Computer Based Information System Journal

ISSN (Print): 2337-8794 | E- ISSN : 2621-5292
web jurnal : http://ejournal.upbatam.ac.id/index.php/cbis

PERBANDINGAN PERFORMANSI DATABASE MONGODB DAN MYSQL DALAM APLIKASI FILE MULTIMEDIA BERBASIS WEB

Mesri Silalahi, Didi Wahyudi
Universitas Putera Batam, Indonesia.

INFORMASI ARTIKEL

Diterima Redaksi: 1 Februari 2018
Diterbitkan *Online*: 31 Februari 2018

KATA KUNCI

MongoDB, MySQL, BLOB, Execution Time, Processor, Memory, Virtual Memory

KORESPONDENSI

E-mail: naomi.766hi@gmail.com

A B S T R A C T

Database appeared and began to develop in line with the needs of processing and data storage to meet the information needs. Database is part of an important building block in an information system. In addition to a relational database (SQL), which stores structured datas in tables with defined schemes, there is a non-relational databases (NoSQL) with a dynamic scheme or unstructured. This study will compare the performance between NoSQL database (MongoDB) and SQL database (MySQL) for a web-based multimedia file storage application that stores files as BLOBs. Performance comparison is based on the speed of execution and the computer resources usage (CPU, memory, and virtual memory).

I. Latar Belakang

Basis data merupakan bagian penting dari sistem informasi yaitu sebagai sumber informasi serta sebagai sarana mencapai sistem informasi yang efektif dan efisien [1]. Teknologi basis data yang sangat populer saat ini adalah *Relational Database Management System* (RDBMS) yang berisi data-data yang terstruktur, dimana antara satu tabel dengan tabel yang lainnya terhubung melalui *primary key* [2]. RDBMS ini telah digunakan sejak tahun 1970an dikarenakan RDBMS memenuhi syarat-syarat penting seperti *confidentiality*, *integrity*, dan *availability*. Terlebih lagi transaksi-transaksi di RDBMS bersifat ACID (*Atomic*, *Consistency*, *Isolation*, dan *Durability*) yang menjamin reliabelnya transaksi. Namun RDBMS menghadapi permasalahan performa pada pemrosesan data tidak terstruktur yang bertambah secara eksponensial, seperti dokumen, email, multimedia atau media sosial. Untuk mengatasi permasalahan RDBMS, Carlo Strozzi pada tahun 1998 memperkenalkan istilah "NoSQL" yang mengacu ke *database* non-relasional. Dan pada tahun 2009 Eric Evans memperkenalkannya kembali [3]. Google, Amazon, Facebook dan LinkedIn adalah perusahaan-perusahaan yang pertama kali menemukan keterbatasan teknologi *database*

relasional sejalan dengan kebutuhan aplikasi mereka [4].

Kelebihan dari NoSQL adalah kemampuan untuk menangani data yang tidak terstruktur secara efisien. *Database* non-relasional tidak menggunakan prinsip-prinsip RDBMS dan tidak menyimpan data di dalam tabel, skemanya tidak tetap dan model datanya sangat sederhana. Beberapa struktur data yang digunakan dalam berbagai *database* NoSQL adalah *Key-Value Store*, *Column-Family Store*, *Document Store*, dan *Graph Database*[5].

Penelitian ini membandingkan performaansi *database* NoSQL (yang diwakilkan oleh MongoDB) dan *database* SQL (yang diwakilkan oleh MySQL). MongoDB dan MySQL akan diletakkan di dalam sebuah server Linux dengan konfigurasi minimum, yang dibantu oleh server web Apache dan interpreter PHP. File-file yang tersimpan akan dapat diakses menggunakan aplikasi *Multimedia* berbasis web. Data yang akan disimpan di basis data MongoDB dan MySQL adalah atribut atau *metadata* dari file multimedia yaitu judul, artis, kategori, jenis file, dan ukuran file – dan data *binary* dari file multimedia itu sendiri.

Performa basis data diukur ketika mengakses data *binary* file dalam beberapa kelompok ukuran file. Jadi ketika ada permintaan akan sebuah file multimedia, *script* PHP akan mengambil *metadata* file tersebut dan kemudian mengirimkan data *binary* file yang bersangkutan untuk kemudian dikirimkan ke *browser* pengguna. Seberapa besar data yang diambil dari *database* dan seberapa besar sumber daya server yang terpakai tergantung permintaan pengguna. Contoh website yang menyajikan file-file multimedia di internet adalah youtube.com, midiworld.com, wavsource.com, dramaload.ch, dewamovie.com, dan lain sebagainya. Tujuan dalam penelitian ini sebagai berikut :

1. Merancang aplikasi *Multimedia* berbasis web dengan basis data MongoDB.
2. Merancang aplikasi *Multimedia* berbasis web dengan basis data MySQL.
3. Membandingkan performaansi *database* MongoDB dan MySQL untuk aplikasi *Multimedia* berbasis web.

II. Kajian Literatur

Aplikasi Web

Aplikasi web dibangun menggunakan 3 level arsitektur yang terdiri dari level atau lapisan [6] :

1. *Client* (antar-muka pengguna). Pada lapisan ini, browser digunakan untuk mengakses aplikasi berbasis web dan mempunyai 2 fungsi. Browser juga bertanggung jawab untuk menyediakan form untuk masukan dari pengguna.
2. Server aplikasi. Lapisan ini berisi server web, logika aplikasi, dan koneksi ke basis data.
3. Server basis data. Lapisan ini menggunakan DBMS yang berisi basis data.

Software Development Life Cycle (SDLC)

SDLC [7] sering disebut *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik).

Tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut :

1. Inisiasi (*initiation*), ditandai dengan pembuatan proposal proyek perangkat lunak
2. Pengembangan konsep sistem (*system concept development*) mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

3. Perencanaan (*planning*), mengembangkan rencana manajemen proyek dan dokumen perencanaan lainnya
4. Analisis kebutuhan (*requirements analysis*), menganalisis kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan *user*.
5. Desain (*design*), mentransformasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem focus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.
6. Pengembangan (*development*), Mengkonversi desain ke sistem informasi yang lengkap.
7. Integrasi dan pengujian (*integration and test*), Mendemonstrasikan sistem perangkat lunak dengan diarahkan oleh staf penjamin kualitas (*quality assurance*) dan *user*.
8. Implementasi (*implementation*), Implementasi perangkat lunak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang teridentifikasi dari fase integrasi dan pengujian.
9. Operasi dan pemeliharaan (*operations and maintenance*), mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan produksi, termasuk implementasi akhir dan masuk pada proses peninjauan.
10. Disposisi (*disposition*), mendeskripsikan aktifitas akhir dari pengembangan sistem dan membangun data yang sebenarnya sesuai aktifitas *user*.

Pemrograman Terstruktur

Pemrograman [7] terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer. Fungsi-fungsi dan prosedur-prosedur ditulis secara sekuensial atau terurut dari atas ke bawah sesuai dengan kebergantungan antar fungsi atau prosedur.

Basis Data

Basis data [1] adalah satu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media. Data disimpan dengan cara-cara tertentu sehingga mudah digunakan atau ditampilkan kembali. Data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal serta dapat disimpan tanpa mengalami ketergantungan dengan program yang menggunakannya.

Basis Data SQL (Relasional)

Menurut Bhugul [5], basis data relasional adalah koleksi *data item* yang diorganisasikan sebagai seperangkat tabel yang terdeskripsi secara formal dimana data dapat diakses atau disusun kembali dengan banyak cara tanpa harus mengorganisasikan kembali tabel-tabel basis data. Basis data relasional mengorganisasikan data ke satu atau lebih tabel (atau relasi) yang berisi kolom dan baris, dengan kunci yang unik untuk setiap barisnya. Menurut Aghi[8], relasi bermanfaat untuk menjaga kelompok data sebagai koleksi yang tetap dengan bantuan tabel data yang berisi informasi yang terstruktur, menghubungkan semua masukan dengan cara memberikan nilai ke atribut. Terdapat beberapa DBMS relasional, contohnya Oracle, DB2, Sybase, MySQL, MS.SQL Server dan MS Access.

Basis Data NoSQL (Non-Relasional)

Patel dan Eltaieb dalam penelitiannya [2]tentang *"Big Data"* menyatakan, perusahaan-perusahaan Web 2.0 yang besar telah mengembangkan atau mengadopsi berbagai jenis basis data NoSQL untuk data mereka yang terus bertambah dan untuk kebutuhan infrastruktur mereka, misalnya Google (BigTable), LinkedIn (Voldemort),

Facebook (Cassandra), Amazon (Dynamo), dan sebagainya.

Penggunaan NoSQL memungkinkan para pengembang aplikasi untuk mengembangkan tanpa harus mengkonversi struktur di memori ke struktur relasional. Basis data NoSQL dirancang untuk mengatasi isu *"Big Data"* dengan memanfaatkan mesin-mesin yang terdistribusi. Basis data NoSQL juga sering disebut basis data *cloud* yang dikembangkan untuk mengatasi permasalahan basis data relasional yang mengalami masalah performa dalam pemrosesan data tidak terstruktur yang terus bertambah secara eksponensial, seperti dokumen, *email*, *multimedia* atau media sosial.

Menurut Bhugul [5] NoSQL menyimpan dan mengambil data dalam format-format yang berbeda. Terdapat 4 kategori basis data NoSQL, yaitu :

1. *Key value store*, merupakan penyimpanan data NoSQL yang paling sederhana untuk digunakan dari perspektif API. Penyimpanan data berupa blob yang hanya menyimpan begitu saja tanpa peduli atau perlu mengetahui apa yang disimpan.
2. *Column-family*. Sebuah sistem *sparse matrix* yang menggunakan baris dan kolom sebagai kunci.
3. *Graph databases*. Data disimpan menggunakan struktur grafis dengan *nodes* (entitas), *properties* (informasi tentang entitas) dan garis (hubungan antar entitas).
4. *Document stores*, merupakan mekanisme untuk menyimpan struktur data hierarki langsung ke basis data.

Database Life Cycle (DBLC)

Menurut A.S dan Shalahuddin[7], dalam membuat perencanaan basis data juga memiliki alur hidup atau *Database Life Cycle* (DBLC). Fase-fase DBLC antara lain :

1. Analisis kebutuhan / *requirement analysis*
2. Desain lojik basis data / *logical database design*, pada tahap ini dilakukan pembuatan *Conceptual Data Model* (CDM).
3. Desain fisik basis data / *physical database design*, pada tahap ini dilakukan pembuatan *Physical Data Model* (PDM).
4. Implementasi, pada tahap ini dilakukan pembuatan *Query SQL* dan penerapannya ke DBMS.

MongoDB

Menurut Bhugul [5], MongoDB adalah basis data dokumen yang menyediakan performa tinggi dan ketersediaan tinggi, skalabilitas yang mudah. MongoDB adalah sebuah basis data *open source* yang banyak digunakan untuk menangani data yang besar. MongoDB memberikan performa yang tinggi karena penggunaan *indexing*, *aggregation*, *load balancing*, dan sebagainya. MongoDB[9] dibuat menggunakan bahasa C++ dan dirilis 2009. MongoDB juga merupakan [9] basis data berorientasi dokumen yang terdiri dari kumpulan koleksi dan sebuah koleksi terdiri dari kumpulan dokumen. MongoDB[4] menyimpan dokumen dalam format BSON (bentuk *binary* dari JSON). BSON mendukung tipe data yang berbeda seperti *integer*, *float*, *string*, *Boolean*, *date*, dan sebagainya. Sedangkan Bhugul[5] menyatakan bahwa ide dasarnya adalah untuk mengganti konsep dari "baris" dengan model yang lebih fleksibel, yaitu "dokumen". MongoDB dioptimalkan untuk operasi CRUD. MongoDB mempunyai fitur-fitur sebagai berikut:

1. *Document Oriented*, MongoDB tidak mengambil dan memecah entitas menjadi beberapa struktur relasional, tetapi MongoDB menyimpannya dalam jumlah dokumen yang minimal.
2. *Ad Hoc Queries*, MongoDB mendukung pencarian berdasarkan *field*, *range queries*, dan *regular expression*. Hasil dari *query* dapat berupa *field-field* tertentu dari

dokumen, termasuk penggunaan fungsi-fungsi JavaScript.

3. *Indexing*, Setiap *field* di dokumen MongoDB dapat diberi indeks yang menyediakan efisiensi dalam pencarian data.
4. *Replication*, MongoDB memberikan ketersediaan yang tinggi untuk kumpulan-kumpulan replika. Sebuah replika berisi dua atau lebih salinan data.
5. *Load Balancing*, Skalabilitas MongoDB bersifat horizontal menggunakan *sharding*. Pengguna memilih sebuah kunci *shard*, untuk menentukan bagaimana data dalam sebuah koleksi akan didistribusikan.

MySQL

MySQL adalah [10] RDBMS *open source* dan *multithreaded* yang dibuat oleh Michael "Monty" Widenius pada 1995. Pada tahun 2000, MySQL dirilis dengan lisensi ganda yang mengijinkan publik untuk menggunakannya secara gratis di bawah lisensi GNU GPL (*General Public License*) yang menyebabkan popularitasnya melambung. Perusahaan yang memiliki dan mengembangkan MySQL adalah MySQL AB (AB = *aktiebolag*, istilah Swedia untuk perusahaan saham), yang sekarang menjadi anak perusahaan dari Sun Microsystems. Saat ini MySQL AB memperkirakan ada lebih dari 6 juta instalasi MySQL di seluruh dunia, dan melaporkan rata-rata jumlah unduhan instalasi MySQL dari situsnya dan situs *mirror* sebanyak 50.000 per hari. Keberhasilan MySQL sebagai basis data terkemuka adalah tidak hanya karena harganya, tetapi juga karena kehandalan, kinerja dan fitur-fiturnya.

Banyaknya fitur MySQL membuat database ini tetap menjadi sistem basis data yang hebat. Kecepatan adalah salah satu fiturnya yang menonjol. Dalam perbandingan oleh eWeek pada beberapa basis data (MySQL, Oracle, MS SQL, IBM DB2, dan Sybase ASE), MySQL dan Oracle menunjukkan performa dan skalabilitas terbaik. MySQL mampu menangani puluhan ribu tabel dan miliaran baris data dengan cepat dan lancar.

Mesin penyimpanan, yang menangani *query* dan menghubungkan pernyataan SQL pengguna dengan penyimpanan, adalah perangkat lunak yang sangat penting dalam semua DBMS. MySQL menawarkan beberapa mesin penyimpanan dengan keunggulan yang berbeda. Beberapa diantaranya adalah mesin penyimpanan *transaction-safe* yang memperbolehkan pengembalian data ke keadaan sebelumnya (*rollback*). Selain itu, MySQL mempunyai banyak sekali fungsi-fungsi di dalamnya. MySQL juga sangat terkenal karena kecepatan dan peningkatan kestabilan.

Performa DBMS

Kaladi and Ponnusamy [11] dalam penelitiannya menuliskan bahwa sistem manajemen basis data perlu efisien dalam hal penyimpanan dan kecepatan. Penambahan dan penghapusan data secara dinamis dari basis data merupakan tantangan untuk mempertahankan mekanisme pengambilan data yang efisien. Meskipun kecepatannya terbatas, sistem basis data perlu untuk mencapai kecepatan penuh melalui penyimpanan dan teknik pengambilan yang efisien.

Keandalan, ketersediaan dan toleransi kesalahan merupakan masalah besar bagi sistem basis data. Keandalan suatu sistem umumnya ditingkatkan melalui redundansi. Bisnis modern tidak boleh kehilangan data atau menyajikan data yang salah. Kegiatan bisnis modern sangat bergantung pada data elektronik. Sistem basis data modern perlu dirancang dengan mekanisme keandalan yang tinggi.

Faktor-faktor yang Mempengaruhi Performa

Faktor-faktor yang mempengaruhi performa DBMS adalah:

1. Waktu Respon (*Response Time*). Rentang waktu dari mulai perintah diberikan ke sistem sampai munculnya jawaban.
2. *Throughput*. Kemampuan keseluruhan komputer dalam memproses data, yang merupakan kombinasi kecepatan IO, kecepatan CPU, kemampuan paralel dan efisiensi sistem operasi dan perangkat lunak sistem.
3. Sumber Daya (*Resources*). Perangkat keras dan perangkat lunak, termasuk memori, kecepatan *disk*, *cache controller* dan sebagainya.
4. *Memory*. Jumlah total memori yang dibutuhkan untuk menyelesaikan eksekusi. Nilai ini diambil setelah berakhirnya eksekusi.

File Multimedia

Sebuah file multimedia pada dasarnya adalah sebuah file digital[12], dimana file berisi kode *binary*, yang merupakan bahasa universal dari komputer. Format file adalah aturan yang mengatur bagaimana cara membaca instruksi dan data dalam sebuah file komputer. Tanpa sebuah format spesifik, kode *binary* tidak mempunyai arti. File data multimedia menghadapi dua tantangan utama. Pertama, apakah format file kompatibel antar *platform*. Sebuah format yang tidak kompatibel antar *platform* tidak dapat digunakan di *platform* lainnya. Contohnya, Microsoft mengembangkan format gambar BMP dan Apple mengembangkan format PICT. Kedua format ini tidak kompatibel antar *platform* dan pengembang tidak akan menggunakan gambar PICT untuk aplikasi di komputer Windows. Maka gambar PICT tersebut akan dikonversi ke format Windows atau ke format yang dapat digunakan oleh kedua *platform*, misalnya TIFF.

Masalah kompatibilitas untuk data multimedia yang kedua adalah apakah program aplikasi yang berbeda di suatu *platform* dapat memproses format tersebut. Pengembang multimedia membuat atau menyunting berbagai elemen di aplikasi mereka pada suatu perangkat lunak spesifik, seperti pengolah kata, program grafis, atau aplikasi *video-editing*. Pada beberapa kasus, pengembang akan bekerja menggunakan program terpisah untuk membuat suatu komponen media. Misalnya pengembang akan bekerja dengan gambar menggunakan aplikasi *photo-editing* dan kemudian menambahkan efek khusus menggunakan program lainnya. Berbagai elemen yang sudah dibuat atau diimpor kemudian digabungkan menggunakan perangkat lunak *authoring*. Dalam prosesnya, sangatlah penting untuk memastikan bahwa file-file tersebut kompatibel. Format yang lebih baru dan yang biasa digunakan untuk keperluan tertentu mungkin belum didukung secara luas, masih butuh waktu. Misalnya format PNG, sebagai lawan dari format GIF, untuk didukung oleh kebanyakan program *image-editing*.

Penyimpanan File Multimedia di MongoDB

Penyimpanan file multimedia di MongoDB menggunakan fasilitas GridFS. GridFS adalah spesifikasi untuk menyimpan dan mengambil file yang melebihi batasan ukuran dokumen BSON yaitu 16 MB. GridFS tidak menyimpan file ke dokumen tunggal, tetapi membagi sebuah file menjadi potongan-potongan, atau *chunk*, dan menyimpan setiap *chunk* sebagai dokumen terpisah. Secara default GridFS membatasi ukuran *chunk* sebesar 255 KB. GridFS menggunakan 2 (dua) buah koleksi untuk menyimpan file. Sebuah koleksi untuk menyimpan *chunk*, dan yang lainnya untuk menyimpan *metadata* file. Ketika *query* dilakukan terhadap sebuah file, *driver* atau *client* akan menyusun kembali potongan file. Informasi juga dapat diakses dari bagian tertentu file, yang memungkinkan untuk lompat ke bagian tengah dari file video atau audio.

GridFS berguna tidak hanya untuk menyimpan file yang melebihi 16 MB, tetapi juga menyimpan file apapun tanpa harus memuat keseluruhan file ke memori. Pada beberapa situasi, menyimpan file berukuran besar mungkin

lebih efisien menggunakan basis data MongoDB daripada menggunakan *filesystem*, karena :

1. *Filesystem* membatasi jumlah file di sebuah direktori, sedangkan GridFS dapat digunakan untuk menyimpan file sebanyak-banyaknya.
2. MongoDB dapat mendistribusikan file dan *metadata*-nya secara otomatis ke sejumlah *server*.
3. Akses terhadap informasi dari bagian file yang berukuran besar dapat dilakukan tanpa memuat seluruh file ke memori.

Penyimpanan File Multimedia di MySQL

Penyimpanan file multimedia di MySQL menggunakan pendekatan bahwa file dianggap sebagai data yang sejajar dengan data lainnya, hanya saja tipe datanya berbeda. Di MySQL file multimedia dianggap sebagai objek *binary* dengan tipe BLOB (*Binary Large Object*), yaitu sebuah tipe data yang menyimpan *byte strings* yang tidak mempunyai *character set*. Terdapat 4 jenis BLOB berdasarkan ukuran maksimalnya di MySQL [10], yaitu :

1. TINYBLOB, dengan ukuran maksimum 255 *bytes*.
2. BLOB, dengan ukuran maksimum 65535 *bytes*.
3. MEDIUMBLOB, dengan ukuran maksimum 16777215 *bytes*.
4. LONGBLOB, dengan ukuran maksimum 4GB.

Karena nilai BLOB bisa sangat panjang, maka akan muncul keterbatasan sebagai berikut :

1. Hanya sejumlah max_sort_length bytes dari kolom yang digunakan ketika pengurutan. Nilai defaultnya adalah 1024. Nilai ini dapat diubah ketika *server* dimulai dan saat sedang berjalan
2. Menyertakan kolom BLOB pada *query* yang diproses menggunakan tabel sementara (*temporary*) menyebabkan server menggunakan tabel di *disk* daripada memori karena *memory storage engine* tidak mendukung tipe data tersebut. Penggunaan dari *disk* akan berakibat menurunnya performa, jadi sertakan kolom BLOB hanya saat dibutuhkan. Misalnya, hindari penggunaan SELECT *, yang memilih semua kolom.
3. Ukuran maksimum BLOB ditentukan oleh tipenya, tetapi jumlah terbesar yang sebenarnya dapat ditransmisikan antara *client* dan *server* tergantung dari besarnya memori yang tersedia dan ukuran *buffer* komunikasi. Ukuran *buffer* pesan dapat diubah dengan mengganti nilai dari variabel max_allowed_packet, tapi harus dilakukan baik di *server* maupun di program *client*.

PHP

PHP pertama kali diciptakan oleh seorang pria berkewarganegaraan Denmark yang bernama Rasmus Lerdorf pada tahun 1995[13]. PHP (*Hypertext Preprocessor*) adalah bahasa *scripting* yang *open source*, yang sangat cocok untuk pengembangan *web* dan dapat disisipkan ke dalam HTML.

Contoh :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <title>Example</title>
   </head>
   <body>

      <?php
         echo "Hi, I'm a PHP script!";
      ?>

   </body>
```

</html>

Penggunaan PHP

1. *Server-side scripting*. Dalam hal ini diperlukan 3 hal, yaitu *parser* PHP (CGI atau modul *server*), *server* web dan *browser* web.
2. *Command line scripting*. *Script* PHP dapat dijalankan menggunakan *parser* PHP tanpa *server* atau *browser*. Jenis penggunaan ini cocok untuk *script* yang secara berulang dieksekusi oleh cron (di *nix atau Linux) atau *Task Scheduler* (Windows). *Script* PHP juga dapat digunakan untuk pemrosesan tugas berbasis teks sederhana.
3. Membuat aplikasi *desktop* dengan menggunakan PHP-GTK yang merupakan sebuah ekstensi dari PHP.

PHP dapat berkomunikasi ke layanan lain yang menggunakan protocol seperti LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (di Windows) dan masih banyak lagi. PHP juga dapat membuat *raw sockets* dan berinteraksi menggunakan protocol lainnya. PHP mendukung pertukaran data kompleks WDDX antar seluruh bahasa pemrograman web. Mengenai interkoneksi, PHP mendukung penggunaan objek Java sebagai objek PHP secara transparan. PHP juga mempunyai fitur-fitur pemrosesan teks yang berguna, termasuk PCRE (*Pearl Compatible Regular Expressions*), dan banyak ekstensi dan *tool* untuk menguraikan dan mengakses dokumen XML. PHP menggunakan libxml2 sebagai standar dari ekstensi-ekstensi XML, dan memperluasnya dengan menambahkan dukungan untuk SimpleXML, XMLReader dan XMLWriter. PHP masih mempunyai banyak ekstensi menarik lainnya. Dan ekstensi-ekstensi PECL yang mungkin tidak terdokumentasi, misalnya XDebug.

Keamanan

PHP secara spesifik dirancang lebih aman dari Perl atau C, dan dengan pilihan yang benar saat melakukan *compile*, konfigurasi, dan pembuatan kode, maka kebebasan dan keamanan akan didapatkan. Pendekatan yang sering dilakukan adalah menyeimbangkan resiko dan kegunaan. Internet dipenuhi oleh orang-orang yang mencoba menembus kode, membuat situs menjadi *crash*, mengirimkan *content* yang tidak pantas, dan sebagainya. Sebuah situs, baik kecil maupun besar, akan menjadi target hanya karena dapat diakses secara *online*.

III. Metodologi

Analisis Penggunaan

Pada aplikasi *multimedia file storage* berbasis web, pengguna mengakses file-file multimedia dan admin mengunggah file-file tersebut ke *database*. Pertama kali pengguna membuka halaman depan, pengguna akan melihat daftar kategori file multimedia dan daftar file-file multimedia yang ada. Di halaman depan ini pengguna akan melihat beberapa file yang terakhir diunggah. Pengguna dapat masuk ke kategori untuk melihat daftar file sesuai kategori yang diinginkan, memilih menampilkan jenis file audio atau video saja, memilih file berdasarkan kategorinya, dan memilih file multimedia berdasarkan artisnya. Selain itu pengguna dapat juga melakukan pencarian dengan kata kunci tertentu terhadap judul (atau nama) file multimedia dan nama artis. Setelah menemukan file yang diinginkan, pengguna dapat memainkan file multimedia tersebut. Karena penelitian ini membahas tentang perbandingan performa antara dua jenis basis data yang berbeda, maka pengguna dapat memilih untuk memainkan file baik dari MongoDB maupun MySQL jika file tersebut tersedia. Dari sisi pengguna, perbedaan perangkat lunak *server* basis data untuk penyimpanan file multimedia tidak berpengaruh terhadap proses memainkan file *multimedia* yang dilakukan.

Untuk menampilkan sebuah file multimedia ke aplikasi berbasis web ini, terlebih dahulu admin harus mengunggah file tersebut. Pertama, admin membuka halaman yang berisi

form unggah file. Kemudian admin akan memilih sebuah file multimedia, mengisi judul atau nama file tersebut, memilih jenis file multimedia (musik atau video), kemudian menekan tombol unggah.

Spesifikasi Kebutuhan Perangkat Lunak

Dari analisis penggunaan di atas, maka dibutuhkan sebuah aplikasi yang sesuai dengan harapan. Aplikasi *Multimedia File Storage* berbasis web menggunakan perangkat lunak yang sama dengan aplikasi web lainnya. Hanya saja terdapat perbedaan basis data yang digunakan, dimana pada aplikasi web yang ditempatkan di *server hosting* pada umumnya basis data yang dipakai biasanya hanya MySQL, sedangkan penelitian ini juga menggunakan sebuah basis data lain selain MySQL yaitu MongoDB.

Kebutuhan Perangkat Lunak Server

Perangkat lunak *server* yang dibutuhkan untuk keperluan aplikasi ini adalah sebuah *web server* dengan *interpreter* PHP, serta perangkat lunak basis data MySQL dan MongoDB. Perangkat-perangkat lunak tersebut harus dijalankan pada sistem operasi yang mampu menjalankannya seperti Windows atau Linux. Semua perangkat lunak tersebut juga sebaiknya menggunakan versi baru karena versi baru berisi perbaikan dari versi lama, dan juga adanya fitur-fitur tambahan. Misalnya untuk basis data MySQL, terdapat tambahan *engine* InnoDB. Sebelumnya, *engine* yang biasa digunakan adalah MyISAM. Untuk aplikasi *Multimedia File Storage* ini, sangatlah dimungkinkan adanya pembacaan secara bersamaan terhadap sebuah *chunk* (yang disimpan dalam sebuah *record*) file multimedia. Pada *engine* MyISAM, jika terjadi hal seperti itu, maka *record* yang sedang dibaca akan dikunci secara eksklusif (*exclusive lock*) sehingga tidak dimungkinkan sesi pembacaan yang lain pada waktu bersamaan, yang pada akhirnya *server* akan gagal mengirim *chunk* yang dibutuhkan. Masalah ini sudah diatasi di *engine* InnoDB yang ditambahkan sebagai *plugin* di MySQL 5.1 dan menjadi *default engine* di MySQL 5.5. Selain itu, InnoDB juga menawarkan reliabilitas yang lebih baik dan kemungkinan kerusakan data juga berkurang.

Sedangkan untuk *interpreter* PHP di *server*, penulis menggunakan ekstensi MySQLi yang baru hadir di PHP versi 5.0. Selain itu penulis juga banyak menggunakan definisi *array* dalam bentuk singkat (*short syntax*) yang cukup menggunakan tanda [dan] yang sebelumnya *array* harus didefinisikan dalam bentuk array(). Fitur ini baru ada di PHP 5.4. Aplikasi *Multimedia File Storage* berbasis web ini juga menggunakan basis data MongoDB melalui ekstensi mongo di PHP. Ekstensi php-mongo ini dapat ditambahkan pada PHP versi 5.3 ke atas.

Kebutuhan Perangkat Keras Server

Kebutuhan perangkat keras *server* juga harus diperhatikan. Aplikasi *Multimedia File Storage* dan aplikasi-aplikasi sejenis yang mengirimkan data yang banyak ke pengguna, setiap potongan data (*chunk*) yang akan dikirim (*stream*) dibaca dari penyimpanan dan kemudian disimpan sementara di memori dan baru akan dihapus dari memori setelah terkirim dan kemudian aplikasi akan melakukan hal yang sama terhadap potongan data berikutnya, demikian seterusnya. Dalam penelitian ini, besarnya *chunk* yang dipakai dapat diubah (besar *chunk* standar yang dipakai MongoDB adalah 255 KB atau 261120 *bytes*), jadi *server* harus mengalokasikan minimal memori sebesar *chunk* untuk seorang pengguna. Banyaknya jumlah akses terhadap file multimedia dari aplikasi ini akan meningkatkan kebutuhan memori secara signifikan.

Dengan besarnya data yang ditransmisikan, maka otomatis membutuhkan komputer *server* yang mampu memprosesnya. Kebutuhan perangkat keras harus disesuaikan dengan tingkat pemrosesan yang dilakukan. Di internet sering dijumpai website yang tidak bisa

diakses sementara waktu karena website-website tersebut ditaruh di *shared server* (sebuah *server* yang berisi banyak website) sehingga alokasi CPU menjadi kecil untuk setiap website. Ketika terjadi banyak akses terhadap sebuah website di *shared server*, batasan alokasi penggunaan CPU terhadap website tersebut akan terpakai semuanya sehingga website tidak bisa diakses sementara waktu (*suspended*). Pada penelitian ini konsumsi CPU menjadi sangat tinggi ketika admin mengunggah sebuah file dan kemudian aplikasi akan memasukkan *chunk* demi *chunk* file tersebut ke basis data. Jika file multimedia yang diunggah sangat besar, maka keadaan tersebut akan berlangsung lama dan akan mengganggu akses terhadap *server*.

Kebutuhan Perangkat Lunak dan Perangkat Keras Pengguna

Di sisi *client*, pengguna harus menggunakan *browser* yang kompatibel dengan HTML 5 – misalnya Google Chrome atau Mozilla Firefox terbaru – agar dapat memainkan file multimedia langsung dari *browser* tanpa membutuhkan perangkat lunak tambahan. Kompatibilitas antar *browser* terhadap HTML 5 dengan dukungan terhadap jenis-jenis file multimedia tentu berbeda-beda. Masalah kompatibilitas *browser* masih menjadi masalah besar bagi pengembang website karena masih banyak pengguna internet yang tidak memperbarui perangkat lunak di komputernya, termasuk *browser* yang digunakan.

Inkonsistensi tampilan website (*browser quirks*) adalah masalah pertama yang muncul. Dalam pembuatan website, setidaknya ada 3 bahasa program yang digunakan yaitu HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheet*), dan JavaScript. Kompatibilitas ketiga bahasa tersebut berlainan pada berbagai *browser*, apalagi versi-versi dari ketiganya terus meningkat seiring dengan perbaikan dan fitur-fitur baru yang ditambahkan.

Contoh inkonsistensi tampilan di penelitian ini adalah ketika menggunakan *browser* lama yaitu Internet Explorer 8 di sistem operasi Microsoft Windows XP. Fitur CSS *border-radius* atau *rounded corner* dimana pojok halaman seharusnya melengkung, di *browser* IE 8 tetap tampil kotak. Kemudian ketika pengguna menekan tombol untuk memainkan file multimedia, IE 8 akan menampilkan dialog untuk mengunduh file tersebut karena tidak mampu memainkan file langsung dari *browser*.

Struktur Data

Perancangan struktur data dalam penelitian ini dibagi menjadi dua bagian karena penelitian ini menggunakan dua buah basis data yang akan dibandingkan, yaitu MySQL dan MongoDB. Basis data MySQL menggunakan struktur data relasional dan basis data MongoDB menggunakan struktur data dokumen.

Aplikasi ini membutuhkan penyimpanan untuk 3 hal di bawah ini :

1. Penyimpanan keterangan file atau *meta data* dari file di tabel file di MySQL sebagai pelengkap tampilan di aplikasi.
2. Penyimpanan data *binary* file multimedia di tabel file_data MySQL. Tabel ini adalah kumpulan *record* yang berisi *chunk* data *binary* sebesar 255 KB (dapat diubah). Jadi sebuah file berukuran 13.81 MB akan disimpan dalam 56 *record* di tabel ini jika menggunakan *chunk* sebesar 255 KB.
3. Penyimpanan data *binary* file multimedia di GridFS MongoDB. GridFS menggunakan 2 buah koleksi yaitu koleksi files yang berisi *metadata* dari file multimedia dan koleksi chunks yang berisi data *binary*. Di koleksi files, meta data yang disimpan hanya nama file. Nama file ini harus sama dengan nama file di tabel file MySQL. Jika tidak sama, maka aplikasi akan menganggap file tersebut tidak ada di MongoDB.

Struktur Data GridFS MongoDB

GridFS menggunakan struktur data standar dengan menggunakan 2 buah koleksi, yaitu koleksi files dan koleksi chunks dengan struktur sebagai berikut :

Tabel 1. Koleksi files di GridFS MongoDB

Atribut	Tipe	Keterangan
_id	<ObjectId>	Identitas unik dokumen.
length	<num>	Ukuran dokumen dalam *bytes*.
chunkSize	<num>	Ukuran *chunk* dalam *bytes* (ukuran default adalah 255 KB).
uploadDate	<timestamp>	Tanggal saat pertama dokumen disimpan di GridFS.
md5	<hash>	*Hash* MD5 file.
filename	<string>	Nama file.
contentType	<string>	Tipe MIME.
aliases	<string array>	Array dari alias.
metadata	<dataObject>	Informasi tambahan yang ingin disimpan.

Tabel 2. Koleksi chunks di GridFS MongoDB

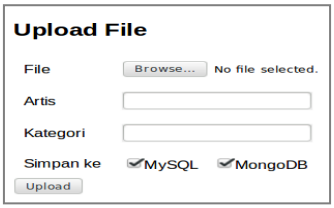
Atribut	Tipe	Keterangan
`_id`	`<Object_id>`	`ObjectId` unik *chunk*.
`files_id`	`<Object_id>`	`_id` dari dokumen induk yang ada di koleksi `files`.
`n`	`<num>`	Nomor urut *chunk*.
`data`	`<binary>`	Data *binary*.

IV. Pembahasan

Aplikasi File Multimedia berbasis Web

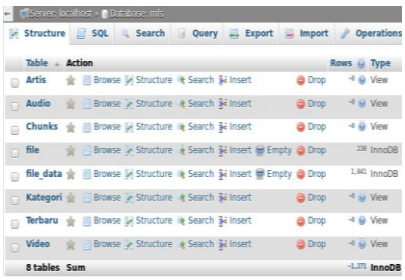
Gambar 1. Halaman Play

Form Upload.



Gambar 2. Form Upload

Basis Data Aplikasi File Multimedia Berbasis Web



Gambar3. Daftar Tabel dan View MySQL di phpMyAdmin

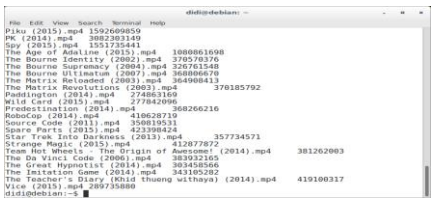
Data yang sudah masuk ke basis data MySQL dapat juga dilihat melalui phpMyAdmin yang sangat memudahkan untuk

mengadministrasi basis data MySQL menggunakan *web browser*

id	jenis	nama	ukuran	tanggal	artis	kategori
1	audio/mpeg	Ada Band - 1001 Cara.mp3	4443290	2016-02-05 17:35:50	Ada Band	Musik
2	audio/mpeg	Ada Band - Bakinya.mp3	2921563	2016-02-05 17:37:13	Ada Band	Musik
3	audio/mpeg	Ada Band - Haruskah Ku Mati.mp3	561173	2016-02-05 17:43:22	Ada Band	Musik
4	audio/mpeg	Ada Band - Jalan cahaya.mp3	4613015	2016-02-05 17:43:46	Ada Band	Musik
5	audio/mpeg	Ada Band - Karena Wanita Ingin Dimengerti.mp3	4323429	2016-02-05 17:44:22	Ada Band	Musik
6	audio/mpeg	Ada Band - Langit 7 Bidadari.mp3	3921259	2016-02-05 17:44:59	Ada Band	Musik
7	audio/mpeg	Ada Band - Manusia Bodoh.mp3	3983655	2016-02-05 17:45:26	Ada Band	Musik
8	audio/mpeg	Ada Band - Masih.mp3	4196639	2016-02-05 17:45:51	Ada Band	Musik
9	audio/mpeg	Ada Band - Nyawa Hidupku.mp3	3913728	2016-02-05 17:46:17	Ada Band	Musik
10	audio/mpeg	Ada Band - Pemain Cinta.mp3	3887254	2016-02-05 17:47:59	Ada Band	Musik
11	audio/mpeg	Ada Band - Pemujamu (CD RIP).mp3	4426220	2016-02-05 17:48:35	Ada Band	Musik
12	audio/mpeg	Ada Band - Senandung Lagu Cinta.mp3	2033920	2016-02-05 17:51:37	Ada Band	Musik
13	audio/mpeg	Ada Band - Setengah Hati.mp3	4248939	2016-02-05 17:51:56	Ada Band	Musik
14	audio/mpeg	Ada Band - Suara Kecewa.mp3	4248372	2016-02-05 17:52:55	Ada Band	Musik
15	audio/mpeg	Ada Band - Surga Cinta.mp3	4335616	2016-02-05 17:53:26	Ada Band	Musik
16	audio/mpeg	Ada Band - Tak Bisa Lagi Menyayangimu.mp3	1852112	2016-02-05 17:54:11	Ada Band	Musik
17	audio/mpeg	Andra And The Back Bone - Main Hati.MP3	3859809	2016-02-05 17:55:14	Andra & The Back Bone	Musik
18	audio/mpeg	Ari Lasso - Cinta Sejati.mp3	4612368	2016-02-05 17:56:48	Ari Lasso	Musik

Gambar 4. Beberapa Contoh Data di Tabel File MySQL

Sedangkan untuk basis data MongoDB, file-file multimedia yang sudah masuk dapat dilihat melalui perintah mongofiles yang biasa digunakan untuk administrasi file GridFS MongoDB. Mongofiles adalah salah satu *tools* yang didapatkan setelah melakukan instalasi MongoDB.



Gambar 5. Data yang tersimpan di GridFS MongoDB

Pengujian Basis Data

Operasi Uji

Jenis operasi uji yang dilakukan adalah *Create (Insert)*, *Read*, dan *Delete*. Operasi *Update* tidak dilakukan karena data yang tersimpan dalam basis data aplikasi ini adalah file *binary* yang tidak mungkin diubah isinya dengan menggunakan perintah standard PHP. Operasi uji yang dilakukan:

1. Operasi *Create*, dalam perintah SQL disebut juga *Insert*. Operasi *Insert* dilakukan untuk mengunggah file ke basis data. Ketika sebuah file diunggah, maka aplikasi ini akan memecah file tersebut sesuai besarnya *chunk*. Setelah dipecah maka setiap pecahan file akan dijadikan sebuah record tersendiri di basis data.
2. Operasi *Read*, dilakukan dengan cara mengunduh banyak file secara bersamaan untuk mensimulasikan penggunaan aplikasi secara bersamaan.
3. Operasi *Delete*, operasi ini digunakan untuk menghapus setiap *chunk* dari file multimedia berukuran besar yang tersimpan di basis data. Semakin besar ukuran file, maka, penghapusan data akan semakin lambat karena banyaknya *chunk*.

Bahan Uji

Bahan uji dalam penelitian ini adalah file-file yang dikelompokkan menurut ukurannya. Setiap kelompok file berisi 20 buah file dengan ukuran yang berbeda. Dari setiap kelompok akan diambil rata-rata nilai hasil ujinya. Kelompok bahan uji tersebut adalah :

1. Ukuran kecil : 0-5 MB
2. Ukuran Sedang : 6 – 100 MB
3. Ukurang Besar : > 100 MB

Variabel Pengukuran

Variable yang diukur dalam penelitian ini adalah Penggunaan *Processor*, Penggunaan Memori Fisik, Penggunaan Memori Virtual, dan Kecepatan Eksekusi. Tetapi khusus untuk operasi uji *Read*, variable Kecepatan Eksekusi tidak dihitung karena kecepatan unduh dapat saja berbeda tergantung jenis dan kondisi jaringan. Misalnya uji *Read* dilakukan melalui komputer lain di jaringan lokal, maka waktu unduhnya jauh lebih cepat apabila melalui internet. Dan ketika pengunduhan dilakukan melalui komputer lain di internet, kecepatan unduh dipengaruhi oleh perbedaan kecepatan *upload* dari koneksi

internet server aplikasi *Multimedia File Storage* dan kecepatan *download* dari koneksi internet komputer yang sedang mengunduh.

Metode Pengukuran

Pengukuran variabel uji penggunaan *processor*, penggunaan memori, penggunaan memori virtual dilakukan menggunakan perintah dstat di *console* linux. Sedangkan kecepatan eksekusi didapatkan dari halaman konfirmasi setelah operasi uji dilakukan. Dstat adalah perangkat lunak untuk memonitor sistem yang dapat diunduh di https://github.com/dagwieers/dstat, yang memberikan semua penggunaan sumber daya sistem secara instan. Hasil keluaran dstat dapat dipilih sesuai keperluan. Misalnya dalam penelitian ini, dstat diatur agar hanya menampilkan penggunaan *processor*, memori, dan memori virtual saja.

Hasil keluaran dstat dapat disimpan ke dalam file teks. Selain itu dapat juga disimpan ke dalam format CSV sehingga dapat diimpor ke dalam Microsoft Excel agar mudah dalam penghitungan data dan pembuatan grafik. Format perintah yang digunakan dalam penelitian ini adalah dstat -tcms --output test.csv untuk menyimpan ke dalam format CSV atau dstat -tcms >> test.txt untuk menyimpan ke dalam format teks. Perintah dengan format di atas dijalankan tepat sebelum operasi uji dilakukan dan dihentikan (dengan menekan tombol Ctrl+C) setelah operasi uji selesai. Di bawah ini adalah gambar contoh hasil perintah di atas :

```
didi@debian:~$ dstat -tcms
----system---- ----total-cpu-usage---- ------memory-usage----- ----swap---
     time     |usr sys idl wai hiq siq| used  buff  cach  free| used  free
12-02 05:36:15|  6   4  89   2   0   0|2342M 34.7M 1428M 37.6M| 139M   14G
12-02 05:36:16|  4   2  93   1   0   0|2342M 34.7M 1428M 37.3M| 139M   14G
12-02 05:36:17|  3   3  93   2   0   0|2343M 34.7M 1428M 36.9M| 139M   14G
12-02 05:36:18|  2   2  96   0   0   0|2342M 34.7M 1428M 37.9M| 139M   14G
12-02 05:36:19|  3   2  94   0   0   0|2342M 34.7M 1428M 37.8M| 139M   14G
12-02 05:36:20|  5   5  90   1   0   0|2342M 34.7M 1427M 37.8M| 139M   14G
12-02 05:36:21|  5   2  92   1   0   0|2342M 34.7M 1427M 37.9M| 139M   14G
12-02 05:36:22|  3   3  92   2   0   0|2342M 34.7M 1427M 38.2M| 139M   14G
12-02 05:36:23|  2   3  95   0   0   0|2341M 34.7M 1427M 38.9M| 139M   14G
12-02 05:36:24|  2   2  96   0   0   0|2341M 34.7M 1427M 39.2M| 139M   14G
12-02 05:36:25|  6   3  89   2   0   0|2342M 34.7M 1427M 38.4M| 139M   14G
12-02 05:36:26|  5   3  92   1   0   0|2341M 34.7M 1427M 39.1M| 139M   14G
```

Gambar 6. Hasil Pengukuran

Dari hasil pengukuran di atas, penggunaan *processor* dapat diambil dari usr pada total-cpu-usage. Kemudian penggunaan memori dapat diambil dari used pada memory-usage. Dan yang terakhir adalah penggunaan memori virtual atau *swap* yang dapat dilihat di used pada bagian swap.

Sedangkan cara mengambil nilai hasil pengukuran adalah sebagai berikut :

1. Penggunaan *processor* diambil dari nilai tertinggi dikurangi nilai terendah. Contoh (dari gambar di atas) :

   nilai tertinggi = 6

   nilai terendah = 2

   maka penggunaan *processor*-nya adalah 6 – 2 = 4

2. Penggunaan memori diambil dari nilai tertinggi dikurangi nilai terendah.

   Contoh (dari gambar di atas) :

   nilai tertinggi = 2343M

   nilai terendah = 2341M

   maka penggunaan memorinya adalah 2343M – 2341M = 2M

3. Penggunaan memori virtual diambil dari nilai tertinggi selama pengukuran.

   Dari gambar di atas : 139M

Hasil Uji Basis Data MongoDB dan MySQL

Operasi Create / Insert

Tabel 3. Hasil Operasi Insert File Berukuran Kecil

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	25	3	48	10	0	0	0.118708	0.75613
2	29	11	20	1	0.042969	0	0.144371	0.026276
3	20	6	21	3	0.105469	0	0.144525	0.028425
4	20	8	9	5	0.109375	0	0.144572	0.036418
5	26	4	11	0	0.09375	0	0.155935	0.034271
6	24	28	15	15	0.152344	0	0.21341	0.039987
7	27	22	22	26	0.132813	0	0.199548	0.04518
8	23	23	23	24	0.082031	0	0.210201	0.044526
9	22	21	13	18	0.070313	0	0.211344	0.056054
10	20	28	22	18	0.097656	0	0.244483	0.048732
11	21	23	11	35	0.066406	0	0.245326	0.064342
12	33	26	33	16	0.066406	0	0.246099	0.065561
13	26	18	14	14	0.066406	0	0.244592	0.057876
14	26	16	21	27	0.0625	0	0.311087	0.076811
15	28	30	14	34	0.21875	0	0.302706	0.072206
16	12	28	15	14	0.078125	0	2.139735	0.07961
17	15	21	24	18	0.089844	0	0.378649	0.078886
18	41	33	17	22	0.101563	0	0.34413	0.09618
19	13	30	15	4	0.140625	0	0.346228	0.089322
20	26	34	54	30	0.203125	0	0.584878	0.106226
	23.85	20.65	21.1	16.7	0.099023	0	0.346526	0.095151

Dari tabel di atas terlihat bahwa pemakaian *Processor* (CPU) dan memori untuk MongoDB lebih sedikit daripada MySQL. Dan MongoDB juga rata-rata 3,6 kali lebih cepat daripada MySQL dalam penyimpanan file berukuran kecil. Jadi dapat disimpulkan bahwa MongoDB lebih cepat dan lebih irit dalam hal pemakaian sumber daya komputer daripada MySQL untuk operasi *INSERT* atau *CREATE* data *binary*.

Tabel 4. Hasil Operasi Insert File Berukuran Sedang

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	7	7	86	19	1165	409	3.476743	1.2549
2	15	8	21	2	0	408	0.877605	0.218272
3	16	7	36	18	0	408	0.921814	0.331327
4	14	11	39	4	0.128906	388	1.165534	0.232279
5	15	9	21	6	0.089844	392	0.943982	0.313019
6	12	7	70	26	0.085938	392	1.544793	0.201332
7	10	5	59	357	0.085938	390	4.357289	0.870136
8	16	10	48	12	0.085938	389	1.17692	0.377571
9	14	16	39	13	0.085938	388	1.265624	0.213503
10	20	11	52	14	0.085938	388	1.244093	0.248795
11	16	14	42	11	0.085938	371	1.477015	0.316518
12	16	11	68	24	7.136719	374	8.557555	0.719889
13	16	8	85	12	8.214844	391	4.214284	1.465814
14	21	7	75	19	9.253906	392	4.387183	1.045734
15	11	19	87	21	14	394	9.4808	2.274673
16	15	17	70	10	17	396	4.526862	1.300158
17	14	7	77	20	23	403	7.510885	2.544604
18	15	11	82	19	28	412	9.601188	1.819042
19	15	50	88	41	36	419	13.03361	3.572581
20	19	21	91	18	37	426	6.498529	3.571212
	14.85	12.9	61.8	33.3	67.26699	396.5	4.313115	1.154578

Dari tabel di atas terlihat bahwa MongoDB masih lebih unggul dalam pemakaian sumber daya komputer seperti *processor* dan memori, dan rata-rata masih lebih cepat 3,7 kali daripada MySQL untuk keperluan menyimpan file berukuran sedang. Tetapi di tabel tersebut juga terlihat bahwa MongoDB lebih mengandalkan memori virtual (*swap*) untuk keperluan ini.

Ketika menyimpan file-file berukuran besar, penggunaan *processor* dan konsumsi memori tidak berbeda jauh antara MongoDB dan MySQL. Hanya saja pemakaian memori virtual MongoDB jauh lebih tinggi daripada MySQL. Dari segi kecepatan, rata-rata MongoDB lebih cepat 2,48 kali daripada MySQL.

Tabel 5. Hasil Operasi Insert File Berukuran Besar

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	14	11	88	14	50	439	17.91756	5.01898
2	18	9	72	93	64	452	17.34601	4.245268
3	24	15	65	16	74	466	20.55279	4.325304
4	15	17	80	103	87	510	25.22348	11.2805
5	16	17	98	137	98	547	26.06525	15.73952
6	17	18	84	24	115	583	31.30751	15.98525
7	15	17	79	42	127	613	31.27036	15.26582
8	18	24	104	83	144	696	44.5641	17.58041
9	16	21	100	64	162	758	44.56931	19.44328
10	15	18	93	44	180	783	45.88321	18.86455
11	14	24	96	89	198	856	48.96113	19.61754
12	17	23	97	66	220	905	54.25631	23.89993
13	16	25	80	76	239	965	59.43757	24.60933
14	14	17	84	50	271	1022	75.80439	29.99122
15	14	22	92	84	290	1083	65.23303	25.8181
16	15	20	107	63	321	1126	68.19556	24.78371
17	15	22	87	71	350	1100	70.33137	25.64794
18	32	24	74	59	372	1141	74.38825	32.09079
19	18	24	95	39	398	1178	76.27536	28.38252
20	15	24	87	391	439	1207	77.78863	30.12082
	16.9	19.6	88.1	80.4	209.95	821.5	48.79356	19.63554

Operasi Read

Pada pengujian operasi *Read*, penulis menggunakan komputer terpisah dan secara bergantian mengunduh semua file berukuran kecil, sedang dan besar (yang khusus digunakan untuk percobaan) dari aplikasi *Multimedia File Storage* ini menggunakan *download manager*. Pemakaian *processor* di *server* aplikasi tidak berbeda jauh untuk file berukuran kecil dan sedang. Tetapi ketika mengunduh file berukuran besar, pemakaian *processor* MySQL melonjak tinggi. Konsumsi memori MySQL untuk file berukuran sedang lebih sedikit daripada MongoDB. Dan ketika melayani file berukuran sedang dan besar, penggunaan memori virtual MySQL lebih besar.

Tabel 6. Hasil Operasi Read

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
K	4	3	225	140	0	0
S	6	6	385	749	337	0
B	26	5	522	82	331	0

Operasi Delete

Tabel 7. Hasil Operasi DELETE Data Ukuran Kecil

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	2	6	0	5	299	299	0.395922	1.959466
2	10	9	1	5	299	298	0.177619	0.228072
3	12	8	2	1	299	298	0.165392	0.1567
4	10	14	4	2	299	298	0.181857	0.141448
5	7	8	6	6	299	298	0.203979	0.137044
	8.2	9	2.6	3.8	299	298.2	0.224954	0.524546

Tabel 8. Hasil Operasi DELETE Data Ukuran Sedang

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	7	13	3	5	298	298	0.454813	0.388715
2	8	9	1	0	298	298	0.387118	0.249506
3	6	8	1	3	298	298	0.432026	0.277997
4	9	10	0	2	298	298	0.434069	0.382032
5	8	9	0	1	298	298	0.372628	0.300188
	7.6	9.8	1	2.2	298	298	0.416131	0.319688

Tabel 9. Hasil Operasi DELETE Data Ukuran Besar

Data	Processor (usr)		Memory (MB) (used)		Swap (MB) (used)		Time (s)	
	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB	MySQL	MongoDB
1	9	11	10	7	298	307	9.195436	2.902609
2	12	10	12	3	300	307	10.82128	1.960059
3	12	11	15	3	302	307	13.45522	2.063635
4	12	8	11	5	302	307	21.54709	2.888499
5	10	11	20	6	307	307	20.92196	3.177332
	11	10.2	13.6	4.8	301.8	307	15.1882	2.598427

Dari percobaan operasi *DELETE* sebanyak 5 kali untuk setiap jenis ukuran file di atas, pemakaian sumber daya komputer berimbang antara MySQL dan MongoDB. Kecepatan juga tidak begitu terasa berbeda (walaupun untuk file berukuran kecil MySQL lebih unggul). Kecepatan terasa berbeda ketika menghapus file berukuran besar.

Sehubungan dengan operasi DELETE, di luar pengujian penulis juga menemukan masalah ketika akan menghapus data *binary* suatu file yang tersebar dalam beberapa *chunk* di MySQL, karena penulis menggunakan kunci pencarian menggunakan atribut yang tidak diindeks. Ini hanya terjadi pada tabel yang mempunyai atribut BLOB.

V. Kesimpulan

1. Aplikasi *File Multimedia* berbasis web yang dirancang menggunakan PHP dengan basis data MongoDB dapat berjalan dengan efektif dan efisien.
2. Aplikasi *File Multimedia* berbasis web yang dirancang menggunakan PHP dengan basis data basis data MySQL dapat berjalan dengan efektif dan efisien.
3. Aplikasi *File Multimedia* berbasis web menggunakan PHP, dengan basis data MongoDB lebih cepat dalam operasi *CREATE* untuk semua ukuran file dan lebih cepat dalam operasi *DELETE* untuk file berukuran besar, serta lebih irit dalam penggunaan sumber daya komputer *server* untuk operasi *READ*.

Ucapan Terima Kasih

Dengan terselesaikannya Karya Ilmiah ini, penulis mengucapkan terimakasih yang sedalam-dalamnya kepada semua pihak yang sudah membantu dalam penyelesaian penelitian ini baik dalam bentuk moril maupun materil

Daftar Pustaka

[1] E. Sutanta, *Basis Data dalam Tinjauan Konseptual*. Yogyakarta: Andi, 2011.

[2] dan T. E. Patel, Tejal, “Relational Database vs NoSQL,” *J. Multidiscip. Eng. Sci. Technol. (University Bridg.*, vol. 2, no. 4, pp. 691–695, 2015.

[3] S. George, “NOSQL - NOTONLY SQL,” *Int. J. Enterp. Comput. Bus. Syst.*, vol. 2, no. 2, pp. 1–11, 2013.

[4] B. P. Pore, Supriya S dan Swalaya, “Comparative Study of SQL & NoSQL Databases,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 4, no. 5, pp. 1747–1753, 2015.

[5] A. M. Bhugul, “Comparative Study of SQL & NOSQL Databases,” *Int. J. Sci. Res. Dev.*, vol. 3, no. 2, pp. 1496–1498, 2015.

[6] dan G. D. Ward, Patricia, “Database Management Systems,” *Walaa Bakry dan Alan Murphy. London: Thomson Learning*, p. 266, 2006.

[7] dan M. S. A.S, Rosa, *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Informatika, 2015.

[8] dan N. B. Aghi, Rajat, Sumeet Mehta, Rahul Chauhan, Siddhant Chaudhary, “A Comprehensive Comparison of SQL and MongoDB Databases,” *Int. J. Sci. Res. Publ.*, vol. 5, no. 2, pp. 1–3, 2015.

[9] dan R. A. Punia, Yogesh, “Implementing

Information System Using MongoDB and Redis," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 3, no. 2, pp. 16–20, 2014.

[10] R. J. . Dyer, *MySQL in a Nutshell*, Second Edi. Andy Oram. Sebastopol: O'Reilly, 2008.

[11] dan P. P. Kaladi, Aparna, "Performance Evaluation of Database Management Systems by The Analysis of DBMS Time and Capacity," *Int. J. Mod. Eng. Res.*, vol. 2, no. 2, pp. 67–72, 2012.

[12] dan K. E. V. Savage, T M, *An Introduction to Digital Multimedia*, 2nd ed. Manchester: Jones & Bartlett Learning, 2014.

[13] dan K. N. Adelheid, Andrea, *Buku Pintar Menguasai PHP MySQL*. Jakarta Selatan: Mediakita, 2012.