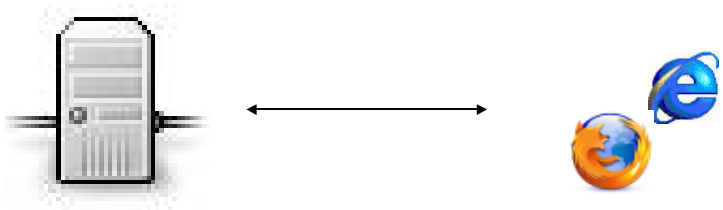
Korppi

Tekniikka ja kehittäminen

25.10.2004
Minna Hillebrand
Pauli Kujala

Korppi

- www-järjestelmä https://korppi.jyu.fi
- Nykyversiota aloitettu syksyllä 2000.
- Kehittäjiä 4-12, koodirivejä ainakin 255 000.

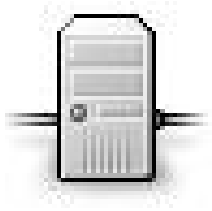


palvelin asiakas

Tekniikka (palvelin)

- RedHat
- Apache
  - HTTP-pyynnöt
- Tomcat
  - Java Server Pages
  - Java-luokat
  - Valmiit kirjastot
- PostgreSQL

- Lisäksi shell-skriptejä, croneja…



palvelin

Asiakaspään vaatimukset

- WWW-selain (https)
- Evästeet (cookie) ja istunto (sessio, palvelimella)

- ”selainriippumaton”
- CSS-tyylit, selainten ominaisuudet, oma tyylitiedosto, osien piilottaminen

asiakas

Kapasiteetti

- 2001 syksy – 2002 alku
  - 2 kpl 500 MHz, 256 MB, 2 kpl 18 Gt
  - noin 40 yhtäaikaista käyttäjää, rekisteröityneitä lähinnä it-tdk:sta
- 2002 alku – 2004 alku
  - 2 kpl 1200 MHz, 1024 MB, 2 kpl 18 Gt
  - noin 100 yhtäaikaista käyttäjää, rekisteröityneitä 12 000
  - kielikeskus, avoin yo
- 2004 alku –
  - 2 x 2 kpl 2133 MHz, 2048 MB, 4 kpl 36 Gt
  - ainakin 250 käyttäjää, rekisteröityneitä reilut 27 000
  - kaikki opiskelijat

Korppi EXIT @ ?

Käyttöaste

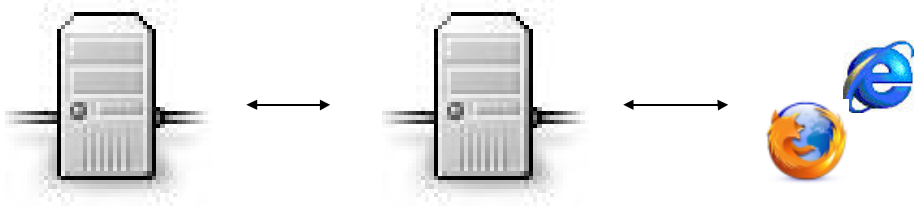
- Jatkuvasti ainakin 20 kirjautunutta.
- Lokia generoituu gigan verran normaalikäytössä !

- http://korppi.jyu.fi/statistics

- Nykyinen käyttökuorma kestetään.
- Tosin syyskuun alkupäivinä oli pieniä ongelmia.

Parempia arkkitehtuureja?

- Palvelimen kuorman hajauttaminen.



tietokanta-palvelin

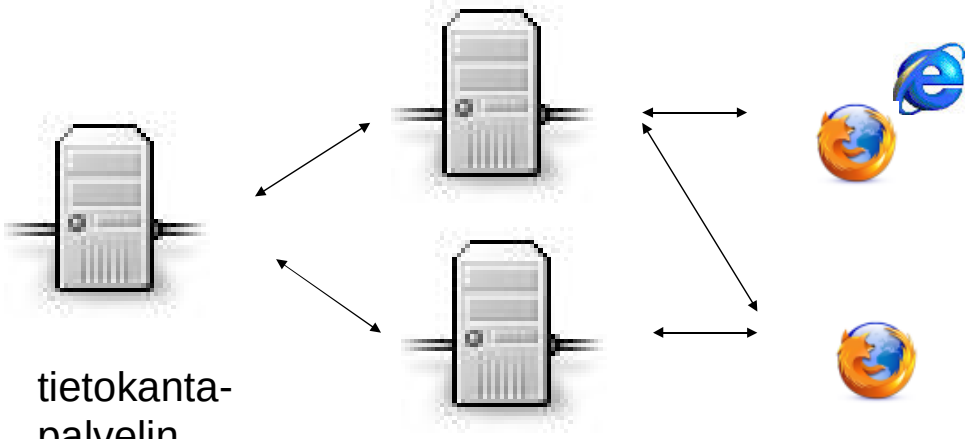
palvelin

asiakas

Korppi EXIT @ ?

Vielä parempia arkkitehtuureja?

- Palvelimen kuorman hajauttaminen.



tietokanta-palvelin

palvelin

asiakas

Muita mahdollisia/tulevia ratkaisuja

- Tietokannan hajautus usealle palvelimelle.
  - Vaikeaa, synkronointi on ongelma
  - Osittainen hajautus?
  - Toteutustekniikatkin tuottavat haasteita hajautukseen.
- Lokin hajauttaminen
  - Suurten tiedostojen käsittely hankalaa, aiheuttaa ongelmia jo Javan buffereissakin.
  - Loki joka tunnille?

Korppi

Kehittämisstrategia

1. Tehdään mitä pyydetään.
2.* Tehdään paremmin.

* valinnainen, tehdään pyynnöstä

Tavallinen tarina

- Tietokantahaku
  - Aloitetaan kyselyllä, joka tekee tehtävänsä.
  - Joku valittaa hitaudesta.
  - Optimoidaan SQL-kyselyä.
  - Lisätään kantaan indeksejä.
  - Muokataan tulosjoukon käsittelyä.
- Ongelmia
  - Hitautta ei havaita pienellä datamäärällä.
  - Alikyselyissä IN vs. EXISTS riippuu mm. datamäärästä.
  - Tulosjoukon modifiointimahdollisuus teknisesti nuori.

Korppi

Prosessimalli

- Code and Fix, vai olisiko sittenkin iteratiivista ?
- Dokumentaatio olematonta.

+ Saadaan tuloksia
- Testaus puutteellista

Perustelut

- Asiakkaita on paljon.
  - Paljon toiveita.
  - Muutokset toimitettava nopeasti (”maksaja määrää”).
- Palautetta tulee paljon.
  - Sähköpostia noin 50 viestiä / päivä.
  - Syksyllä 150-200 viestiä / päivä parin-kolmen viikon ajan.
  - Virheraportteja harvoin, kerran kahdessa viikossa.
  - Kannustusta kerran kahdessa kuussa .
- Nopea toimitus on ainoa valtti.
  - Korppi ei ole vakiintunut järjestelmä. Kilpailijoita on.

Korppi

Lisää perusteluita

- Korppia kehutaan muunneltavaksi.
  - Sovellusprojekteina tuotetaan uusia moduuleita.
  - Moduulit pyritään integroimaan, mutta (järkevät) keinot nähdään vasta myöhemmin.
- Ei resursseja dokumentaation manuaaliseen ylläpitoon.
- Hyvä koodi on yhtä nopeaa kirjoittaa kuin huono koodi.
  - Hyvän koodin keksimiseen menee paljon kauemmin.
  - Huono koodi muuntuu paremmaksi 2-vaiheessa.

Kehitysympäristö

- Pyritään tuotantoa vastaavaan kokoonpanoon.
- Kehityspalvelin, testauspalvelin ja ”esikorppi” (tuotannossa)
- Sovelluskehittimissä myös omat palvelimet.
  - JBuilder, NetBeans
  - Kehitystä voidaan kuitenkin tehdä tekstieditoreilla (-2004).
- Muutamia tuotantotietokannan kopioita.

Korppi

Kehittimien merkitys

- Sovelluskehittimenä JBuilder/NetBeans (2002-)
  - Raaka koodaus.
  - Toiminnallisuuden testaus (lokaali Tomcat).
- Versionhallintana CVS (2002-)
  - Integroitu sovelluskehittimiin, sekä komentoriviltä.
  - Pääasiallinen jakelukanava ja koodin synkronointi.
- Tehtävien hallinnointiin Bugzilla (2003-)
  - Bugit ja kehitysideat.
  - Tehtävien jako ja vaiheen seuranta.

Koodin matka ideasta käyttöön

1. Oma kehitysympäristö: kehitys ja testaus.
2. CVS: koodi muiden saataville ja hyödynnettäväksi.
3. Kehityskorppi.
4. Testikorppi: myös muut kehittäjät testaavat. Bugzillan bugit kuitataan toteutetuiksi yleensä tämän vaiheen jälkeen.
5. Esikorppi: tuotantokäytössä rajatulla käyttäjäjoukolla.
6. Tuotantokorppi.

Korppi

Prosessin pullonkaulat

- Testauksen vähyys on suurin ongelma.
  - Isossa järjestelmässä pienikin muutos voi vaikuttaa yllättäviin paikkoihin. Onko kaikki mahdollinen jo testattu?
- Testaajia ei ole!
  - Kehittäjä ei ole testaaja.
  - Automaattinen testaaminen hankalaa.
- Historian painolasti
  - Esim. muutokset parametrilistoissa.
  - Vanha toiminnallisuus säilytettävä.

Lisää pullonkauloja

- Kommunikointi ja priorisointi.
  - Päällekkäinen työ?
  - Priorisointi kehittäjän mielen mukaan.
- Yhteinen tietokanta.
  - Muutokset tietokannan rakenteessa täytyy olla koodin suhteen synkronissa.

Korppi

Korppi isolla koolla

- Kehittäjätkään eivät enää tunne koko järjestelmää.
  - Päällekkäiset toiminnallisuudet.
  - Ristiriitaiset toiminnallisuudet.
- Tietokannan tauluja paljon.
  - Oppimisen mahdottomuus.
  - Saman taulun kuormittaminen moneen merkitykseen.
- Päivittäminen vaikeaa.
  - Monella kehittäjällä koodi elää jatkuvasti.
  - Milloin riittävän vakaa päivitykseen? Tilanteen jäädyttäminen.

Korppi uudelle kehittäjälle

- Paljon opiskeltavaa.
  - Tekniikka + työkalut.
  - Luokkahierarkia (paketteja 16).
  - Luokkaviidakko.
- Asenne.
  - Ei voi tehdä uutta, vaan kaikki tehdään olemassa olevan päälle. Vanhaa ei saa rikkoa!
  - Yritäpäs testata n+1 paikkaa joiden olemassa olosta et edes tiedä…
- Uutta saa ja kannattaa kuitenkin aina kokeilla.

Korppi

”Best” Practices – asioita joita luultiin hyviksi ratkaisuiksi…

- Sivupyyntöön tallennettu tietokantayhteys.
  - Käyttäjälle taataan tietokantayhteys.
  - Käytännössä rajattu määrä yhteyksiä.
  - Joskus tarvetta sisäkkäisille kyselyille.

”Best” Practices…

- Autonumber
  - Tietokantojen SQL92-standardissa ei ollut juoksevaa integer-kenttää (automaattista avainkenttää), joten tehtiin itse.
  - Taulu, joka ylläpitää maksimiarvoa.
  - Varaa tietokannasta seuraava vapaa arvo: katso viimeinen, ja päivitä taulua.
- Tietokannan triggerit lisäävät uusia rivejä.
  - Maksimiarvojen taulua päivitetään tietokannan sisältä, mikä ohittaa Javan lukitukset.
- Hajautus
  - Joudutaanko määräämään int-alueita?

Korppi

”Best” Practices…

- Oikeudet ja oikeuksien hallinta
  - Alkuun riitti opiskelija, opettaja ja sihteeri.
  - Käyttäjään liittyy tieto organisaatiosta, ja organisaatioilla on hierarkinen rakenne. Näiden perusteella saadaan karkea rajaus.
- Opiskelija ja/tai opettaja monessa tiedekunnassa?
  - Hätäratkaisuna erillinen opiskelija- ja opettajatunnus, tai monta tunnusta?
- Kuka ylläpitää käyttäjätasoja?
  - Sihteerit ”höveleitä”, eivätkä halua lisää työtä.
- Mahdollisesti ratkaisuna ryhmäpohjainen oikeusjako.

”Best” Practices…

- Käyttöliittymän ja toiminnallisuuden erottaminen toisistaan.
  - JSP on käyttöliittymää ja Java-luokat toiminnallisuutta varten.
  - Aiemmin JSP-sivut olivat pitkiä ja Java-luokat lyhyitä (ei juurikaan toimintaa). Tietokannan käsittely oli JSP:ssä.
  - Nyt JSP-sivu on pahimmillaan `class.showStuff()`
  - Menty yli pyrkimyksessä komponenttimaisuuteen.

Korppi

”Best” Practices…

- Olio pitää huolen omasta tilastaan.
  - Tietokannan tietoa alustetaan hakutuloksesta olioksi.
  - Usein olion alustaminen vaatii useiden kyselyiden tuloksia; esimerkiksi kurssille perustiedot, opettajat, opetusryhmät…
  - Olion luominen hidasta!
  - Alustetaan tarpeettomia tietoja.
  - Konstruktoreille SMALL_MODE –parametreja.

Uloskirjautuminen

Olet kirjautunut ulos Korppi-järjestelmästä.

Tervetuloa takaisin.

Korppi-FAQ: nyt on hyvä hetki kysyä.