelma ei tartu poikkeukseen, tulkitsija tulostaa kutsupinon. Lähde tason virheenkorjaus mahdollistaa paikallisen ja globaalin muuttujien tarkastelun, arvojen mielivaltaisia ilmaisuja, pysäytyspisteiden asettamisen, koodin katsastamisen rivi kerrallaan jne. Virheenkorjaus itsessään on kirjoitettu Pythonilla, näin todistaen Pythonin introspektiivisen voiman. (What is Python? Executive Summary. 2014)

3.7.3 PyCairo

PyCairo on Pythonin liitännäinen, Cairon grafiikka kirjastolle. Joten Cairo on siis 2D - grafiikan kirjasto, monen laitteen ulostulo tuella. Cairo on suunniteltu tuottamaan johdonmukaista ulostuloa kaikille ulostulo medioille, samalla käyttäen hyväkseen näyttölaitteiston kiihtyvyyttä silloin kun se on saatavilla. (Cairo. 2012.) (Chaplin, S. 2008)

3.7.4 Django

Django on korkeantason Python verkkorakenne, joka rohkaisee nopeaa kehitystä ja puhdasta, käytännöllistä muotoilua. Se kehiteltiin nopeasti liikkuviin verkkouutisten operaatioihin. Django suunniteltiin käsittelemään kahta haastetta: uutishuoneiden intensiivisiä määräaikoja ja kokeneiden verkkokehittäjien tiukkoja vaatimuksia, jotka sen kirjoittivat. Se antaa käyttäjän rakentaa korkeasti suoriutuvia, tyylikkäitä verkkosovelluksia nopeasti. (Meet Django. n, d)

3.7.5 Twisted

Twisted projekti on saanut suosiotaan tehokkaana ja vakaavampana tapana verkkosovellusten täytäntöönpanona. Sen keskuksessa, Twisted on asynkroninen verkkorakenne. Toisin kuin muut samankaltaiset rakenteet, Twisted tarjoaa monipuolisen valikoiman integroituja kirjastoja käsittelemään yleisiä protokollia ja ohjelmoinnin tehtäviä, kuten käyttäjän autentikointia ja jopa etäolioiden välitystä. Yhtenä filosofiana Twisted:in takana pidetään, on pilkkoa perinteisiä erotteluja työkalujen keskuudessa. Sillä sama palvelin, joka tarjoaa verkkosisällön, voi ratkaista DNS hakuja. Vaikka paketti itsessään on aika suuri, sovellusten ei tarvitse tuoda kaikkia Twisted:in komponentteja, jotta käytönkuormitus pidetään minimaallisena. (Kinder, K. 2005)

3.7.6 Apache

Apache on yksi suosituimmista verkkopalvelin sovelluksista. Se mahdollistaa tietokoneen isännöidä yhtä tai useampaa verkkosivua, johon voidaan yhdistää Internetin kautta käyttämällä verkkoselainta. Ensimmäinen Apachen versio julkaistiin 1995 ”Apache Grouping” johdosta. Vuonna 1999 ”Apache Groupista” muodostui ”Apache Software Foundation”, ei voittoa tavoitteleva organisaatio, joka yleisesti ylläpitää Apachen verkkopalvelin sovelluksen kehittämistä. (Apache. 2011)

Apachen suosio verkkopalvelimena johtunee suurimmalta osin siitä, että se on täysin avointa lähdekoodia ja on täten ilmainen käyttää. Luonnollisesti verkkopalvelin yhtiöt voivat tarjota Apache pohjaista verkko isännöinti vaihtoehtoa minimaallisella kustannuksella. Voidaan verrata esimerkiksi Windows palvelin ohjelmistoon, joka vaatii kaupallisen lisenssin. Apache tukee useita käyttöjärjestelmiä kuten Linux, Windows ja Macintosh. Kuitenkin monet Linux käyttöjärjestelmät ovat avointa lähdekoodia, Apache yhdistettynä Linuxiin, on muodostunut suosituin verkko isännöinti konfiguraatio kombinaatio. (Apache. 2011)

Apache voi isännöidä staattisia verkkosivuja ja dynaamisia verkkosivuja, jotka käyttävät palvelin puolen koodikieltä, kuten PHP, Python ja Perl. Näiden ja muidenkin kielien tukeminen implementoidaan moduuleilla, asentamalla paketteja, jotka on lisätty standardi Apachen asennuksessa. Apache tukee myös muitakin moduuleja joilla voidaan lisätä turvallisuusvaihtoehtoja, tiedostojen hallintaa, ja muita tarvittavia ominaisuuksia. (Apache. 2011)

WSGI

WSGI (*Web Server Gateway Interface*) on *Pythonin* standardi web-sovelluksille ja se tarkoittaa web palvelimen portin rajapintaa. Sitä käytetään rajapintana web-palvelimen ja sovelluksen välillä, jota halutaan käyttää. Jokaiselle projektille vaaditaan omistautunut WSGI käsittelijä prosessoimaan sovelluksen pyyntöjä. Käsittelijä on yleensä pieni skripti, luokiteltu objektin instanssi, joka toimitetaan projektin mukana. (WSGI. 2011)

3.8 Työkalut, jotka toimivat Graphitin kanssa

Graphitella on mahdollisuutena iso joukko erilaisia työkaluja, jotka voidaan integroida sen kanssa. Liitteestä 2 on luetteloitu näistä suurin osa ja niitä on jaoteltu seuraavanlaisesti:

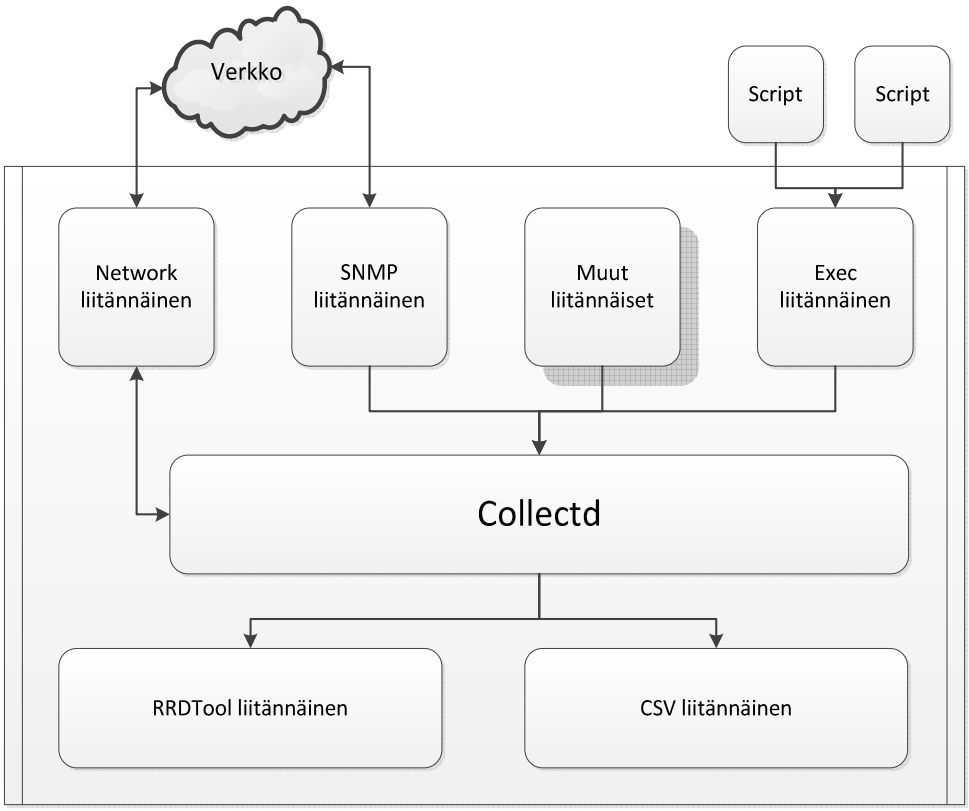
- Kerääminen
- Lähettäminen
- Visualisointi
- Monitorointi
- Muut

(Davis, C. 2014)

4 COLLECTD

4.1 Yleistä

Collectd on Daemon, joka kerää järjestelmän suorituskyvyn määräaikaista statistiikkaa ja tarjoaa mekanismeja arvojen varastoinniksi monella eri tapaa. Esimerkiksi RRD tiedostoja. Näitä statistiikkoja voidaan käyttää nykyisten suorituskyvyn pullokaulojen etsimiseen ja ennustamaan tulevaisuuden järjestelmän kuormitusta. Kaikki toiminnollisuudet Collectd:ssa tapahtuvat liitännäisten kautta, kuten kuviossa 7 on esitetty Collectd:n arkkitehtuurinen rakenne. Paitsi konfiguraatiotiedoston jäsentäminen. Tämä tarkoittaa sitä, että olennaisella Daemonilla ei ole ulkoisia riippuvuuksia ja pitäisi suorittua millä tahansa, jolla on tekemistä POSIXin kanssa. (Collectd – The system statistics collection daemon. n, d) (Features. n, d)



Kuvio 7. Collectd - arkkitehtuuri

4.2 Arkkitehtuurin liitännäiset

4.2.1 Liitännäinen SNMP

SNMP (*Simple Network Management Protocol*) liitännäinen käyttää *Net-SNMP* kirjastoa lukeakseen arvoja verkkolaitteista käyttämällä SNMP:tä. Se on laajasti levinnyt standardi, joka tarjoaa hallinnollista tietoa laitteilta kuten kytkimiltä, reitittimiltä, telineiden palvelinkaapeista, UPS:sta (*Uninterruptible power supplies*) jne. Vaikkakin teoreettisesti arvojen kerääminen samanaikaisesti muilta tietokoneilta ei ole suosittua tämän protokollan kautta. Suositaankin Collectd:n omaa protokollaa, *Network* - liitännäisellä toteutettuna. (Plugin:SNMP.2010)

4.2.2 Liitännäinen Exec

Ecec liitännäinen toteuttaa skriptejä / sovelluksia ja lukee takaisin arvoja, jotka tulostetaan *STDOUT*:iin ohjelman toimesta. Tämä mahdollistaa Daemonin laajentamisen helposti ja joustavalla tavalla. Huonona puolena on, että täytäntöönpanolliset ulkoiset ohjelmat ovat erittäin tehottomia ja aina potentiaalinen turvallisuusriski. Collectd yrittää torjua näitä ongelmia seuraavanlaisilla vaihtoehdoilla tai mekanismilla. (Plugin:Exec. 2011)

On varmistettu, että yksi ohjelman tapaus toteutetaan Collectd toimesta milloin tahansa. Koska toteutetut ohjelmat voivat lausua niin monta arvoa kuin ne haluavat. On yleistä jättää nämä mukautetut ohjelmat suoriutumaan silmukassa, jotta niitä ei tarvitsisi käynnistää säännöllisesti. Erityisesti skriptien kanssa tämä on suuri suorituskyvyllinen voitto, koska tulkitsijaa ei tarvitse käynnistää uudelleen ja uudelleen ja skripti jäsennetään vain kertaalleen. Kuitenkin vaadittava muisti tulkitsijan pysymiselle muistissa on eräänlaista vaihtokauppaa. Tämä liitännäinen on yleinen liitännäinen, eli toisin sanoen se ei voi toimia ilman konfiguraatiota, koska sillä ei ole kohtuullista oletus käyttäytymistä. (Plugin:Exec. 2011)

4.2.3 Liitännäinen CSV

CSV - liitännäinen kirjoittaa arvoja tekstimuoto tiedostoihin, hyvin tunnetulla CVS (Comma Seperated Values) formaatilla. Nämä tiedoston voidaan siten tuoda moniin muihin ohjelmistoihin, kuten suosittuihin taulukkolaskenta ohjelmistoihin. (Plugin:CVS. 2009)

Esimerkki

```
#<Plugin csv>
#      DataDir "/opt/collectd/var/lib/collectd/csv"
#      StoreRates false
#</Plugin>
```

Attribuutit:

DataDir, hakemisto: Asettaa hakemiston minne CVS - tiedostot varastoidaan. Oletusarvona on, että CVS - tiedostot on generoitu Daemonin työskentely hakemiston alla, toisin sanoen päähakemisto. Erikoisempia lankoja *stdout* ja *stderr* voidaan käyttää kirjoittamaan standardi ulostulo ja standardi virhe kanavat, vastaavasti. Tässä on järkeä silloin, kun Collectd pyörii etuala- tai ei-Daemon-tilassa.

StoreRates, (False|True): Mikäli asetetaan *True*, eli todeksi, muuttaa laskurin arvot asteiksi. Mikäli asetetaan *False*, eli epätosi, myös oletusarvo. Laskurin arvot varastoidaan sellaisenaan, eli kasvavana kokonaislukuna.

(Plugin:CVS. 2009)

4.3 Ominaisuudet

Collectd konfiguraatio on pidetty niin yksinkertaisena kuin mahdollista. Sen lisäksi, kun valitaan mitä liitännäisiä ladataan, ei tarvitse konfiguroida mitään muuta. Mutta Daemon voidaan toteuttaa haluamallaan tavalla mikäli tarvetta. Collectd ei ole scripti, mutta se on kirjoitettu selkokielisellä C-koodilla, suorituskykyä ja siirrettävyyttä varten. Aivan kuten Daemon se pysyttelee muistissa, joten ei ole tarvetta käynnistää raskasta

tulkitsijaa joka kerta, kun arvoja kirjataan. Tämä mahdollistaa Collectd:lle 10 sekunnin oletus erotuskyvyn, samalla se kohtelee järjestelmää miellyttävästi. (Features. n, d)

4.4 Hienostunut verkkokoodi

4.4.1 Verkkokoodi

Collectd käyttää tiedon työntö mallinnusta, toisin sanoen tieto kerätään Collectd:n toimesta ja lähetetään *ryhmälähetys* ryhmälle tai palvelimelle. Näin ei ole keskitettyä tapausta, joka kyselisi eri arvoja. Verkkokoodi käyttää hyödykseen IPv6 ja *ryhmälähetys* verkkotekniikoita. Collectd:ssa voidaan konfiguroida *yksittäislähetys* siten, että määritellään tiedonsiirrossa lähettäjät ja vastaanottaja erikseen. Eli toisin sanoen määritellään asiakkaat, jotka lähettävät tietonsa keskitetylle palvelimelle. (Features. n, d)

Seuraavat asetukset voidaan määrittää:

- *Ei käytetä verkkoa*: Mikäli *verkko* – liitännäistä ei ladata, on verkon käyttö kokonaan otettu pois käytöstä. Tällöin ei ole kantaa, ei ylikuuluvuutta, joten ei ongelmia.

- *Ryhmälähetys:* Tietoa voidaan lähettää ja vastaanottaa *ryhmälähetys* ryhmässä. Tämä on helpoin vaihtoehto, jos on monta *asiakasta* ja yksi tai useampi palvelin lähiverkossa. *Ryhmälähetyksen* käyttö on triviaalia, yksinkertaisesti merkitään *ryhmälähetys* osoite ja palvelin automaattisesti tunnistaa sen ja liittyy tunnettuun ryhmään.

  Ryhmälähetyksen käyttämistä voidaan ajatella ”automaattisena löytämisenä”. Palvelimen ei tarvitse tietää mitä asiakkaita on olemassa ja asiakkaiden ei tarvitse tietää palvelimien IP – osoitteita. Itse asiassa ne eivät edes tiedä montako palvelinta on olemassa. Tätä voidaan ajatella, kuin radio lähetykseksi. Kun kanava on asetettu, voidaan vastaanottaa kaikki lähetetty tieto lähettäjän toimesta, riippumatta siitä missä ollaan.

- *Yksittäislähetys:* Voidaan määrittää yksittäinen isäntä, johon lähetetään tietoa. Erinomainen silloin, kun on hajautetut isännät.

- *Välitys toimenpide:* Voidaan asettaa lähettämään tietoa, joka on vastaanotettu verkosta. Käyttämällä tätä voidaan lähettää tietoa, joka on vastaanotettu *ryhmälähetyksellä* ja lähettää se *yksittäislähetyksenä* määriteltyyn osoitteeseen. Kaksi erillistä *ryhmälähetys* ryhmää voidaan yhdistää ilman reititystä.

(Features. n, d)

4.4.2 Binääriprotokolla

Verkkoprotokollasta on suunniteltu niin kevyt kuin mahdollista, jottei tiedon kerääminen hitaammassakin verkossa olisi ongelma. Protokolla on laajennettavissa, joten tulevaisuutta ajatellen se on ”takaisinpäin” yhteensopiva. Binääriprotokolla on protokolla, jonka *verkko* - liitännäinen toteuttaa. Ulkoiset toteutukset vaihtelevat kerättyä tietoa, joka kerätään Collectd:lla tai lähettävät tietoa Collectd:hen. (Features. n, d) (Binary protocol. 2012)

Protokollan rakenne

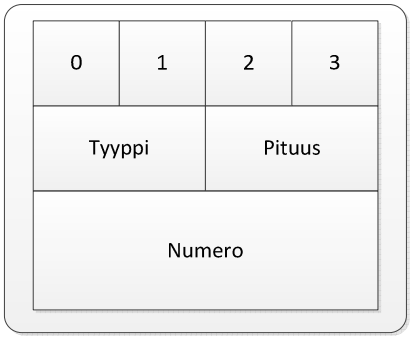
Jokainen paketti koostuu yhdestä tai useammasta niin sanoituista ”osista” (ks. kuvio 8). Jokainen osa alkaa samoilla neljällä tavulla. Rakenne muodostuu kahdesta tavusta, jotka määrittävät ”*osan tyypin*” eli millaista tietoa on liitetty mukaan tähän osaan. Kahdesta jälkimmäisistä tavuista, jotka määrittävät ”*osan pituuden*”, sisältäen otsikon neljä tavua itsessään. Molemmat *tyyppi* ja *pituus* tulisi olla koodattuna verkko järjestyksessä. (Binary protocol. 2012)

Kuvio 8. Rakenteen osat

Maksimaallinen kuorman koko on siis missä tahansa osassa 65 551 tavua. Käyttämällä tätä rakennetta, asiakkaat voivat päättää sellaisen osan koon mitä ne eivät tiedä ja ohittaa tuntematon tieto. Tämä tekee protokollasta eteenpäin yhteensopivan, joten uusia ominaisuuksia voidaan lisätä helposti. On olemassa kaksi rakenne osaa, joita käytetään pariin ”*tyyppiin*”, *numeeriseen* ja *langalliseen*. (Binary protocol. 2012)

Numeeriset osat

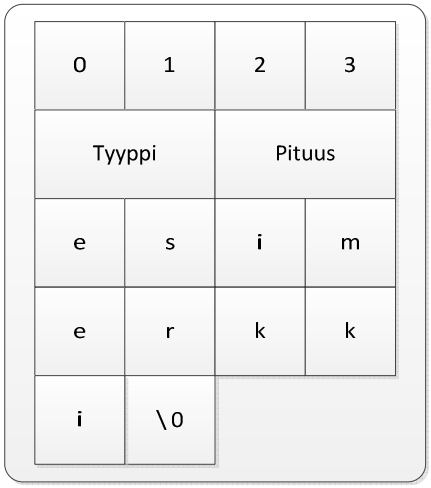
Numeeriset kokonaislukujen arvot (ks. kuvio 9), toisin sanoen aikaväli ja aika arvot, lähetetään käyttämällä kahdeksan tavun kokonaislukuja. *Pituus* kenttä noissa osissa tulee aina asettaa kokonaisluvuksi 12. (Binary protocol. 2012)



Kuvio 9. Numeerisen osan rakenne

Langalliset osat

Langat lähetetään sisältäen lopussa nolla tavun (ks. kuvio 10). Esimerkissä (ks. kuvio 10) nähdään langan *esimerkki* koodaus. Lanka on yhdeksän merkkiä pitkä, perässä heti nolla tavu. Liitettynä otsikon neljään tavuun, kokonaispituus on tälle osalle 14 tavua. (Binary protocol. 2012)



Kuvio 10. Langallist osat, esim. "esimerkki”- koodaus

Arvo osat

Arvo osat koodaavat todelliset tiedon näytteet. Edellisen ajan, liitännäisen, liitännäisen tapauksen, tyypin, ja tyypin tapauksen osat täytyvät luoda tiedon näytteille kontekstin. Arvo osat muodostuvat seuraavanlaisesti:

- Tyyppi 0x0006
- Pituus (16 bittinen kenttä)
- Arvojen lukumäärä (16 bittinen kenttä)
- Arvot jokaiselle
    - Tiedon tyypin koodi (8 bittinen kenttä)
        - *Laskuri* → 0
        - *Mittari* → 1
        - *Polveutua* → 2
        - *Absoluuttinen* → 3
    - Arvo (64 bittinen kenttä)
        - *Laskuri* → Verkko, *ei määritelty kokonaisluku*

- *Mittari* → x86, *kaksinkertainen*
- *Polveutua* → Verkko, *määritelty kokonaisluku*
- *Absoluuttinen* → Verkko, ei määritelty

Monilla tiedon näytteillä on yksi arvo, kuten ”CPU”, mutta muilla on useita arvoja, kuten ”disk_merged”, jolla on useita luku ja kirjoitus arvoja.

Mikäli on useita arvoja, tyypit pitää määrittää ensimmäisenä, sitten vasta arvot, tähän tyyliin:

[type] [type] [type] [value] [value] [value]

eikä näin:

[type] [value] [type] [value] [type] [value]

(Binary protocol. 2012)

Osien tyypit

Liitteessä 4 on taulukko osien tyypeistä joita käytetään. Näitä numeerisia tyyppejä käytetään ”*osien*” tunnistukseen.

5 CABOT

Cabot on ilmainen, avoimen lähdekoodin, itsenäinen seuranta alusta infrastruktuuri, joka tarjoaa *PagerDutyn, Server Densityn, Pingdomin* ja *Nagioksen* parhaimpia puolia, ilman näiden monimutkaisuutta ja kustannuksia. Se tarjoaa verkkokäyttöliittymän, joka mahdollistaa palvelujen seurannan ja lähettää puhelimeen tekstiviestin tai Hipchat/sähköposti hälytyksen päivystyksessä olevalle henkilölle tai tiimille, mikäli palvelu alkaa käyttäytyä huonosti tai kaatuu. Erittäin hyvä ominaisuus on, että voidaan käyttää tietoa, jota jo hyödynnetään Graphitessa. Näistä voidaan luoda hälytyksiä, ilman aivan uudenlaisen tiedon kerääjän toteutusta ja ylläpitoa.

Hälytyksiä voidaan perustaa seuraavista:

- Graphite metriikat
- Tilanne koodit ja verkko päätepisteiden vastauksista
- Jenkinsin “töiden” tiloista

(Cabot. 2014)

Cabot on rakennettu Arachyksella jouluprojektina, koska he eivät voineet ”kääriä” päätänsä Nagioksen ympärille, ja mikään muu toteutus ei tuntunut sopivan heidän tapaukseensa. He julkaisivat sen avoimena lähdekoodina, ja toivovat, että se on myös muille toteutuksille yhtä käytännöllinen. Cabot on kirjoitettu *Pythonilla* ja käyttää *Djangoa*, *Bootsrrappia*, *Font Awsomia* ja muita käytännöllisiä ohjelmia taustalla. (Cabot. 2014)

6 TYÖN TOTEUTUS

6.1 Toteutus yleiskuvallisesti

Musicinfon palvelimet käyttävät Ubuntu käyttöjärjestelmää, joten koko toteutus on toteutettu täysin Linux ympäristössä. Tähän toteutukseen on yhdistetty laaja valikoima avoimen lähdekoodin ohjelmistoja, joista on kasattu toiminnallinen valvontajärjestelmä sisältäen Graphiten, Collectd, Cabot ja monia muita.

Graphiten toiminnollisuus ei ole ensi alkuunsa helppo ymmärtää. Joten käsitellään sen toiminnollisuutta ensin yleiskuvallisesti. Kuviossa 11 on esitetty koko opinnäytetyön toteutuksen toiminnollisuus kokonaisuudessaan ja sen on tarkoitus antaa hyvin yksinkertainen käsitys siitä miten työ on toteutettu, joten avataan kuvion tarkoitusta.

Muille palvelimille tiedon ”kerääjänä” toteutettiin Collectd - ohjelmisto. Sen tarkoitus toimia muiden palvelimien ”asiakkaana” ja kerätä, tallentaa ja lähettää järjestelmän tietoja pääpalvelimelle, jossa Graphite on toteutettu. Collectd:ssä on laaja tarjonta liitännäisiä, joita voidaan käyttää tähän käyttötarkoitukseen.

HUOM! On suositeltavaa asentaa toteutusympäristöön pääpalvelimelle Collectd asennus liitteen 3, ”*lähteestä asentamisen*” mukaan mikäli ollaan integroimassa Collectd:tä Graphiten toteutukseen, koska *Write_Graphite* – liitännäinen toiminta on parhaimmillaan uusimmassa versiossa (5.4+). Aikaisimmissa versioissa liitännäisen ominaisuudet ovat puutteellisia

Tiedon saapuessa pääpalvelimelle, se vastaanotetaan pääpalvelimen Collectd:n verkkoliitännäisellä. Collectd lähettää tiedon Carbon varastointijärjestelmälle käyttäen liitännäistään ”*Write_Graphite*”. Se lähettää myös pääpalvelimen tiedot, jotka se on myös kerännyt.

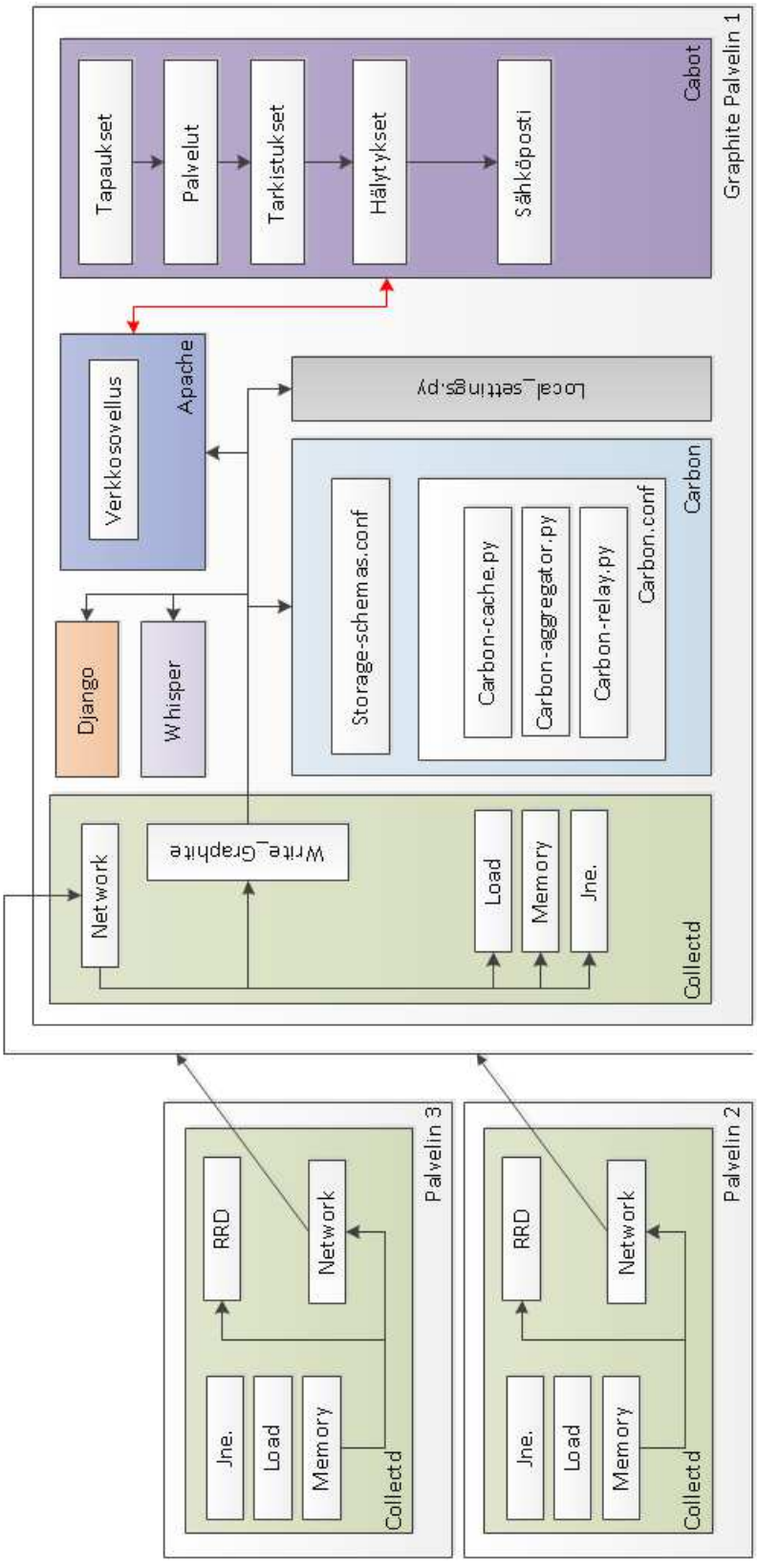
Carbon vastaanottaa tiedot, tallentaa ne sen omien konfiguraatioiden mukaisesti. Konfiguraatiotiedostot ”*Carbon.*conf”, joka sisältää Carbonin *Cahce/Relay/Aggregator* toiminnallisuudet ja ”*Storage-Schema.*conf”, jossa määritellään varastoinnin konfiguraatiot. Graphite käyttää varastointipalvelua Whisper, joka keskustelee Carbonin kanssa.

Graphiten omia tärkeimpiä konfiguraatioita perustoiminnollisuuden kannalta on ”*Local_settings.py*” tiedosto. Tiedostosta ja sen konfiguraatioista löytyy tarkempi avaus edempänä sen omassa osiossaan.

Nyt kun tiedon kerääminen ja varastointi on määritelty, on tiedon esittämisen vuoro. Kaikista edellisistä määritteistä Carbon syöttää tiedot Graphiten verkkosovellukseen, jossa tiedot voidaan nähdä selkeinä graafisina kuvioina. Tästä huolehtivat Apache verkkosovelluspalvelin ja sen kanssa keskustelee Django, joka huolehtii verkkosovelluksen tiedon tallentamisen, esimerkiksi käyttäjätunnukset ja salasanat.

Nyt meillä on kokonaisuus, jossa kerätään, lähetetään, tallennetaan ja esitetään tietoa. Kuitenkaan valvonnan kannalta se tärkein puuttuu, eli määriteltyjen arvojen pohjalta toteutetut hälytykset, joiden pohjalta voidaan ennakoida tapahtumia, joita halutaan välttää.

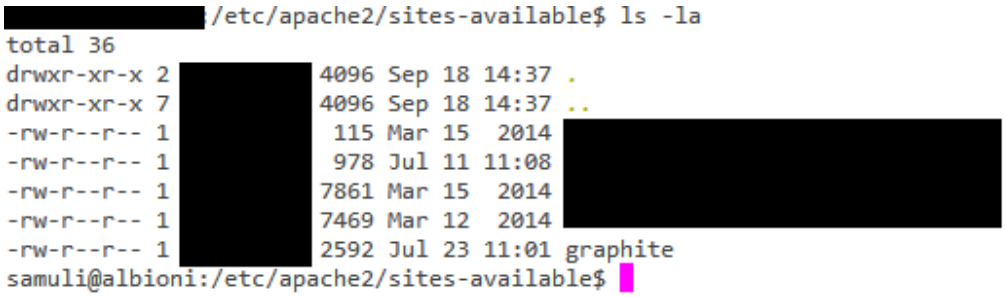
Tässä mukaan astuu hälytysjärjestelmä Cabot. Cabotille on annettu käyttäjätunnukset Graphiteen, joilla se kirjautuu sisään. Sille kerrotaan Graphiten metriikoiden avulla mitä metriikkaa sen tulee seurata ja tehdä hälytykset määritysten mukaisesti. Hälytykset lähetetään sähköpostiin. Cabotin sisäinen hierarkia koostuu karkeasti Tapaukset - - > Palvelut - - > Tarkistukset. Näiden määrittely ja toiminnollisuudet on kerrottu paremmin Cabotin omassa osiossa myöhempänä.



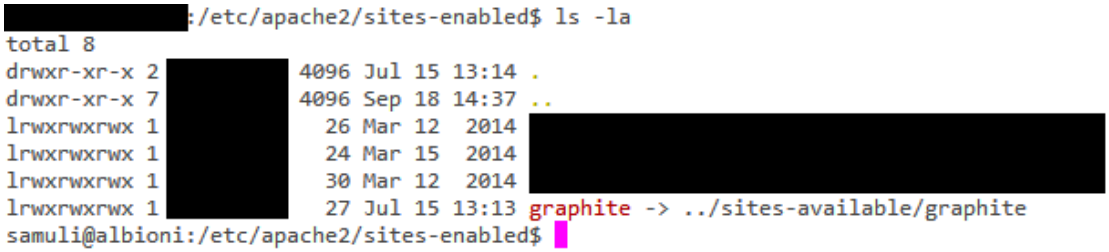
Kuvio 11. Toiminnollisuus

6.2 Apache

Mikäli Graphite on asennettu ohjeiden mukaisesti, jotka löytyvät liitteestä 1. Kuvioissa 12 ja 13 on kaappaukset *Graphite* Apachen verkkosovelluksen konfiguraatioista. Konfiguraatiotiedostossa on käytetty pohjana liitteen 1 asennusohjeiden mukaisesti tiedostoa, joka on hakemistossa verkkosovelluksen alla esimerkkitiedosto *example-graphite-vhost.conf.* Kuten liitteessä yksi on selitetty, on näillä kahdella *symlink* yhteys toisiinsa. Eli, kun Graphite verkkosovellus tiedosto sisällytetään hakemistoon *sites-available,* se kopioidaan *symlinkillä* myös hakemistoon *sites-anabled*.



Kuvio 12. Apache sites-available



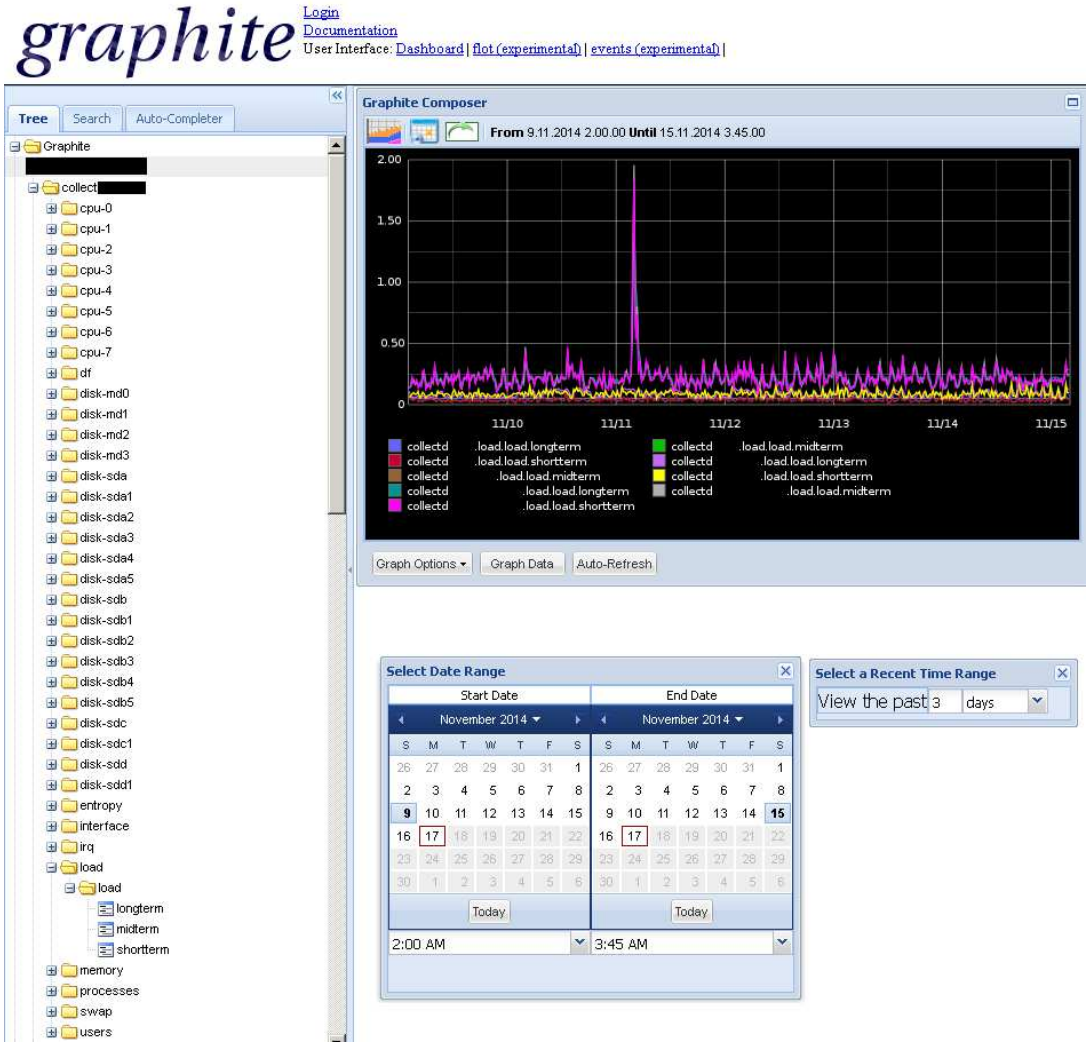
Kuvio 13. Apache sites-enabled

Kuten liitteessä 5 on nähtävissä, *ports.conf* tiedostoon on lisättävä sen portin numero, jota halutaan verkkosovelluksessa käyttää seuraavanlaisesti:

Listen "haluttu portti"

6.3 Graphite - verkkosovellus

Asentamisen jälkeen Graphiten verkkosovellus on osoitteessa *localhost:portti*, portti on se portti joka määritellään *Apachen* konfiguraatiotiedostossa. Konfiguraatiotiedosto löytyy kokonaisuudessaan liitteestä 6. Kuviossa 14 on kaappaus Graphiten näkymästä. Vasemmalla on hakemisto, joka muistuttaa *Windows:in* hakemistopolku rakennetta. Hakemistosta voidaan avata ja sulkea monia eri näkymiä oikealla näkyvään *Composeriin*. Aikaa voidaan muuttaa alaoikealla näkyvillä aikavalikoilla. Vasemmanpuoleisella voidaan vaihtaa päivittäin ja määrittää kellonaika. Oikeanpuolimmaisella voidaan vaihtaa viimeaikaiset kuviot. Esim, kuten kuviossa 14, valitaan viimeisen kolmen päivän ajalta.



Kuvio 14. Graphite – yleisnäkymä

Local_settings.py

Graphiten verkkosovellus sisällyttää *settings.py* moduulista *local_settings.py* tiedoston. Tästä tiedostosta Graphiten verkkosovelluksen konfiguraatiot ladataan. *local_settings.py* tiedosto sijaitsee Graphite moduulin alla, jossa verkkosovelluksen koodi suorittaa toimintansa. Asennuksen yhteydessä tiedosto löytyy oletushakemistosta:

/opt/graphite/webapp/graphite/local_setting.py

Tiedosto on nähtävissä kokonaisuudessaan liitteessä 11. Tärkeimmät asetukset joihin puututtiin, tärkein niistä Djangon tietokannan hakemiston kertominen, olivat tässä tiedostossa. (Davis, C. 2014)

Lokit

Esimerkki palvelimelta

```
# Logging
LOG_RENDERING_PERFORMANCE = True
LOG_CACHE_PERFORMANCE = True
LOG_METRIC_ACCESS = True
```

Attribuutit:

LOG_RENDERING_PERFORMANCE (False|True): Oletusarvona “false” eli epätosi. Mikäli “true” eli tosi, laukaistaan rendering.log tiedoston luominen, jolla seurataan ajoitus pyyntöjä Render URL APIsta.

LOG_CACHE_PERFORMANCE (False|True): Oletusarvona “false” eli epätosi. Mikäli “true” eli tosi, laukaistaan cache.log tiedoston luominen, jolla seurataan carbon-cachen kauko-kutsuja kuin myös Request Cachen osumia ja ohilyöntejä.

LOG_METRIC_ACCESS (False|True): Oletuksarvona “false” eli epätosi. Mikäli “true” eli tosi, laukaistaan metricaccess.log tiedoston luominen, joka seuraa Whisperin ja RRD:n tietoihin pääsyä. (Davis, C. 2014)

Virheiden korjaus

Esimerkki palvelimelta

```
# Enable full debug page display on exceptions (Internal Server Error pages)
DEBUG = True
```

Attribuutit:

Debug (False|True): Oletusarvona “false” eli epätosi. Mikäli “true” eli tosi, mahdollistetaan yksityiskohtainen Django virhesivun luominen. Hyvä olla päällä Graphite verkkoympäristön kehittämisessä.

Etäkäyttäjä

Esimerkki palvelimelta

```
## REMOTE_USER authentication. See: https://docs.djangoproject.com/en/dev/howto/auth-remote$
USE_REMOTE_USER_AUTHENTICATION = True
```

Attribuutit:

USE_REMOTE_USER_AUTHENTICATION (False|True): Oletusarvona “false” eli epätosi. Mikäli “true” mahdollistetaan Djangon *Taka-alanEtäkäyttäjä* taka-alan autentikointi.

Tietokanta

Tämä konfiguraatio asettaa Djangon tietokanta asetukset. Graphite käyttää tietokantaa varastoidakseen käyttäjäprofiileja, kojelautoja ja tapahtumien toiminnollisuuksiin. Se käyttää SQLite tietokantatiedostoa, joka sijaitsee oletuksena polussa:

```
/opt/graphite/webapp/graphite/graphite.db
```

Mikäli käytetään useita Graphite verkkosovelluksia, vaaditaan PostgresSQL tai MySQL tietokanta, jotta kaikki tapaukset voivat jakaa saman tietolähteen. (Davis, C. 2014)

Esimerkki palvelimelta

```
DATABASES = {
   'default': {
      'NAME': '/opt/graphite/webapp/graphite/graphite.db',
      'ENGINE': 'django.db.backends.sqlite3',
      'USER': '',
      'PASSWORD': '',
      'HOST': '',
      'PORT': ''
   }
}
```

Oletuksena on vain tyhjä hakemisto ´default´ { }. Hakemisto sisältää kaikki asetukset tietokannoille joita käytetään Djangon kanssa. Se muodostuu sisäkkäisistä hakemistoista, joiden sisältö kartoittaa tietokantojen aliaksia hakemistoon, joka sisällyttää vaihtoehdot yksittäisille tietokannoille. Tietokannan asetukset tulee konfiguroida oletustietokanta (default), voidaan myös määrittää useampia tietokantoja. Kun yhdistetään muuhun taka-alan tietokantaan, kuten MySQL, Oracle tai PostgreSQL, lisä attribuutteja vaaditaan. Aivan kuten yllä olevassa esimerkissä on havaittavissa. (Settings. 2014)

Attribuutit:

NAME: Oletuksena tyhjä lanka (` `). Tulee käyttää tietokannan nimeä, mutta SQLiteä käytettäessä, koko tietokannan hakemiston polku ja tietokannan nimi tulee määrittää. Määritettäessä tulee käyttää aina kauttaviivaa, esimerkiksi

/opt/graphite/webapp/graphite/graphite.db.

ENGINE: Oletuksena tyhjä lanka (` `). Tietokannan taka-ala, jota käytetään. Sisäänrakennettuja taka-alan tietokantoja, joita voidaan käyttää:

- `django.db.backends.postgresql_psycopg2`
- `django.db.backends.mysql`
- `django.db.backends.sqlite3`

- `django.db.backends.oracle`

Voidaan käyttää muitakin taka-alan tietokantoja, joita ei toimiteta Djangon kanssa asettamalla ENGINE täysin pätevään polkuun.

USER: Oletuksena tyhjä lanka (` `). Käyttäjänimi, jota käytetään yhdistäessä tietokantaan. Ei käytetä tätä vaihtoehtoa SQLiten kanssa.

PASSWORD: Oletuksena tyhjä lanka (` `). Salasana, jota käytetään yhdistettäessä tietokantaan. Ei käytetä tätä vaihtoehtoa SQLiten kanssa.

HOST: Oletuksena tyhjä lanka (` `). Määritetään mitä isäntää käytetään yhdistettäessä tietokantaan. Tyhjäksi jättäminen merkitsee *localhostia*. Ei käytetä tätä vaihtoehtoa SQLiten kanssa.

PORT: Oletuksena tyhjä lanka (` `). Portti, jota käytetään yhdistettäessä tietokantaan. Tyhjä lanka tarkoittaa oletusporttia. Ei käytetä tätä vaihtoehtoa SQLiten kanssa

Muita vaihtoehdollisia attribuutteja:

ATOMIC_REQUEST (False|True): Oletuksena “false” eli epätosi. Mikäli tosi “true” eli tosi. Jokaisen HTTP pyynnön toimitus tässä tietokannassa paketoidaan.

AUTOCOMMIT (False|True): Oletuksena “true” eli tosi. Mikäli “false” eli epätosi. Poistetaan Djangon toimituksen hallinta ja implementoidaan oma valinta.

CONN_MAX_AGE: Oletuksena “0”. Tietokannan yhteyden elinikä sekunteina. Mikäli käytetään arvoa “nolla” sulkeakseen yhteys jokaisen pyynnön lopussa. “None” asetusta loputtomiin yhtämittaisiin yhteyksiin.

TEST: Oletuksena “{ }”. Testi tietokannan hakemiston asetukset.

(Settings. 2014)

6.4 Carbon

6.4.1 Carbon.conf

Tämä tiedosto löytyy oletuksena hakemistopolusta ***/opt/graphite/conf*** ja se on nähtävissä kokonaisuudessaan liitteessä 8. Tämä on Carbonin pää konfiguraatiotiedosto. Se määrittää asetukset jokaiselle Carbonin Daemonille. Tähän tiedostoon on tehty dokumentaatiota kommenttien avulla. Asetukset ovat segmentoitu jokaiselle Daemonille erikseen.

- Carbon-cache ohjautuu osion [cache] konfiguraatiossa
- Carbon-relay ohjautuu osion [relay] konfiguraatiossa
- Carbon-aggeregator ohjautyy osion [aggregaror] konfiguraatiossa

Kuitenkin suositeltavaa on jättää [cache] osio sellaisekseen, eikä muuttaa sen konfiguraatioita. (Davis, C. 2014)

6.4.2 Storage-schemas.conf

Tämä konfiguraatiotiedosto tarkentaa tallennettavien metriikoiden tallentamisen. Liitteessä 9 on ote käytetyn konfiguraation tiedosto kokonaisuudessaan. Se yhdistää metriikkapolkuja kuvioihin ja kertoo Whisperille mitä taajuutta ja miten tietopisteiden historiaa tallennetaan. On tärkeää muistaa tästä tiedostosta seuraavaa.

- Tässä tiedostossa voi olla monta osioista.
- Osiot sovelletaan järjestyksessä ylhäältä alaspäin.
- Kuviot ovat säännöllisiä lausekkeita. Ensimmäinen kuvio, joka sopii metriikan nimeen, käytetään.
- Säilyttäminen asetetaan heti, kun ensimmäinen metriikka lähetetään.

- Tämän tiedoston muokkaaminen ei vaikuta jo luotuihin .wsp – tiedostoihin. Tämä voidaan muuttaa Whisperin resize.py tiedostossa.

(Davis, C. 2014)

Esimerkki konfiguraatiotiedostosta

[everything_1min_12months]
priority = 100
*pattern = .**
retentions = 1m:365d

Säännöt, joita käytetään, koostuvat kolmesta rivistä:

- Nimi, määritelty hakasuluissa
- Regex, määritelty “pattern=” attribuutin jälkeen
- Muistiin säilyttäminen määritellään “retentions=” attribuutin jälkeen

Muistiin säilyttämisen riville voidaan määrittää useita ajallisia attribuutteja. Nimi [everything_1min_12months] on pääasiallisesti dokumentointi tarkoitusta varten, ja esiintyy *creates.log* – tiedostossa, kun metriikka luodaan, joka sopii tähän osioon. Jokainen muisti *taajuus:historia* erotellaan kaksoispisteellä. Taajuudet ja historiat määritellään käyttämällä seuraavia suffikseja:

- s – ”second”, sekuntti
- m – ”minute”, minuutti
- h – ”hour”, tunti
- d – ”day”, päivä
- y – ”year” vuosi

(Davis, C. 2014)

6.5 Collectd.conf

Tämä konfiguraatiotiedosto kontrolloi, kuinka järjestelmän statistiikan keräämisen käytettävä Collectd Daemon käyttäytyy. Kaikista merkittävin vaihtoehto on LoadPlugin, joka hallitsee mitä liitännäisiä ladataan. Nämä liitännäiset määrittävät lopullisen Collectd:n käyttäytymisen. Tämän konfiguraatiotiedoston syntaksi on samankaltainen, kuin mitä suositussa verkkopalvelin sovelluksessa Apachessa. (Foster, F. n, d)

Jokainen rivi sisältää vaihtoehdon, avaimen ja listan yhdestä tai useammasta arvosta, tai tietyn osion alun ja lopun. Tyhjät rivit ja kaikki mitkä tulevat hash "#" - symbolin, eli kommentoidaan, jälkeen jätetään huomioimatta. Avaimet ovat noteerattomia lankoja, jotka koostuvat aakkosnumeerista merkeistä ja alaviiva "_" - merkistä. Avaimia kohdellaan tapauskohtaisesti Collectd:n toimesta ja kaikki liitännäiset, jotka siihen kuuluvat. Arvot voivat olla noteerattomia lankoja, noteerattuja lankoja, numero tai boolean ilmaisu. Noteeraamattomat langat muodostuvat ainoastaan aakkosnumeerista merkeistä ja alaviivoista ja niitä ei tarvitse noteerata. Noteeratut langat ovat suljettuina kaksois noteerauksella ("). Numerot voidaan määritellä desimaaleilla ja liukulukuesityksellä, käyttämällä pistettä "." desimaalien erottajana, HEXA desimaalia käyttämällä 0x etuliitteellä tai oktaalia johtavalla nollalla (0). Boolean arvot ovat joko totta (true) tai tarua (false). (Foster, F. n, d)

Konfiguraatiot luetaan ja käsitellään järjestyksessä, ylhäältä alaspäin. Joten liitännäiset ladataan siinä järjestyksessä, kuin ne on listattuna konfiguraatio tiedostossa. Hyvänä periaatteena pidetään, että ladataan mitkä tahansa *log* - liitännäiset ensimmäisenä, seuraamaan viestintää liitännäisiltä konfiguraation aikana. Myös LoadPluging vaihtoehto tulee esiintyä ennen asianmukaista <plugin…> blokkia.

Esimerkki pääpalvelimelta

```
##############################################################
# Global                                                     #
#------------------------------------------------------------#
# Global settings for the daemon.                            #
##############################################################

Hostname    "localhost"
```

```
#FQDNLookup   true
#BaseDir     "/opt/collectd/var/lib/collectd"
#PIDFile     "/opt/collectd/var/run/collectd.pid"
PluginDir   "/opt/collectd/lib/collectd"
TypesDB     "/opt/collectd/share/collectd/types.db"
#Interval     10
#Timeout      2
#ReadThreads  5
```

Attribuutit:

Hostname, nimi: Asettaa isäntänimen, joka identifioi isännän. Mikäli vaihtoehto jätetään pois, tai kommentoidaan, isäntänimi päätetään käyttämällä *gethostname(2)* järjestelmä kutsua.

FQDNLookup (False|True): Mikäli isäntänimi määritellään automaattisesti. Tämä asetus hallinnoi pitäisikö Daemonin selvittää FQDN (Fully Quolified Domain Name). Tämä tapahtuu käyttämällä nimeä, jonka *gethostname* hakun palauttaa. Tämä vaihtoehto on oletuksena käytössä.

BaseDir, hakemisto: Asettaa perus hakemiston. Tämän hakemiston alle luodaan kaikki RRD-tiedostot. Mahdollisesti luodaan lisää alihakemistoja. Tämä on myös työskentely hakemisto Daemonille.

PIDFile, tiedosto. Asettaa minne PID tiedosto luodaan. Tämä tiedosto ylikirjoitetaan silloin kun se on jo olemassa ja poistetaan silloin kun ohjelma pysähtyy.

PluginDir, hakemisto: Polku Collectd liitännäisiin.

TypesDB, tiedosto [tiedosto…]: Asettaa yhden tai useamman tieto-asetus kuvauksen sisältävän tiedoston.

Interval, sekunneissa: Konfiguroi aikavälin, jossa kyselemällä luetaan liitännäiset. Itsestäänselvyytenä on, että mitä pienempi arvo sitä korkeampi järjestelmän kuormituksen Collectd aiheuttaa. Korkeammat arvot taas johtavat karkeampiin tuloksiin.

VAROITUS: Tämä arvo tulee asettaa vain kerran ja olla muuttamatta sitä. Mikäli muutetaan arvoa, tulee poistaa kaikki RRD tiedostot. Olettaen, että käytetään RRDTool tai RRDCacheD liitännäistä.

Timeout, toistot: Oletetaan, että arvolista on "kadoksissa" silloin kuin päivitystä ei ole luettu tai vastaanotettu toistojen toistoon. Oletuksena Collectd pitää arvolistaa hukkuneena, mikäli päivitystä ei ole vastaanotettu kahdella päivitys aikavälillä. Koska tämä asetus käyttää toistoja. Suurin sallittu aika ilman päivitystä riippuu aikaväli informaatiosta, jonka jokainen arvolista sisältää.

ReadThreads, numeerinen: Liitännäisiä lukevien lankojen lukumäärä. Oletusarvona on lukumäärä viisi. Tätä arvoa voidaan nostaa mikäli on useampi, kuin viisi liitännäistä ja näiden lukeminen vie paljon aikaa. Yleensä ne ovat ne liitännäiset, jotka suorittavat verkon-I/O toimintoja. Ei suositella asettamaan tätä arvoa suuremmaksi, kuin mitä takaisin tulevien rekisteröityjen kutsujen lukumäärä.

Muita vaihtoehdollisia attribuutteja:

AutoLoadPlugin (False|True): Mikäli asetettuna "false", epätosi ja on myös oletusarvo. Jokainen liitännäinen tulee ladata nimenomaisesti käyttämällä *LoadPlugin* asetusta. Mikäli kohdataan *<Plugin...>* blokki ja yhtään kokoonpano käsittelyn takaisin kutsua tälle liitäiselle ei ole vastaanotettu. Lokitetaan tällöin hälytys tapahtuma ja blokki jätetään huomioimatta.

Mikäli asetetaan "true" eli todeksi, nimenomaista *LoadPlugin* asetusta ei tarvita. Jokainen *<Plugin…>* blokki toimii välittömästi niin kuin se olisi asetettu *LoadPlugin* asetuksen mukaisesti. *LoadPlugin* asetukset vaaditaan silti liitännäisiin, jotka eivät tarjoa minkäänlaista konfiguraatiota.

Include, polku [kaava]: Jos *polku* viittaa tiedostoon, kyseinen tiedosto sisällytetään. Jos *polku* viittaa hakemistoon, rekursiivisesti kaikki tiedostot kyseisessä hakemistossa ja niiden alihakemistoissa sisällytetään. Alkaen versiosta 5.3, tämä voi myös olla oma blokkinsa, mihin vaikuttavat vaihtoehdot *Includen* käyttäytymiseen voidaan määrittää. Seuraavanlainen vaihtoehto sallitaan:

```
<Include "/etc/collectd.d">
  Filter "*.conf"
</Include>
```

Filter, kaava: Mikäli *fnmatch* - funktio on käytettävissä järjestelmässä, shell - tyyppinen villikorttinen *kaava* voidaan määrittää suodattamaan mitkä tiedostot sisällytetään. Esimerkiksi kaikki tiedostot, jotka ovat yhteensopivia **.conf* attribuutin kanssa, missä tahansa alikansiossa hakemistossa */etc/collectd.d*:

```
Include "/etc/collectd.d" "*.conf"
```

Mikäli useampi kuin yksi tiedosto sisällytetään yhdellä *Include* attribuutilla, tiedostot sisällytetään sanakirjamaisessa järjestyksessä. Kuitenkin voidaan käyttää numeerisia etuliitteitä määrittämään tiedostoille järjestys missä ne ladataan. Estääkseen silmukoita, pesintä on rajoitettu kahdeksaan eri tasoon, joiden pitäisi olla riittäviä yleiseen käyttöön.

WriteThreads, numero: Lankojen lukumäärä, jotka aloittavat liitännäisten lukemisen. Oletusarvona on viisi, mutta arvoa voidaan nostaa, mikäli on useampi kuin viisi liitännäistä joiden kirjoittaminen kestää suhteellisen kauan.

WriteQueueLimitHigh, korkea numero. Ja **WriteQueueLimitLow**, matala numero: Metriikoita luetaan *luku lankojen* toimesta ja asetetaan jonoon jossa niitä käsittelee *kirjoitus langat.* Mikäli yksi *kirjoitus liitännäinen* on hidas, tämä jono kasvaa. Jotta voitaisiin välttää muistiongelmiin kyseisessä tilanteessa, voidaan jonon kokoa rajoittaa.

Oletusarvona on, ettei ole minkäänlaista rajoitusta ja muistin käyttö voi kasvaa loputtomiin. Tämä ei ole ongelma itsessään asiakkaille, jotka käsittelevät ainoastaan paikallisia metriikoita. Palvelimille suositellaan asetettavaksi ei-nolla arvo.

Voidaan asettaa rajoituksia käyttämällä *WriteQueueLimitHigh:ta* ja *WriteQueueLimitLow:ta*. Jokainen näistä ottaa vastaan numeerisia argumentteja, joka on metriikan numero jonossa. Jos jonossa on *HighNum* metriikka, kaikki muut metriikat tiputetaan. Jos on kuitenkin vähemmän kuin *LowNum* metriikoita jonossa, kaikki uudet metriikat uudelleen asetetaan jonoon. Jos metriikoiden lukumäärä jonossa on *LowNumin* ja *HighNumin* välistä, kaikki metriikat tiputetaan todennäköisyydellä, joka vastaa jonossa olevien metriikoiden lukumäärää.

Mikäli *WriteQueueLimitHigh* asetetaan ei-nolla arvoon ja *WriteQueueLimitLow* ei ole määritetty, tällöin jälkimmäinen on oletuksena puolet *WriteQueueLimitHigh* arvosta. Mikäli ei haluta satunnaisesti tiputtaa arvoja jonossa jolloin sen koko on *LowNum* ja *HighNum* välistä, tulee *WriteQueueLimitHigh* ja *WriteQueueLimitLow* asettaa samaan arvoon.

LoadPlugin, liitännäinen: Lataa liitännäisen liitännäisen. Tämä vaaditaan, jotta voidaan ladata liitännäisiä, ellei AutoLoadPlugin vaihtoehto ole käytössä. Ilman yhtään ladattua liitännäistä, Collectd on käytännössä täysin hyödytön

Ainoastaan ensimmäinen *LoadPlugin* asetus tai blokki tietyllä liitännäisen nimellä on vaikutusta. Tämä on hyödyllistä silloin kun halutaan jakaa konfiguraatio pienemmiksi tiedostoiksi ja halutaan jokaisen tiedoston olevan itsenäinen. Toisin sanoen sisältää *liitännäisen* blokin ja sitten asianmukaisen *LoadPlugin* lausunnon. Huonona puolena taas on, mikäli on useita ristiriitaisia *LoadPlugin* blokkeja, esimerkiksi silloin kun ne on määritelty eri väliajoin, ainoastaan ensimmäinen joka kohdataan vaikuttaa ja muut jätetään huomioimatta.

LoadPlugin voi olla yksinkertainen konfiguraatio asetus tai blokki lisä vaihtoehdoilla, jotka vaikuttavat *LoadPluginin* käyttäytymiseen. Yksinkertaisesti asetus näyttää tältä:

LoadPlugin “cpu”

Vaihtoehdot *LoadPluginin* blokin sisällä voi yliajaa oletusarvot ja vaikuttaa siihen kuinka liitännäisiä ladataan, esimerkiksi:

```
<LoadPlugin perl>
  Globals true
 Interval 60
</LoadPlugin>
```

Seuraavat vaihtoehdot kirjataan <**LoadPlugin**> lohkojen kanssa

Globals (False|True): Mikäli käytössä, Collectd vie vientinä kaikki liitännäisen globaalit symbolit. Näin tekemällä symboleista saatavia, tuodaan ne ratkaisemaan ratkaisematta jääneitä symboleita lastattaviin liitännäisiin, mikäli se on tuettuna järjestelmässä.

Tämä on kätevää, esimerkiksi ladattaessa liitännäistä, joka upottaa hieman koodikieltä Daemoniin. Koodikielet yleensä tarjoavat keinoja ladata liitännäisiä, jotka on kirjoitettu C:llä. Nämä liitännäiset vaativat symbooleja, jotka tulkitsija tarjoaa, jotka ladataan vastaaviin Collectd:n liitännäisille riippuvuuksina.

Oletusarvona on, että tämä on poissa käytöstä. Erikoisena poikkeuksen, jos liitännäisen nimi on *perl* tai *python*, oletusarvo vaihdetaan vastakkaiseen. Jotta normaalin käyttäjän ei koskaan tarvitsisi työskennellä näin matalatasoisen linkitys jutun kanssa.

Interval, sekunneissa: Asettaa liitännäis-erityisen aikavälin kerättäville metriikoille. Tämä yliajaa globaalit asetukset. Jos liitännäinen tarjoaa omaa määrittelemään tukeaan aikavälille, kyseinen asetus astuu voimaan

6.6 Collectd liitännäiset

6.6.1 Liitännäinen LogFile

LogFile liitännäinen vastaanottaa loki viestejä Daemonilta ja kirjoittaa ne tekstitiedostoon. Jotta, muut liitännäiset voivat raportoida virheistä ja varoituksista alustuksen aikana, *LogFile* liitännäinen tulisi ladata yhtenä ensimmäisenä liitännäisenä, jottei aivan ensimmäisenä. Tämä tarkoittaa sitä, että sen *LoadPlugin* rivi tulisi olla yksi ensimmäisistä riveistä konfiguraatio tiedostossa. (Plugin:LogFile. 2009)

```
<Plugin logfile>
    LogLevel info
    File STDOUT
    Timestamp true
    # PrintSeverity false
</Plugin>
```

Attribuutit:

LogLevel (debug|info|notice|warning|err): Määrittää lokien tason. Mikäli, esimerkiksi asetetaan *notice*, niin kaikki tapahtumat *notice, warning,* tai *err* kirjoitetaan lokitiedostoon. Huomioitavaa *debug* vaihtoehdossa on, että se on avoinna vain jos Collectd on koottu vikasietotilan tuella.

File, tiedosto: Määrittää tiedoston mihin loki viestit kirjoitetaan. Erityisiä lankoja *stdout* ja *stderr* voidaan käyttää kirjoittamaan standardi ulostulo ja standard virhe kanavat, vastaavasti. Tässä on kuitenkin enemmän järkeä mikäli Collectd suoritetaan etuala- tai ei-Daemon-tilassa.

Timestamp (False|True): Kaikilla riveillä etuliitteenä nykyinen aika. Oletusarvona *true*, eli tosi.

PrintSeverity (False|True): Mikäli toteutetaan, kaikki rivit saavat etuliitteen vakavuuden mukaan, esimerkiksi “*warning*”. Oletuksena *false*, eli epätosi.

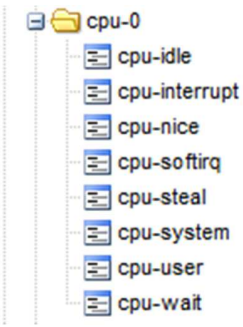
(Plugin:LogFile. 2009)

6.6.2 Liitännäinen CPU

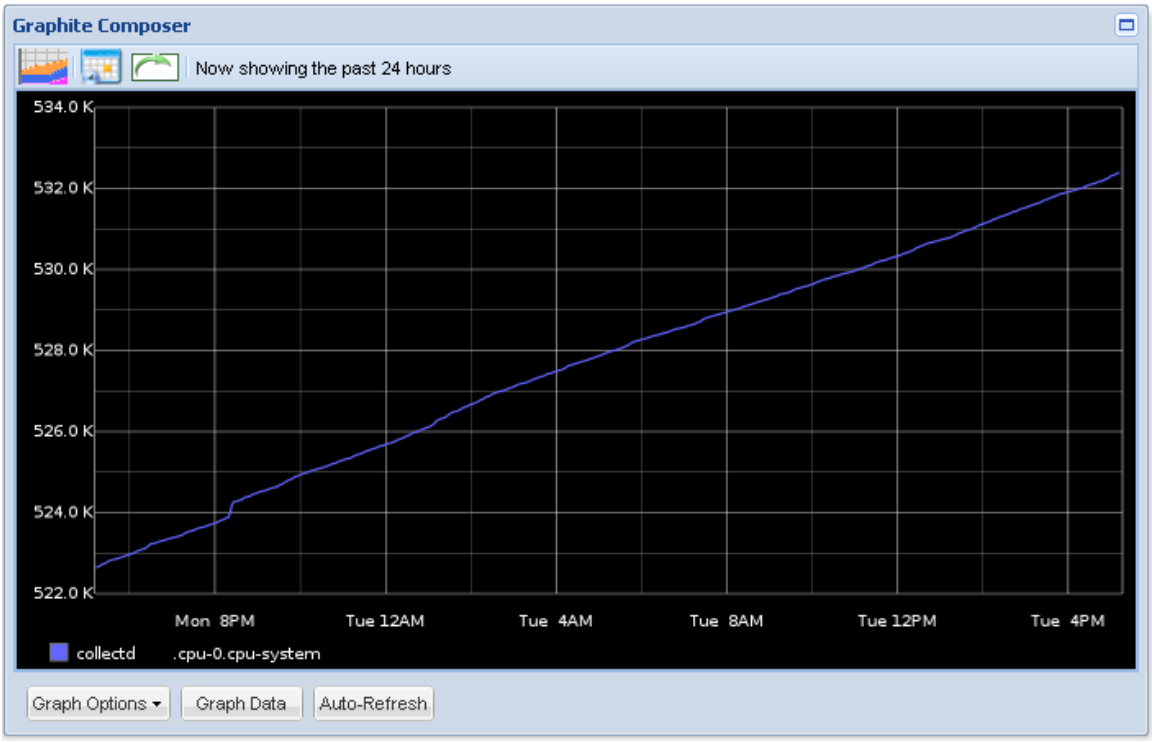
CPU liitännäinen kerää, kuinka kauan CPU käyttää aikaa erilaisissa tiloissa. Etenkin kun se toteuttaa käyttäjän koodia, järjestelmän koodia, odottaa I/O-tapahtumia ja on tyhjäkäynnillä. Tämä liitännäinen ei kerää prosenttiosuuksia. Vaan se kerää niin sanottuja *”jiffeja”* eli yksiköiden aikataulutusta. Monissa Linux järjestelmissä on noin 100 *”jiffiä”* sekunnissa, mutta tämä ei tarkoita, että päästäisiin prosenttiosuuksiin. Riippuen järjestelmän kuormasta, raudasta, onko järjestelmä tehty virtuaalisesti ja monista muista syistä, voi olla enemmän tai vähemmän, kuin 100 *”jiffiä”* sekunnissa. (Plugin:CPU. 2012)

Tässä liitännäisessä ei ole ainuttakaan vaihtoehtoa saatavilla. Se lukee seuraavaa polkua */sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq*, saadakseen nykyisen CPU taajuuden. (Foster, F. n, d)

Kuviossa 15 on esitettynä CPU näkymien hakemistopolku. Kuviossa 16 on esitettynä yksi näistä vaihtoehdoista ”CPU-system”. Toiminnollisuus valikossa on samanlainen kuin muissakin liitännäisten hakemistoissa, klikkaamalla näkymää voidaan valita ja poistaa se käytöstä. Useita näkymiä voidaan valita kerrallaan, mutta tämä saattaa vääristää näkymää.



Kuvio 15. CPU - hakemistopolku



Kuvio 16. CPU – kaavio

6.6.3 Liitännäinen DF

DF liitännäinen kerää informaatiota tiedostojärjestelmän käytöstä. Pohjimmiltaan, kuinka paljon tilaa on liitetyllä osiolla käytetty, ja kuinka paljon sitä on vapaana. Se on nimetty UNIX - komennon mukaan. Kuitenkin kaikki osiot, eivät välttämättä ole kiinnostuksen kohteena. Esimerkiksi */proc* ja */dev* eivät yleensä täyty ja niiden koossa ei yleensä ole paljon järkeä. Tästä syystä DF liitännäinen tarjoaa valinnan tietyille laitteille, liitäntäkohdille tai tiedostojärjestelmätyypeille. (Plugin:DF. 2013)

Esimerkki pääpalvelimelta

```
<Plugin df>
    Device "/dev/esimerkki"
    MountPoint "/"
    FSType "ext3"
#    IgnoreSelected false
#    ReportByDevice false
#    ReportReserved false
```

```
#       ReportInodes false
</Plugin>
```

Attribuutit:

Device laite: Valittu osio, liittyen laitteennimeen.

MountPoint hakemisto: Valittu osio perustuen asennettuun polkuun.

FSType: Valittu osio perustuen tiedostojärjestelmän tyyppiin.

IgnoreSelected (False|True): Käänteinen valinta: Jos asetetaan “true” eli todeksi. Kaikki osiot **paitsi** ne, jotka sopivat yhteenkin kriteeriin, kerätään. Oletuksena on, että ainoastaan valitut osiot kerätään, mikäli valinta suoritetaan. Jos yhtään valintaa ei ole konfiguroitu ollenkaan, kaikki osiot valitaan.

ReportByDevice (False|True): Raportoidaan käyttämällä laitteenimeä, ennemmin kuin liitäntäkohtaa. Toisin sanoen oletusarvolla “false”, se raportoi levykkeen juurena eli “roottina”, mutta “true” eli totena, se on “sda1” tai vastaava.

ReportReserved: Jos tämä vaihtoehto konfiguroidaan “false” - vaihtoehdoksi, on myös oletusarvo, vapaa levytila, joka on varattuna “root” käyttäjälle, raportoidaan vapaana. Se voi siis johtaa siihen, että käyttäjän prosessit eivät voi kirjoittaa osiolle huolimatta siitä, että kuvaaja näyttää jäljellä olevaa vapaata levytilaa.

ReportInodes (False|True): Otetaan käyttöön tai poistetaan käytöstä raportointi vapaista, varatuista ja käytetyistä inodeista. Oletuksena inode kerääminen on poistettu käytöstä. Käytetään tätä vaihtoehtoa, mikäli inodet ovat harvinaisia resursseja. Yleensä siksi, koska pienet tiedostot ovat varastoituna levylle. Tämä on yleinen skenaario postin lähettämisen agenteille ja verkko kätköille.

Muita vaihtoehdollisia attribuutteja:

ValuesAbsolute (False|True): Otetaan käyttöön tai poistetaan käytöstä raportointi vapaasta, käytössä olevasta ja käytetystä levytilasta 1k-lohkoina. Oletuksena on “true” eli tosi.

ValuesPercentage (False|True): Otetaan käyttöön tai poistetaan käytöstä raportointi vapaasta, käytössä olevasta ja käytetystä levytilasta prosentteina. Oletuksena on “false”. Tämä on hyödyllinen käyttöönottaessa Collectd pilvessä, missä koneita erikokoisilla levykkeillä voi olla olemassa. Joten tämä on paljon käytännöllisempää konfiguroida kynnysarvoihin, jotka perustuvat suhteelliseen levykokoon.

(Foster, F. n, d) (Plugin:DF. 2013)

6.6.4 Liitännäinen Disk

Disk liitännäinen kerää kovalevyn suorituskyvyn tilastotietoja ja niissä joissa tuetaan, osioiden. Vaikka “*oktetit*” ja “*operaatiot*” ovat suora suuntaisia, kaksi muuta tietosettiä vaatii aukaisemista:

- “*Yhdistää*” on määrä operaatioita, jotka ovat jo jonossa olevia operaatioita ja ne voidaan yhdistää toisiinsa, toisin sanoen yksi fyysinen levylle pääseminen palvelee kahta tai useampaa loogista operaatiota.

- “Aika” on I/O operaatioiden suorituksen keskimääräinen aika.

(Plugin:Disk)

Esimerkki palvelimelta

```
#<Plugin disk>
#      Disk "/^[hs]d[a-f][0-9]?$/"
#      IgnoreSelected false
#</Plugin>
```

Attribuutit:

Käyttämällä kahta seuraavaa vaihtoehtoa voidaan sivuuttaa jotkin levykkeet tai konfiguroida kerääminen vain määriteltyihin levykkeisiin.

Disk, nimi: Valitaan levykkeen *nimi*. Onko se kerätty tai sivuutettu riippuu täysin *IgnoreSelected* asetuksesta. Aivan kuten liitännäiset, jotka käyttävät Daemonin sivuutuslistan toiminnollisuutta, lanka, joka alkaa ja päättyy kauttaviivaan tulkitaan säännöllisenä lausekkeena. Esimerkiksi:

Disk “sdd”
Disk “/hda[34]/”

IgnoreSelected (False|True): Asettaa onko valitut levykkeet ne, jotka on määriteltynä *Disk* ilmaisussa, jotka sivuutetaan tai kaikki muut levykkeet sivuutetaan. Käyttäytyminen on toivottavasti intuitiivinen: mikäli ainuttakaan *Disk* vaihtoehtoa ei konfiguroida, kaikilta levykkeiltä kerätään tietoa. Mikäli annetaan edes yksi *Disk* vaihtoehto ja ei ole asetettu *IgnoreSelected* tai asetettu *false*, eli epätosi, ainoastaan täsmäävät levyt kerätään. Mikäli *IgnoreSelected* asetetaan *true*, eli todeksi, kaikki levyt kerätään paitsi ne jotka täsmäävät.

6.6.5 Liitännäinen Interface

Rajapinta liitännäinen kerää liikenteen informaatiota, kuinka paljon oktetteja liikkuu sekunnissa, paketteja sekunnissa ja rajapinnan virheitä, tietenkin montako virhettä tapahtuu sekunnissa. (Plugin:Interface. 2014)

Esimerkki Pääpalvelimelta

```
<Plugin interface>
    Interface "eth0"
    IgnoreSelected false
</Plugin>
```

Attribuutit:

Interface rajapinta: Valitaan kyseinen rajapinta. Oletusarvona on, että nämä rajapinnan asetuksen jälkeen kerätään.

IgnoreSelected (False|True): Jos tätä konfiguraatiota ei anneta, Liikenne - liitännäinen kerää tiedot kaikilta rajapinnoilta. Tämä ei välttämättä ole käytännöllistä, erityisesti loopback - tai vastaaville rajapinnoille. Joten voidaan käyttää Interface - vaihtoehtoa rajapinnoille, joita halutaan valvoa. Kuitenkin helpompaa on kerätä tietoa kaikilta rajapinnoilta kuin mitä muutamamilta tietyiltä. Tämä vaihtoehto antaa mahdollisuuden tehdä juurikin niin. Asettamalla IngnoreSelection “true”, eli todeksi, vaikutus rajapinnoilla on käänteinen; Kaikki valitut rajapinnat jätetään huomioimatta ja muilta rajapinnoilta kerätään tietoa.

(Foster, F. n, d)

6.6.6 Liitännäinen Load

Liitännäinen Load kerää järjestelmän kuormituksen tietoa. Numeeriset arvot antavat karkean käsityksen koneen käyttöasteesta, vaikka niiden tarkoitus on yleensä yliarvostettua. Järjestelmän kuorma on määritelty käynnissä olevien tehtävien määräksi juoksujonossa ja on tarjottavissa monissa käyttöjärjestelmissä yhden, viiden tai viidentoista minuutin keskiarvolla. (Plugin:Load. 2009)

6.6.7 Liitännäinen Memory

Memory liitännäinen kerää fyysisen muistin käyttöä. Arvot raportoidaan käyttöjärjestelmän käytön mukaan. Linuxin alla kategoriat ovat:

- Käytetty
- Puskuroitu
- Välimuistissa
- Vapaana

Vapaa muisti on muistia, josta maksetaan. Se kuluttaan energiaa ja eikä tee mitään hyödyllistä. On normaalia, että käyttöjärjestelmä asettaa tämän muistin hyödylliseen käyttöön. (Plugin:Memory. 2014)

6.6.8 Liitännäinen Network

Mikäli kerätään suorituskyvyn tietoja useammalta kuin yhdeltä isännältä, luultavasti halutaan, että tieto keskitetyssä paikassa sen sijaan, että sitä levitettäisiin tuhansille palvelimille. Juuri tähän tarkoitukseen *Network* - liitännäinen on olemassa. (Plugin:Network. 2010)

Network - liitännäinen kykenee lähettämään tietoja toiselle palvelimelle ja vastaavasti vastaanottamaan arvoja toiselta palvelimelta. Kumman toiminnon palvelin tekee, riippuu se konfiguraatiosta. Alkaen versiosta 4.7 *Network* - liitännäinen tarjoaa mahdollisuuden allekirjoittaa tai salata tietoliikenteen. (Networking introduction. 2012) (Plugin:Network. 2010)

Esimerkki Pääpalvelimelta

```
<Plugin network>
#     # client setup:
#     Server "ff18::efc0:4a42" "25826"
#     <Server "239.192.74.66" "25826">
#           SecurityLevel Encrypt
#           Username "kayttaja"
#           Password "salasana"
#           Interface "eth0"
#     </Server>
#     TimeToLive "128"
#
#     # server setup:
#     Listen "ff18::efc0:4a42" "25826"
#     <Listen "46.4.14.203" "25826">
#           SecurityLevel Sign
#           AuthFile "Kotihakemisto/passwd"
#           Interface "eth0"
#     </Listen>
```

```
    Listen "0.0.0.0"

#     MaxPacketSize 1024
#
#     # proxy setup (client and server as above):
#     Forward true
#
#     # statistics about the network plugin itself
#     ReportStats false
#
#     # "garbage collection"
#     CacheFlush 1800
</Plugin>
```

Attribuutit:

Server ja **Listen**:ia voidaan molempia käyttää yksittäisenä valintana tai lohkona. Käytettäessä lohkona, annetut valinnat ovat päteviä vain tässä kannassa

Server, isäntä [portti]: Palvelimen tiedonanto/lohko asettaa mihin palvelin lähettää tietopaketteja. Tiedonanto voi esiintyä useampaan kertaan, lähettääkseen jokaisen tietosähkeen useampaan kohteeseen. Argumentti Isäntä voi olla isäntänimi, IPv4-osoite tai IPv6-osoite. Vaihtoehtoinen toinen argumentti määrittää portin numeron tai palvelun nimen. Mikäli ei määritetä, oletusarvona on porttinumero 25826.

Seuraavat vaihtoehdot kirjataan **<Server>** lohkojen kanssa:

> **SecurityLevel** (Encrypt|Sign|None): Asettaa verkkoyhteydelle turvallisuusasetuksen. Kun turvallisuustaso asetetaan *enrcypt*, tieto lähetetään verkon ylitse käyttämällä AES-256 salausta. Salattujen pakettien eheys varmistetaan SHA-1:lla. Kun turvallisuustaso asetetaan *Sign*, lähetetty tieto allekirjoitetaan käyttämällä HMAC-SHA-256 viestin autentikointikoodia. Kun turvallisuustaso asetetaan *None*, tieto lähetetään ilman minkäänlaista turvallisuutta.
>
> **Username**, käyttäjänimi: Asettaa lähetykselle käyttäjänimen. Palvelin käyttää tätä katsoakseen salasanan. *AutFile*:ssa on määritelty asetukset. Kaikki turvallisuustasot, paitsi *None* vaativat tämän asetuksen.

Password, salasana: Asettaa salasanan tälle kannalle. Kaikki turvallisuustasot, paitsi *None* vaativat tämän asetuksen.

Nämä toiminnollisuudet ovat saatavilla vain, jos *Network* kytke on linkattuna *libcrypt:*in kanssa

Interface, rajapinnan nimi: Asettaa IP-paketeille ulosmenevän rajapinnan. Tämä koskettaa ainakin vähintään IPv6 paketteja ja mikäli mahdollista myös IPv4 paketteja. Jos tämä vaihtoehto ei ole sovellettavissa, määrittelemättömissä tai olemattoman rajapinnan nimi on määritelty. Tällaisessa tilanteessa on oletus käyttäytymisenä, että annetaan kernelin valita asianmukainen rajapinta. Yksittäislähetykselle manuaalisesti valittu rajapinta on välttämätön vain harvinaisissa tapauksissa.

Listen, isäntä [portti]: Listen - lausunto sitoo rajapinnat. Kun useita lausuntoja löytyy, daemon yhdistää nämä useat rajapinnat. Argumentti *Host* voi olla isäntänimi, IPv4-osoite tai IPv6-osoite. Mikäli argumentti on ryhmälähetysosoite, Daemon liittyy ryhmälähetysryhmään. Vaihtoehtoinen *[portti]*, argumentti määrittää portin numeron tai palvelun nimen. Mikäli ei ole määritelty, oletuksen on portti 25826.

Seuraavat vaihtoehdot kirjataan **<Listen>** lohkojen kanssa

SecurityLevel (Encrypt|Sign|None): Asettaa turvallisuustason verkkoyhteydelle. Mikäli taso on asetettu *Encrypt* - tasolle, ainoastaan salattu tieto hyväksytään. Salattujen pakettien eheys varmistetaan SHA-1:lla. Mikäli taso on asetettu *Sign* - tasolle, ainoastaan allekirjoitettua ja salattua tietoa hyväksytään. Mikäli taso on asetettu *None* - tasolle, kaikki tieto hyväksytään. Mikäli annetaan *AuthFile* vaihtoehto, salatun tiedon salaus puretaan mikäli mahdollista.

Nämä toiminnollisuudet ovat saatavilla vain, jos *Network* - liitännäinen on linkattuna *libcrypt:*in kanssa.

AuthFile, tiedostonnimi: Asettaa tiedoston, jossa käyttäjänimet on kartoitettu salasanoiksi. Näitä salasanoja käytetään todentamaan allekirjoituksia ja verkonpakettien tiedon salauksen purkamiseen. Tämä on valinnaista mikäli *SecurityLevel* on asetettu *None*. Mikäli on asetettu, allekirjoitettu tieto todennetaan ja salatut paketit avataan. Muussa tapauksessa allekirjoitettu tieto hyväksytään ilman, että allekirjoitusta ja salattua tietoa voidaan purkaa. Muille turvallisuustasoille tämä vaihtoehto on pakollinen.

Tiedostomuoto on erittäin yksinkertainen. Jokainen rivi koostuu käyttäjänimestä, jonka perässä kaksoispiste, mistä tahansa määrästä välilyöntejä, jota seuraa salasana. Esimerkki tiedosto voisi näyttää tältä:

tunnus1: esimerkki1

tunnus2: esimerkki2

Interface, rajapinnannimi: Asettaa nimenomaisesti sisään tulevan rajapinnan IP-paketeille. Tämä koskettaa ainakin vähintään IPv6 paketteja ja mikäli mahdollista myös IPv4 paketteja. Jos tämä vaihtoehto ei ole sovellettavissa, määrittelemättömissä tai olemattoman rajapinnan nimi on määritelty. On tällaisessa tilanteessa oletus käyttäytymisenä, että annetaan kernelin valita asianmukainen rajapinta. Sisään tuleva liikenne hyväksytään vain, jos se saapuu määriteltyyn rajapintaan.

TimeToLive, 1-255: Asettaa lähetettävien pakettien TTL -arvon. Tämä pätee kaikkiin, yksittäis- ja ryhmälähetykseen, ja IPv4 ja IPv6 paketteihin. Oletuksena on, ettei tätä arvoa muuteta. Se tarkoittaa sitä, että useissa käyttöjärjestelmissä ryhmälähetys paketit lähetetään TTL arvolla 1.

MaxPacketSize 1024-65535: Määrittää tietosähkeen maksimi koon verkon ylitse. Paketit, jotka ovat suurempia kuin asetettu arvo, katkaistaan. Oletusarvo on 1452 tavua. Palvelimen puolelta tämä raja-arvo tulisi olla suurempi kuin yhdenkään asiakkaan. Samoin,

kuten asiakkaan puolen arvon ei saisi olla suurempi kuin palvelimen puolen, tai tieto hävitetään.

Yhteensopivuus: versiot ennen versiota 4.8 käyttävät kiinteätä 1024 tavun puskurin kokoa. Lähtien versioista 4.8, 4.9 ja 4.10 käytetään oletusarvona 1024 tavua välttääkseen ongelmat tiedon lähettämisessä vanhemmille palvelimille.

Forward (False|True): Mikäli asetetaan “true” eli todeksi, kirjoitetaan paketit, jotka vastaanotettiin *Network* - liitännäisellä, lähetyskantaan. Tämä tuli aktivoida ainoastaa silloin kun *Listen* - ja *Server* - lausunnot eroavat. Muuten paketit voidaan lähettää useaan kertaa samalle ryhmälähetys ryhmälle. Vaikkakin tämä johtaakin suurempaan verkkoliikenteeseen, se ei ole ongelma, sillä liitännäisellä on duplikaatti tarkistus, joten arvot eivät joudu silmukkaan.

ReportStats (False|True): Network - liitännäinen ei kykene ainoastaan vastaanottamaan ja lähettämään statistiikkaa, se kykenee luomaan statistiikkaa myös itsestään. Kerätty tieto sisältäen lähetettyjen ja vastaanotettujen oktettien ja pakettien määrä, vastaanottojonon pituus ja käsiteltävien arvojen määrä. Mikäli asetetaan “true” eli tosi, *Network* - liitännäinen luo kyseiset statistiikat saataville. Oletusarvona on “false” eli epätosi.

(Foster, F. n, d)

6.6.9 Liitännäinen Processes

Processes liitännäinen kerää useita prosesseja niiden tilan perusteella ryhmiin, esimerkiksi juoksevat, nukkuvat, horroksessa jne. Sen lisäksi voidaan valita tarkempaa statistiikkaa valituista prosesseista, ryhmiteltynä nimellä. (Plugin:Processes. 2010)

Esimerkki palvelimelta

```
#<Plugin processes>
#      Process "name"
#      ProcessMatch "foobar" "/usr/bin/perl foobar\\.pl.*"
#</Plugin>
```

Process, nimi: Valitaan määritellyn nimen perusteella tarkempaa statistiikkaa prosessista. Näistä valituista prosesseista kerätty statistiikka on RSS:n koko, CPU:n käyttö, prosessien ja lankojen lukumäärä, I/O tieto ja pienet ja suuret sivunpuutokset.

ProcessMatch, nimi, säännöllinen lauseke: samankaltainen *Process* vaihtoehdon kanssa. Tämä mahdollistaa tarkempien prosessien statistiikkojen

(Plugin:Processes. 2010)

6.6.10 Liitännäinen RRDTool

Huom! RRDTool on käytössä ainoastaan palvelimilla 2 ja 3. Siksi, että näillä palvelimilla ei ole Whisperia tuomassa omia ominaisuuksiaan, joilla se voisi korvata RRDTool:in heikkoudet.

RRDTool - liitännäinen on collectd:n yksi monimutkaisemmista liitännäisistä. Syy tähän on sen rakenteellinen ominaisuus toimia hyvin suurissa asetelmissa, joissa RRD - tiedostot aiheuttavat vakavia I/O ongelmia. (Inside the RRDtool plugin. 2012)

I/O Inferno

RRD - tiedostojen päivittäminen on intensiivistä, koska ainoastaan erittäin vähän tietoa kirjoitetaan jokaisessa päivityksessä ja paikkoihin joihin päästään, eivät ole peräkkäisiä. Niin kauan, kuin RRD - tiedostot mahtuvat muistiin, toiminnallisen järjestelmän kätkö suoriutuu hyvin ja yleisesti I/O ei ole tällöin ongelma. Mikäli kuitenkin on useampia tiedostoja, törmätään ongelmiin, koska HDD - levykkeet eivät ole hyviä toiminnollisuudeltaan satunnaisessa pääsyssä ja NAND-flash tekniikkaan perustuvat SSD - levykkeet eivät myöskään ole siinä hyviä vielä. (Inside the RRDtool plugin. 2012)

Kaikkia näitä pieniä, ei-juoksevia I/O - operaatioita viitataan joskus "*I/O Infernoksi*". Tavalla, jolla kovalevyt toimivat, on erittäin vaikeaa tällaisen käyttötavan kanssa ja niiden siirtonopeus voi tipahtaa kahteen megatavuun sekunnissa. Tietenkin voitaisiin kasvattaa

aikaväliä, jolla tietoa kerättäisiin, mutta se ei olisi kauhean käytännöllistä. (Inside the RRDtool plugin. 2012)

I/O Infernosta selviytyminen

Sen sijaan, että nostettaisiin aikaväliä tiedolle, jota kerätään, nostetaankin sen tiedon aikaväliä mitä kirjoitetaan levykkeelle. Yhden arvon päivittäminen RRD - tiedostossa, vaatii kahdeksan tavun kirjoittamisen tiedostoon. Näiden kahdeksan tavun muuttamiseen tiedostossa, yksi blokki on luettava levykkeeltä, päivitettävä ja kirjoitettava uudelleen levykkeelle. Oletetaan ettei laitteessa ole tarpeeksi muistia näiden kaikkien tiedostojen muistissa pitämiseen, tämän tiedoston päivittäminen 50 kertaan vaatii 50 luku- ja kirjoitusoperaatiota. (Inside the RRDtool plugin. 2012)

Päivittäessä 50:ntä arvoa samanaikaisesti, kirjoitetaan 400 tavua tiedostoon. Tähän, yksi tai kaksi blokkia tarvitsee lukea levykkeeltä ja kirjoittaa takaisin, johtaen enintään kahteen luku- ja kirjoitusoperaatioon. Toinen blokki on aivan ensimmäisen jälkeen, joten kirjoittamista ja lukemista ei oteta huomioon, koska se on sarjalukuinen ja täten erittäin nopea. (Inside the RRDtool plugin. 2012)

Kätkeäkseen arvoja muistiin, liitännäinen käyttää "*itsetasapainottuvaa binääristä hakupuuta*" (ks. kuvio 18). Jokainen solmu on sopusoinnussa RRD – tiedoston kanssa ja pitää arvoja, joita ei vielä kirjoitettu tiedostoon. Kun uusi arvo lisätään solmuun, liitännäinen vertailee aikaleimoja vanhemmasta arvosta, siihen aikaleiman arvoon, jota ollaan sijoittamassa. Mikäli eroavaisuus on liian suuri, solmun "ikä", solmu sijoitetaan *päivitettyyn jonoon*. (Inside the RRDtool plugin. 2012)

Mikäli solmu vastaanottaa arvon, joka on jo päivitetyssä jonossa. Se solmu, joka on *jonossa* päivitetään, jotta odottavat kirjoitusoperaatiot sisältäisivät uusimman arvon myös. Se ei kuitenkaan ole toteutettu näin, mutta solmu on joko *kätkössä* tai *jonossa*, muttei koskaan kummassakin yhtäaikaisesti. Erillinen *jonottavat langat* poistavat arvoja *päivitetystä jonosta* ja kirjoittamaan niitä tarkoituksenmukaiseen RRD – tiedostoon. (Inside the RRDtool plugin. 2012)

Mikäli arvot poistetaan *päivitetystä* jonosta korkeammalla tahdilla, kuin mitä *jonottavat langat* pystyvät niitä poistamaan ja kirjoittamaan RRD – tiedostoihin, uudet arvot menevät solmuihin, jotka on jo poistettu ja useat arvot yhdistetään yhteen päivitykseen RRD-tiedostoa varten. Joten vaikka laitteisto ei pysyisikään kaiken tiedon perässä, jota halutaan kirjoittaa levykkeelle, Collectd kykenee ja toimii dynaamisesti kasvamana puskurina arvojen ja RRD – tiedostojen välillä levykkeellä. (Inside the RRDtool plugin. 2012)

I/O Infernosta karkaaminen

Järjestelmän pitäisi pystyä melkein käsittelemään mielivaltaisen määrän tietoa, mutta *jonottavat langat* suorittaa toimintoja jatkuvasti, ja se on erittäin hyvä siinä mitä se tekee. Järjestelmä on kuitenkin kiireinen kirjottamaan RRD – tiedostoja, eikä sillä mahdollisesti kyetä tekemään oikein mitään muuta. Kaavioiden tuottaminen RRD – tiedostoista tällaisella järjestelmällä ei ole kuitenkaan mukavaa. (Inside the RRDtool plugin. 2012)

Mikäli *jonottavat langat* ei suorittaisi toimintojaan jatkuvasti, esimerkiksi, koska on asetettu aikalisälle korkea arvo, kaikki arvot yleensä saavuttavat oikean ”iän” samanaikaisesti. Kuvitellaan, että kaikki päivitykset jaettiin tasaisesti viidessä minuutissa, sen jälkeen kuin ne on kirjoitettu levykkeelle. Aamulla kuitenkin varmuuskopiointi kuitenkin suoritetaan ja I/O on paljon hitaampi. *Päivitetty jono* kasvaa ja, kun varmuuskopiointi on suoritettu, useimmat arvot saavuttavat aikalisä iän samanaikaisesti. (Inside the RRDtool plugin. 2012)

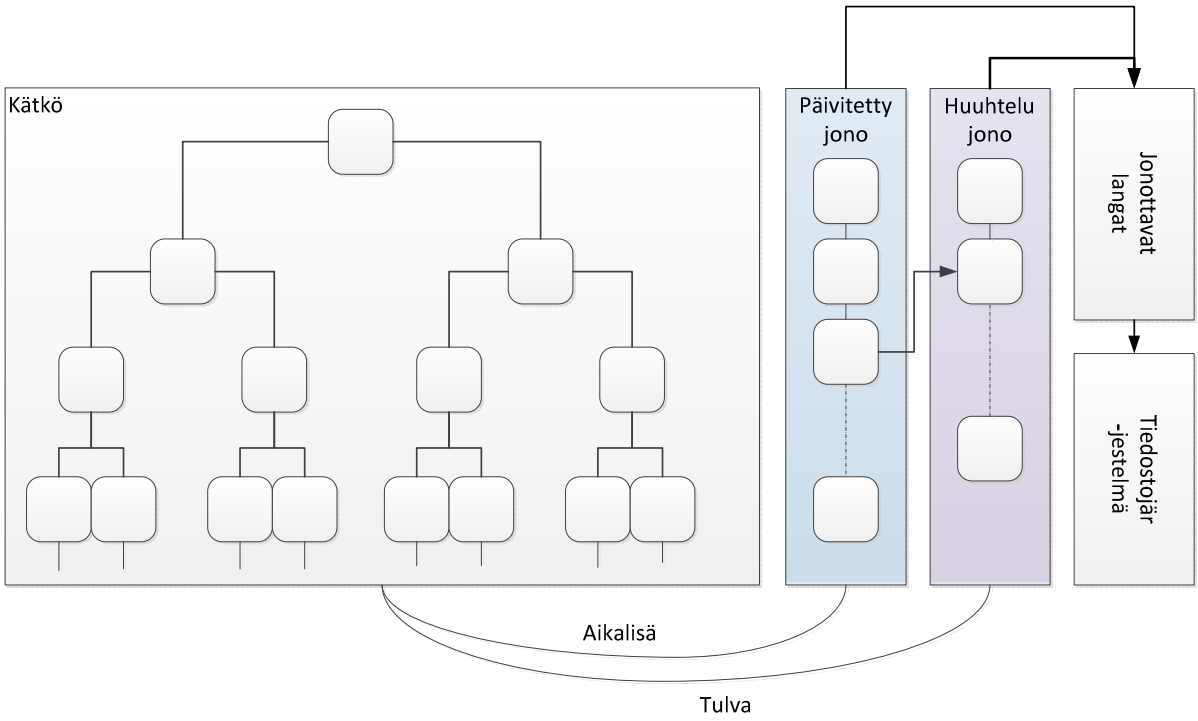
Ratkaisu tähän ongelmaan on kuristaa nopeutta, jolla RRD – tiedostot kirjoitetaan. Tämä ei kuitenkaan ole sitä mitä *RRDTool* – liitännäinen tekee. Koska nopeus, jolla solmuja poistetaan *päivitetystä jonosta* on rajoitettu, mutta kuitenkin sillä on samankaltainen vaikutus. Joten esimerkiksi, jos järjestelmä kykenee suorittamaan 100 päivitystä sekunnissa, kuristamalla liitännäinen 50 päivitykseen sekunnissa, vie täten kauemmin aikaa kirjoittaa arvoja levykkeelle, mutta järjestelmä säilyttää käytettävyytensä. (Inside the RRDtool plugin. 2012)

I/O Infernon hallitseminen

Edellinen kohta saattaa herättää ajatuksia, että tiedon näkeminen kestää nyt kauemmin, ennen kuin se on nähtävissä? Mitä hyötyä siis on korkeasta resoluutiosta, jos kestää tunteja saada tieto kaavioihin? Ei tietenkään mitään, ja tässä viimeinen käsite astuu mukaan, *huuhtelu*. (Inside the RRDtool plugin. 2012)

Huuhtelun ideana on, että arvojen suuruus on paljon suurempi kuin mitä, joku varsinaisesti katsoo tiedoista luotuja kaavioita. Miksi siis Daemonin tarvitsisi kirjoittaa järjestelmän tiedostoon kymmenen sekunnin välein, jos kaaviota katsastellaan vain muutaman kerran päivässä? Tuntuu siis järkevämmältä kirjoittaa järjestelmän tiedostoon vain silloin kuin sitä varsinaisesti tarvitaan. Juuri tätä *huuhtelu* varsinaisesti tarkoittaa. (Inside the RRDtool plugin. 2012)

RRDTool - liitännäiselle voidaan kertoa, että kirjoittaa kaikki arvot yhteen tiedostoon levykkeelle määritetyllä ajankohdalla, eli *huuhdella* arvot. Mikäli *HUUHTELU* – komento solmulle vastaanotetaan, se asetetaan *huuhtelu jonoon* (ks. kuvio 18). Jos solmu on jo *päivitys jonossa*, se poistetaan sieltä ja siirretään *huuhtelu jonoon*. *Jonottavat langat* käsittelee *huuhtelu jonoa* ehdottomalla prioriteetilla. Toisin sanoen solmut poistetaan *päivitys jonosta*, mikäli *huuhtelu jono* on tyhjä. Tämän takia *päivitys jonon* poistamista voidaan rajoittaa: Kaikki tiedostot, jotka *huuhdeltiin*, kirjoitetaan levykkeelle korkeimmalla mahdollisella nopeudella, joka ei riipu *päivitys jonon* nopeusrajoituksesta. (Inside the RRDtool plugin. 2012)



Kuvio 17. Itsetasapainottuva binäärinen hakupuu

Esimerkki palvelimelta

```
<Plugin rrdtool>
    DataDir "/var/lib/collectd/rrd"
#     CacheTimeout 120
#     CacheFlush 900
#     WritesPerSecond 30
#     RandomTimeout 0
#
# The following settings are rather advanced
# and should usually not be touched:
#     StepSize 10
#     HeartBeat 20
#     RRARows 1200
#     RRATimespan 158112000
#     XFF 0.1
</Plugin>
```

Attribuutit:

DataDir, hakemisto: Määrittää hakemiston, jonka alle RRD tiedostot tallennetaan. Oletusarvona on, että tiedostot luodaan työskentelevän Daemonin alle.

Muita vaihtoehdollisia attribuutteja:

CacheTimeout, sekunneissa: Mikäli tämän vaihtoehdon arvo on suurempi kuin nolla, *RRDTool* liitännäinen tallentaa arvoja kätköihin. Kirjoittamalla useita arvoja yhtäaikaisesti vähennetään I/O operaatioita ja täten vähentää kuormaa tiedostojen päivittämisessä. Vaihtokauppana on kuitenkin, että kaaviot laahustavat perässä ja enemmän muistia käytetään.

CacheFlush, sekunneissa: *RRDTool* liitännäisen käyttäessä kätkentää se kirjoittaa arvoja tiettyyn RRD - tiedostoon, mikäli vanhin arvo on vanhempi kuin tai sama kuin määritetty arvo sekunteina. Mikäli joitakin RRD - tiedostoja ei päivitetä jostain tuntemattomasta syystä, esimerkiksi palvelin on sammunut, jotkin arvot voivat vielä löytyä kätkön sisältä. Mikäli *CacheFlush* asetetaan, koko kätkö tarkastellaan lävitse vanhempien arvojen kuin *CacheTimeout* asetusten mukaisesti ja kirjoitetaan levykkeelle määriteltyjen *sekuntien* mukaisesti. Tämä on kuitenkin arvokasta ja ei tee mitään normaaleissa tilanteissa. Tämä arvo ei saa olla liian pieni. Oletusarvoa *900* pidetään hyvänä arvona, toisaalta asettaessa tämän arvon suuremmaksikin, esimerkiksi arvoon 7 500 ei normaalisti tuota mitään harmia.

WritesPerSecond, päivitykset: Kerättäessä paljon tilastollista tietoa Collectd:lla ja *RRDTool* liitännäisellä, törmätään suorituskyky ongelmiin. *CacheFlush* asetus ja sisäinen päivitys jono varmistaa siitä, että Collectd jatkaa työskentelyään raskaan kuorman alla, mutta järjestelmästä saattaa tulla erittäin “välinpitämätön” ja hidas. Tämä on ongelma, mikäli kaavioita luodaan RRD - tiedostoista samalla koneelle.

Tämä asetus on suunniteltu erittäin laajalle toteutukselle. Tämän asetuksen arvo on asetettu 25 ja 80 päivityksen välille sekunnissa, riippuen komponenteista, jättää palvelimen tarpeeksi myötäiseksi piirtää kaavioita samalla, kun kätkettyjä arvoja kirjoitetaan levylle.

Huuhdellut arvot, eli arvot, jotka pakotetaan levylle *FLUSH* - komennon toimesta, ei ole vaikutusta tästä rajoituksesta. Ne kirjoitetaan niin nopeasti, kuin mahdollista, joten verkon käyttöliittymät ovat ajan tasalla, kun piirretään kaavioita.

RandomTimeout, sekunneissa: Mikäli asetetaan, varsinainen aikalisä jokaiselle arvolle valitaan satunnaisesti *CacheTimeout - RandomTimeout* ja *CacheTimeout + RandomTimeout* arvojen väliltä. Tarkoituksena on välttää korkeaa kuormaa tilanteissa, jotka esiintyvät, kun monien arvojen aikalisä on asetettu samaan aikaan. Tämä on ongelma erityisesti sen jälkeen, kun Daemon käynnistyy, koska kaikki arvot lisätään sisäiseen kätköön samanaikaisesti.

Seuraavat asetukset ovat edistyksellisempään toteutukseen ja näihin ei yleensä kosketa:

StepSize, sekunneissa: Määrittää askeleen koon uusiin RRD - tiedostoihin. Oletusarvona ja ihanteena on, ettei tätä asetusta asetettaisi ja askelkoko asetetaan aikavälille sille tiedolle, joka kerätään. Ei ole suositeltavaa käyttää tätä vaihtoehtoa mikäli ei ole erittäin hyvää syytä. Tämän asetuksen asettaminen saattaa aiheuttaa ongelmia *snmp* - , *exec* - liitännäisten tai Daemoniin, joka on asetettu vastaanottamaan tietoa muilta isänniltä.

HeartBeat, sekunneissa: Pakottaa “sydämmen lyönnit” uusiin RRD - tiedostoihin. Ei ole suositeltavaa käyttää tätä vaihtoehtoa mikäli ei ole erittäin hyvää syytä.

RRARows, NumeroRivit: RRDtool - liitännäinen laskee PDP:ten määrän jokaista CDP:tä kohden, perustuen *StepSize* arvoon. Pohjimmiltaan ei tule asettaa pienemmäksi kuin kuvioiden leveys pikseleissä. Oletusarvona on 1200.

RRATimespan, sekunteina: lisää RRA:n eliniän, annetaan sekunteina. Vaihtoehtoa käyttämällä useita kertoja saadaan enemmän kuin yksi RRA. Mikäli tätä vaihtoehtoa ei koskaan käytetä, sisäänrakennettuja oletuksia “3600, 86400, 604800, 2678400, 31622400” käytetään.

XFF, muuttuja: Asettaa "*XFiles Factorin*". Oletusasetuksena on 0.1. Mikäli on epävarmuutta käytöstä, ei ole suositeltavaa asettaa tätä vaihtoehtoa. Muuttujan tulee olla alueella 0.0 - 1.0. Nolla on kaikki sisältävä ja yksi on yksilöllinen.

(Foster, F. n, d)

6.6.11 Liitännäinen Write_Graphite

HUOM! On suositeltavaa asentaa pääpalvelimelle Collectd liitteen 3, "*lähteestä asentamisen*" mukaan, koska *Write_Graphite* – liitännäinen toiminta on parhaimmillaan uusimmassa versiossa (5.4+). Aikaisimmissa versioissa liitännäisen ominaisuudet ovat puutteellisia.

Write_Graphite - liitännäinen varastoi arvoja Carboniin, mikä on Graphiten varastointi kerros. Tämä kytkentä tähtää korkeaan kyvykkyyteen. Se pitää TCP - yhteyden Carboniin avoinna, tarkoituksenaan minimoida yhteyden kättelyn kustannuksia. Se puskuroi tietoa puskuriin lähettääkseen monta linjaa kerralla, mieluummin kuin generoimalla paljon pieniä verkkopaketteja. Puskurin koko 1428 tavua on mitoitettu niin, että puskuri mahtuu TCP ja IP otsikoiden kanssa yhteen ethernet kehykseen ja voidaan kuljettaa ilman pirstoutumista. (Plugin:Write Graphite. 2014)

Esimerkki Pääpalvelimelta

```
<Plugin write_graphite>
 <Carbon>
   Host "localhost"
   Port "2003"
   Prefix "esimerkki"
#    Postfix "esimerkki"
   StoreRates false | true
   AlwaysAppendDS false | true
   EscapeCharacter "_"
 </Carbon>
</Plugin>
```

Attribuutit:

Tämä konfiguraatio koostuu yhdestä tai useammasta **<Carbon>** lohkosta. Lohkon sisällä, seuraavat vaihtoehdot ovat kirjattavissa.

Host: Isäntänimi tai osoite mihin yhdistetään. Oletus on “localhost”

Port: Palvelun nimi tai portin numero mihin yhdistetään. Oletusportti on “2003”

Prefix lanka: Kun asetetaan, lanka lisätään isäntänimen eteen. Pisteet ja “valkoiset välit” eivät karkaa tässä langassa

Postfix lanka: Kun asetetaan, lanka esiintyy isäntänimessä. Pisteet ja “valkoiset välit” eivät karkaa tässä langassa

StoreRates (False|True): Mikäli asetetaan “true” eli todeksi, muunnetaan laskurin arvot asteeksi. Mikäli asetetaan “false” eli vääräksi, laskurin arvot varastoidaan sellaisina kuin ne ovat, eli kokonaislukuna.

AlwaysAppendDS (False|True): Mikäli asetetaan “true”, liitetään tiedon lähteen nimi, “Data Source” (DS) , metriikka tunnisteeseen. Mikäli asetetaan “false” , on myös oletusarvo. Tämä tehdään silloin kuin on useampi kuin yksi tiedon lähde.

EscapeCharacter: Carbon käyttää pistettä (.) pakenevana merkkinä ja ei anna tyhjän väliä tunnisteeseen. *EscapeCharacter* vaihtoehto määrittää mikä merkki pisteytetään, tyhjällä välillä ja hallinta merkit korvataan. Oletuksena alaviiva (_).

Muita vaihtoehdollisia attribuutteja:

Protocol: Protokolla jolla yhdistetään Graphiteen. Oletusarvo on TCP.

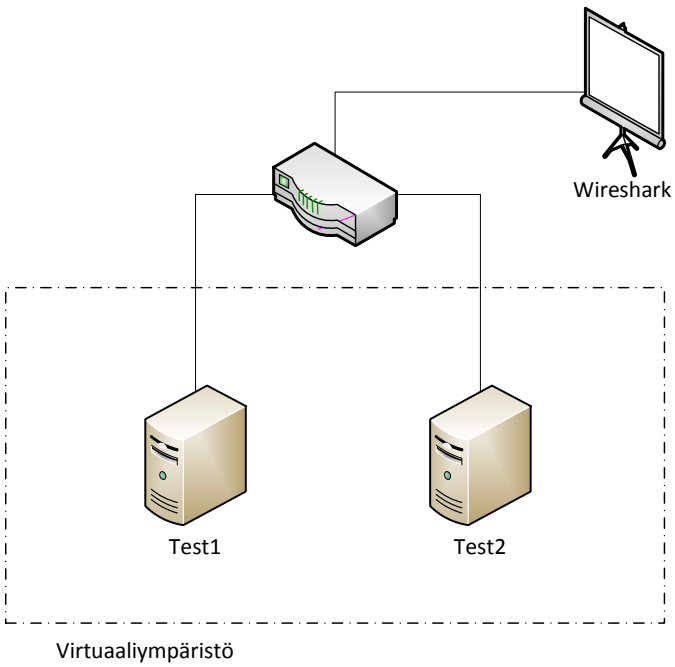
LogSendErros (False|True) : Mikäli asetetaan “true” eli todeksi, mikä on myös oletusarvo. Tällöin lähetettävän tiedon virheet lokitetaan. Mikäli asetetaan “false” , luonnollisesti virheitä ei lokiteta. Tämä on erittäin käytännöllistä silloin kuin käytetään UDP protokollaa, sillä monesti halutaan käyttää “tulita-ja-unohda” lähestymistä ja lokituksen virheen täyttäisivät syslog -tiedoston tarpeettomilla viesteillä.

SeperateInstances (False|True): Mikäli asetetaan “true”, eli todeksi kytkennän tapaus ja tyypin tapaus on niiden omissa polku komponenteissa, esimerkiksi *host.cpu.0.cpu.idle*. Mikäli “false”, on myös oletusarvo, kytkentä ja kytkennän tapaus asetetaan samaan komponenttiin, esimerkiksi *host.cpu-0.cpu-idle*

(Foster, F. n, d)

6.7 Collectd liikenteen toteaminen

Collectd:n käyttämän *Network* – liitännäisen lähettämän liikenteen toteaminen osoittautui hieman ongelmalliseksi todeta toteutetussa ympäristössä. Joten, jotta saataisiin vaivattomasti kaapattua liitännäisen lähettämää tietoa, toteutettiin testiympäristö. Testiympäristö on nähtävissä kuviossa 18. Ympäristössä on kaksi virtuaalikonetta *test1* ja *test2*. Koneille on toteutettu vastaava Collectd ympäristö, jossa *test1* toimii palvelimen roolissa ja *test2* asiakkaan roolissa. Tieto lähetetään *test2:lta* ryhmälähetyksenä, jotta voitaisiin kaapata se Wiresharkilla.



Kuvio 18. Virtuaaliympäristö

Konfiguraatio test1 palvelimelta

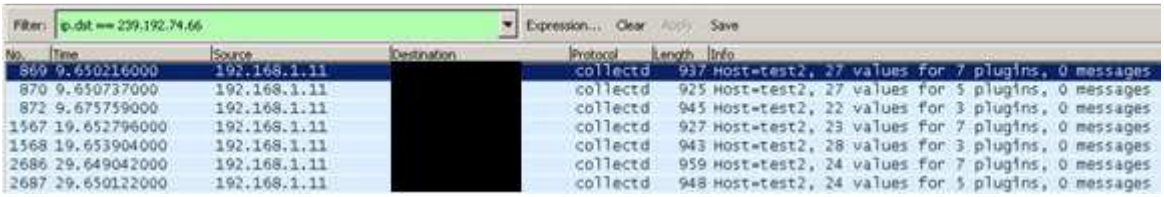
```
<Plugin network>
#     # server setup:
#     Listen "ff18::efc0:4a42" "25826"
    <Listen "239.192.74.66" "25826">
#           SecurityLevel Sign
#           AuthFile "Kotihakemisto/passwd"
#           Interface "eth0"
    </Listen>
</Plugin>
```

Konfiguraatio test2 asiakas koneelta

```
<Plugin network>
#     # client setup:
#     Server "ff18::efc0:4a42" "25826"
    <Server "239.192.74.66" "25826">
#           SecurityLevel Encrypt
#           Username "kayttaja"
#           Password "salasana"
#           Interface "eth0"
    </Server>
</Plugin>
```

Yläpuolella on kummankin virtuaalikoneen *Network* – liitännäisestä kaapatut konfiguraatiot. Kuten on nähtävissä konfiguraatiosta, *test1* on asetettu palvelimen rooliin. Yksinkertaisesti sille on vain kerrottu mitä ryhmälähetys osoitetta ja mitä porttia kuunnella. *Test2:n* konfiguraatiosta ilmenee selvästi, että se on asetettu asiakkaan rooliin. Sille on yksinkertaisesti kerrottu mitä ryhmälähetys osoitetta ja porttia käyttää tiedon lähettämisessä.

Kuviossa 19 on kaappaus liikenteestä. Ensimmäisenä kaappauksessa on suodatettu muu liikenne pois asettamalla ainoastaan ryhmälähetysosoite seurantaan. Siinä on nähtävissä mistä osoitteesta paketit tulevat ja mihin osoitteeseen niitä lähetetään. Protokollana on Collectd oma protokolla. Pakettien koko nähdään, sekä muuta informaatiota, jossa on kerrottu isäntä, josta tieto tulee (*Host=test2*), montako arvoa päivitetään tietylle määrälle liitännäisiä.



Kuvio 19. Wireshark - kaappaus liikenteestä

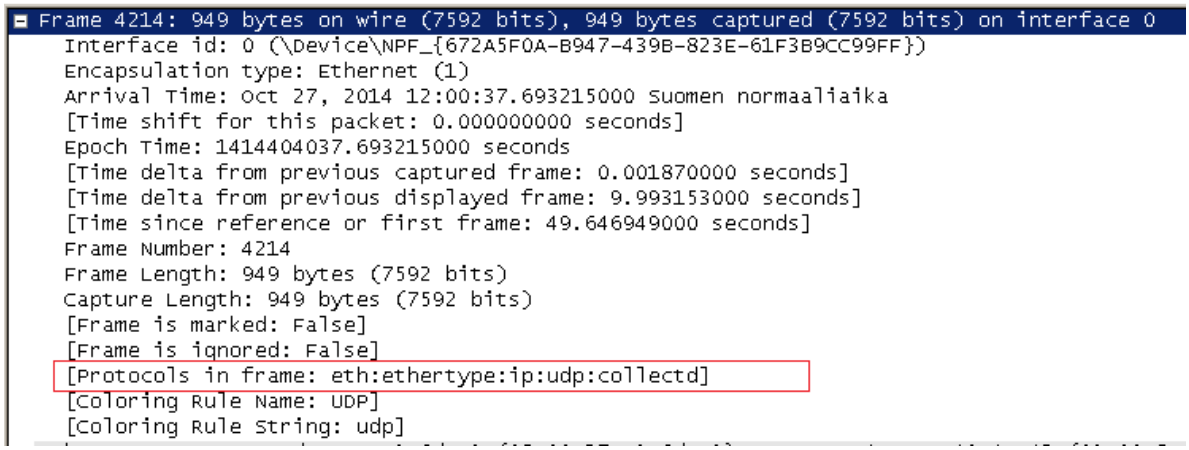
Kuviossa 20 on kaappaus yhdestä paketista. Kaappauksesta nähdään kehyksen koko. Kaappauksessa on avattuna *ethernet II* kohta, josta voidaan nähdä paketin olevan IPv4 ryhmälähetystä. Seuraava kohta kertoo, että käytetään IPv4 protokollaa. Sekä lähde- ja kohdeosoite. Käytetään UDP – protokollaa, jossa kerrotaan lähdeportti ja kohdeportti. Viimeisenä tulee Collectd:n paketti, jossa on sen keräämät tiedot, jotka lähetetään tässä paketissa.

Kuvio 20. Protokollat

Kuviossa 21 on kaappaus kehyksestä. Kaappauksessa on avattu kehyspaketti. Tärkeimpänä tässä kaappauksessa on *kehyksen protokollat*. Protokollat järjestyksessä ovat:

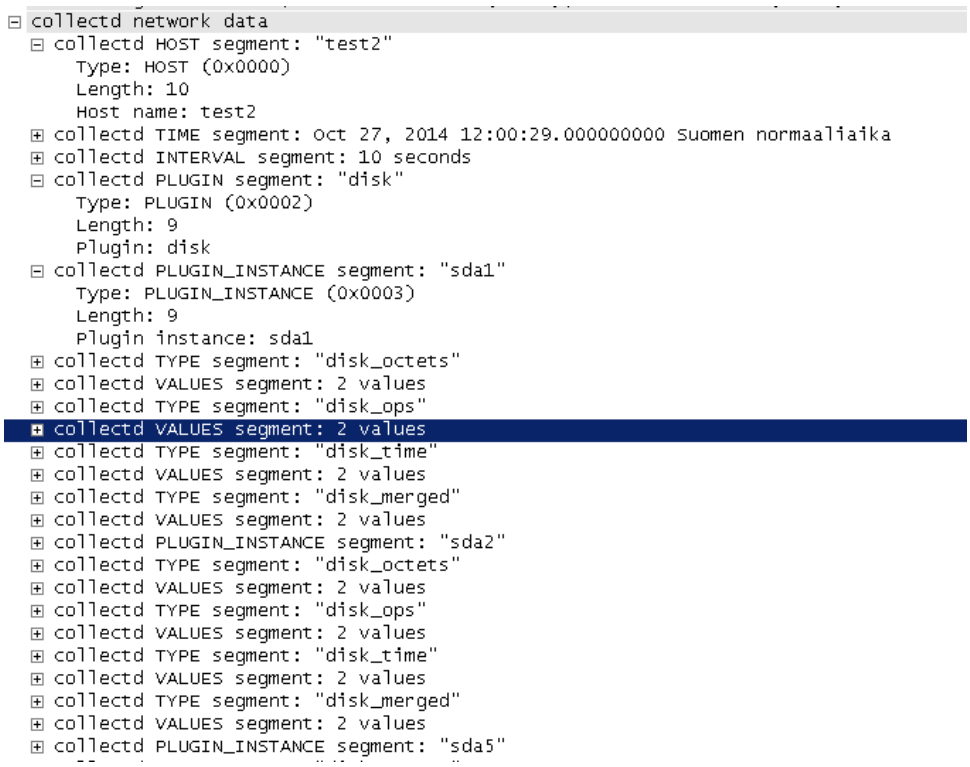
- Ethernet
- Ethernet tyyppi
- IP
- UDP
- Collectd

Paketti siis muodostuu näistä protokollista.



Kuvio 21. Kaapattu kehys

Kuviossa 22 on avattu Collectd:n lähettämä tieto paketissa. Kuten on nähtävissä, tieto lähetetään selkokielisenä, aivan kuten kaappauksesta on huomattavissa. Esimerkiksi Collectd isännännimi (*collectd HOST segment: ”test2”*) on selvästi luettavissa paketista.



Kuvio 22. Kaapatut collectd - arvot

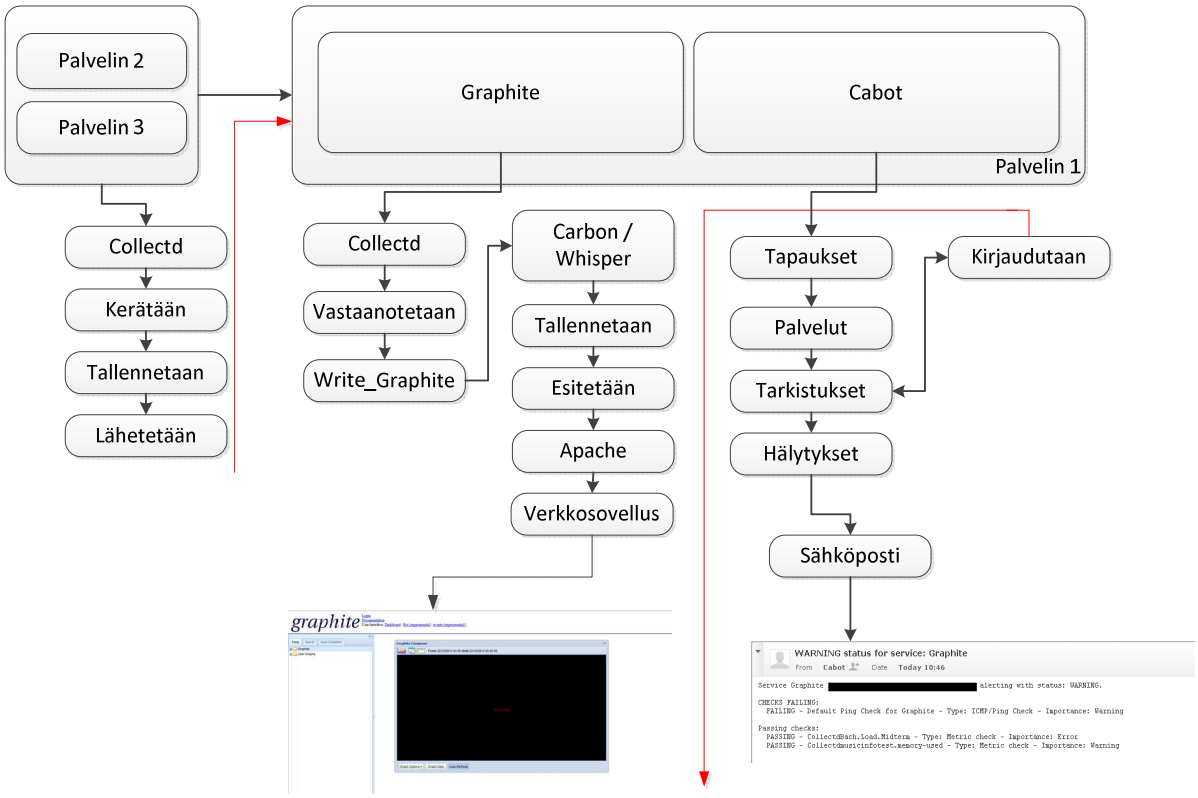
6.8 Cabot

6.8.1 Toteutus

Graphiten ja Collectd:n konfiguraation ja asetusten jälkeen tuotiin järjestelmään mukaan Cabot hälytysohjelma tuomaan mukaan hälytys ominaisuudet. Cabot asetettiin seuraamaan palvelimien CPU:ita, levyjen käyttöjä, kuormaa ja muistin käyttöä. Arvoista muodostettiin hälytykset, jotka lähetetään sähköpostilla.

yleiskuvallisesti kuvion 23 tarkoitus on esittää, kuinka Cabotin toiminnollisuus vaikuttaa koko järjestelmään ja, kuinka se on tuotu siihen mukaan. Kuten kuviossa 11, on kuviossa 23 esitetty palvelimet kaksi ja kolme, joilla kerääjänä toimii Collectd. Collectd kerää, tallentaa ja lähettää tietonsa pääpalvelimella, palvelin 1, olevalle Collectd:lle. Se vastaanottaa tiedot ja siirtää ne *Write_Graphite* liitännäisellä Carbonille. Carbon tallentaa ne Whisperiin, sekä lähettää ne Graphiten verkkosovellukselle, jossa ne esitetään kaavioina.

Cabotille on määritelty tapaukset, palvelut ja tarkistukset. Nämä tuovat hierarkkista toiminnollisuutta Cabotin rakenteeseen. Cabotille on määritelty käyttäjätunnukset, joilla se voi kirjautua Graphiteen. Näin Cabot voi hakea arvoja, joita tarkistusasetuksissa sille määritellään tarkastelemaan. Tarkistuksien perusteella muodostetaan hälytykselle asetukset. Hälytyksistä lähetetään automaattinen sähköposti pääkäyttäjille.



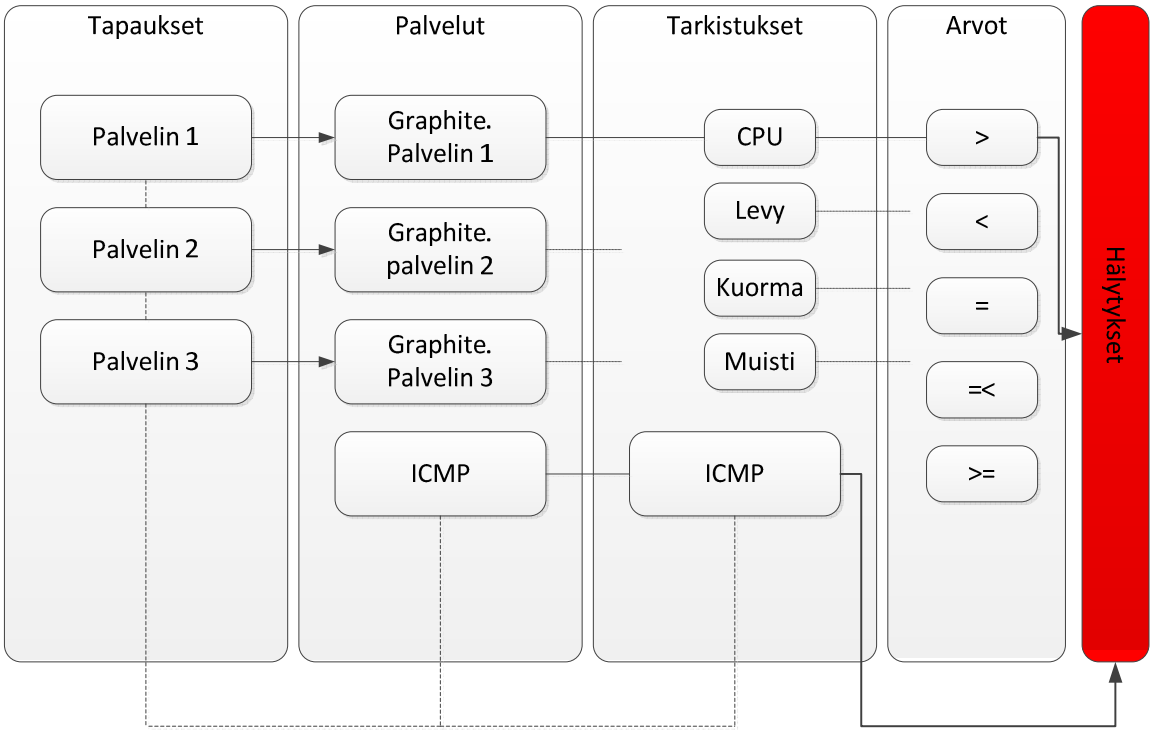
Kuvio 23. Cabotin toiminnallisuus

Cabot antoi mahdollisuuden toteuttaa hierarkkista suunnittelua monella eri tapaa. Päädyimme kuitenkin kuvion 24 mukaiseen hierarkia ajatteluun.

- **Tapaukset**, korkeimmalle tasolle suunnittelimme yksittäisesti kaikki palvelimet. Näin voitaisiin seurata palvelimia yksittäisesti, eikä koko kokonaisuutta. Jokaisen palvelimen alle määriteltiin sitä koskevien arvojen valvonta.

- **Palvelut**, keskimmäiselle tasolle suunnittelimme palvelin kohtaiset Graphiten arvojen tarkistukset. Liitimme tälle tasolle myös ICMP tarkistuksen, jolla valvotaan jokaisen palvelimen olemassaoloa ICMP viesteillä.

- **Tarkistukset**, nyt olimme hierarkkisesti määritelleet kaksi ylimäistä tasoa. Alimmalla tasolla on itse tarkistukset. Tarkistukset liitettiin palvelin kohtaiseen palveluun. Tarkistuksiin kuului kaikki palvelin kohtaiset Graphite arvot. Esim, palvelin ykkösen CPU arvojen seuranta. Graphitesta haetuille arvoille määriteltiin vielä hälytysrajat, joko suurempi, pienempi tai yhtä suuri kuin, joku tietty määritelty arvo. Näiden pohjalta syntyi hälytysarvot.



Kuvio 24. Cabot – jaottelu

6.8.2 Hälytysten toteutus

Edellisten määritysten jälkeen muodostettiin hälytykset, jotka perustuivat arvoihin, jotka haettiin Graphitesta. Näitä arvoja muokattiin muutamalla funktiolla tarpeille paremmiksi.

Funktioiden selitykset

derivative(seriesList), Vastakkainen integraalifunktion kanssa. Kätevä silloin, kun otetaan yhteenlasketut metriikan ja lasketaan delta myöhempien tietopisteiden väliltä.

sumSeries(seriesList), lyhyemmin sum(). Tämä laskeen kaikki metriikat yhteen ja palauttaa summan jokaisesta tietopisteestä.

asPercent(seriesList, total=None), Laskee prosentin villinkortinsarjoista. Mikäli "total" määritellään, jokainen sarja lasketaan prosentteina kyseiseen kokonaissummaan. Mikäli ei määritellä, kaikkien pisteiden summaa villinkortinsarjassa käytetään sen sijaan.

CPU

Funktio

sum(derivative(x..y))*

Muuttujat:
X = Isäntä, eli sen isännän nimi Graphitesta, jota halutaan seurata
Y = Muuttuja, se muuttuva arvo, jota halutaan seurata

Hakemistopolun tai muuttujan paikalle voidaan asettaa "*" - merkki ilmaisemaan, että haetaan kaikki mahdolliset hakemistopolut/muuttajat kyseisestä kohdasta

ESIM CPU

sum(derivative(isäntä1..cpu-user))*

Haetaan isännältä1 kaikkista CPU hakemistopoluista (ks. kuvio 25) CPU:t, joita kyseisellä isännällä on. Ja kaikki "cpu-user" -arvot, derivoidaan ne, ja lopulta lasketaan ne yhteen yhdeksi arvoksi.

Kuvio 25. Esim CPU

DF

Funktio

asPercent(x.z.z.y.free, sum (x.z.y.))*

Muuttujat:

X = Isäntä, eli sen isännän nimi Graphitesta, jota halutaan seurata

Z = Hakemistopolku Graphiten näkymässä

Y = Muuttuja, se muuttuva arvo, jota halutaan seurata

Hakemistopolun tai muuttujan paikalle voidaan asettaa "*" - merkki ilmaisemaan, että haetaan kaikki mahdolliset hakemistopolut/muuttajat kyseisestä kohdasta

ESIM DF

asPrecent(isäntä1.df.df-data.free, sum (isäntä1.df.df-data,))*

Haetaan isännältä1 (ks. kuvio 26) hakemistopolusta "*df.df-data*" kaikki vapaana oleva tila. Ennen sitä kuitenkin summataan samasta hakemistopolusta, sekä vapaa, että varattu tila. Verrataan summan arvoa vapaan tilan arvoon ja muutetaan lopuksi prosenttiarvoksi "*asPrecent*" funktiolla.

Kuvio 26. Esim DF

Kuorma

Funktio

x.z.z.y

Muuttujat:

X = Isäntä, eli sen isännän nimi Graphitesta, jota halutaan seurata

Z = Hakemistopolku Graphiten näkymässä

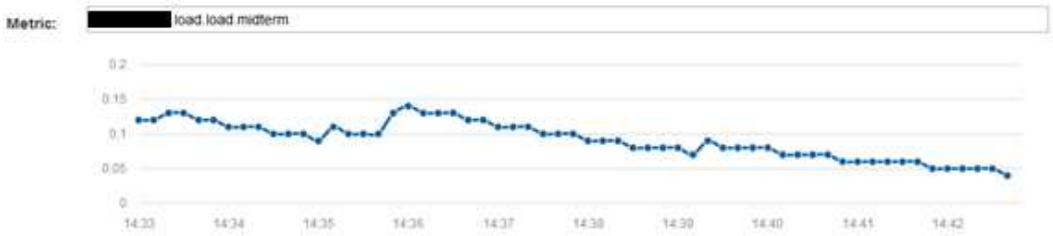
Y = Muuttuja, se muuttuva arvo, jota halutaan seurata

Hakemistopolun tai muuttujan paikalle voidaan asettaa "*" - merkki ilmaisemaan, että haetaan kaikki mahdolliset hakemistopolut/muuttajat kyseisestä kohdasta.

ESIM Kuorma

isäntä1.load.load.midterm

Haetaan isännältä1 (ks. kuvio 27) kuorman "midterm" – arvot



Kuvio 27. Esim kuorma

Muisti

Funktio

x.z.y

Muuttujat:

X = Isäntä, eli sen isännän nimi Graphitesta, jota halutaan seurata

Z = Hakemistopolku Graphiten näkymässä

Y = Muuttuja, se muuttuva arvo, jota halutaan seurata

Hakemistopolun tai muuttujan paikalle voidaan asettaa “*” - merkki ilmaisemaan, että haetaan kaikki mahdolliset hakemistopolut/muuttajat kyseisestä kohdasta

ESIM Muisti

isäntä1.memory.memory-used

Haetaan isännältä1 (ks. kuvio 28) käytetyn muistin käyttämän tilan arvot

Kuvio 28. Esim muisti

7 POHDINTA

Avoin lähdekoodi antaa järjettömän määrän mahdollisuuksia toteutuksiin. Yhteisön takia avoin lähdekoodi elää koko ajan, näin saadaan useita toimivia kokonaisuuksia. Voidaan poimia omaan tarpeeseen tarvittavat kokonaisuudet. Kuten myös Graphiten kohdalla tuli huomattua. Sen avoimen lähdekoodin tuomat ominaisuudet olivat huomattavan laajat ja antoivat mahdollisuuden, vaikka minkälaiseen ja monipuoliseen toteutukseen.

Graphite on suunniteltu todella suurille kokoonpanoille, joissa on satoja palvelimia. Näkisin se oivana keskitettynä valvontajärjestelmänä. Graphite antaa myös ominaisuuksia jaettuun keskitykseen, jossa voidaan suorittaa jaottelua todella suurissakin ympäristöissä.

Kuitenkin tullaan avoimen lähdekoodin kanssa sen suurimpaan heikkouteen. Sanonta ” mitä useampi kokki, sen huonompi soppa” tuntui kuvaavan Graphiten kokonaisuuden toteuttamista alkuvaiheessa. Liian monta eri toteutusta löytyi ohjeiden puolelta. Etsiskelyyn meni aikaa, ehkä turhankin paljon. Vaikka Graphiten wiki – sivusto tarjosikin oivalliset puitteet asennuksen kannalta, tarvitsi sitä kuitenkin käydä lukemassa mahdollisia ongelmatilanteita muiden toteutuksista. Voidaan sanoa, että ohjeet voisivat kaivata parempaa yksityiskohtaisuutta joidenkin asioiden kohdalla.

Voin myös sanoa, että Graphiten asennus on yksilöllistä, eli jokaisen palvelimen perusta voi erota huomattavasti toisistaan. Joten sanoisin, että liian monta liikkuvaa osaa on vielä huomioitava Graphiten asennuksen ja toteutuksen puolelta. Pelkkien riippuvuuksien kanssa taistelu tuottaa jo harmaita hiuksia. Jotkin näistä vaativat sen tietyn version, sillä joillakin ohjelmilla oli versioiden välillä suuriakin puutteita tai muutoksia.

Kaiken kaikkiaan lopputulos Graphiten kohdalla onkin aivan mainio, vaikka se onkin tarkoitettu suurille toteutuksille, en kuitenkaan näe miksi sitä ei voitaisiin käyttää pienemmissäkin toteutuksissa. Laajenemiselle on tällöin oivalliset puitteet. Graphiten yksi parhaimpia ominaisuuksia on sen laajentamisen mahdollisuus. Sille on annettu kasa sovelluksia, joita voidaan integroida toteutukseen ja tuoda vielä parempia kokonaisuuksia.

Tässäkin toteutuksessa vain kahdella Graphiten kanssa toimivalla sovelluksella tuotiin suurta arvoa toteutukseen. Eikä niiden tuominen mukaan Graphiten toteutukseen olleet aivan ylipääsemätöntä. Kuitenkin sovellusten eri versioiden kanssa painiminen tuli näissäkin sovelluksissa esille. Se, että toteutuksessa on yhdellä sovelluksella kaksi eri julkaisu versiota, kertoo monimutkaisesta hallitsemisesta tulevaisuudessa. On mustettava käsitellä näitä versioita yksityiskohtaisesti.

Collectd:n osalta voitaisiin sanoa sen olevan todella kätevä asiakassovellus. Collectd ratkaisi ongelman, jossa piti saada muilta palvelimilta vaivattomasti tiedot pääpalvelimelle. Se voidaan asentaa ja toteuttaa minimissään yhdellä komentorivin komennolla ja sillä on vain yksittäinen konfiguraatiotiedosto, joka on selkeä. Tulevaisuuden kannalta tämä on tärkeää, voidaan suhteellisen nopeasti ja helposti lisätä palvelimia kokonaisuuteen. Kuitenkin pääpalvelimen Collectd versio on käsiteltävä yksityiskohtaisemmin. Sen toimintaan laittaminen ”käsin” tuotti muutamia ylimääräisiä työtunteja, mutta oli todella sen arvoista.

Cabot hälytysjärjestelmä on yksinkertainen käyttää ja hallita. Kuitenkin avoimen lähdekoodin sovellukseksi silläkin on asennuksen ja toteutuksen puolella omat kiekuransa. Puhtaaseen uuteen ympäristöön toteutuksen toteuttamisen, en uskoisi olevan ongelma, mutta Musicinfon ympäristö oli jo monimutkainen. Musicinfon ympäristö sisälsi joitakin sovelluskokonaisuuksia, jotka toivat jo Cabotin toiminnollisuuteen vaativien sovellusten ominaisuuksia. Cabotin kohdalla jouduinkin tukeutumaan eniten toimeksiantajan apuun. Cabot käsitelläänkin vain ominaisuuksien ja toteutuksen kannalta tässä opinnäytetyössä, eikä mennä sen syvemmälle sen toteutuksessa. Halusin tuoda sen kuitenkin opinnäytetyöhön tästä huolimatta.

Toivonkin toteutetun ympäristön tuovan Musicinfolle tulevaisuudessa hyödyllistä ja elin tärkeää arvoa palvelimien valvonnassa ja hallitsemisessa. Toteutusta tehtäessä Musicinfon ympäristöön oli mukavaa ja tarpeiden mukaisesti tukea sai aina, kun sitä tarvitsi. Työ toi omaan kokemukseen suurta arvoa. Ei sitä usein pääse rakentelemaan aivan nollasta järjestelmää, joka tulee suoraan tuotantoon. Työssä pääsikin näkemään ongelmatilanteita ja haasteita, jotka ilmenevät suoraan tuotantoon tehdyille toteutuksille.

LÄHTEET

Apache. 2011. Artikkeli techterms.com sivustolla. Julkaistu 7.1.2011. Viitattu 20.10.2014. http://www.techterms.com/definition/apache

Beal, V. N, d. I/O – input/output. Artikkeli wepopedia.com sivustolla. Viitattu 31.10.2014. http://www.webopedia.com/TERM/I/I_O.html

Binary protocol. 2012. Artikkeli collectd.org sivustolla. Julkaistu 21.11.2012. Viitattu 27.10.2014. https://collectd.org/wiki/index.php/Binary_protocol

Cabot. 2014. Artikkeli github.com sivustolla. Julkaistu 19.9.2014. Viitattu 5.11.2014. https://github.com/arachnys/cabot/blob/master/README.md

Cairo. 2012. Artikkeli cairographics.org sivustolla. Julkaistu 27.3.2012. Viitattu 3.10.2014. http://cairographics.org/

Carbon. 2008. Artikkeli wikidot.com sivustolla. Julkaistu 7.5.2008. Viitattu 2.10.2014. http://graphite.wikidot.com/carbondaemons.html

Collectd – The system statistics collection daemon. N, d. Artikkeli collectd.org sivustolla. Viitattu 24.10.2014. https://collectd.org/index.shtml

CPU. 2014. Artikkeli techterms.com sivustolla. Julkaistu 11.7.2014. Viitattu 31.10.2014. http://www.techterms.com/definition/cpu

Daemon Definition. 2005. Artikkeli linfo.org sivustolla. Julkaistu 16.8.2005. Viitattu 3.11.2014. http://www.linfo.org/daemon.html

Davis, C. 2011. Overview. Artikkeli readthedocs.org sivustolta. Julkaistu 1.1.2011. Viitattu 1.10.2014. https://graphite.readthedocs.org/en/latest/overview.html

Davis, C. 2011. The Carbon Daemons. Artikkeli readthedocs.org sivustolla. Julkaistu 2.1.2011. Viitattu 1.10.2014. http://graphite.readthedocs.org/en/latest/carbon

Davis, C. 2014. Configuring Carbon. Artikkeli readthedocs.org sivustolla. Julkaistu 1.1.2014. Viitattu 20.10.2014. https://graphite.readthedocs.org/en/latest/config-carbon.html#carbon-conf

Davis, C. 2014. Graphite-web's local_settings.py. Artikkeli readthedocs.org sivustolla. Julkaistu 1.1.2014. Viitattu 22.10.2014. https://graphite.readthedocs.org/en/latest/config-local-settings.html

Davis, C. 2014. Installing Graphite. Artikkeli readthedocs.org sivustolla. Julkaistu 1.1.2014. Viitattu 6.10.2014. http://graphite.readthedocs.org/en/latest/install.html

Davis, C. 2014. Tools That Work With Graphite. Artikkeli readthedocs.org sivustolta. Julkaistu 1.1.2014. Viitattu 30.10.2014. https://graphite.readthedocs.org/en/latest/tools.html

Davis, C. N,d. Graphite. Artikkelin aosabook.org sivustolla. Viitattu 1.10.2014. http://www.aosabook.org/en/graphite.html

FAQ. 2009. Artikkeli wikidot.com sivustolla. Julkaistu 4.9.2009. Viitattu 2.10.2014. http://graphite.wikidot.com/faq

Features. N, d. Artikkeli collectd.org sivustolla. Viitattu 24.10.2014. https://collectd.org/features.shtml

Foster, F. N, d. Manpage collectd.conf(5). Artikkeli collectd.org sivustolla. Viitattu 7.10.2014.https://collectd.org/documentation/manpages/collectd.conf.5.shtml

I/O Systems. N, d. Artikkeli cs.uic.edu sivustolla. Viitattu 31.10.2014. http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/13_IOSystems.html

Inside the RRDtool plugin. 2012. Artikkeli collectd.org sivustolla. Julkaistu 30.3.2012. Viitattu 29.10.2014. https://collectd.org/wiki/index.php/Inside_the_RRDtool_plugin

Janssen, C. N, d. Input/Output (I/O). Artikkeli techopedia.com sivustolla. Viitattu 31.10.2014. http://www.techopedia.com/definition/24931/input-output-io

Kinder, K. 2005. Event-Driven Programming with Twisted and Python. Artikkeli linuxjournal.com sivustolla. Julkaistu 26.1.2005. Viitattu 3.11.2014. http://www.linuxjournal.com/article/7871

Meet Django. N, d. Artikkeli djangoproject.com sivustolla. Viitattu 3.11.2014. https://www.djangoproject.com/

Open Source. N, d. Artikkeli webopedia.com sivustolla. Viitattu 20.10.2014. http://www.webopedia.com/TERM/O/open_source.html

Overview. Chaplin, S. 2008. Artikkeli cairographics.org sivustolla. Julkaistu 1.1.2008. Viitattu 2.10.2014. http://cairographics.org/documentation/pycairo/3/overview.html

Plugin:CPU. 2012. Artikkeli collectd.org sivustolla. Julkaistu 5.11.2012. Viitattu 23.10.2014. https://collectd.org/wiki/index.php/CPU

Plugin:CVS. 2009. Artikkeli collectd.org sivustolla. Julkaistu 20.12.2009. Viitattu 11.11.2014. https://collectd.org/wiki/index.php/Plugin:CSV

Plugin:DF. 2013. Artikkeli collectd.org sivustolla. Julkaistu 26.11.2013. Viitattu 7.10.2014. https://collectd.org/wiki/index.php/Df

Plugin:Disk. 2011. Artikkeli collectd.org sivustolla. Julkaistu 28.1.2011. Viitattu 23.10.2014. https://collectd.org/wiki/index.php/Disk

Plugin:Exec. 2011. Artikkeli collectd.org sivustolla. Julkaistu 28.9.2011. Viitattu 11.11.2014. https://collectd.org/wiki/index.php/Exec

Plugin:Interface. 2014. Artikkeli collectd.org sivustolla. Julkaistu 14.5.2014. Viitattu 7.10.2014. https://collectd.org/wiki/index.php/Interface

Plugin:Load. 2009. Artikkeli collectd.org sivustolla. Julkaistu 20.12.2009. Viitattu 8.10.2014. https://collectd.org/wiki/index.php/Load

Plugin:LogFile. 2009. Artikkeli collectd.org sivustolla. Julkaistu 20.12.2009. Viitattu 12.11.2014. https://collectd.org/wiki/index.php/Plugin:LogFile

Plugin:Memory. 2014. Artikkeli collectd.org sivustolla. Julkaistu 14.5.2014. Viitattu 8.10.2014. https://collectd.org/wiki/index.php/Memory

Plugin:Network. 2010. Artikkeli collectd.org sivustolla. Julkaistu 8.3.2010. Viitattu 8.10,2014. https://collectd.org/wiki/index.php/Network

Plugin:Processes. 2010. Artikkeli collectd.org sivustolla. Julkaistu 30.6.2010. Viitattu 12.11.2014. https://collectd.org/wiki/index.php/Processes

Plugin:SNMP. 2010. Artikkeli collectd.org sivustolla. Julkaistu 18.11.2010. Viitattu 11.11.2014. https://collectd.org/wiki/index.php/Snmp

Plugin:Write Graphite. 2014. Artikkeli collectd.org sivustolla. Julkaistu 1.9.2014. Viitattu 7.10.2014. https://collectd.org/wiki/index.php/Plugin:Write_Graphite

Server monitoring. N,d. Artikkeli appperfect sivustolla. Viitattu 6.10.2014. http://www.appperfect.com/products/monitoring/server-monitoring.html

Settings. 2014. Artikkeli djangoproject.com sivustolla. Julkaistu 1.1.2014. Viitattu 22.10.2014. https://docs.djangoproject.com/en/dev/ref/settings/#databases

Stanek, W. 1999. Monitoring Server Performance and Activity. Artikkeli technet.microsoft sivustolla. Julkaistu 1.1.1999. Viitattu 6.10.2014. http://technet.microsoft.com/en-us/library/bb726968.aspx

What does open source mean? 2000. Artikkeli howstuffworks.com sivustolla. Julkaistu 1. 8.2000. Viitattu 20.10.2014. http://computer.howstuffworks.com/question435.htm

What is Musicinfo. n, d. Artikkeli musicinfo.fi sivustolla. Viitattu 24.11.2014. http://www.musicinfo.fi/about-us.html

What is open source? N, d. Artikkeli opensource.com sivustolla. Viitattu 20.10.2014. http://opensource.com/resources/what-open-source

What is Python? Executive Summary. 2014. Artikkeli python.org sivustolla. Julkaistu 1.1.2014. viitattu 2.10.2014. https://www.python.org/doc/essays/blurb/

WSGI. 2011. Artikkeli savage.net sivustolla. Julkaistu 3.3.2011. Viitattu 18.11.2014. https://www.servage.net/wiki/WSGI

Yu, J. 2010. Round Robin Databases. Artikkeli wordpress.com sivustolla. Julkaistu 8.1.2010. Viitattu 20.10.2014. http://jawnsy.wordpress.com/2010/01/08/round-robin-databases/

Zandbergen, P. N, d. Central Processing Unit (CPU): Parts, Definition & Function. Artikkeli education-portal.com sivustolla. Viitattu 31.10.2014. http://education-portal.com/academy/lesson/central-processing-unit-cpu-parts-definition-function.html#lesson

LIITTEET

Liite 1. Graphite asennusohje

Haetaan "Graphiten" tärkeimmät paketit

Sisältää kolme pakettia, Whisper, Carbon ja Graphite - verkkosovellus. Nämä kaikki on saatavilla osoitteesta: https://launchpad.net/graphite/+download/

Voidaan tarkistaa uusimmat lähteet käyttämällä git:tiä:

```
# Master (unstable/alpha) branch
git clone https://github.com/graphite-project/graphite-web.git
git clone https://github.com/graphite-project/carbon.git
git clone https://github.com/graphite-project/whisper.git
```

TAI

```
# 0.9.x (stable) branch
git clone https://github.com/graphite-project/graphite-web.git
cd graphite-web
git checkout 0.9.x
cd ..
git clone https://github.com/graphite-project/carbon.git
cd carbon
git checkout 0.9.x
cd ..
git clone https://github.com/graphite-project/whisper.git
cd whisper
git checkout 0.9.x
cd ..
```

Asennetaan Whisper

```
pushd whisper
sudo python setup.py install
popd
```

Asennetaan Carbon

Oletusarvona on, että kaikki asennetaan hakemistoon

```
/opt/graphite
```

Asennetaan

pushd
python setup.py install
popd

Konfiguroidaan Carbon

pushd /opt/graphite/conf
cp carbon.conf.example carbon.conf
cp storage-schemas.conf.example storage-schemas.conf

Oletuksena arvot ovat esimerkki tiedostossa ihan järkeviä, mutta kuitenkin on suositeltavaa miettiä paljon tietoa halutaan säilyttää. Oletuksena on, että tiedot säilytetään yhden päivän ajan, yhden minuutin aikavälillä. Tämä asetetaan Whisper tiedostoihin yksityiskohtaisesti, ja täällä arvoja vaihtamalla ei muuta jo olemassa olevia metriikoita. Näiden arvojen vaihtamiseen voidaan käyttää myöhemmin Whisperin muuntautumis skriptiä (Whisper-resize.py).

Alla on esimerkki 13 kuukauden säilyttämisen esimerkistä.

[everything_1min_13months]
priority = 100
*pattern = .**
retentions = 1m:395d

Muutettaessa on huomioitava, että vaihdetaan koko *[everything_1min_13months]* osio.

Konfiguroidaan Graphiten verkkosovellus

Tämä on etualan verkkosovellus, joka esittää kuviot.

HUOM! Seuraavassa kohdassa tehdään tarkistus, että vaadittavat riippuvuudet täytetään. Käytä jakelijan pakettienhallintaa tai muuta keinoa asentaa vaadittavat ohjelmat.

HUOM! Tässä kohtaa on myös oltava tarkkana riippuvuuksien versioiden kanssa. Djangolla on ongelmia uusimman version kanssa integroida sitä Graphiten riippuvuuksien kanssa, joten on suositeltavaa käyttää aikaisempaa versiota. Työn toteutuksessa käytetiin versiota Django= = 1.4.5.

Djangon version voi tarkastaa komennolla

pip freeze | grep Django

Tarkistus

pushd graphite-web
python check-dependencies.py
popd

Kun/Mikäli riippuvuudet on hoidettu/täytetty, asennetaan Graphite verkkosovellus.

pushd graphite-web
python setup.py install
popd

Konfiguroidaan Apache

Graphiten hakemistossa on verkkosovelluksen alla esimerkkitiedosto *example-graphite-vhost.conf*. Tätä tiedostoa voidaan käyttää konfiguroimaan Apache. Eri jakelijalla on eri tavat konfiguroida Apache. Esimerkiksi:

- Ubuntu käyttää */etc/apache2/sites-available* ja *sites-enabled/* asian hoitamiseen. Symlinkiä *sites-enabled/* hakemistosta *sites-available/* hakemistoon käytetään sen jälkeen, kun tiedosto on asetettu *sites-available/* hakemistoon.
- ln on komentorivipohjainen työkalu symbolisten ja kovien linkkien tekoon ja hallintaan. Komennolla voidaan suorittaa symbolinen linkki.

  ln –s

  ***-s** - luo symbolisia linkkejä*

Graphite on ”HakemistoJuuressa” verkkopalvelimella, ja ei anna pääsyä avoimeen-HTML:ään alihakemistoissa ilman asianmukaisia konfiguraatioita. Joten on hyvä muokata *example-graphite-vhosts.conf* tiedostoa vaihtamalla portin numeron tai käyttää perinteisiä “*SetHandler None*” - hakemistojen sallia pääsy muihin hakemistoihin.

On pidettävä huoli, että ladataan uudet Apache konfiguraatiot

sudo /etc/init.d/apache2 reload

Alustavan tietokannan luonti

On Djangolle kerrottava, että se luo Graphiten verkkosovelluksen käyttämät tietokannan taulut. Tämä on aika suora johteista, erityisesti silloin, jos käytetään oletusasetuksilla olevaa sqlite:ä.

HUOM! Mikäli käytetään muuta taka-alan tietokantaa, kuin sqlite, on ensin tehtävä *$Graphite_Juuri/webapp/graphite/local_settings.py* tiedosto, joka yliajaa tietokantaan liittyvät asetukset *settings.py* tiedostossa. On hyvä käyttää pohjana tähän tiedostoon *$Graphite_Juuri/webapp/graphite/local_settings.py.example* tiedostoa.

Oletetaan kuitenkin, että käytetään oletusasetuksia. Voidaan luoda tietokanta seuraavalla komennoilla

cd /opt/graphite/webapp/graphite
sudo python manage.py syncdb

HUOM! Tässä kohtaa pyydetään luomaan ADMIN käyttäjä, ja on suositeltavaa luoda kyseinen käyttäjä tässä vaiheessa.

Seuraavaksi on vaihdettava tietokannan omistusoikeuksia niin, että ne ovat samalla käyttäjällä, joka **omistaa** Apache prosessit. Mikäli jakelussa Apachea käyttää käyttäjä "nobody"

sudo chown nobody:nobody /opt/graphite/storage/graphite.db

HUOM! Mikäli käytetään taka-alan ohjelmia kuten mysql tai postgres, TIETOKANNAN_KAYTTAJA, joka luotiin *local_settings.py* tiedostossa, tulee olla oikeudet luoda tauluja tietokannan TIETOKANNAN_NIMI alla.

Uudelleen käynnistä Apache ja Graphiten verkkosovelluksen pitäisi olla nähtävissä sen pääsivulla. Mikäli törmätään ongelmiin, voidaan lisätä virheen korjaukseen tarkkuutta luomalla *Local_settings.py* tiedosto.

cd /opt/graphite/webapp/graphite
cp local_settings.py.example local_settings.py

Poistetaan kommentti seuraavalta riviltä */opt/graphite/webapp/graphite/local_settings.py* tiedostosta.

 # DEBUG = True

HUOM! On hyvä huomioida, että apachen lokit Graphiten verkkosovellukselle GRAPHITEN_APACHE.CONF ovat */opt/graphite/storage/logs/* hakemistossa.

Käynnistetään Carbon

cd /opt/graphite/
./bin/carbon-cache.py start

Nyt Graphite pitäisi olla asennettu onnistuneesti

Liite 2. Työkalut jotka toimivat Graphiten kanssa

Kerääminen

Seuraavat työkalut ovat tarkoitettu tiedon keräämiseen, käsittelyyn ja tallentamiseen:

Bucky, pieni palvelu, joka on toteutettu Pythonilla keräämään ja lähettämään metriikoita Graphiteen. Se kykenee keräämään metriikoita Collectd:n Daemoneista ja StatsD:n asiakkaista

Collectd, Daemoni, joka kerää järjestelmän suorituskyvyn tietoja määräajallisesti ja tarjoaa mekaniikan tietojen tallentamisen monella eri tapaa, sisältäen RRD:n.

Collectl, keräystyökalu järjestelmän metriikoille, joka voidaan suorittaa vuorovaikutteisesti ja Damonina. Sisältää tuen laajaan joukkoon osajärjestelmiä. Collectl sisältää Graphiten käyttöliittymän, joka mahdollistaa tiedon syöttämisen helposti Graphitelle varastoitavaksi.

Diamond, Pythonin Daemoni, joka kerää järjestelmän metriikoita ja julkaisee ne Graphitessa. Se on kykeneväinen keräämään CPU, muisti, verkko, I/O, kuormitusta ja levyke metriikoita.

Ganglia, skaalautuva monitorointi järjestelmä korkean suorituskyvyn järjestelmille. Se kerää järjestelmän suorituskyvyn metriikoita ja varastoi ne RRD:hen, on kuitenkin olemassa liitännäinen, joka mahdollistaa Ganglian lähettää metriikoita Graphiteen.

Graphite PowerShell Functions, joukko toimintoja, joita voidaan käyttää Windowsin suorituskyvyn laskureina ja lähettää ne Graphite palvelimelle. Pääfunktio voidaan suorittaa Windowsin palveluna, ja kaikki voidaan konfiguroida XML – tiedostossa.

HoardD, Node.js, joka on kirjoitettu CoffeeScriptilla lähettämään tietoa palvelimilta Graphiteen. Samaan tyyliin kuin Collectd, mutta tarkoitettu paljon helpommaksi laajentaa ja pienemmällä jalanjäljellä. Sisältää peruskerääjät ja voidaan laajentaa Javascriptilla ja CoffeeScriptilla.

Host sFlow, Vapaan lähdekoodin toteutus sFlow protokollasta, joka tuo standartin asetelman isännän CPU:sta, muistista, levykkeistä, ja verkon I/O metriikoista. Apuohjelma sflow2Graphite muuntaa sFlow:n Graphiten selkeäkieliseksi protokollaksi, mahdollistaen Graphiten vastaanottaa sFlow metriikoita.

jmxtrans, tehokas työkalu, joka toteuttaa JMX kyselyitä JAVA sovelluksista. Se vaatii pienehkön konfiguraation ja on kykeneväinen lähettämään metriikoiden tietoa monelle taka-alan sovellukselle, sisältäen Graphiten.

Logster, apuohjelma Graphiteen loki tiedostojen lukemiseen ja metriikoiden luomiseen. Ihanteellinen visualisoimaan tapahtumia, jotka tapahtuvat sovelluksissa, järjestelmässä tai virhe lokeissa.

metrics-sampler, java ohjelma, säännöllisillä metriikka kyselyillä, konfiguroidusta joukosta sisääntuloja, vaihtoehtoja ja niiden uudelleen nimeämiseen käyttäen säännöllistä ilmaisua ja lähettämällä ne konfiguroituun joukkoon ulosmenoja.

Sensu, monitorointi rakenne, joka reitittää metriikoita Graphiteen. Palvelimet merkitsevät joukon tarkastuksia, joten metriikoiden saaminen uudelta palvelimelta Graphiteen on niin yksinkertaista, että asennetaan vain Sensu asiakas ja merkitään joukko tarkastuksia.

SqlToGraphite, agentti windowsille, joka on kirjoitettu .net:lla keräämään metriikoita. Se käyttää metriikoita pollaamalla päätepisteet SWL kyselyllä ja työntämällä tulokset Graphiteen. Se käyttää joko paikallista tai keskitettyä konfiguraatiota http:n ylitse.

SSC Serv, Windowsin palvelu, joka määräajoin julkaisee järjestelmän metriikoita. Esimerkiksi CPU, muisti ja levynkäyttöä. Se kykenee varastoimaan tietoa Graphiteen käyttämällä nimeämismallinnusta, joka on identtinen Collectd:n kanssa.

Lähettäminen

Seuraavat työkalut ovat tarkoitettu tiedon lähettämiseen:

Backstop, yksinkertainen taka-ala sovellus metriikoiden lähettämiseen Graphiteen. Se hyväksyy JSON:in tietoa http POST:in kautta ja valtuuttaa tiedon yhdelle tai useammalla Carbonin/Graphiten kuuntelijalle.

Evenflow, yksinkertainen palvelu sFlow:n datagrammien lähettämiseen. Se hyväksyy sFlow datagrammit useista verkkolaitteista ja valtuuttaa tiedon Carbonin kuuntelijalle.

Graphite-Newrelic, Graphiten tiedon saaminen New Relic:iin, New Relic – liitännäisen avulla.

Graphite-relay, nopea Graphite rele.

Graphios, pieni Pythonin Daemoni, joka lähettää Nagioksen suorituskyvyn tietoa Graphiteen.

Grovkets, node.js sovellus, joka tarjoaa JSON:in tietovuon HTTP:n ylitse Graphiteen.

Ledbetter, yksinkertainen kirjoitus Nagioksen ongelmien statistiikan keraamiseen ja lähettämiseen ne Graphiteen. Se keskittyy tiivistettyyn statistiikkaan ja kirjoittaa ne nagious.problem metriikkaan Graphiten sisällä.

pipe-tp-Graphite, pieni komentorivi kirjoitus, joka tekee muiden CLI ohjelmistojen raportoimisen Graphiteen.

statsd, yksinkertainen Daemoni helpoon aggregaattiin.

Visualisointi

Seuraavat työkalut ovat tarkoitettu tiedon visualisointiin:

Charcoal, yksinkertainen Sinatra kojelauta Graphiten taka-alalle tai mille tahansa tilanne seuranta palvelulle, joka voi luoda kuvioita suoraan URL:ista.

Descartes, Sinatra pohjainen kojelauta, joka mahdollistaa käyttäjän korreloida useita metriikoita yhdellä kaaviolla, katsastella pitkän aikavälin trendejä yhden tai useamman kaavion kautta, ja tehdä yhteistyötä muiden kanssa yhdistetyillä kojelaudoilla.

Firefly, verkkosovellus, joka on suunniteltu voimakkaaseen ja joustavaan aikajaksollisen grafiikan piirtämiseen verkon kehittelijöille.

Gdash, yksinkertainen Graphite kojelauta kehitelmä, joka käyttää Twitterin Bootstrappia.

Giraffe, Graphiten reaaliajan kojelauta, joka perustuu Rickshaw:iin ja ei vaadi palvelimen taka-alaa.

Graphana, Graphiten kojelaudan korvaaja, monipuolisella kaavio editoinnilla ja kojelaudan rajapinnan muokkaamisella. Sisältää uniikin Graphite kohteen parserin, joka mahdollistaa helpon metriikoiden ja funktioiden muokkaamisen.

graphitus, asiakkaan puolen kojelauta Graphiten toteutukselle, joka käyttää bootstrappia ja underscore.js:ää.

Graph-Explorer, Graphiten kojelauta, joka käyttää liitännäisiä lisäämään leimoja ja metatietoja metriikkoihin ja jonokielen, joka mahdollistaa suodattaa metriikoita ja kuvioiden manipuloinnin lennosta.

Graph-Index, kaavioiden indeksi Diamondille.

Graphene, Graphiten kojelauta työkalu perustuen D3.js:ään ja Backbone.js:ää, joka luotiin tarjoamaan erittäin esteettinen reaaliajan kojelauta. Graohene tarjoaa tuhansien datapisteiden esittämisen reaaliajassa.

Graphite-Observer, reaaliajan kojelauta Graphiteen

Graphite-Tattle, itsepalvelu kojelauta taka-alaan Graphiteen ja Gangliaan.

Graphiti, tehokas etualan kojelauta, joka keskittyy pääsyn, elpymisen, säätämisen ja manipuloinnin helppouteen.

Graphitoid, android sovellus, joka mahdollistaa Graphiten kaavioiden selailun android laitteella.

Graphsky, joustava ja helposti konfiguroitava PHP – pohjainen kojelauta.

Hubot, ”campfire” botti, joka on kirjoitetti Node.js ja CoffeeScript:illa.

Leonardo, Graphiten kojelauta, joka on innoittunut Gdash:ista. Se on kirjoitettu Pythonilla käyttämällä Flash rakennetta. Rajapinta on rakennettu Bootstrapilla. Kaaviot ja kojelaudat konfiguroidaan YAML tiedostoilla.

Orion, tehokas työkalu kojelautojen luomiseen, katseluun ja hallintaan Graphitessa. Se mahdollistaa helpon implementoinnin autentikointi käytänteisiin.

Pencil, monitoroinnin etuala Graphiteen. Se toimii verkkopalvelimella, joka jakaa hienoja Graphite URL:eja kiinnostavilla ja intuitiivisilla ulkoasuilla.

Seyren, Hälytyskojelauta Graphiteen.

Tasseo, kevyt helposti konfiguroitavissa, reaaliajan kojelauta Graphiten metriikoille.

Tassera, joustava etuala kojelautojen luomiselle laajalla valikoimalla tiedon esitykseen.

TimeseriesWidget, lisää aikajaksoja kaavioihin verkkosivulle/kojelaudalle käyttämällä yksinkertaista API:a. Se keskittyy vuorovaikutteisuuteen ja moderneihin ominaisuuksiin.

Monitorointi

Cabot, itsestään isännöivä monitorointi ja hälytys palvelin, joka seuraa Graphiten metriikoita ja voi hälyttää niistä puhelimella, tekstiviestillä, Hipchatilla tai sähköpostilla.

graphite-beacon, yksinkertainen hälytyssovellus Graphiteen. Se on asynkroninen ja lähettää huomautuksia perustuen Graphiten metriikoihin.

rearview, reaaliajan monitorointi kehys, joka istuu Graphiten ajanjaksollisen tiedon päälle. Tämä mahdollistaa käyttäjä luoda monitoreita, jotka ovat sekä visuaalisia ja hälyttää tiedoista, joita se jakaa Graphitesta. Hälytyksiä voidaan lähettää sähköpostin, ”pagerdytin” tai ”campfire” kautta.

Rocksteady, järjestelmä, joka sitoo yhteen Graphiten, RabbitMQ, ja Esperin.

Shinken, järjestelmän monitorointi ratkaisu, joka on yhteensopiva Nagioksen kanssa, joka painottaa skaalattavuutta, joustavuutta ja helppoa toteutusta. Shinken tarjoaa täydellistä integraatiota Graphiten kanssa prosessointiin ja suoritustiedon esittämiseen

Muut

Muut mahdolliset työkalut:

Therry, yksinkertainen verkkopalvelu, joka kätkee Graphiten metriikoita ja paljastaa ne takapäässä tai etsii niitä osamerkkijonoilla.

Liite 3. Collectd asennus

Normaali

HUOM! Suositeltavaa käyttää tätä asennusmenetelmää Collectd:n ”asiakkaille”.

Normaali asennus voidaan suorittaa komennolla:

apt-get install collectd

Konfiguraatiotiedostot löytyvät kansiosta:

/etc/collectd/

Lähteestä asentaminen

HUOM! Suositeltavaa käyttää tätä asennusmenetelmää toteutuksen Collectd:n ”isännälle”. Mikäli ollaan integroimassa Collectd:tä Graphiten kanssa. Liitännäinen *Write_Graphite* sisältää puutteita aikaisemmissa versioissa, joten on suositeltavaa käyttää Collectd versiota (5.4+). Tämä asennus on vaativa, mutta vaaditaan edellä mainitun toiminnollisuuden kannalta.

Mikäli jakelijalla ei ole binääripakettia, tai olemassa oleva on vanhentunut, voidaan koota lähteistä. Debianin käyttäjät voivat yksinkertaisesti asentaa *build-essentieal* – paketin.

apt-get install build-essential

Haetaan ja puretaan paketti lähteestä. Voidaan käyttää mitä tahansa hakemistoa paketille, mutta esimerkiksi haetaan paketti hakemistoon */tmp/*. Komennoissa kohta *x.y.z* viittaa haluttuun pakettiin, joka halutaan hakea.

cd /tmp/
wget http://collectd.org/files/collectd-x.y.z.tar.bz2
tar jxf collectd-x.y.z.tar.bz2
cd collectd-x.y.z

Asennetaan perinteisellä komennolla paketit

./configure

Sen jälkeen kun *configure* - skripti on valmis, se esittää yhteenvedon kirjastoista, jotka se löysi ja ei löytänyt, sekä ne liitännäiset, jotka on mahdollistettu. Oletuksena on, että kaikki liitännäiset, joiden riippuvuudet täyttyvät, toteutetaan. Mikäli liitännäinen, jota halutaan käyttää puuttuu, asennetaan vaadittavat kehityspaketit ja ajetaan *configure* uudelleen.

Oletuksena on, että asennus suoritetaan hakemistoon */opt/collectd*. Mikäli suositaankin muuta hakemistoa, kutsutaan *configure* skriptiä - - *prefix* vaihtoehdolla

Viimeisenä kootaan ja asennetaan ohjelma.

make all install
cd /opt/collectd/

Konfigurointi

Konfigurointitiedosto löytyy hakemistosta *<prefix>/etc/collectd.conf*

Daemonin käynnistäminen

Mikäli ollaan valmiita konfiguraatioiden kanssa, tulee käynnistää/uudelleen käynnistää Daemon. Mikäli asennettiin binääripaketit *init* – skriptin pitäisi sijaita ”jossain”. Debianissa komento on

/etc/init.d/collectd restart

Vaihtoehtoisesti voidaan Daemoni käynnistää ”käsin”. Tämä suoritetaan toteuttamalla

/opt/collectd/sbin/collectd

Tai mikäli käytetään binääripaketteja

/usr/sbin/collectd

Liite 4. Collectd binääriprotokollan osien tyypit

Osien tyypit			
ID	NIMI	TIETO TYYPPI	KOMMENTTI
0x0000	Isäntä	Lanka	Isännän nimi, joka osakkaana tiedon arvoille
0x0001	Aika	Numero	Aikaleima seuraaville osakkaana oleville tieto arvoille, unix aika muodossa
0x0008	Aika (korkea resoluutio)	Numero	Aikaleima joka on osakkaana tieto arvoissa. Aika määritellään 2^-30 sekunnilla
0x0002	Liitännäinen	Lanka	Liitännäisen nimi, joka on osakkaana tiedon arvoille. Esim "CPU"
0x0003	Liitännäisen tapaus	Lanka	Liitannäisen tapauksen nimi, joka on osakkaana tieto arvoille. Esim "1"
0x0004	Tyyppi	Lanka	Tyypin nimi, joka on osaakkaana tiedon arvoille. Esim "CPU"

0x0005	Tyypin tapaus	Lanka	Tyypin tapauksen nimi, joka on osaakkaana tiedon arvoille. Esim "idle"
0x0006	Arvoja	*Muu*	Tiedon arvoja
0x0007	Aikaväli	Numero	Aikaväli jota käytetään asettamaan "askelma", jolloin luodaan uusi RRD, ellei *RRDTool* liitännäinen pakota *StepSizea*. Käytetään myös *aikalisä* arvojen havaitsemiseen
0x0009	Aikaväli (korkea resoluutio)	Numero	Aikaväli jolloin seuraavat tieto arvot kerätään. Aikavälin annetaan 2^-30 sekunttina
0x0100	Viesti (huomautukset)	Lanka	
0x0101	Kovuus	Numero	
0x0200	Allekirjoitus (HMAC-SHA-256)	*Muu*	
0x0210	Salaus (AES-256/OFB/SHA-1)	*Muu*	

Liite 5. Apache konfiguraatiotiedostot

Polusta **/etc/apache2** hakemistopolusta löytyy apachen tarvittavat konfiguraatiotiedostot ja kansiot:

apache2.conf
conf.d
envvars
httpd.conf
magic
mods-available
mods-enabled
ports.conf
sites-available
sites-enabled

Liite 6. Graphite Apache konfikuraatiotiedosto

```
# in this file to your chosen install location.

<IfModule !wsgi_module.c>
   LoadModule wsgi_module modules/mod_wsgi.so
</IfModule>

# XXX You need to set this up!
```

```
# Read http://code.google.com/p/modwsgi/wiki/ConfigurationDirec-
tives#WSGISocketPrefix
WSGISocketPrefix /etc/httpd/wsgi

<VirtualHost *:PORTTI>
#       ServerAdmin webmaster@localhost.fi
      ServerName localhost
      DocumentRoot "/opt/graphite/webapp"
      ErrorLog /opt/graphite/storage/log/webapp/error.log
      CustomLog /opt/graphite/storage/log/webapp/access.log common

      # I've found that an equal number of processes & threads tends
      # to show the best performance for Graphite (ymmv).
          WSGIDaemonProcess graphite processes=5 threads=5 display-
          name='%{GROUP}' inactivity$
      WSGIProcessGroup graphite
      WSGIApplicationGroup %{GLOBAL}
           WSGIImportScript /opt/graphite/conf/graphite.wsgi process-
          group=graphite applicatio$

      # XXX You will need to create this file! There is a graphite.wsgi.example
      # file in this directory that you can safely use, just copy it to graphite.wgsi
      WSGIScriptAlias / /opt/graphite/conf/graphite.wsgi

      Alias /content/ /opt/graphite/webapp/content/
      <Location "/content/">
            SetHandler None
      </Location>

      # XXX In order for the django admin site media to work you
      # must change @DJANGO_ROOT@ to be the path to your django
      # installation, which is probably something like:
      # /usr/lib/python2.6/site-packages/django
      Alias /media/ "@DJANGO_ROOT@/contrib/admin/media/"
      <Location "/media/">
            SetHandler None
      </Location>

      # The graphite.wsgi file has to be accessible by apache. It won't
      # be visible to clients because of the DocumentRoot though.
      <Directory /opt/graphite/conf/>
            Order deny,allow
            Allow from all
      </Directory>

</VirtualHost>
```

Liite 7. Graphite konfiguraatiotiedostot

Polusta **/opt/graphite/conf** hakemistopolusta löytyy Graphiten tarvittavat konfiguraatiotiedostot:

aggregation-rules.conf.example
blacklist.conf.example
carbon.amqp.conf.example
carbon.conf
carbon.conf.example
dashboard.conf
dashboard.conf.example
graphite.wsgi
graphite.wsgi.example
graphTemplates.conf.example
relay-rules.conf
relay-rules.conf.example
rewrite-rules.conf.example
storage-aggregation.conf
storage-aggregation.conf.example
storage-schemas.conf
storage-schemas.conf.example
whitelist.conf.example

Liite 8. Carbon.conf

Carbon.conf tiedoston sisältö kokonaisuudessaan:

```
[cache]
# Configure carbon directories.
#
# OS environment variables can be used to tell carbon where graphite is
# installed, where to read configuration from and where to write data.
#
#   GRAPHITE_ROOT       - Root directory of the graphite installation.
#                         Defaults to ../
#   GRAPHITE_CONF_DIR   - Configuration directory (where this file lives).
#                         Defaults to $GRAPHITE_ROOT/conf/
#   GRAPHITE_STORAGE_DIR - Storage directory for whipser/rrd/log/pid files.
#                         Defaults to $GRAPHITE_ROOT/storage/
#
# To change other directory paths, add settings to this file. The following
# configuration variables are available with these default values:
#
#   STORAGE_DIR    = $GRAPHITE_STORAGE_DIR
#   LOCAL_DATA_DIR = STORAGE_DIR/whisper/
#   WHITELISTS_DIR = STORAGE_DIR/lists/
```

```
#   CONF_DIR      = STORAGE_DIR/conf/
#   LOG_DIR       = STORAGE_DIR/log/
#   PID_DIR       = STORAGE_DIR/
#
# For FHS style directory structures, use:
#
#   STORAGE_DIR    = /var/lib/carbon/
#   CONF_DIR       = /etc/carbon/
#   LOG_DIR        = /var/log/carbon/
#   PID_DIR        = /var/run/
#
#LOCAL_DATA_DIR = /opt/graphite/storage/whisper/

# Specify the user to drop privileges to
# If this is blank carbon runs as the user that invokes it
# This user must have write access to the local data directory
USER =

# Limit the size of the cache to avoid swapping or becoming CPU bound.
# Sorts and serving cache queries gets more expensive as the cache grows.
# Use the value "inf" (infinity) for an unlimited cache size.
MAX_CACHE_SIZE = inf

# Limits the number of whisper update_many() calls per second, which effectively
# means the number of write requests sent to the disk. This is intended to
# prevent over-utilizing the disk and thus starving the rest of the system.
# When the rate of required updates exceeds this, then carbon's caching will
# take effect and increase the overall throughput accordingly.
MAX_UPDATES_PER_SECOND = 500

# Softly limits the number of whisper files that get created each minute.
# Setting this value low (like at 50) is a good way to ensure your graphite
# system will not be adversely impacted when a bunch of new metrics are
# sent to it. The trade off is that it will take much longer for those metrics'
# database files to all get created and thus longer until the data becomes usable.
# Setting this value high (like "inf" for infinity) will cause graphite to create
# the files quickly but at the risk of slowing I/O down considerably for a while.
MAX_CREATES_PER_MINUTE = 50

LINE_RECEIVER_INTERFACE = 0.0.0.0
LINE_RECEIVER_PORT = 2003

# Set this to True to enable the UDP listener. By default this is off
# because it is very common to run multiple carbon daemons and managing
# another (rarely used) port for every carbon instance is not fun.
ENABLE_UDP_LISTENER = False
UDP_RECEIVER_INTERFACE = 0.0.0.0
UDP_RECEIVER_PORT = 2003
```

PICKLE_RECEIVER_INTERFACE = 0.0.0.0
PICKLE_RECEIVER_PORT = 2004

 # Per security concerns outlined in Bug #817247 the pickle receiver
 # will use a more secure and slightly less efficient unpickler.
 # Set this to True to revert to the old-fashioned insecure unpickler.
USE_INSECURE_UNPICKLER = False

CACHE_QUERY_INTERFACE = 0.0.0.0
CACHE_QUERY_PORT = 7002

 # Set this to False to drop datapoints received after the cache
 # reaches MAX_CACHE_SIZE. If this is True (the default) then sockets
 # over which metrics are received will temporarily stop accepting
 # data until the cache size falls below 95% MAX_CACHE_SIZE.
USE_FLOW_CONTROL = True

 # By default, carbon-cache will log every whisper update. This can be excessive and
 # degrade performance if logging on the same volume as the whisper data is stored.
LOG_UPDATES = False

 # On some systems it is desirable for whisper to write synchronously.
 # Set this option to True if you'd like to try this. Basically it will
 # shift the onus of buffering writes from the kernel into carbon's cache.
WHISPER_AUTOFLUSH = False

 # By default new Whisper files are created pre-allocated with the data region
 # filled with zeros to prevent fragmentation and speed up contiguous reads and
 # writes (which are common). Enabling this option will cause Whisper to create
 # the file sparsely instead. Enabling this option may allow a large increase of
 # MAX_CREATES_PER_MINUTE but may have longer term performance implications
 # depending on the underlying storage configuration.
 # WHISPER_SPARSE_CREATE = False

 # Enabling this option will cause Whisper to lock each Whisper file it writes
 # to with an exclusive lock (LOCK_EX, see: man 2 flock). This is useful when
 # multiple carbon-cache daemons are writing to the same files
 # WHISPER_LOCK_WRITES = False

 # Set this to True to enable whitelisting and blacklisting of metrics in
 # CONF_DIR/whitelist and CONF_DIR/blacklist. If the whitelist is missing or
 # empty, all metrics will pass through
 # USE_WHITELIST = False

```
# By default, carbon itself will log statistics (such as a count,
# metricsReceived) with the top level prefix of 'carbon' at an interval of 60
# seconds. Set CARBON_METRIC_INTERVAL to 0 to disable instrumentation
# CARBON_METRIC_PREFIX = carbon
# CARBON_METRIC_INTERVAL = 60

# Enable AMQP if you want to receve metrics using an amqp broker
# ENABLE_AMQP = False

# Verbose means a line will be logged for every metric received
# useful for testing
# AMQP_VERBOSE = False

# AMQP_HOST = localhost
# AMQP_PORT = 5672
# AMQP_VHOST = /
# AMQP_USER = guest
# AMQP_PASSWORD = guest
# AMQP_EXCHANGE = graphite
# AMQP_METRIC_NAME_IN_BODY = False

# The manhole interface allows you to SSH into the carbon daemon
# and get a python interpreter. BE CAREFUL WITH THIS! If you do
# something like time.sleep() in the interpreter, the whole process
# will sleep! This is *extremely* helpful in debugging, assuming
# you are familiar with the code. If you are not, please don't
# mess with this, you are asking for trouble :)
# MANHOLE_PUBLIC_KEY = ssh-rsa AAAAB3NzaC1yc2EAAAABi-
wAaAIEAoxN0sv/e4eZCPpi3N3KYvyzRaBaMeS2R$

# Patterns for all of the metrics this machine will store. Read more at
# http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol#Bindings
#
# Example: store all sales, linux servers, and utilization metrics
# BIND_PATTERNS = sales.#, servers.linux.#, #.utilization
#
# Example: store everything
# BIND_PATTERNS = #

# To configure special settings for the carbon-cache instance 'b', uncomment this:
#[cache:b]
#LINE_RECEIVER_PORT = 2103
#PICKLE_RECEIVER_PORT = 2104
#CACHE_QUERY_PORT = 7102
# and any other settings you want to customize, defaults are inherited
# from [carbon] section.
# You can then specify the --instance=b option to manage this instance
```

```
[relay]
LINE_RECEIVER_INTERFACE = 0.0.0.0
LINE_RECEIVER_PORT = 2013
PICKLE_RECEIVER_INTERFACE = 0.0.0.0
PICKLE_RECEIVER_PORT = 2014

# To use consistent hashing instead of the user defined relay-rules.conf,
# change this to:
# RELAY_METHOD = consistent-hashing
RELAY_METHOD = rules

# If you use consistent-hashing you may want to add redundancy
# of your data by replicating every datapoint to more than
# one machine.
REPLICATION_FACTOR = 1

# This is a list of carbon daemons we will send any relayed or
# generated metrics to. The default provided would send to a single
# carbon-cache instance on the default port. However if you
# use multiple carbon-cache instances then it would look like this:
#
# DESTINATIONS = 127.0.0.1:2004:a, 127.0.0.1:2104:b
#
# The general form is IP:PORT:INSTANCE where the :INSTANCE part is
# optional and refers to the "None" instance if omitted.
#
# Note that if the destinations are all carbon-caches then this should
# exactly match the webapp's CARBONLINK_HOSTS setting in terms of
# instances listed (order matters!).
#
# If using RELAY_METHOD = rules, all destinations used in relay-rules.conf
# must be defined in this list
DESTINATIONS = 127.0.0.1:2014

# This defines the maximum "message size" between carbon daemons.
# You shouldn't need to tune this unless you really know what you're doing.
MAX_DATAPOINTS_PER_MESSAGE = 500
MAX_QUEUE_SIZE = 10000

# Set this to False to drop datapoints when any send queue (sending datapoints
# to a downstream carbon daemon) hits MAX_QUEUE_SIZE. If this is True (the
# default) then sockets over which metrics are received will temporarily stop ac-
cepting
# data until the send queues fall below 80% MAX_QUEUE_SIZE.
USE_FLOW_CONTROL = True
```

```
# Set this to True to enable whitelisting and blacklisting of metrics in
# CONF_DIR/whitelist and CONF_DIR/blacklist. If the whitelist is missing or
# empty, all metrics will pass through
# USE_WHITELIST = False

# By default, carbon itself will log statistics (such as a count,
# metricsReceived) with the top level prefix of 'carbon' at an interval of 60
# seconds. Set CARBON_METRIC_INTERVAL to 0 to disable instrumentation
# CARBON_METRIC_PREFIX = carbon
# CARBON_METRIC_INTERVAL = 60


[aggregator]
LINE_RECEIVER_INTERFACE = 0.0.0.0
LINE_RECEIVER_PORT = 2023

PICKLE_RECEIVER_INTERFACE = 0.0.0.0
PICKLE_RECEIVER_PORT = 2024

# This is a list of carbon daemons we will send any relayed or
# generated metrics to. The default provided would send to a single
# carbon-cache instance on the default port. However if you
# use multiple carbon-cache instances then it would look like this:
#
# DESTINATIONS = 127.0.0.1:2004:a, 127.0.0.1:2104:b
#
# The format is comma-delimited IP:PORT:INSTANCE where the :INSTANCE
part is
# optional and refers to the "None" instance if omitted.
#
# Note that if the destinations are all carbon-caches then this should
# exactly match the webapp's CARBONLINK_HOSTS setting in terms of
# instances listed (order matters!).
DESTINATIONS = 127.0.0.1:2004

# If you want to add redundancy to your data by replicating every
# datapoint to more than one machine, increase this.
REPLICATION_FACTOR = 1

# This is the maximum number of datapoints that can be queued up
# for a single destination. Once this limit is hit, we will
# stop accepting new data if USE_FLOW_CONTROL is True, otherwise
# we will drop any subsequently received datapoints.
MAX_QUEUE_SIZE = 10000

# Set this to False to drop datapoints when any send queue (sending datapoints
# to a downstream carbon daemon) hits MAX_QUEUE_SIZE. If this is True (the
```

```
# default) then sockets over which metrics are received will temporarily stop ac-
cepting
# data until the send queues fall below 80% MAX_QUEUE_SIZE.
USE_FLOW_CONTROL = True

# This defines the maximum "message size" between carbon daemons.
# You shouldn't need to tune this unless you really know what you're doing.
MAX_DATAPOINTS_PER_MESSAGE = 500

# This defines how many datapoints the aggregator remembers for
# each metric. Aggregation only happens for datapoints that fall in
# the past MAX_AGGREGATION_INTERVALS * intervalSize seconds.
MAX_AGGREGATION_INTERVALS = 5

# Set this to True to enable whitelisting and blacklisting of metrics in
# CONF_DIR/whitelist and CONF_DIR/blacklist. If the whitelist is missing or
# empty, all metrics will pass through
# USE_WHITELIST = False

# By default, carbon itself will log statistics (such as a count,
# metricsReceived) with the top level prefix of 'carbon' at an interval of 60
# seconds. Set CARBON_METRIC_INTERVAL to 0 to disable instrumentation
# CARBON_METRIC_PREFIX = carbon
# CARBON_METRIC_INTERVAL = 60

#----------------------------------------->
```

Liite 9. Storage-schema.conf

Storage.schema.conf tiedoston sisältö kokonaisuudessaan:

```
# Schema definitions for Whisper files. Entries are scanned in order,
# and first match wins. This file is scanned for changes every 60 seconds.
#
#  [name]
#  pattern = regex
#  retentions = timePerPoint:timeToStore, timePerPoint:timeToStore, ...

# Carbon's internal metrics. This entry should match what is specified in
# CARBON_METRIC_PREFIX and CARBON_METRIC_INTERVAL settings
#[carbon]
#pattern = ^carbon\.
#retentions = 60:90d

#[default_1min_for_1day]
#pattern = .*
#retentions = 60s:1d
```

```
[collectd]
pattern = ^collectd.*
retentions = 10s:1d,1m:7d,10m:1y

[everything_1min_12months]
priority = 100
pattern = .*
retentions = 1m:365d
```

Liite 10. Graphite webapp konfiguraatiotiedostot

Polusta **/opt/graphite/webapp/graphite** hakemistopolusta löytyy Graphiten webappin tarvittavat konfiguraatiot:

```
account
app_settings.py
app_settings.pyc
browser
cli
composer
dashboard
events
graphite.db
graphlot
_init_.py
_init_.pyc
local_settings.py
local_settings.pyc
local_settings.py.example
logger.py
logger.pyc
manage.backup.py
manage.py
manage.pyc
metrics
remote_storage.py
remote_storage.pyc
render
settings.py
settings.pyc
templates
thirdparty
urls.py
urls.py
util.py
util.pyc
version
```

views.py
views.pyc
whitelist

Liite 11. Local_settings.py

Local_settings.py tiedoston sisältö kokonaisuudessaan:

```
## Graphite local_settings.py
# Edit this file to customize the default Graphite webapp settings
#
# Additional customizations to Django settings can be added to this file as well

#####################################
# General Configuration #
#####################################
# Set your local timezone (Django's default is America/Chicago)
# If your graphs appear to be offset by a couple hours then this probably
# needs to be explicitly set to your local timezone.
#TIME_ZONE = 'Finland/Helsinki'

# Override this to provide documentation specific to your Graphite deployment
#DOCUMENTATION_URL = "http://graphite.readthedocs.org/"

# Logging
LOG_RENDERING_PERFORMANCE = True
LOG_CACHE_PERFORMANCE = True
LOG_METRIC_ACCESS = True

# Enable full debug page display on exceptions (Internal Server Error pages)
DEBUG = True

# If using RRD files and rrdcached, set to the address or socket of the daemon
#FLUSHRRDCACHED = 'unix:/var/run/rrdcached.sock'

# This lists the memcached servers that will be used by this webapp.
# If you have a cluster of webapps you should ensure all of them
# have the *exact* same value for this setting. That will maximize cache
# efficiency. Setting MEMCACHE_HOSTS to be empty will turn off use of
# memcached entirely.
#
# You should not use the loopback address (127.0.0.1) here if using clustering
# as every webapp in the cluster should use the exact same values to prevent
# unneeded cache misses. Set to [] to disable caching of images and fetched data
#MEMCACHE_HOSTS = ['10.10.10.10:11211', '10.10.10.11:11211',
'10.10.10.12:11211']
#DEFAULT_CACHE_DURATION = 60 # Cache images and data for 1 minute
```

```
#####################################
# Filesystem Paths #
#####################################
# Change only GRAPHITE_ROOT if your install is merely shifted from
/opt/graphite
# to somewhere else
#GRAPHITE_ROOT = '/opt/graphite'

# Most installs done outside of a separate tree such as /opt/graphite will only
# need to change these three settings. Note that the default settings for each
# of these is relative to GRAPHITE_ROOT
#CONF_DIR = '/opt/graphite/conf'
#STORAGE_DIR = '/opt/graphite/storage'
#CONTENT_DIR = '/opt/graphite/webapp/content'

# To further or fully customize the paths, modify the following. Note that the
# default settings for each of these are relative to CONF_DIR and STOR-
AGE_DIR
#
## Webapp config files
#DASHBOARD_CONF = '/opt/graphite/conf/dashboard.conf'
#GRAPHTEMPLATES_CONF = '/opt/graphite/conf/graphTemplates.conf'

## Data directories
# NOTE: If any directory is unreadable in DATA_DIRS it will break metric
browsing
#WHISPER_DIR = '/opt/graphite/storage/whisper'
#RRD_DIR = '/opt/graphite/storage/rrd'
#DATA_DIRS = [WHISPER_DIR, RRD_DIR] # Default: set from the above vari-
ables
#LOG_DIR = '/opt/graphite/storage/log/webapp'
#INDEX_FILE = '/opt/graphite/storage/index'  # Search index file

#####################################
# Email Configuration #
#####################################
# This is used for emailing rendered Graphs
# Default backend is SMTP
#EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
#EMAIL_HOST = 'localhost'
#EMAIL_PORT = 25
#EMAIL_HOST_USER = ''
#EMAIL_HOST_PASSWORD = ''
#EMAIL_USE_TLS = False
# To drop emails on the floor, enable the Dummy backend:
#EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'
```

```
#####################################
# Authentication Configuration #
#####################################
## LDAP / ActiveDirectory authentication setup
#USE_LDAP_AUTH = True
#LDAP_SERVER = "ldap.mycompany.com"
#LDAP_PORT = 389
#       OR
#LDAP_URI = "ldaps://ldap.mycompany.com:636"
#LDAP_SEARCH_BASE = "OU=users,DC=mycompany,DC=com"
#LDAP_BASE_USER = "CN=some_readonly_account,DC=my-
company,DC=com"
#LDAP_BASE_PASS = "readonly_account_password"
#LDAP_USER_QUERY = "(username=%s)"  #For Active Directory use "(sA-
MAccountName=%s)"
#
# If you want to further customize the ldap connection options you should
# directly use ldap.set_option to set the ldap module's global options.
# For example:
#
#import ldap
#ldap.set_option(ldap.OPT_X_TLS_REQUIRE_CERT, ldap.OPT_X_TLS_AL-
LOW)
#ldap.set_option(ldap.OPT_X_TLS_CACERTDIR, "/etc/ssl/ca")
#ldap.set_option(ldap.OPT_X_TLS_CERTFILE, "/etc/ssl/mycert.pem")
#ldap.set_option(ldap.OPT_X_TLS_KEYFILE, "/etc/ssl/mykey.pem")
# See http://www.python-ldap.org/ for further details on these options.

## REMOTE_USER authentication. See: https://docs.djangopro-
ject.com/en/dev/howto/auth-remote$
USE_REMOTE_USER_AUTHENTICATION = True

# Override the URL for the login link (e.g. for django_openid_auth)
#LOGIN_URL = '/account/login'

############################
# Database Configuration #
############################
# By default sqlite is used. If you cluster multiple webapps you will need
# to setup an external database (such as MySQL) and configure all of the webapp
# instances to use the same database. Note that this database is only used to store
# Django models such as saved graphs, dashboards, user preferences, etc.
# Metric data is not stored here.
#
# DO NOT FORGET TO RUN 'manage.py syncdb' AFTER SETTING UP A NEW
DATABASE
#
# The following built-in database engines are available:
```

```
#  django.db.backends.postgresql          # Removed in Django 1.4
#  django.db.backends.postgresql_psycopg2
#  django.db.backends.mysql
#  django.db.backends.sqlite3
#  django.db.backends.oracle
#
# The default is 'django.db.backends.sqlite3' with file 'graphite.db'
# located in STORAGE_DIR
#
# Users with Django 1.2 or greater should use the new dictionary
# specification as the old database specification style is removed in 1.4
DATABASES = {
   'default': {
      'NAME': '/opt/graphite/webapp/graphite/graphite.db',
      'ENGINE': 'django.db.backends.sqlite3',
      'USER': '',
      'PASSWORD': '',
      'HOST': '',
      'PORT': ''
   }
}
#
# Users still on Django 1.1 must use the old method instead:
#DATABASE_ENGINE = 'django.db.backends.mysql'
#DATABASE_NAME = 'graphite'      # Or path to the database file if using sqlite3
#DATABASE_USER = 'graphite'
#DATABASE_PASSWORD = 'graphite-is-awesome'
#DATABASE_HOST = 'mysql.mycompany.com'
#DATABASE_PORT = '3306'


############################
# Cluster Configuration #
############################
# (To avoid excessive DNS lookups you want to stick to using IP addresses only in
this enti$
#
# This should list the IP address (and optionally port) of the webapp on each
# remote server in the cluster. These servers must each have local access to
# metric data. Note that the first server to return a match for a query will be
# used.
#CLUSTER_SERVERS = ["10.0.2.2:80", "10.0.2.3:80"]

## These are timeout values (in seconds) for requests to remote webapps
#REMOTE_STORE_FETCH_TIMEOUT = 6   # Timeout to fetch series data
#REMOTE_STORE_FIND_TIMEOUT = 2.5  # Timeout for metric find requests
#REMOTE_STORE_RETRY_DELAY = 60    # Time before retrying a failed re-
mote webapp
```

```
#REMOTE_FIND_CACHE_DURATION = 300 # Time to cache remote metric
find results

## Remote rendering settings
# Set to True to enable rendering of Graphs on a remote webapp
#REMOTE_RENDERING = True
# List of IP (and optionally port) of the webapp on each remote server that
# will be used for rendering. Note that each rendering host should have local
# access to metric data or should have CLUSTER_SERVERS configured
#RENDERING_HOSTS = []
#REMOTE_RENDER_CONNECT_TIMEOUT = 1.0

# If you are running multiple carbon-caches on this machine (typically behind a
relay using
# consistent hashing), you'll need to list the ip address, cache query port, and in-
stance n$
# instance on the local machine (NOT every carbon-cache in the entire cluster).
The default$
# and a common scheme is to use 7102 for instance b, 7202 for instance c, etc.
#
# You *should* use 127.0.0.1 here in most cases
#CARBONLINK_HOSTS = ["127.0.0.1:7002:a", "127.0.0.1:7102:b",
"127.0.0.1:7202:c"]
#CARBONLINK_TIMEOUT = 1.0

#######################################
# Additional Django Settings #
#######################################
# Uncomment the following line for direct access to Django settings such as
# MIDDLEWARE_CLASSES or APPS
#from graphite.app_settings import *
```

Liite 12. Collectd.conf

Collectd.conf tiedoston sisältö kokonaisuudessaan:

```
#
# Config file for collectd(1).
# Please read collectd.conf(5) for a list of options.
# http://collectd.org/
#

##############################################################################
# Global                                                     #
#----------------------------------------------------------------------------#
# Global settings for the daemon.                                            #
##############################################################################
```

```
Hostname    "localhost"
#FQDNLookup   true
#BaseDir     "/opt/collectd/var/lib/collectd"
#PIDFile     "/opt/collectd/var/run/collectd.pid"
PluginDir   "/opt/collectd/lib/collectd"
TypesDB     "/opt/collectd/share/collectd/types.db"
#Interval     10
#Timeout      2
#ReadThreads  5

##############################################################################
# Logging                                                    #
#----------------------------------------------------------------------------#
# Plugins which provide logging functions should be loaded first, so log     #
# messages generated when loading or configuring other plugins can be        #
# accessed.                                                  #
##############################################################################

LoadPlugin syslog
LoadPlugin logfile

<Plugin logfile>
      LogLevel info
      File STDOUT
      Timestamp true
      PrintSeverity false
</Plugin>

#<Plugin syslog>
#       LogLevel info
#</Plugin>

##############################################################################
# LoadPlugin section                                         #
#----------------------------------------------------------------------------#
# Lines beginning with a single `#' belong to plugins which have been built  #
# but are disabled by default.                                 #
#                                                            #
# Lines begnning with `##' belong to plugins which have not been built due   #
# to missing dependencies or because they have been deactivated explicitly.  #
##############################################################################

#LoadPlugin cpufreq
#LoadPlugin csv
##LoadPlugin curl
##LoadPlugin curl_json
##LoadPlugin curl_xml
```

##LoadPlugin dbi
LoadPlugin df
#LoadPlugin disk
##LoadPlugin dns
#LoadPlugin email
#LoadPlugin entropy
#LoadPlugin ethstat
#LoadPlugin exec
#LoadPlugin filecount
#LoadPlugin fscache
##LoadPlugin gmond
#LoadPlugin hddtemp
LoadPlugin interface
##LoadPlugin iptables
##LoadPlugin ipmi
#LoadPlugin ipvs
#LoadPlugin irq
##LoadPlugin java
##LoadPlugin libvirt
LoadPlugin load
##LoadPlugin lpar
#LoadPlugin madwifi
#LoadPlugin mbmon
#LoadPlugin md
##LoadPlugin memcachec
#LoadPlugin memcached
LoadPlugin memory
##LoadPlugin modbus
#LoadPlugin multimeter
##LoadPlugin mysql
##LoadPlugin netapp
##LoadPlugin netlink
LoadPlugin network
#LoadPlugin nfs
##LoadPlugin nginx
##LoadPlugin notify_desktop
##LoadPlugin notify_email
#LoadPlugin ntpd
#LoadPlugin numa
##LoadPlugin nut
#LoadPlugin olsrd
##LoadPlugin onewire
#LoadPlugin openvpn
##LoadPlugin oracle
##<LoadPlugin perl>
Globals true
##</LoadPlugin>
##LoadPlugin pinba

```
##LoadPlugin ping
#LoadPlugin postgresql
#LoadPlugin powerdns
#LoadPlugin processes
#LoadPlugin protocols
#<LoadPlugin python>
#  Globals true
#</LoadPlugin>
##LoadPlugin redis
##LoadPlugin routeros
#LoadPlugin rrdcached
LoadPlugin rrdtool
##LoadPlugin sensors
#LoadPlugin serial
##LoadPlugin snmp
#LoadPlugin swap
#LoadPlugin table
#LoadPlugin tail
##LoadPlugin tape
#LoadPlugin tcpconns
#LoadPlugin teamspeak2
#LoadPlugin ted
#LoadPlugin thermal
##LoadPlugin tokyotyrant
#LoadPlugin unixsock
#LoadPlugin uptime
#LoadPlugin users
#LoadPlugin uuid
##LoadPlugin varnish
#LoadPlugin vmem
#LoadPlugin vserver
#LoadPlugin wireless
LoadPlugin write_graphite
##LoadPlugin write_http
##LoadPlugin write_redis
##LoadPlugin write_mongodb
##LoadPlugin xmms
##LoadPlugin zfs_arc

##############################################################################
# Plugin configuration                                       #
#----------------------------------------------------------------------------#
# In this section configuration stubs for each plugin are provided. A desc-  #
# ription of those options is available in the collectd.conf(5) manual page. #
##############################################################################

#<Plugin "amqp">
#  <Publish "name">
```

```
#   Host "localhost"
#   Port "5672"
#   VHost "/"
#   User "guest"
#   Password "guest"
#   Exchange "amq.fanout"
#   RoutingKey "collectd"
#   Persistent false
#   StoreRates false
#  </Publish>
#</Plugin>

#<Plugin apache>
#  <Instance "local">
#   URL "http://localhost/status?auto"
#   User "www-user"
#   Password "secret"
#   CACert "/etc/ssl/ca.crt"
#  </Instance>
#</Plugin>

#<Plugin apcups>
#       Host "localhost"
#       Port "3551"
#</Plugin>

#<Plugin ascent>
#       URL "http://localhost/ascent/status/"
#       User "www-user"
#       Password "secret"
#       CACert "/etc/ssl/ca.crt"
#</Plugin>

#<Plugin "bind">
#  URL "http://localhost:8053/"
#  ParseTime       false
#  OpCodes         true
#  QTypes          true
#
#  ServerStats     true
#  ZoneMaintStats  true
#  ResolverStats   false
#  MemoryStats     true
#
#  <View "_default">
#   QTypes        true
#   ResolverStats true
#   CacheRRSets   true
```

```
#
#    Zone "127.in-addr.arpa/IN"
#  </View>
#</Plugin>

#<Plugin csv>
#       DataDir "/opt/collectd/var/lib/collectd/csv"
#       StoreRates false
#</Plugin>

#<Plugin curl>
#  <Page "stock_quotes">
#    URL "http://finance.google.com/finance?q=NYSE%3AAMD"
#    User "foo"
#    Password "bar"
#    MeasureResponseTime false
#    <Match>
#      Regex "<span +class=\"pr\"[^>]*> *([0-9]*\\.[0-9]+) *</span>"
#      DSType "GaugeAverage"
#      Type "stock_value"
#      Instance "AMD"
#    </Match>
#  </Page>
#</Plugin>

#<Plugin curl_json>
## See: http://wiki.apache.org/couchdb/Runtime_Statistics
#  <URL "http://localhost:5984/_stats">
#    Instance "httpd"
#    <Key "httpd/requests/count">
#      Type "http_requests"
#    </Key>
#
#    <Key "httpd_request_methods/*/count">
#      Type "http_request_methods"
#    </Key>
#
#    <Key "httpd_status_codes/*/count">
#      Type "http_response_codes"
#    </Key>
#  </URL>
## Database status metrics:
#  <URL "http://localhost:5984/_all_dbs">
#    Instance "dbs"
#    <Key "*/doc_count">
#      Type "gauge"
#    </Key>
#    <Key "*/doc_del_count">
```

```
#     Type "counter"
#   </Key>
#   <Key "*/disk_size">
#     Type "bytes"
#   </Key>
#  </URL>
#</Plugin>

#<Plugin "curl_xml">
#  <URL "http://localhost/stats.xml">
#   Host "my_host"
#   Instance "some_instance"
#   User "collectd"
#   Password "thaiNg0I"
#   VerifyPeer true
#   VerifyHost true
#   CACert "/path/to/ca.crt"
#
#   <XPath "table[@id=\"magic_level\"]/tr">
#     Type "magic_level"
#     #InstancePrefix "prefix-"
#     InstanceFrom "td[1]"
#     ValuesFrom "td[2]/span[@class=\"level\"]"
#   </XPath>
#  </URL>
#</Plugin>

#<Plugin dbi>
#      <Query "num_of_customers">
#            Statement "SELECT 'customers' AS c_key, COUNT(*) AS c_value
FROM customers_$
#            <Result>
#                  Type "gauge"
#                  InstancesFrom "c_key"
#                  ValuesFrom "c_value"
#            </Result>
#      </Query>
#      <Database "customers_db">
#            Driver "mysql"
#            DriverOption "host" "localhost"
#            DriverOption "username" "collectd"
#            DriverOption "password" "AeXohy0O"
#            DriverOption "dbname" "custdb0"
#            #SelectDB "custdb0"
#            Query "num_of_customers"
#            #Query "..."
#      </Database>
#</Plugin>
```

```
<Plugin df>
#       Device "/dev/md2"
#       Device "192.168.0.2:/mnt/nfs"
#       MountPoint "/"
#       FSType "ext3"
#       IgnoreSelected false
#       ReportByDevice false
#       ReportReserved false
#       ReportInodes false
</Plugin>

#<Plugin disk>
#       Disk "/^[hs]d[a-f][0-9]?$/"
#       IgnoreSelected false
#</Plugin>

#<Plugin dns>
#       Interface "eth0"
#       IgnoreSource "192.168.0.1"
#       SelectNumericQueryTypes true
#</Plugin>

#<Plugin email>
#       SocketFile "/opt/collectd/var/run/collectd-email"
#       SocketGroup "collectd"
#       SocketPerms "0770"
#       MaxConns 5
#</Plugin>

#<Plugin ethstat>
#       Interface "eth0"
#       Map "rx_csum_offload_errors" "if_rx_errors" "checksum_offload"
#       Map "multicast" "if_multicast"
#       MappedOnly false
#</Plugin>

#<Plugin exec>
#       Exec "user:group" "/path/to/exec"
#       NotificationExec "user:group" "/path/to/exec"
#</Plugin>

#<Plugin filecount>
#       <Directory "/path/to/dir">
#               Instance "foodir"
#               Name "*.conf"
#               MTime "-5m"
#               Size "+10k"
```

```
#             Recursive true
#             IncludeHidden false
#       </Directory>
#</Plugin>

#<Plugin "gmond">
#  MCReceiveFrom "239.2.11.71" "8649"
#  <Metric "swap_total">
#    Type "swap"
#    TypeInstance "total"
#    DataSource "value"
#  </Metric>
#  <Metric "swap_free">
#    Type "swap"
#    TypeInstance "free"
#    DataSource "value"
#  </Metric>
#</Plugin>

#<Plugin hddtemp>
#  Host "127.0.0.1"
#  Port "7634"
#</Plugin>

<Plugin interface>
       Interface "eth0"
       IgnoreSelected false
</Plugin>

#<Plugin ipmi>
#       Sensor "some_sensor"
#       Sensor "another_one"
#       IgnoreSelected false
#       NotifySensorAdd false
#       NotifySensorRemove true
#       NotifySensorNotPresent false
#</Plugin>

#<Plugin iptables>
#       Chain table chain
#</Plugin>

#<Plugin irq>
#       Irq 7
#       Irq 8
#       Irq 9
#       IgnoreSelected true
#</Plugin>
```

```
#<Plugin "java">
#       JVMArg "-verbose:jni"
#       JVMArg "-Djava.class.path=/opt/collectd/share/collectd/java/collectd-
api.jar"
#
#       LoadPlugin "org.collectd.java.Foobar"
#       <Plugin "org.collectd.java.Foobar">
#         # To be parsed by the plugin
#       </Plugin>
#</Plugin>

#<Plugin libvirt>
#       Connection "xen:///"
#       RefreshInterval 60
#       Domain "name"
#       BlockDevice "name:device"
#       InterfaceDevice "name:device"
#       IgnoreSelected false
#       HostnameFormat name
#       InterfaceFormat name
#</Plugin>

#<Plugin lpar>
#       CpuPoolStats   false
#       ReportBySerial false
#</Plugin>

#<Plugin madwifi>
#       Interface "wlan0"
#       IgnoreSelected false
#       Source "SysFS"
#       WatchSet "None"
#       WatchAdd "node_octets"
#       WatchAdd "node_rssi"
#       WatchAdd "is_rx_acl"
#       WatchAdd "is_scan_active"
#</Plugin>

#<Plugin mbmon>
#       Host "127.0.0.1"
#       Port "411"
#</Plugin>

#<Plugin md>
#       Device "/dev/md0"
#       IgnoreSelected false
#</Plugin>
```

```
#<Plugin memcachec>
#       <Page "plugin_instance">
#               Server "localhost"
#               Key "page_key"
#               <Match>
#                       Regex "(\\d+) bytes sent"
#                       ExcludeRegex "<lines to be excluded>"
#                       DSType CounterAdd
#                       Type "ipt_octets"
#                       Instance "type_instance"
#               </Match>
#       </Page>
#</Plugin>

#<Plugin memcached>
#       Host "127.0.0.1"
#       Port "11211"
#</Plugin>

#<Plugin modbus>
#       <Data "data_name">
#               RegisterBase 1234
#               RegisterType float
#               Type gauge
#               Instance "..."
#       </Data>
#
#       <Host "name">
#               Address "addr"
#               Port "1234"
#               Interval 60
#
#               <Slave 1>
#                       Instance "foobar" # optional
#                       Collect "data_name"
#               </Slave>
#       </Host>
#</Plugin>

#<Plugin mysql>
#       <Database db_name>
#               Host "database.serv.er"
#               User "db_user"
#               Password "secret"
#               Database "db_name"
#               MasterStats true
#       </Database>
```

```
#
#     <Database db_name2>
#         Host "localhost"
#         Socket "/var/run/mysql/mysqld.sock"
#         SlaveStats true
#         SlaveNotifications true
#     </Database>
#</Plugin>

#<Plugin netapp>
#     <Host "netapp1.example.com">
#         Protocol      "https"
#         Address       "10.0.0.1"
#         Port          443
#         User          "username"
#         Password      "aef4Aebe"
#         Interval      30
#
#         <WAFL>
#             Interval 30
#             GetNameCache   true
#             GetDirCache    true
#             GetBufferCache true
#             GetInodeCache  true
#         </WAFL>
#
#         <Disks>
#             Interval 30
#             GetBusy true
#         </Disks>
#
#         <VolumePerf>
#             Interval 30
#             GetIO      "volume0"
#             IgnoreSelectedIO      false
#             GetOps     "volume0"
#             IgnoreSelectedOps     false
#             GetLatency "volume0"
#             IgnoreSelectedLatency false
#         </VolumePerf>
#
#         <VolumeUsage>
#             Interval 30
#             GetCapacity "vol0"
#             GetCapacity "vol1"
#             IgnoreSelectedCapacity false
#             GetSnapshot "vol1"
#             GetSnapshot "vol3"
```

```
#                 IgnoreSelectedSnapshot false
#             </VolumeUsage>
#
#             <System>
#                 Interval 30
#                 GetCPULoad     true
#                 GetInterfaces  true
#                 GetDiskOps     true
#                 GetDiskIO      true
#             </System>
#     </Host>
#</Plugin>

#<Plugin netlink>
#     Interface "All"
#     VerboseInterface "All"
#     QDisc "eth0" "pfifo_fast-1:0"
#     Class "ppp0" "htb-1:10"
#     Filter "ppp0" "u32-1:0"
#     IgnoreSelected false
#</Plugin>

<Plugin network>
#     # client setup:
#     Server "ff18::efc0:4a42" "25826"
#     <Server "239.192.74.66" "25826">
#         SecurityLevel Encrypt
#         Username "user"
#         Password "secret"
#         Interface "eth0"
#     </Server>
#     TimeToLive "128"
#
#     # server setup:
#     Listen "ff18::efc0:4a42" "25826"
#     <Listen "239.192.74.66" "25826">
#         SecurityLevel Sign
#         AuthFile "/opt/collectd/etc/passwd"
#         Interface "eth0"
#     </Listen>

     Listen "239.192.74.66"

#     MaxPacketSize 1024
#
#     # proxy setup (client and server as above):
#     Forward true
#
```

```
#       # statistics about the network plugin itself
#       ReportStats false
#
#       # "garbage collection"
#       CacheFlush 1800
</Plugin>

#<Plugin nginx>
#       URL "http://localhost/status?auto"
#       User "www-user"
#       Password "secret"
#       CACert "/etc/ssl/ca.crt"
#</Plugin>

#<Plugin notify_desktop>
#       OkayTimeout 1000
#       WarningTimeout 5000
#       FailureTimeout 0
#</Plugin>

#<Plugin notify_email>
#       SMTPServer "localhost"
#       SMTPPort 25
#       SMTPUser "my-username"
#       SMTPPassword "my-password"
#       From "collectd@main0server.com"
#       # <WARNING/FAILURE/OK> on <hostname>. beware! do not use not
more than two %s in th$
#       Subject "Aaaaaa!! %s on %s!!!!!"
#       Recipient "email1@domain1.net"
#       Recipient "email2@domain2.com"
#</Plugin>

#<Plugin ntpd>
#       Host "localhost"
#       Port 123
#       ReverseLookups false
#</Plugin>

#<Plugin nut>
#       UPS "upsname@hostname:port"
#</Plugin>

#<Plugin olsrd>
#       Host "127.0.0.1"
#       Port "2006"
#       CollectLinks "Summary"
#       CollectRoutes "Summary"
```

```
#       CollectTopology "Summary"
#</Plugin>

#<Plugin onewire>
#       Device "-s localhost:4304"
#       Sensor "F10FCA000800"
#       IgnoreSelected false
#</Plugin>

#<Plugin openvpn>
#       StatusFile "/etc/openvpn/openvpn-status.log"
#       ImprovedNamingSchema false
#       CollectCompression true
#       CollectIndividualUsers true
#       CollectUserCount false
#</Plugin>

#<Plugin oracle>
#  <Query "out_of_stock">
#    Statement "SELECT category, COUNT(*) AS value FROM products WHERE
in_stock = 0 GROUP B$
#    <Result>
#      Type "gauge"
#      InstancesFrom "category"
#      ValuesFrom "value"
#    </Result>
#  </Query>
#  <Database "product_information">
#    ConnectID "db01"
#    Username "oracle"
#    Password "secret"
#    Query "out_of_stock"
#  </Database>
#</Plugin>

#<Plugin perl>
#       IncludeDir "/my/include/path"
#       BaseName "Collectd::Plugins"
#       EnableDebugger ""
#       LoadPlugin Monitorus
#       LoadPlugin OpenVZ
#
#       <Plugin foo>
#               Foo "Bar"
#               Qux "Baz"
#       </Plugin>
#</Plugin>
```

```
#<Plugin pinba>
#       Address "::0"
#       Port "30002"
#       <View "name">
#               Host "host name"
#               Server "server name"
#               Script "script name"
#       </View>
#</Plugin>

#<Plugin ping>
#       Host "host.foo.bar"
#       Interval 1.0
#       Timeout 0.9
#       TTL 255
#       SourceAddress "1.2.3.4"
#       Device "eth0"
#       MaxMissed -1
#</Plugin>

#<Plugin postgresql>
#       <Query magic>
#               Statement "SELECT magic FROM wizard WHERE host = $1;"
#               Param hostname
#               <Result>
#                       Type gauge
#                       InstancePrefix "magic"
#                       ValuesFrom magic
#               </Result>
#       </Query>
#       <Query rt36_tickets>
#               Statement "SELECT COUNT(type) AS count, type \
#                               FROM (SELECT CASE \
#                                       WHEN resolved = 'epoch' THEN 'open' \
#                                       ELSE 'resolved' END AS type \
#                                       FROM tickets) type \
#                               GROUP BY type;"
#               <Result>
#                       Type counter
#                       InstancePrefix "rt36_tickets"
#                       InstancesFrom "type"
#                       ValuesFrom "count"
#               </Result>
#       </Query>
#       <Database foo>
#               Host "hostname"
#               Port "5432"
#               User "username"
```

```
#             Password "secret"
#             SSLMode "prefer"
#             KRBSrvName "kerberos_service_name"
#             Query magic
#       </Database>
#       <Database bar>
#             Interval 60
#             Service "service_name"
#             Query backend # predefined
#             Query rt36_tickets
#       </Database>
#</Plugin>

#<Plugin powerdns>
#  <Server "server_name">
#    Collect "latency"
#    Collect "udp-answers" "udp-queries"
#    Socket "/var/run/pdns.controlsocket"
#  </Server>
#  <Recursor "recursor_name">
#    Collect "questions"
#    Collect "cache-hits" "cache-misses"
#    Socket "/var/run/pdns_recursor.controlsocket"
#  </Recursor>
#  LocalSocket "/opt/collectd/var/run/collectd-powerdns"
#</Plugin>

#<Plugin processes>
#       Process "name"
#</Plugin>

#<Plugin protocols>
#       Value "/^Tcp:/"
#       IgnoreSelected false
#</Plugin>

#<Plugin python>
#       ModulePath "/path/to/your/python/modules"
#       LogTraces true
#       Interactive true
#       Import "spam"
#
#       <Module spam>
#             spam "wonderful" "lovely"
#       </Module>
#</Plugin>

#<Plugin redis>
```

```
#   <Node example>
#     Host "redis.example.com"
#     Port "6379"
#     Timeout 2000
#   </Node>
#</Plugin>

#<Plugin routeros>
#       <Router>
#               Host "router.example.com"
#               Port "8728"
#               User "admin"
#               Password "dozaiTh4"
#               CollectInterface true
#               CollectRegistrationTable true
#               CollectCPULoad true
#               CollectMemory true
#               CollectDF true
#               CollectDisk true
#       </Router>
#</Plugin>

#<Plugin rrdcached>
#       DaemonAddress "unix:/tmp/rrdcached.sock"
#       DataDir "/opt/collectd/var/lib/collectd/rrd"
#       CreateFiles true
#       CollectStatistics true
#</Plugin>

#<Plugin rrdtool>
#        DataDir "/opt/collectd/var/lib/collectd/rrd"
#      CacheTimeout 120
#      CacheFlush   900
#</Plugin>

#<Plugin sensors>
#       SensorConfigFile "/etc/sensors.conf"
#       Sensor "it8712-isa-0290/temperature-temp1"
#       Sensor "it8712-isa-0290/fanspeed-fan3"
#       Sensor "it8712-isa-0290/voltage-in8"
#       IgnoreSelected false
#</Plugin>

#<Plugin snmp>
#   <Data "powerplus_voltge_input">
#       Type "voltage"
#       Table false
#       Instance "input_line1"
```

```
#     Values "SNMPv2-SMI::enterprises.6050.5.4.1.1.2.1"
#   </Data>
#   <Data "hr_users">
#     Type "users"
#     Table false
#     Instance ""
#     Values "HOST-RESOURCES-MIB::hrSystemNumUsers.0"
#   </Data>
#   <Data "std_traffic">
#     Type "if_octets"
#     Table true
#     Instance "IF-MIB::ifDescr"
#     Values "IF-MIB::ifInOctets" "IF-MIB::ifOutOctets"
#   </Data>
#
#   <Host "some.switch.mydomain.org">
#     Address "192.168.0.2"
#     Version 1
#     Community "community_string"
#     Collect "std_traffic"
#     Interval 120
#   </Host>
#   <Host "some.server.mydomain.org">
#     Address "192.168.0.42"
#     Version 2
#     Community "another_string"
#     Collect "std_traffic" "hr_users"
#   </Host>
#   <Host "some.ups.mydomain.org">
#     Address "192.168.0.3"
#     Version 1
#     Community "more_communities"
#     Collect "powerplus_voltge_input"
#     Interval 300
#   </Host>
#</Plugin>

#<Plugin "swap">
#     ReportByDevice false
#</Plugin>

#<Plugin "table">
#     <Table "/proc/slabinfo">
#          Instance "slabinfo"
#          Separator " "
#          <Result>
#               Type gauge
#               InstancePrefix "active_objs"
```

```
#                   InstancesFrom 0
#                   ValuesFrom 1
#             </Result>
#             <Result>
#                   Type gauge
#                   InstancePrefix "objperslab"
#                   InstancesFrom 0
#                   ValuesFrom 4
#             </Result>
#       </Table>
#</Plugin>

#<Plugin "tail">
#  <File "/var/log/exim4/mainlog">
#    Instance "exim"
#    <Match>
#      Regex "S=([1-9][0-9]*)"
#      DSType "CounterAdd"
#      Type "ipt_bytes"
#      Instance "total"
#    </Match>
#    <Match>
#      Regex "\\<R=local_user\\>"
#      ExcludeRegex "\\<R=local_user\\>.*mail_spool defer"
#      DSType "CounterInc"
#      Type "counter"
#      Instance "local_user"
#    </Match>
#  </File>
#</Plugin>

#<Plugin tcpconns>
#       ListeningPorts false
#       LocalPort "25"
#       RemotePort "25"
#</Plugin>

#<Plugin teamspeak2>
#       Host "127.0.0.1"
#       Port "51234"
#       Server "8767"
#</Plugin>

#<Plugin ted>
#       Device "/dev/ttyUSB0"
#       Retries 0
#</Plugin>
```

```
#<Plugin thermal>
#       ForceUseProcfs false
#       Device "THRM"
#       IgnoreSelected false
#</Plugin>

#<Plugin tokyotyrant>
#       Host "localhost"
#       Port "1978"
#</Plugin>

#<Plugin unixsock>
#       SocketFile "/opt/collectd/var/run/collectd-unixsock"
#       SocketGroup "collectd"
#       SocketPerms "0660"
#       DeleteSocket false
#</Plugin>

#<Plugin uuid>
#       UUIDFile "/etc/uuid"
#</Plugin>

#<Plugin varnish>
#   This tag support an argument if you want to
#   monitor the local instance just use </Instance>
#   If you prefer defining another instance you can do
#   so by using <Instance "myinstance">
#   <Instance>
#     CollectCache true
#     CollectBackend true
#     CollectConnections true
#     CollectSHM true
#     CollectESI false
#     CollectFetch false
#     CollectHCB false
#     CollectSMA false
#     CollectSMS false
#     CollectSM false
#     CollectTotals false
#     CollectWorkers false
#   </Instance>
#</Plugin>

#<Plugin vmem>
#       Verbose false
#</Plugin>

<Plugin write_graphite>
```

```
  <Carbon>
    Host "localhost"
    Port "2003"
    Prefix "collectd"
#   Postfix "collectd"
    StoreRates false
    AlwaysAppendDS false
    EscapeCharacter "_"
  </Carbon>
</Plugin>

#<Plugin write_http>
#       <URL "http://example.com/collectd-post">
#               User "collectd"
#               Password "weCh3ik0"
#               VerifyPeer true
#               VerifyHost true
#               CACert "/etc/ssl/ca.crt"
#               Format "Command"
#               StoreRates false
#       </URL>
#</Plugin>

#<Plugin write_redis>
#       <Node "example">
#               Host "localhost"
#               Port "6379"
#               Timeout 1000
#       </Node>
#</Plugin>

#<Plugin write_redis>
#       <Node "example">
#               Host "localhost"
#               Port "6379"
#               Timeout 1000
#       </Node>
#</Plugin>

#<Plugin write_mongodb>
#       <Node "example">
#               Host "localhost"
#               Port "27017"
#               Timeout 1000
#               StoreRates false
#       </Node>
#</Plugin>
```

```
##############################################################################
# Filter configuration                                                       #
#----------------------------------------------------------------------------#
# The following configures collectd's filtering mechanism. Before changing   #
# anything in this section, please read the `FILTER CONFIGURATION' section   #
#                                                                            #
# in the collectd.conf(5) manual page.                                       #
##############################################################################

# Load required matches:
#LoadPlugin match_empty_counter
#LoadPlugin match_hashed
#LoadPlugin match_regex
#LoadPlugin match_value
#LoadPlugin match_timediff

# Load required targets:
#LoadPlugin target_notification
#LoadPlugin target_replace
#LoadPlugin target_scale
#LoadPlugin target_set
#LoadPlugin target_v5upgrade

#----------------------------------------------------------------------------#
# The following block demonstrates the default behavior if no filtering is   #
# configured at all: All values will be sent to all available write plugins. #
#----------------------------------------------------------------------------#

#<Chain "PostCache">
#  Target "write"
#</Chain>

##############################################################################
# Threshold configuration                                                    #
#----------------------------------------------------------------------------#
# The following outlines how to configure collectd's threshold checking      #
# plugin. The plugin and possible configuration options are documented in    #
# the collectd-threshold(5) manual page.                                     #
##############################################################################

#LoadPlugin "threshold"
#<Plugin "threshold">
#  <Type "foo">
#    WarningMin    0.00
#    WarningMax 1000.00
#    FailureMin    0.00
#    FailureMax 1200.00
#    Invert false
```

```
#   Instance "bar"
#  </Type>
#
#  <Plugin "interface">
#   Instance "eth0"
#    <Type "if_octets">
#     FailureMax 10000000
#     DataSource "rx"
#    </Type>
#  </Plugin>
#
#  <Host "hostname">
#    <Type "cpu">
#     Instance "idle"
#     FailureMin 10
#    </Type>
#
#    <Plugin "memory">
#      <Type "memory">
#       Instance "cached"
#       WarningMin 100000000
#      </Type>
#    </Plugin>
#
#    <Type "load">
#      DataSource "midterm"
#      FailureMax 4
#      Hits 3
#      Hysteresis 3
#    </Type>
#  </Host>
#</Plugin>
```