2016-07-06

Jämförelse av svarstider för olika bilddata-baser för Javabaserade http-servrar

Staffan Bäcklin

EXAMENSARBETE
Datateknik
Kandidatnivå G2E, 15 hp

Institutionen för ingenjörsvetenskap, Högskolan Väst

EXAMENSARBETE

Jämförelse av svarstider för olika bilddatabaser för Javabaserade http-servrar

Sammanfattning

Denna kandidatuppsats berör databaser i javabaserade bildhanteringssystem där bilderna lagras och hämtas som binära objekt. I MySQL och en del andra databashanterare kallas detta format för Blob(Binary large object). För att bildhanteringssystemet skall fungera bra krävs det att man använder en snabb databas.

Syftet har varit att av ett urval databaser utse den databas som är snabbast i avseende på svarstider för hämtning av bilder som lagras som binära objekt i databaser. Databaserna är de fyra välkända databashanterarna MySQL, MariaDB, PostGreSQL och MongoDB.

Testerna har utförts med databaserna integrerade i Javabaserade klient-server moduler för att så mycket som möjligt spegla de villkor som förekommer i ett bildhanteringssystem. De testverktyg som har använts är JMeter som är en avancerad applikation för mätning av svarstider och PerfMon som övervakar åtgång av systemresurser.

MongoDB var den snabbaste bilddatabasen men det finns många osäkerhetsfaktorer som måste beaktas vilket också beskrivs i denna kandidatuppsats.

Trots att många åtgärder för att motverka osäkerhetsfaktorerna har gjorts, förblir mätosäkerheten stor. Mer åtgärder för att isolera databasernas del av svarstiderna i ett klient-server system måste göras. Förslag på åtgärder redogörs i denna kandidatuppsats.

Datum:	2016-07-06
Författare:	Staffan Bäcklin
Examinator:	Thomas Lundqvist
Handledare:	Lars-Åke Edgardh, Spacemetric AB
Program:	Kandidatexamen inom Datateknik, 180 hp
Huvudområde:	Datateknik
Utbildningsnivå:	Grundnivå
Kurskod:	EXD500, 15 hp
Nyckelord:	*Bilddatabas, Java, Servlet, http, MySQL, MariaDB, PostGreSQL, MongoDB, Benchmarking, Svarstid, JMeter, SQL, NoSQL, Blob*
Utgivare:	Högskolan Väst, Institutionen för ingenjörsvetenskap 461 86 Trollhättan Tel: 0520-22 30 00 Fax: 0520-22 32 99, www.hv.se

Benchmark of different image databases for Java-based http-servers

Summary

This bachelor thesis concerns databases in Java-based imaging system where the images are stored and retrieved as binary objects. In MySQL and in some other database management systems this format is called Blob (Binary Large Object). For the imaging system to work well, it is necessary to use a fast database.

The aim has been that out of a sample of databases designate the database that is the fastest in terms of response times for downloading images stored as binary objects in databases. The databases are the four well-known database management systems MySQL, MariaDB, PostgreSQL and MongoDB.

The tests have been conducted with the databases integrated into Java-based client-server modules in order to as much as possible mirror the conditions prevailing in an imaging system. The test tool that has been used are JMeter which is an advanced application for measuring response times and PerfMon to monitor the consumption of system resources.

MongoDB was the fastest image database, but there are many uncertainties that must be considered, which is also explained in this bachelor thesis.

Although many measures to counter the uncertainties have been made, the measurement uncertainty remains big. Further measures to isolate the database part of the response times in a client-server system must be made. Proposed measures are described in this bachelor thesis.

Date:	July 6, 2016
Author:	Staffan Bäcklin
Examiner:	Thomas Lundqvist
Advisor:	Lars-Åke Edgardh, Spacemetric AB
Programme:	Computer Engineering and System Development, 180 HE credits
Main field of study:	Computer Engineering
Education level:	First cycle
Course code:	EXD500, 15 HE credits
Keywords:	*Image database, Java, Servlet, http, MySQL, MariaDB, PostGreSQL, MongoDB, Benchmarking, response time, JMeter, SQL, NoSQL, Blob*
Publisher:	University West, Department of Engineering Science SE-461 86 Trollhättan, Sweden Phone: + 46 520 22 30 00, Fax: + 46 520 22 32 99, www.hv.se

Förord

Våren 2014 tog jag kontakt med Spacemetric [1] som utvecklar bildhanteringssystem för bilder tagna av satellit och luftburna sensorer såsom flygplan och UAV för att göra mitt examensarbete. Förslaget jag fick från Spacemetric var att ta fram en snabbare bildcache. Bildcache avsågs i det här fallet egentligen ett persistent bildlager för företagets bildhaneringssystem.

Huvudsyftet med detta examensarbete var att genom olika mätningar utse en databas av ett urval som kan anses snabbast med avseende på hur databaserna används. Databaserna ingår var och en i en Java-baserad klient-server lösning som testas som helhet. Beställaren av examensarbetet är Lars-Åke Edgardh på Spacemetrics som ville veta om det fanns snabbbare databaser som SpaceMetric kan använda för sin produkt KeyStone. Inriktningen har varit att genom mätningar kunna göra bra jämförelser för utvärdering med databasen MySQL som utgångspunkt.

Förutom att redovisa mätresultaten så har metodik och förhållande som råder för att göra jämförelsetester gjorts.

Stort tack till Lars-Åke Edghard på SpaceMetric som kommit med bra idéer och synpunkter samt gett mig ett mycket intressant examensarbete.

Innehållsförteckning

Bilagor

A. Mätresultat

Nomenklatur

Relationsdatabas - Databas där informationen är organiserad i relationer

Blob – Binary large object. Lagring av fil i databas som hel enhet.

Post - Enhet av sammanhängande data.

SQL - Structured Query Language, är ett standardiserat programspråk för att hämta och modifiera data i en relationsdatabas.

NoSQL - Databas som frångår principen att lagra information i relationer och som har mer flexibel struktur hur data lagras.

Dokumentorienterad - Lagrar enheter av data som dokument istället för poster så som det lagras i relationsdatabaser.

Svarstid - Den tid det tar från att en förfrågan till ett system/enhet skickas till dess att den är besvarad.

Servletcontainer - Javabaserad server som hanterar servlets.

Servlet - Javaklass inuti en servletcontainer som tar emot förfrågningar via http och avger respons i önskat format som exempelvis HTML, XML, filer mm.

GIS – Geographic Information System som är datorbaserat system för att samla in, lagra, analysera och presentera geografiska data

NASA - USA:s federala myndighet för rymdfart men är också ledande i utveckling av GIS och hantering av geografisk data.
Tråd – Del av process(program) som delar gemensamma resurser som bl.a. minne som används av en process.

Genomsnittlig standardavvikelse = $\sqrt{(1/N \sum(\text{Standard Avvikelse})^2)}$.

1 Inledning

Spacemetric utvecklar ett bildhanteringssystem som heter KeyStone. Keystone är ett javabaserat klient-server system för lagring och visning av geospatiala bilder. Att bilderna klassas som geospatiala innebär att i bilderna eller i anslutning till bilderna finns telemetrisk information.

2014 består bildlagret av en MySQL databas som lagrar bilder i form av bildrutor som kallas för Tile på engelska. Bildrutorna är kvadratiska och används i geospatiala visualiseringar i 2D, bl.a. World Wind som visar en jordglob och vartefter man panorerar och zoomar visas skarpare och skarpare bilder. Mer detaljerat om bildrutesystemet som tillämpas i WorldWind kommer senare i denna uppsats.

Den stora frågan var om det fanns andra bildlagringssystem som gör åtkomsten till bilderna snabbare, d.v.s. kortare svarstid. Krav att bildlagrena kan hantera flera samtidiga anrop finns dels för att KeyStone för o fylla en bild med bildrutor gör minst 6 st. http anrop, 1 per bildruta för att få bildrutorna så snart som möjligt och för att KeyStone oftast har flera samtidiga användare.

För att ta fram ett snabbare bildlager skulle några lagringssystem som Spacemetric var nyfikna på utvärderas, bl.a. MariaDB som är en annan databasserver med öppen källkod. Eventuellt andra lagersystem än relationsdatabaser var också intressant. Det skall också vara databasservrar som fungerar med javabaserade servrar och då i synnerhet Tomcat som KeyStone använder.

1.1 Syfte

Ändamålet med arbetet är att utvärdera vilken av 4 välkända databashanterare (som utvecklas med öppen källkod eller som har sitt ursprung från utveckling med öppen källkod) som är snabbast, d.v.s. med de kortaste svarstiderna som kan användas som bilddatabas för en javabaserad http-server.

1.1.1 Mätvärde

Huvudfrågan är vilken databas som är snabbast så det som mätts, behandlats och analyserats är svarstid. Ju kortare svarstid ju bättre.

1.1.2 Databasservrar/Bildlager

De 4 databashanterarna är MySQL, MariaDB, PostGreSQL och MongoDB som är en s.k. NoSQL databashanterare.

1.2 Problemställningar

Med projektet kom flera problem och problemställningar. Vad gör att en databasserver klassas som snabbare än andra? Vilka kriterier skall tillämpas?

Vad skall mätas och hur skall det mätas? Vilka mätmetoder finns? Vilka verktyg finns, tillförlitlighet. Vad för underlag skall samlas in? Detta var en aspekt av problem. Andra problem är vilka osäkerhetsfaktorer och slutligen så innebar varje databas att sätta sig in i hur den fungerar och hur programmera testmoduler som skulle använda databasservrarna och det på ett så enhetligt o rättvisande som möjligt att villkoren är så lika som möjligt.

2 Bakgrund

Spacemetric ville utröna om det finns andra databaser som kan ge kortare svarstider för deras bildhanteringssystem KeyStone.

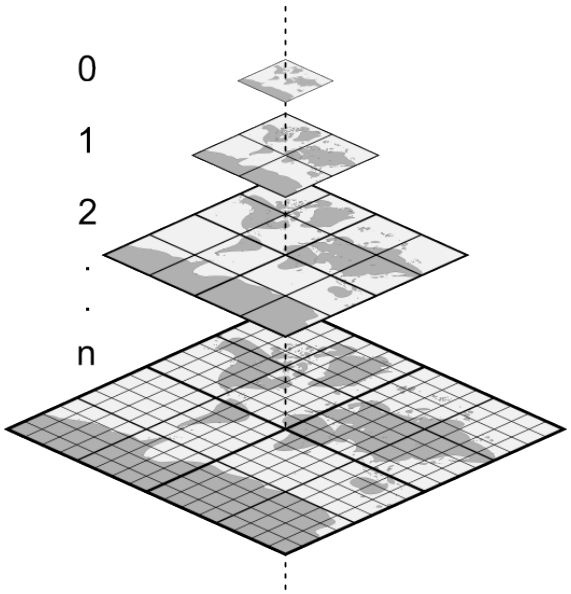
2.1 Bildrutesystemet

KeyStone har en Javabaserad klient baserad på Eclipse som är baserad på öppen källkod. Eclipse var från början ett utvecklingsverktyg för Java men har genom åren fått många andra användningar där man vill ha Javabaserade klientprogram. Eclipse har ett tekniskt ramverk som gör det lätt att modifiera samt skapa olika tillägg för utökad funktionalitet.

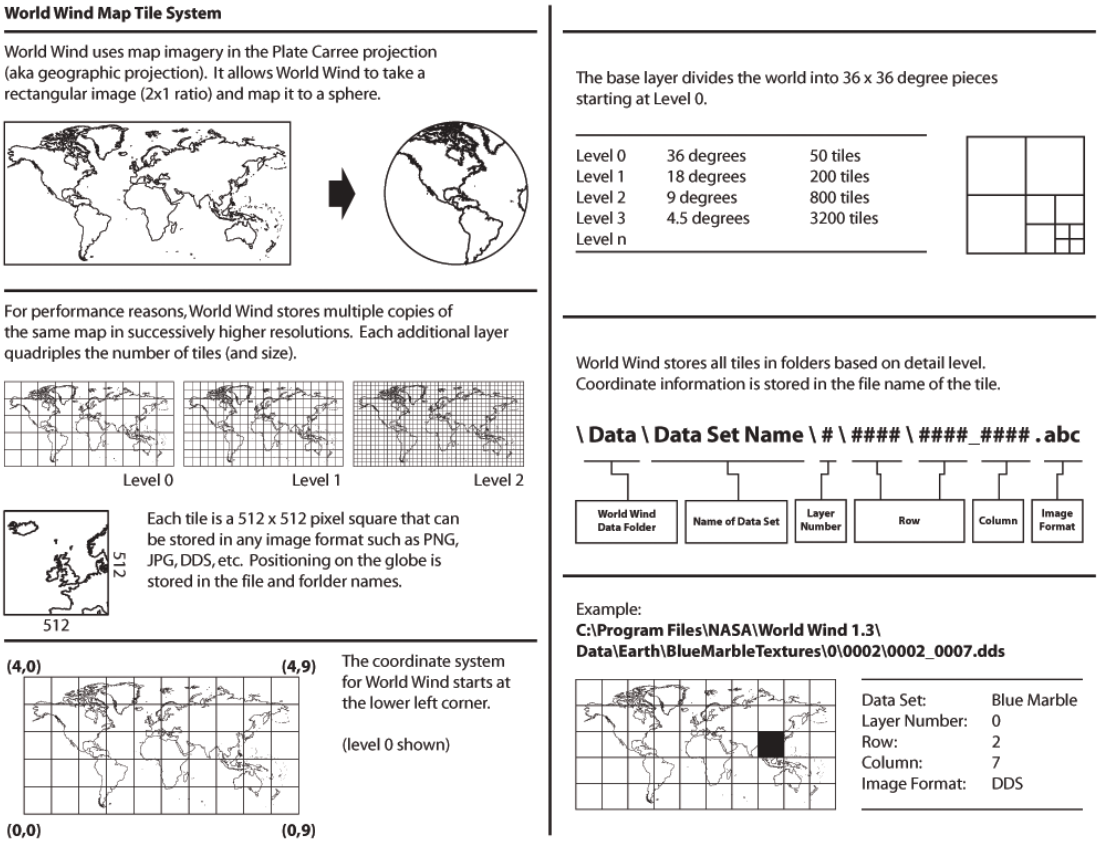
I KeyStone klienten panorerar och zoomar användaren en gränsnittsmodul som heter WorldWind [3] och som är utvecklat av NASA och som också är utvecklad med öppen källkod. I WorldWind så är rutan som är en rasterbaserad bild med en till flera olika bildlager uppdelad i bildrutor (Tiles) där varje bildruta är kvadratisk. När helst användaren panorerar eller zoomar så tar WorldWind via anrop till server eller filsystem fram de nya bildrutor som efterfrågas. Det gör att istället för att ta fram en till flera stora bilder så kan några mindre bilder anropas vilket minskar utnyttjande av nätverkstrafik samt arbetsminne mm. Bildrutorna är större bilder uppdelade på kvadrater med ett bestämt antal pixlar, oftast 512 pixlar/sida. I WorldWind sätts dessa samman till en komplett bild. Varje pixel i varje bild kan kopplas till en geografisk position.

KeyStone server är en Tomcat server som i nuläget hämtar bildrutorna från en MySQL databas.

Bildrutorna är indelade i zoom/detaljnivåer där 0 är den lägsta nivån. I WorldWind så är består första uppsättningen av 5x10 rutor som sedan fördubblas varefter zoom/detaljupplösningsnivån ökar. Det börjar från nedre vänstra hörnet med 0,0.



Källa: http://data.webglearth.com/doc/webgl-earthch1.html



Källa: http://www.worldwindcentral.com/wiki/Tiling_System

2.2 Databashanterare

Samtliga databashanterare som utvärderats i detta examensarbete har sitt ursprung i utveckling med öppen källkod och är kända för att kunna hantera stora mängder data.

2.2.1 MySQL

Databashanterare som använder frågespråket SQL och baseras på öppen källkod. MySQL licensieras under GNU General Public License. Skapad av MySQL AB i Uppsala men ägs och hanteras sedan 2008 av Oracle. MySQL finns som fri programvara och som kommersiellt licensierad produkt.

Finns för flera olika plattformar som Windows, Linux, Unix m.m.

2.2.2 MariaDB

Databashanterare som använder frågespråket SQL och baseras på öppen källkod. MariaDB licensieras under GNU General Public License. MariaDB utvecklas som en fortsättning av MySQL utav de som utvecklade MySQL sedan det såldes till Oracle 2008. MariaDB publicerades första gången 2009. All programkod är fri under GPL eller liknande villkor.

Finns för flera olika plattformar som Windows, Linux, Unix m.m.

2.2.3 PostGreSQL

Databashanterare som använder frågespråket SQL i stor utsträckning och baseras på öppen källkod. PostGreSQL har sitt ursprung i ett system som utvecklades på University of California, Berkeley 1986. Databashanteraren släpptes 1997 genom ett projekt med öppen källkod. PostgreSQL utvecklas av PostgreSQL Global Development Group som består av företag och enskilda som bidrar till utvecklingen.

Finns för flera olika plattformar som Windows, Linux, Unix m.m.

2.2.4 MongoDB

Dokumentorienterad databashanterare som baseras på öppen källkod. Utvecklad från början av företaget 10Gen som kommersiell produkt men släpptes 2009 som öppen källkod.

Finns för flera olika plattformar som Windows, Linux, Unix m.m.

2.3 Testverktyg

2.3.1 JMeter

Apache har utvecklat JMeter [4] som är ett program som kan genomföra flera olika tester bl.a. http-anrop där svarstider mäts och genomflöde mäts och statistiska beräkningar utförs.

2.3.2 Perfmon

En miniserver som fångar upp prestandaförbrukning på datorn. Denna server anropas från Jmeter där man kan läsa av olika mått på prestandaförbrukningen under tiden som testerna utförs. Detta för att kunna läsa av datorns ”hälsostatus” under tiden som testerna körs så att man bättre kan bedöma prestandaåtgång på CPU, RAM och Disk.

2.3.3 Log4j2

För att bedöma hur stor del av körningstiden som utgörs av databashantering krävs att olika tidpunkter fångas upp och beräknas. Detta görs med verktyget log4j2 som skriver till en logg fil. I valda delar av källkoden finns metoder som skriver ner tidpunkter och beräkningar till loggfilen.

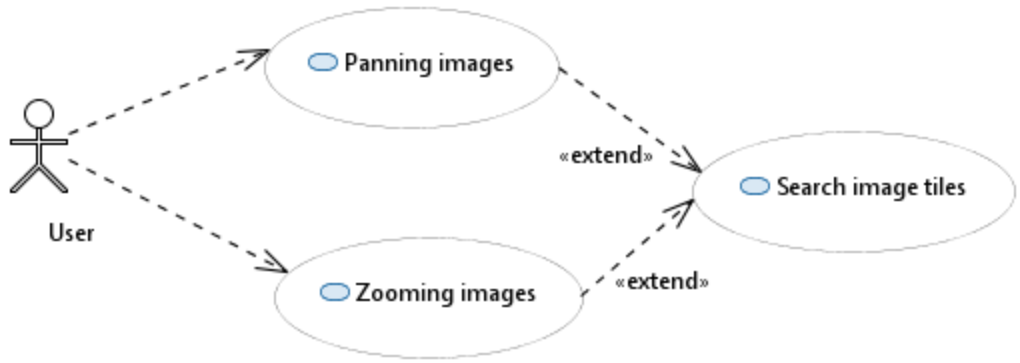
3 Metod

För att jämförelsetesterna skall vara meningsfulla så måste de likna den verklighet som gäller för ett bildhanteringssystem som KeyStone samt att nyckelvärden som är relevanta och som besvarar rätt frågor bestämmas.

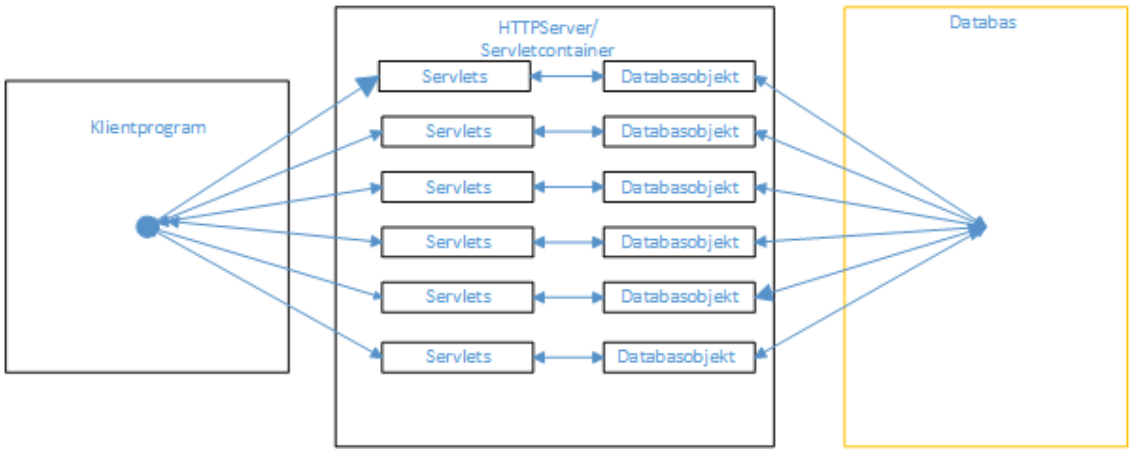
För att simulera detta behövs ett scenario som speglar hur KeyStone med dess klienter används. Scenariot i sin tur utgör grunden för hur testmetoderna skall se ut samt vilka testverktyg som kan användas för att fånga upp rätt information.

3.1 Scenario

En till flera användare använder varsitt klient-program för visning av geospatiala bilder. I det grafiska gränssnittet finns en s.k. ”frame” d.v.s. en ruta som visar en större bild. Denna bild kan vara en karta, satellitfoto/flygfoto eller en jordglob. Dessa i sin tur består av flera mindre kvadratiska bildrutor, s.k. tiles.



Vartefter användare panorerar d.v.s. förflyttar globen/kartan N-S eller Ö-V riktning så tas ny uppsättning bildrutor fram. Likaså om användaren zoomar så tas ny uppsättning bildrutor med för zoom-nivån passande detaljupplösning fram.



För varje bildruta skickas ett http-anrop till en javabaserad http-server. I servern tar en servlet emot anropet och skickar sökfråga till en databas. Finns bilden så returnerar servleten på servern en bildruta till klient-programmet. Om det inte finns en bild med dem parametrarna i databasen returnerar servleten felmeddelande enligt http standard (http 404). Detta innebär att från varje klient-program så kommer flertalet samtidiga http-anrop.

3.2 Nyckelvärden

Hur avgör man om ett system är snabbt eller inte. Grund nyckelvärdet är svarstid som är från att ett http-anrop startas till dess att http-anropet fått tillbaka en respons som kan vara text, fil/bild mm. Vi utgår från den önskade responsen som är en bild från bildlagret som motsvarar sökparametrarna från http-anropet. Problemet är att denna svarstid kan variera stort mellan flera http-anrop. Detta gör det svårt att på några få http-anrop avgöra vilket bildlager som är snabbast. Därför måste flera andra nyckelvärden användas.

Nyckelvärdena är

- Genomsnittlig svarstid: Genomsnittsvärdet av summan av svarstid/antal trådar.
- Total svarstid: Tiden från första http anropet till dess att sista bilden är levererad som respons.
- Genomflöde TPS (Transaktion Per Sekund): Antalet http anrop som får respons. Önskade responsen är bild men kan även förekomma text för felmeddelande.
- Genomflöde KB/S: Mängden KB per sekund som flödar igenom som respons på http-anrop.
- Percentiler: Värde på svarstid som NN % alla anrop är mindre än eller lika med. Med alla databaser kan det hända att några sökningar vid testtillfället tar längre tid än de andra. Genom percentiler vet man i.a. f en svarstid som NN % var mindre än eller lika med.

Vilka nyckelvärden bör man använda? Efter ett flertal mätningar visade det sig att genomflödet i KB/S var den mest användbara.

3.3 Testscenario

De tester som skall utföras skall så mycket som möjligt motsvara scenariot. Det skall vara samma villkor för alla databaser. Det innebär att belastningen i testerna skall vara så lika som möjligt för de olika databaserna. Testerna skall också göras utifrån de förutsättningar som skall gälla för Spacemetrics system.

Varje http-anrop körs i en s.k. tråd i JMeter. En tråd har en början och ett slut. En tråd kan köras enskilt eller samtidigt med andra trådar. I detta examensarbete utförs tester där trådarna körs sekventiellt (en efter en) och där de startar samtidigt. Trådarna har ingen samverkan med andra trådar.

De tre uppsättningarna är

- Kallstart Sekventiell exekvering: Förutom den första posten så har inget anrop sökt någon av de andra posterna. Det innebär att sökresultaten hämtas från filsystemet. Trådar med http anrop som gör sökning mot bildlager startas en efter en när föregående tråd är klar.
- Buffrad Sekventiell exekvering: Alla posterna har varit med i tidigare sökresultat och hämtas oftast från arbetsminnet. Trådar med http anrop som gör sökning mot bildlager startas en efter en när föregående tråd är klar.
- Buffrad Samtidig exekvering: Alla posterna har varit med i tidigare sökresultat och hämtas oftast från arbetsminnet. Trådar med http anrop som gör sökning mot bildlager samtidigt. Varje tråd kör sitt eget http anrop med sökning mot bildlager till dess att bilden är hämtad till den anropade klienten.

3.3.1 Testmiljö

1 bärbar dator

ASUS N53S

Processor: Intel® Core(TM) i7-2670QM CPU @ 2.20 GHz

RAM: 16 GB

Operativsystem: windows 7 Home edition

3.3.2 Testmodul

För varje databas finns en servlet. Servleten använder i sin tur en javaklass för databasåtkomst samt en databasserver med databas med en tabell där testbilderna lagras.

3.4 Osäkerhetsfaktorer

Det visade sig tidigt att kunna göra entydiga mätningar var svårt. Många saker bidrar till att göra mätosäkerheten hög.

Faktorer som bidrar till mätosäkerhet

- Datapool: Javabaserade servrar som Tomcat använder en s.k. datapool där anslutningsdata lagras i en pool så att nästa databasanrop inte skall behöva göra om hela anslutningsprocessen. Första gången en servlet använder en databasanslutning så var svarstiderna oftast mkt längre än de följande vilket påverkar den genomsnittliga svarstiden. Åtgärd: Skapa en första trådgrupp i JMeter med servlet som använde databaskopplingen.
- Övriga processer: Windows egna program och tjänster samt andra program typ Antivirus och andra program använder systemresurser för uppgraderingar, uppdateringar eller andra åtgärder. Det kan innebära att när en databas testas så är åtgången

av systemresurser låg medan vid ett annat tillfälle är åtgången av systemresurser hög vilket innebär mindre av CPU, RAM mm vilket påverkar svarstiderna. Åtgärd: Observera Windows egen systemövervakare som finns i Aktivitetshanteraren flertalet ggr för bilda en ”normalbild”, d.v.s. en situation när åtgång på CPU, RAM och läsning och skrivning till disk var låg. När systemövervakaren visade detta normalläge så sattes testerna igång. Utöver detta användes ett tilläggsprogram till JMeter som övervakar nyttjande av systemresurser under testprojektets körning. På så sätt kan det påvisas att det varit likartade förhållande vid alla testtillfällena.

- Övrig mätosäkerhet: När servlets gör databasanrop första gången så läses data oftast från fil vilket innebär längre svarstider än när datat hämtas från arbetsminnet. Likväl så är mätosäkerheten stor såväl vid de första databassökningarna som vid de senare. Vid varje mättillfälle så var alla databaserna avstängda och vid respektive mätning startades den databas som skulle testas. Dessutom gjordes det flertalet olika trådgrupper för att minimera mätosäkerheten.

Trots dessa åtgärder var det ändå stor variation på testresultaten.

3.5 Uppsättning

3.5.1 Klient-program

Applikation som gör http-anrop till en servlet med parametrar som identifierar bild. Parametrarna är

id: id nummer

x: vågrät position

y: lodrät position

z: zoom-nivå.

3.5.2 HTTP Server med servletcontainer

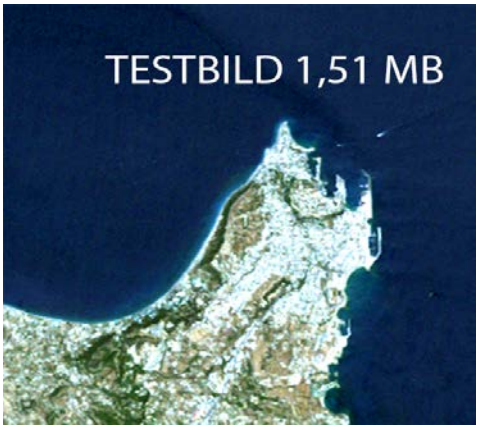
Apache Tomcat är en javabaserad http server som exekverar Servlets och JSP.

3.5.3 Databasserver

SQL server eller icke SQL server som lagrar en större uppsättning bilder. Varje bild har en primärnyckel bestående av id, x, y och z.

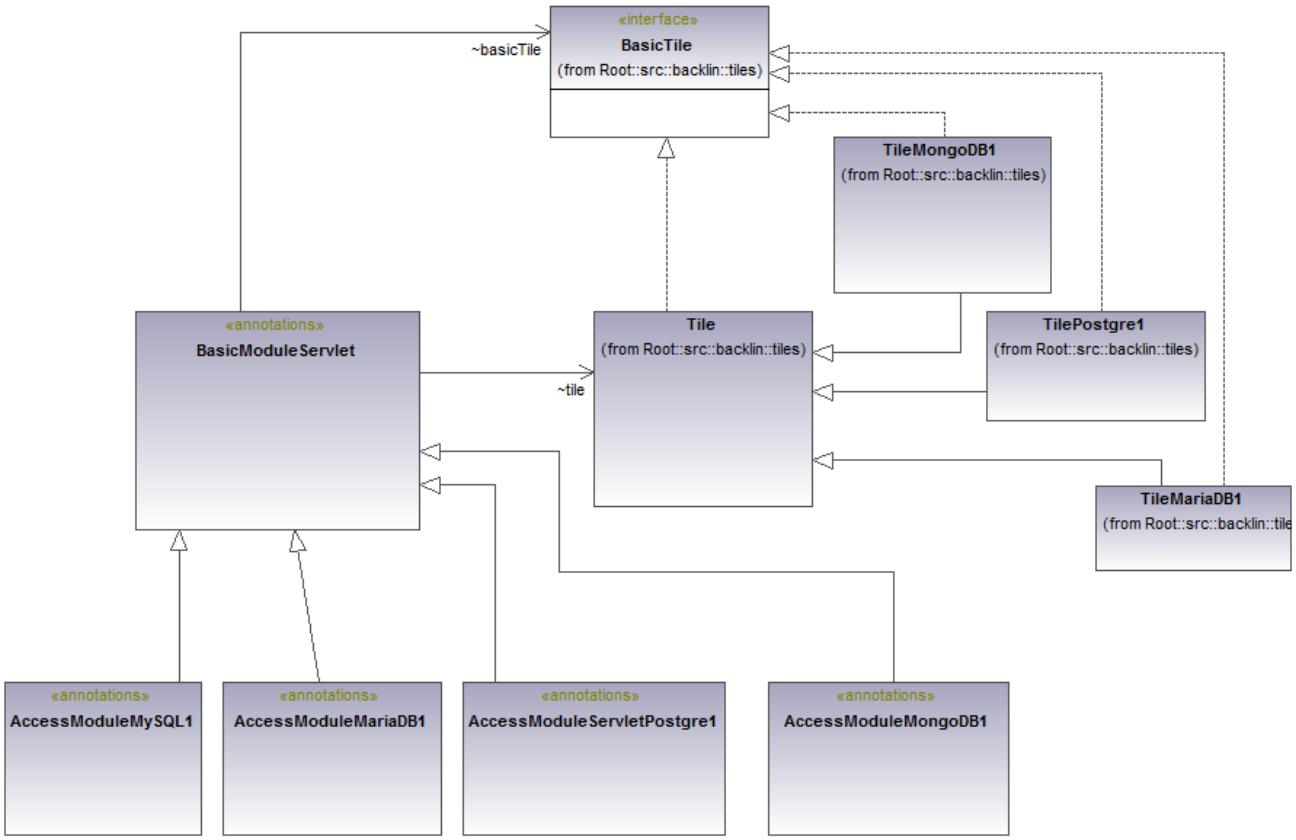
3.5.4 Testbild

För att kunna göra rättvisa jämförelser med de olika databaserna används en och samma testbild. Varje post i databastabellen som är den samma för alla databasservrar har kolumnerna id, x, y, z och bild som är i format på respektive databas för lagring av binär data. Varje databas har samma antal poster. Storleken på bilden är 1,52 MB.



3.6 Klasser

Klassuppsättningen är gjord med tanke på att förutsättningarna mellan de olika servlet-databaserna skall vara så lika som möjligt samt för att nyttja god objektorienterad design och utveckling



BasicModuleServlet

Servlet och basklass som de andra Servletklasserna ärver ifrån. Detta för att skapa så likartade förutsättningar som möjligt.

AccessModuleMySQL1

Ärver av BasicModuleServlet och använder MySQL som databas. Testbilderna lagras som blob i databastabellen.

AccesModuleMariaDB1

Ärver av BasicModuleServlet och använder MariaDB som databas. Testbilderna lagras som blob i databastabellen.

AccessModuleServletPostGree1

Ärver av BasicModuleServlet och använder PostgreSQL som databas. Testbilderna lagras som bytes i databastabellen.

AccessModuleMongoDB1

Ärver av BasicModuleServlet och använder MongoDB som databas.

Samtliga Servlets använder ett Tile objekt som ärver av klassen Tile.

Tile

Basklass för åtkomst och hämtning av bild från databas som andra klasser ärver från. Prefix är Tile<namn>.

BasicTile

Det Java interface som servletklasserna använder för hantering av Tile objekt.

3.7 Mätverktyg

Det behövs mätmetoder och verktyg som fångar upp svarstider, genomflöde samt status på hårdvaran vid mättillfället. Det behövs också fångas upp olika delar av exekveringstiderna för att bedöma hur mkt av svarstiderna som utgörs av sökning mot databasserver.

3.7.1 Eclipse

Eclipse [2] är utvecklingsverktyget användes för programmering av servlets, konfigurationsfiler m.m. men det användes också för att övervaka Tomcat servern. Från Eclipse startades Tomcat servern med servletklasserna vilket medgav att debugtexter som kom från delar av programkoden för Javaklasser och servlets samt systemmeddelande från Tomcat servern kunde övervakas.

3.7.2 Uppsättning och modifiering av Jmeter

En modifiering gjordes i JMeter för att kunna se den Totala Svarstiden. I JMeter sattes ett Testprojekt för varje servlet. Testprojektet hade URL till servletten som kördes. En csv fil med unika värden för parametrarna id, x, y, z värden

I varje Testprojekt ingår s.k. Trådgrupper där man kan ange antal trådar som körs samtidigt eller med fördröjning. Varje trådgrupp gör http-anrop med parametrar från csv filen. Varje rad i csv filen har en unik sammansättning av värden för parametrarna id, x, y och z. Dessa värden användes för sökning av bilder i databasen. Mellan varje trådgrupp sattes det upp en fördröjning

Till slut lägger man till s.k. Lyssnare som samlar in, behandlar och visar olika värden. Med ett musklick kan man på så sätt trigga flertalet olika körningar, samla in mätresultat som man sedan sammanfattar.

3.8 Genomförande

Att bli bekant med JMeter samt hitta moduler och metoder hur genomföra testerna på ett bra sätt som möjliggjorde att de testvärden som genererades tog mycket tid. Att genomföra separata test med de olika aspekter som sekventiell och parallell exekvering visade sig vara omständligt. Istället används en testuppsättning för varje databashanterare som testades.

3.8.1 Testuppsättningen

Innan så kontrollerades det i Windows aktivitetshanterare på den aktuella statusen på åtgång av arbetsminne och processorkraft. Därefter startades Eclipse och den databas som vid tillfället avsåg att testa samt Perfmon. Det finns ett plugin till JMeter som lyssnar av Perfmon. Därefter startades JMeter och Tomcatservern.

I JMetern börjar man med en testplan där man också skapar centrala parametrar mm. I Testplanen skapar man en till flera s.k. trådgrupper.

I trådgruppen anger man antal trådar och hur de skall exekveras. I trådgruppen lägger man sen till en modul för den typ som man skall mäta. I detta fall så är det http modul som anropar en URL till den servlet som skall testas.

I testplanen lägger man också upp olika s.k. lyssnare som fångar upp mätresultat, bearbetar och beräknar dem och visar dem. Dessa går också att spara i csv filer som man sen kan öppna i Excel för vidare bearbetning.

Varje testplan innehöll följande omgångar

1. 1 trådgrupp som gjorde ett HTTP anrop i syfte att initiera datapoolen i Tomcat. Denna räknades sedan bort i testresultaten.
2. 1 trådgrupp om 100 sekventiella trådar som kördes en efter en. Detta i syfte att se om det förekommer skillnader på svarstider då databasen läser data från disk eller när den läser från arbetsminnet.

3. 4 trådgrupper om 100 sekventiella exekveringar där databasen läser från arbetsminnet.
4. 4 trådgrupper med 50 trådar som startar samtidigt.

Mellan varje trådgrupp sattes en s.k. timer med fördröjning för skapa tid mellan körningarna av trådgrupper.

4 Resultat

Detta kapitel sammanfattar resultaten av de mätningar som gjordes enligt den testplan som är beskrivet i föregående kapitel. De resultat som redovisas är genomsnittlig svarstid, transaktioner per sekund (TPS) och genomflöde i KB per sekund (KB/sec). Standardavikelse och genomsnittlig standardavikelse redovisas också. Genomsnittlig standardavvikelse redovisas där flera trådgrupper har körs efter varandra utan stopp.

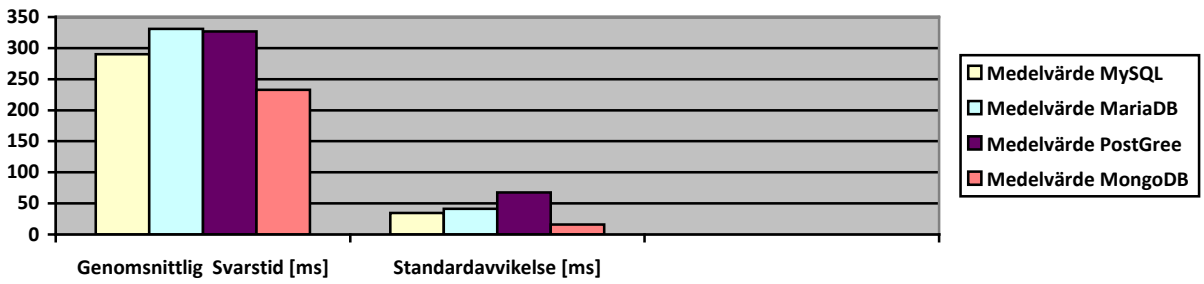
Genomsnittlig standardavvikelse = $\sqrt{(1/N \sum(\text{Standard Avvikelse})^2)}$

4.1 Kallstart sekventiell exekvering

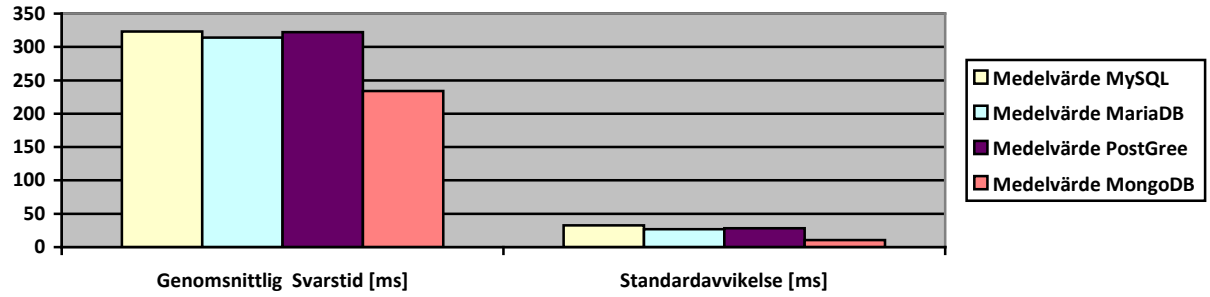
Här presenteras resultaten från omgång 2 i testplanen. I denna omgång exekverade JMeter en trådgrupp om 100 trådar för varje databas där varje tråd läste parametrarna för x, y, z från en textfil och gör ett HTTP anrop till en servlet. Varje rad i csv filen har en unik sammansättning av värden för parametrarna x, y och z. I denna omgång så hämtas data från disk av databaserna i och med att parametervärdena läses för första gången. Databasen har innan den första trådgruppen som exekverade en tråd varit avstängd och således inte haft några databassökningar innan.

MongoDB gav minsta genomsnittliga svarstid, standard avvikelse, och bäst genomflöde i TPS och Genomflöde i KB/sekund. Resultaten för MongoDB stod ut klart från de andra databaserna.

4.1.1 Genomsnittlig svarstid

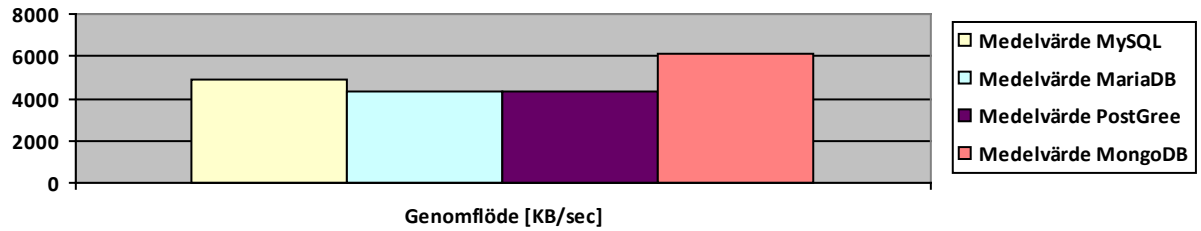


Figur 1:Mättillfälle 1

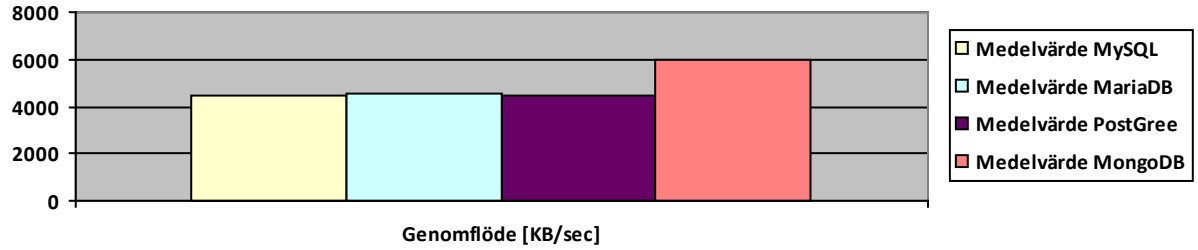


Figur 2: Mättillfälle 2

4.1.2 Genomflöde



Figur 3: Mättillfälle 1

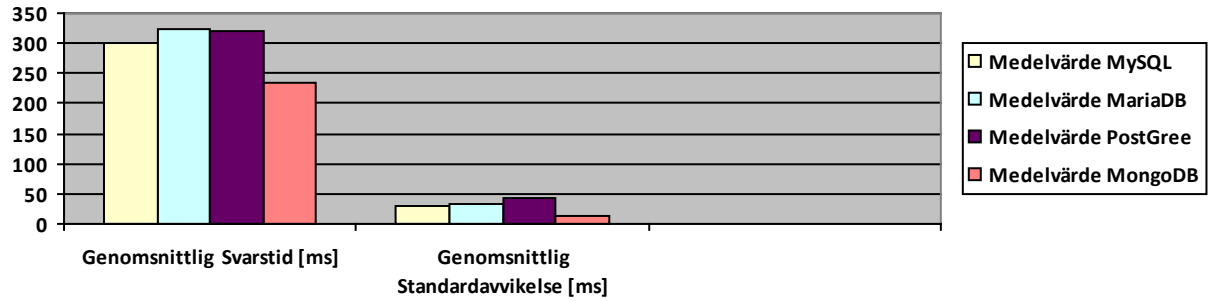


Figur 4: Mättillfälle 2

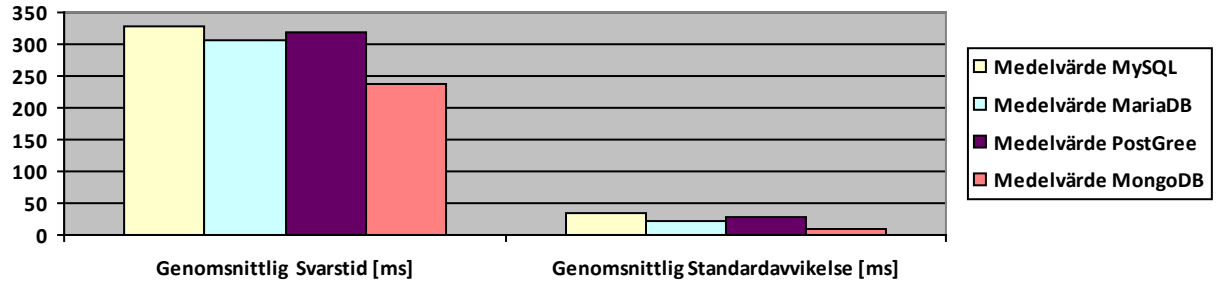
4.2 Buffrad sekventiell exekvering

Här presenteras resultaten från omgång 3 i testplanen. I denna omgång exekverade JMeter 4 stycken trådgrupper med 100 trådar per trådgrupp på samma sätt som omgång 2 i testplanen där varje trådgrupp exekverade trådar sekventiellt, d.v.s. efter varandra. Detta utan att starta om databasen vilket innebar att det data som databasen läste från disk i föregående omgång nu var lagrat i arbetsminnet.

4.2.1 Genomsnittlig svarstid

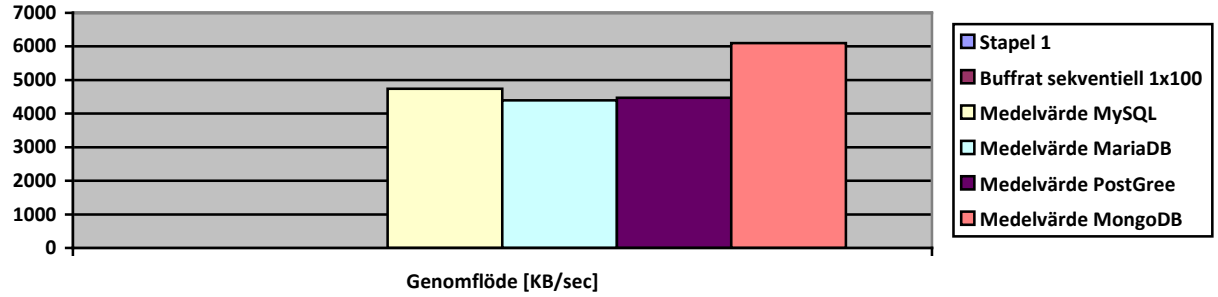


Figur 5 - Mättillfälle 1

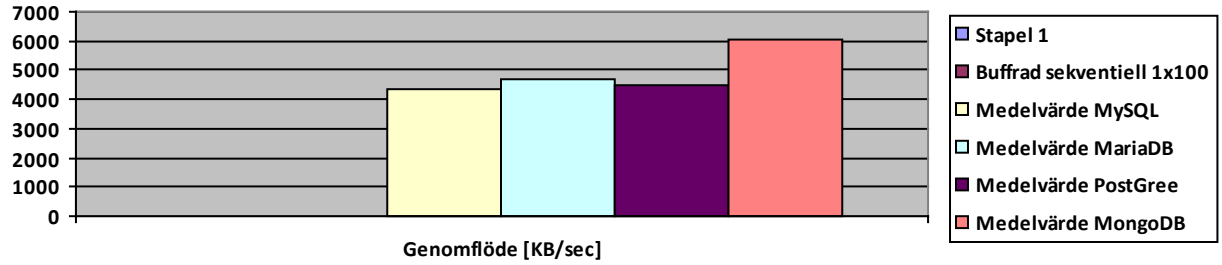


Figur 6 - Mättillfälle 2

4.2.2 Genomflöde



Figur 7 - Mättillfälle 1



Figur 8 - Mättillfälle 2

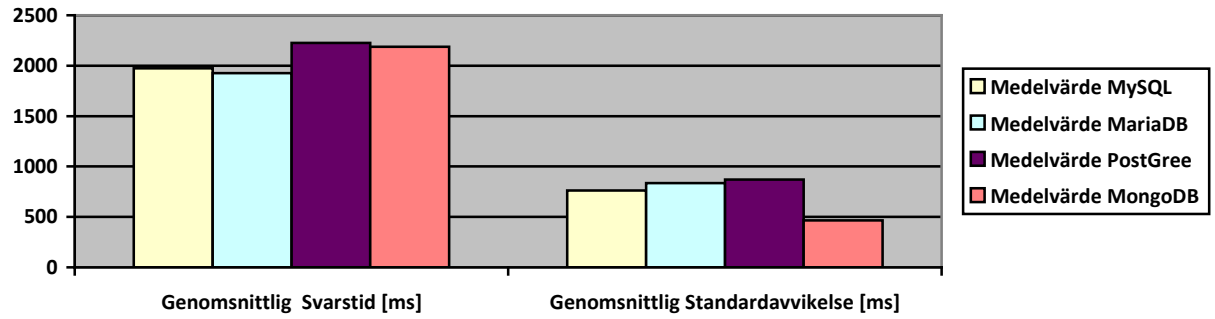
MongoDB gav minsta genomsnittliga svarstid, standard avvikelse och bäst genomflöde i TPS och KB/sekund.

4.3 Buffrad samtidig exekvering

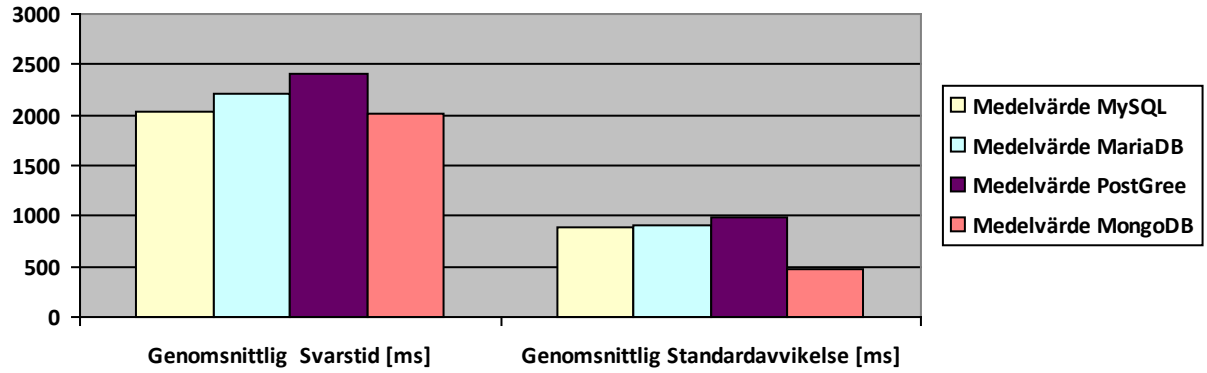
Här presenteras resultaten från den fjärde omgången i testplanen. I denna omgång exekverade JMeter 4 stycken trådgrupper om 50 trådar varje som startade samtidigt. Avsevärd större svarstider men också mycket större genomflöde i KB/sekund. Standardavvikelserna blev på samtliga mycket större än i de föregående omgångarna. Speciellt för SQL databaserna.

MariaDB hade vid första mättillfället minst genomsnittliga svarstid medan MongoDB hade det vid det andra mättillfället. För samtliga SQL databaserna var standardavvikelserna mycket större än för MongoDB. MongoDB framstår som stabilare.

4.3.1 Genomsnittlig svarstid

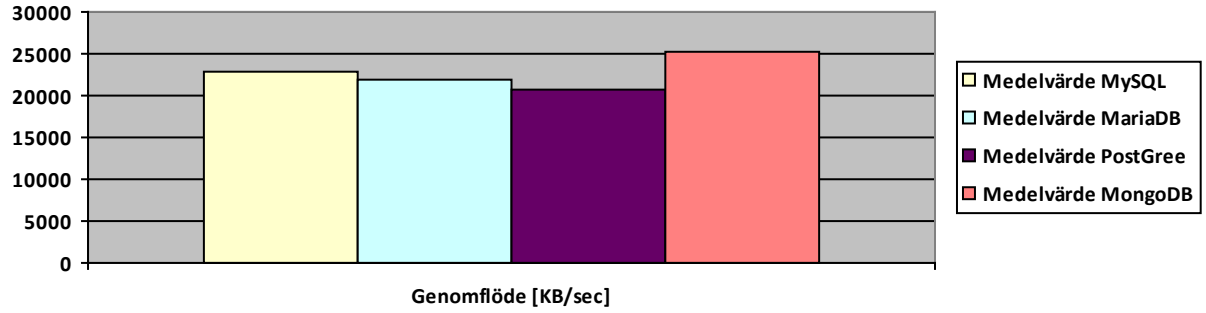


Figur 9 - Mättillfälle 1

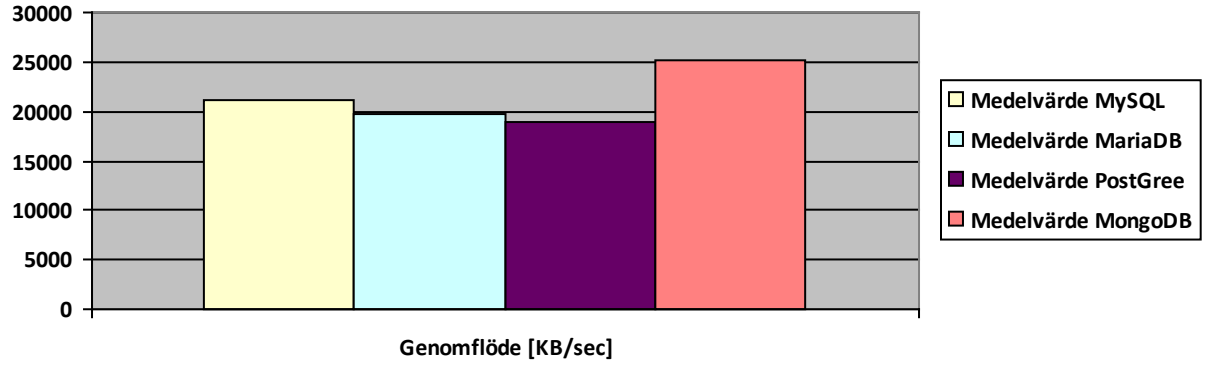


Figur 10 - Mättillfälle 2

4.3.2 Genomflöde



Figur 11 - Mättillfälle 1



Figur 12 - Mättillfälle 2

5 Analys/diskussion

Att genom tester bedöma vilken databas i urvalet som är snabbast är svårt på grund av många faktorer.

Vad är snabbast egentligen? Är det att mäta svarstiden som är grund nyckel värdet? Tyvärr så varierar de uppmätta svarstiderna avsevärt från en mätning till en annan. Ett sätt att parera mätosäkerheten är att göra många mätningar för göra statiska beräkningar som t.ex. den genomsnittliga svarstiden. Säg att man gör 100 http-anrop efter varandra som man mäter svarstiden för och sen beräknar genomsnittsvärdet. Problemet är att även nästföljande mätning på 100 http-anrop kan variera stort. Likaså flera efter varandra eller mätta vid olika dagar visar stor variation, speciellt bland de 3 SQL-databaserna var variationerna stora. Sedan att samla in och bearbeta informationen visade sig vara mycket tidsödande. Efterhand uppkom bra metoder för smidig uppsamling av information men viss mätosäkerhet får man acceptera men genom att kartlägga de faktorer som bidrar till ökad mätosäkerhet så kan också mätosäkerheten göras mindre genom bra testmetoder.

Besvarar mätning av svarstiderna på sekventiella http-anrop Spacemetrics fråga om snabbt bildlager för KeyStone? Det beror på! Framförallt beror det på hur användandet av KeyStone ser ut.

Scenariot är ju att 1 – N användare genom ett klientprogram där WorldWind finns och där i WorldWind panorerar eller zoomar generar flertalet http-anrop som i bildlagret söker fram bilder. Vanligast att anta är att mer än en användare använder Keystone samtidigt. Även med bara en användare så innebär en panorering eller zoomning att flera http-anrop görs samtidigt. Kan man genom mätning av svarstiderna på sekventiella http-anrop avgöra om vilken databas/bildlager i urvalet som kommer att ge kortast svarstider vid flera samtidiga http-anrop?

Var de nyckelvärden som användes tillförlitliga och användbara i arbetet att bedöma vilken databas som var snabbast?

De nyckelvärden som användes var:

- Genomsnittlig svarstid: Genom att flera mätningar används så ökar tillförlitligheten för mätningarna. Dock så kan hög standardavvikelse göra det missvisande då vissa anrop med hög svarstid innebär att vissa användare kommer få vänta länge på sina bilder trots att den genomsnittliga svarstiden var lägre än för andra tester. Mest användbart för mätning av samtidiga anrop.
- Total svarstid: Att mäta tiden från första http-anropen till att sista anropen visar hur snabbt x antal http-anrop hinner bli besvarade. Visade också bättre samband med percentiler och genomflöde än genomsnittlig svarstid. Mest användbart för mätning av samtidiga anrop. Förutsättning är att filerna har samma storlek. Mest användbart för mätning av samtidiga anrop.

- Percentiler: NN % som är mindre än eller lika med ett värde på svarstid. Eftersom det händer att vissa sökningar tar mkt längre tid så visar detta värde att i.a. f NN% var mindre än eller lika med det värdet. Percentiler är bra nyckelvärde för mätning av såväl sekventiella http-anrop som för samtidiga http-anrop. Förutsätter att filerna har samma storlek.
- TPS (Transaktion per sekund): Anger hur många transaktioner per sekund som blir genomförda och i.o.m. att det är samma storlek på alla bilderna så är denna siffra användbar som jämförelsevärde. Mest användbart för mätning av samtidiga anrop. Mest användbart för mätning av samtidiga anrop.
- Genomflöde i KB/S: Kilobyte/sekund: Eftersom det handlar om filer så är detta värde högst användbart som nyckelvärde. Det är användbart också med varierande filformat. Mest användbart för mätning av samtidiga anrop.

För mätning av sekventiella http-anrop med samma filstorlek är genomsnittlig svarstid, Total svarstid och percentiler samt Transaktion Per Sekund användbara.

För mätning av samtidiga http-anrop med samma filstorlek blir svarstiderna generellt mkt högre och likaså mätosäkerheten är Total svarstid, percentiler och genomflöde i Transaktion Per Sekund användbara.

Genomflöde i KB/S är alltid användbart för mätning av sekventiella såväl som samtidiga http-anrop. För de mätningar som gjordes samt för eventuellt kommande mätningar med varierande filstorlek så kommer detta nyckelvärde vara det mest användbara och utslagsgivande.

Vilken av dessa nyckelvärde skall man då använda sig av för bedöma vilken databas som är snabbast? Det är genomflöde KB/S men när samtliga http-anrop förväntas få samma filstorlek kan de andra nyckelvärdena också användas tillsammans som komplement. KB/S fungerar såväl för tester med enhetliga filstorlekar som med varierande filstorlekar. Eftersom det efterfrågades vilken databas som var snabbast för lagring och åtkomst av filer så är KB/s utslagsgivande. *Nyckelvärdet genomflöde KB/s är således det mest relevanta för utse snabbaste bildlagret.*

Var mätningarna tillräckligt utslagsgivande för utse det snabbaste bildlagret? Det fanns en markant skillnad i genomflöde KB/S mellan MongoDB och de tre SQL-databaserna där genomflödet på samtliga tester alltid var signifikant mkt bättre men med SQL databaserna så varierade det mellan vilken som hade bäst genomflöde vid de olika mätningarna och mättillfällena samt att skillnaderna var generellt små. Det går inte att avgöra vilken av SQL databaserna som var snabbast. Möjligen kan man utröna att MySQL och MariaDB konkurrerar om första platsen bland SQL-databaserna. Kan man med fler mättillfällen få till ett avgörande om SQL databaserna. Kanske men det skulle kräva avsevärt många fler mättillfällen med allt vad det innebär med att bearbeta, analysera och sammanställa mätresultaten. Det är knappast görligt för ett företag som letar efter en eventuell ersättare till den databas

som de redan har. Kan man vid de mätningar som gjordes utse en tydlig ”vinnare” så kan man gå vidare i övriga viktiga urval som prestanda och stabilitetstester mm.

Vid andra s.k. ”Benchmarkingtester” [5] så rörde det sig om 1 – 2 mättillfällen med färre mätningar. Svarstiderna för de olika databaserna kan vara olika beroende på om databaserna användes första gången eller flera gånger samt att det kunde vara olika om det var ett http-anrop åt gången eller om det var flera http-anrop samtidigt.

Var filstorleken adekvat? Efter samråd med Spacemetric så var storleken på 1.52 MB adekvat. En liknande jämförelsetest som publicerades 2006 [6] så räknades filstorlekar större än 1 MB som för stora för en SQL-databas men mätningarna visar att de SQL databaserna som användes gott och väl klarar av dem filstorlekarna.

Skillnaderna när http-anropen skedde sekventiellt jämfört när http-anropen skedde samtidigt var mycket stora. Svarstider och standardavvikelser blev avsevärt mycket större för samtidiga http-anrop.

Vad inverkar på svarstiderna?

- Hårdvara: Processorkraft, Arbetsminne, bandbredd på nätverkskort och läshastighet på hårddiskarna. Framförallt läshastigheten på hårddiskarna och nätverkets bandbredd.
- Operativsystem: Hur operativsystemet hanterar hårdvaruresurser men också hur den hanterar trådar och processer.
- http-server: Hur servern hanterar trådar. Vid sekventiella http-anrop stod http-servern för större del av svarstiden. Däremot när det var flera samtidiga anrop så stod databasservrarna generellt för större del av svarstiderna.
- Databasserver: Hur databasservern hanterar trådar, skriver och läser från hårddisk samt hantering av data i arbetsminnet.
- Databasdesign: Användning av index och primärnycklar gör att databasen även när den läser från disk [7] inte behöver läsa igenom hela databasfilen utan läser igenom indexfiler med hjälp av algoritmer. Innan nuvarande databasdesign så var svarstiderna avsevärt mycket högre trots att det var mindre mängd data lagrat i databasen. Framförallt när det gäller SQL databaser så är index [8] en mycket viktig faktor.

Slutliga frågan är detta ett bra sätt att utse den snabbaste bilddatabasen utav ett urval?

- Mätosäkerheterna som förekom trots en rad åtgärder gör skapar viss osäkerhet när skillnaderna inte är tydliga.
- Olika databaser har olika sätt att lagra filer vilket kräver en hel del utvecklingsarbete samt tid att sätta sig in i hur man programmerar mot dessa databaser, förmodligen inte enbart i Java utan även med andra programmeringsspråk.
- Antalet skikt, d.v.s. klient, servlet, databas skapar osäkerhet om vad som påverkar mest i fråga om svarstider.

Vad kan göras annorlunda i liknade projekt?

- Enbart mäta svarstiden för objekt av en Javaklass eller JDBC procedur som hämtar bilder från databas med samma som i detta examensarbete.
- En virtuell maskin eller en testmaskin med ett operativsystem som är avskild från vanligt LAN och Internet och där det enkelt går att styra vilka processer som skall köras samt sätta antalet sådana till ett absolut minimum. Kanske ett speciellt operativ system eller distribution gjord enbart för riktmärkestester.
- Efter att testat med Javaklasser mot databas, komplettera med de testmetoder som tillämpades i detta projekt.
- Önskvärt vore bättre moduler för insamling av testdata från JMeter mm som skapar en komplett rapport, d.v.s. ett rapportverktyg för att spara till csv format och därefter bearbeta alla dessa data var omständligt och tidsödande.

6 Slutsatser

Av de 4 databaserna hade MongoDB de kortaste svarstiderna och bästa genomflödet i Transaktion Per Sekund och KB/Sekund. MongoDB var mycket jämn i prestandan och hade också lägst standardavvikelser och genomsnittlig stardadavvikelse jämfört med de andra SQL databaserna. Vid iakttagelse av PerfMon som monitorerade valda delar av prestandan så var uttaget av arbetsminne större för MongoDB än för SQL databaserna. I övrigt så var skillnaderna i läsning av disk och nyttjande processorkraft inte stora.

Av de 3 vanliga SQL databaserna var MySQL och MariaDB de databaser som hade kortaste svarstider och bästa genomflödena i Transaktion Per Sekund och KB/Sekund. MySQL var dessutom bäst när det gällde att hantera samtidiga sökningar.

Databasindex innebär att skillnaden mellan att databasen hämtar sökt data från hårddisken eller från arbetsminnet blir relativt liten.

Att enkelt och entydigt göra prestandamätningar på databaser som dessutom är inbäddade i andra system såsom servlet-behållare mm är svårt. Mätosäkerheterna är höga p.g.a. flera faktorer. Innan man överväger att byta en befintlig databas så bör man se över andra faktorer såsom hårdvara, konfiguration och databasdesign.

Vid få anrop/sekund så svarade Tomcat servern för större del av svarstiden medan när det var många sökningar/sekund så stod databaserna för den större delen av svarstiden.

7 Referenser

[1] Spacemetric [Online]. http://www.spacemetric.com [2014-05-10]

[2] Eclipse [Online]. http://http://www.eclipse.org[2014-05-14]

[3] Terrain Level of Detail. [Online]. http://www.worldwindcentral.com/wiki/Terrain_Level_of_Detail [2014-06-01]

[4] JMeter [Online]. http://www.jmeter.org[2014-05-14]

[5] A Comparative Benchmark of Large Objects in Relational Databases. Sorin Stancu-Mara, Peter Baumann, Vladislav Marinov School of Engineering and Science Jacobs University [Elektronisk]. http://www.faculty.jacobs-university.de/pbaumann/iu-bremen.de_pbaumann/Papers/blob-report.pdf

[6] To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?. April 2006 . Technical Report MSR-TR-2006-45 [Elektronisk] . http://arxiv.org/ftp/cs/papers/0701/0701168.pdf

[7] Ramez Elmasri, Shamkant B. Navathe: "Fundamentals of Database Systems" i *kapitel 14*, fjärde uplagan, Addison Wesley, 2004. S 455-486.

[8] Ramez Elmasri, Shamkant B. Navathe: "Fundamentals of Database Systems" i *kapitel 16.2*, fjärde uplagan, Addison Wesley, 2004. S 541-547.

A. Mätresultat

	Bästa resultat SQL databas
	Bästa resultat
	Sämsta resultat

Kallstart sekventiell exekvering

Mättillfälle 1						
2015-11-14	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvi-kelse [ms]	TPS	Genomflöde [KB/sec]	Total svarstid [ms]
Kallstart 1x100						
Medelvärde MySQL	1	290	34,81	3,4	4922,07	29179
Medelvärde MariaDB	1	331	41,26	3	4311,52	33311
Medelvärde PostGreSQlL	1	327	67,5	3	4363,13	32917
Medelvärde MongoDB	1	233	16,18	4,2	6098,04	23552

Mättillfälle 2						
2015-12-07	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvikelse [ms]	TPS	Genomflöde [KB/sec]	Total svarstid [ms]
Kallstart 1x100						
Medelvärde MySQL	1	323	32,76	3,1	4422,24	32477
Medelvärde MariaDB	1	314	26,87	3,2	4542,24	31619
Medelvärde PostGreSQL	1	322	28,23	3,1	4437,54	32365
Medelvärde MongoDB	1	234	10,67	4,2	6009,75	23898

Buffrad sekventiell exekvering

4 trådgrupper om 100 trådar i varje som startas sekventiellt.

Mättillfälle 1						
2015-11-14	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvi-kelse [ms]	TPS	Genomflöde [KB/sec]	Total svarstid [ms]
Buffrat sekventiell 1x100						
Medelvärde MySQL	4	301	29,97	3,275	4736,9675	30322,5
Medelvärde MariaDB	4	324	31,96	3,075	4391,6875	32703,75
Medelvärde PostGreSQL	4	319,25	42,78	3,125	4472,63	32167,75
Medelvärde MongoDB	4	233	12,23	4,275	6096,385	23562

Mättillfälle 2						
2015-12-07	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvikelse [ms]	TPS	Genomflöde [KB/sec]	Total svarstid [ms]
Buffrad sekventiell 1x100						
Medelvärde MySQL	4	328,25	35,29	3,05	4347,0175	33 045,25
Medelvärde MariaDB	4	305,75	22,36	3,25	4655,9325	30849,25
Medelvärde PostGreSQL	4	318,25	27,83	3,125	4488,24	32032,75
Medelvärde MongoDB	4	236,25	10,44	4,175	6022,77	23849,75

Buffrad samtidig exekvering

4 trådgrupper om 50 trådar i varje som startas samtidigt.

Mättillfälle 1						
2015-11-14	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvi-kelse [ms]	TPS	Genomflöde [KB/sec]	Total svars-tid [ms]
Buffrat samtidigt 50x1						
Medelvärde MySQL	4	1974,5	762,38	15,925	22843,45	3148
Medelvärde MariaDB	4	1926,25	836,60	15,2	21811,8425	3294,5
Medelvärde PostGreSQL	4	2225,5	872,12	14,425	20718,5425	3470,5
Medelvärde MongoDB	4	2188,75	464,17	17,5	25151,4325	2856,5

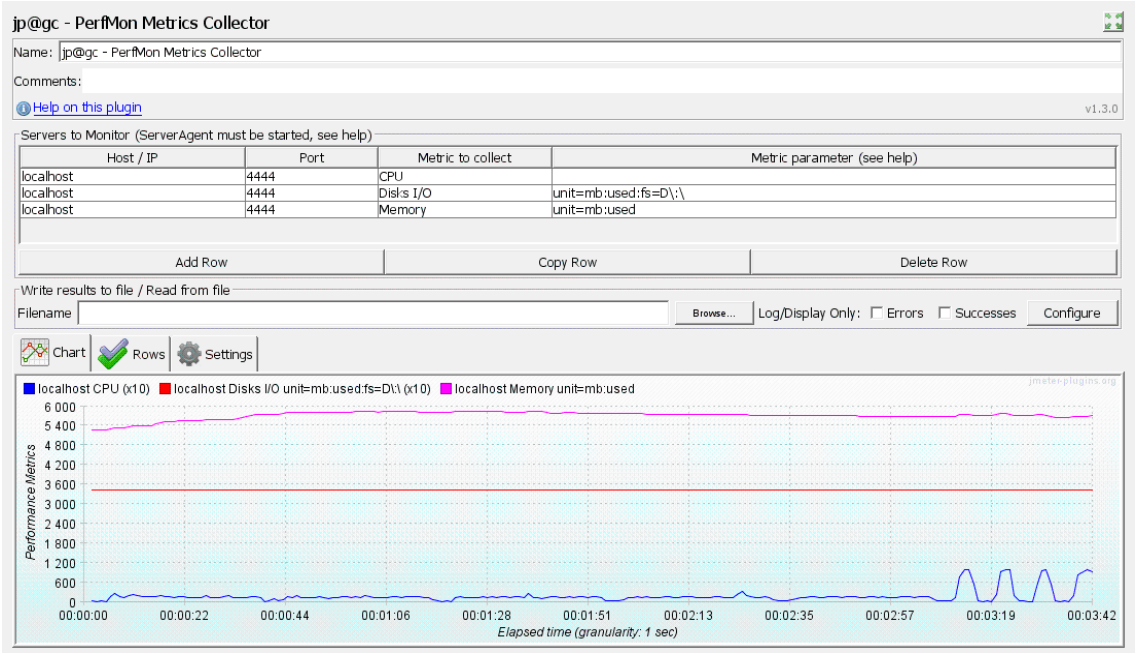
Mättillfälle 2						
2015-12-07	Antal körningar	Genomsnittlig Svarstid [ms]	Genomsnittlig Standardavvikelse [ms]	TPS	Genomflöde [KB/sec]	Total svars-tid [ms]
Buffrad samtidig 50x1						
Medelvärde MySQL	4	2035,5	885,37	14,65	21045,0125	3 414,25
Medelvärde MariaDB	4	2213,5	916,57	13,725	19699,105	3660
Medelvärde PostGreSQL	4	2408	981,85	13,15	18885,635	3 803,25
Medelvärde MongoDB	4	2013,75	479,85	17,6	25247,7525	2 845,5

Resultat från mätning av prestandaåtgång vid det andra mättillfället med PerfMon 2015-12-07

MySQL

MariaDB

PostGreSQL



MongoDB