DESAIN AWAL SPESIFIKASI PERANGKAT KERAS DAN JARINGAN *HIGH PERFORMANCE COMPUTING* UNTUK MENDUKUNG *OPEN GOVERNMENT* DI INDONESIA

PRELIMINARY DESIGN HARDWARE AND NETWORK SPESIFICATION ON HIGH PERFORMANCE COMPUTING TO SUPPORT OPEN GOVERNMENT IN INDONESIA

Anton Siswo R.A[1] dan **Nia Maulidia**[2]

[1]Jurusan Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom,
Jl. Telekomunikasi 01, Bandung Jawa Barat 40287 – Indonesia;

[2]Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta
Jl. Prof. Dr. G.A Siwabessy, Depok, Jawa Barat 16242, Indonesia
e-mail : raharjo@telkomuniversity.ac.id[1], e-mail : raharjotelu@gmail.com[1], *e-mail* :niamaulidia1@gmail.com[2]

Naskah diterima tanggal 24 Maret 2015, direvisi tanggal 27 Mei 2015, disetujui tanggal 10 Juni 2015

Abstract

The need for transparency of data for a variety of public access, as well as guarantee the legal basis of the disclosure made significantly Open Government should be implemented in Indonesia. To support the Open Government thing to consider is the stability of access by the user, while the stability of access is influenced by two main issues, namely the specification of hardware (servers) and network.To optimize the server, we conducted a study of the technique In Memory Computing which is one technology that is rapidly growing in the era of Big Data Processing are included in clumps of High Performance Computing. Such technology is needed in areas that require very high computational intensity, because it has the ability to overcome the bottleneck that occurs in the system, especially on the permanent storage media such as Hard Disk Drive (HDD) and Solid State Disk (SSD). In this scientific study, carried out the design of hardware that can be applied to minimize the bottleneck that occurs and maximize the use of In Memory Computing technology, including the design of the specification as a central processing server to the network design specification required, so that can be easily done escalation when needed.

Keywords*: Open Government, In Memory Computing, Big Processing Data, Preliminary Design, Storage Devices*

Abstrak

Kebutuhan keterbukaan data untuk berbagai akses publik, serta jaminan dasar hukum tentang keterbukaan informasi menjadikan *Open Government* secara *significant* harus segera diimplementasikan di Indonesia. Untuk mendukung *Open Government* hal yang perlu dipertimbangkan adalah kestabilan akses yang dilakukan oleh pengguna, sedangkan kestabilan akses dipengaruhi oleh dua hal utama, yaitu spesifikasi perangkat keras (*server*) dan jaringan. Untuk optimalisasi pada server, kami melakukan kajian tentang teknik *In Memory Computing* yang merupakan salah satu teknologi yang sedang berkembang pesat dalam era *Big Processing Data*. Teknologi tersebut sangat dibutuhkan pada area-area yang membutuhkan intensitas komputasi sangat tinggi, karena memiliki kemampuan untuk mengatasi *bottleneck* yang terjadi di dalam sistem, terutama pada bagian media penyimpanan permanen seperti *Hard Disk Drive* (*HDD)* dan *Solid State Disk* (*SSD)*. Pada kajian ini, dilakukan desain perangkat keras dan spesifikasi jaringan yang dapat diterapkan untuk mendukung *High Performance Computing* yang dapat mendukung *Open Government* di Indonesia dan termasuk di dalamnya adalah desain dari spesifikasi server sebagai pusat pemrosesan sampai dengan desain dari spesifikasi jaringan yang dibutuhkan, agar dapat dengan mudah dilakukan eskalasi ketika dibutuhkan.

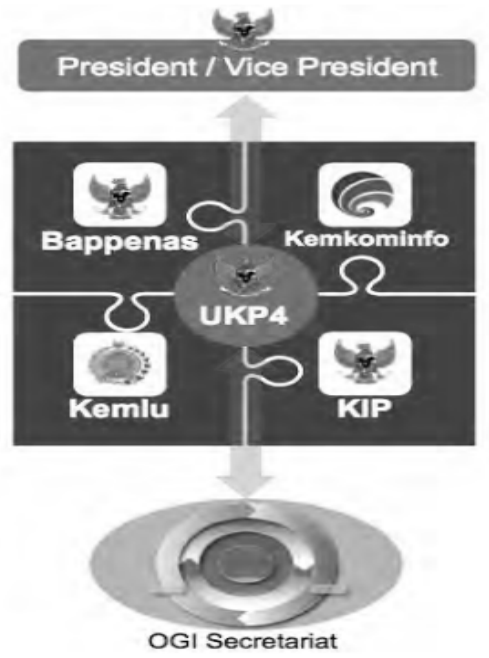
Kata kunci: *Open Government*, *In Memory Computing*, *Big Processing Data*, Desain Awal, Media Penyimpanan

PENDAHULUAN

Kebutuhan akan transparansi data atau yang biasa disebut dengan *Open Government Data* di Indonesia merupakan hal yang sangat mendesak untuk segera diimplementasikan, hal ini dapat dilihat dari ketidakpercayaan rakyat dengan sistem pemerintahan yang sedang berjalan. Semakin tinggi rasa ketidakpercayaan dan ketidaknyamanan masyarakat dengan sistem pemerintahan dikhawatirkan Indonesia akan menjadi negara yang mudah untuk di pecah-belah, dan keutuhan NKRI akan terguncang dengan isu–isu yang tidak bisa dipertanggungjawabkan kebenarannya. Oleh karena itu, pemerintah mempunyai peran yang penting untuk segera mewujudkan *Open Government* yang sebenarnya sudah diatur dalam UU No. 14 Tahun 2008 tentang Keterbukaan Informasi Publik, maka pemerintah wajib melakukan *Open Data* agar dapat terjadi perluasan peran dan partisipasi publik diperlukan dalam pembangunan untuk membuka akses agar publik ikut mengawasi pembangunan digunakan untuk kepentingan masyarakat (OGI 2012).

Open Government sudah diterapkan oleh negara maju salah satunya, yaitu Inggris yang menerapkan transparansi terhadap ke mana uang pajak rakyat yang telah dibelanjakan oleh pemerintah (OGP UK 2013), dan di Denmark yang menerapkan sistem keterbukaan terhadap keterbukaan parlemen dan proses pembuatan undang–undang. Akibat permasalahan sosial yang kompleks dan tingginya tingkat korupsi. Pemerintah Indonesia sudah mulai membentuk *Open Government* pada september tahun 2011 dengan nama OGI (*Open Government Indonesia*) dengan susunan struktural seperti pada Gambar 1.

Rancangan program OGI didasari oleh tiga hal, yaitu transparansi, partisipasi, dan inovasi (OGI 2013). Transparansi dalam OGI bertujuan agar warga mengetahui apa yang dilakukan oleh Pemerintah (*to know what government's doing*), sedangkan partisipasi dimaksudkan agar Pemerintah mendapatkan dukungan gagasan dan keahlian dari warga dalam menjalankan fungsinya (*public to contribute idea and expertise*).



Gambar 1. Susunan Struktural OGI[1]

Kolaborasi melibatkan berbagai level pemerintahan, termasuk sektor swasta untuk meningkatkan efektivitas pemerintah dalam menjalankan fungsinya (*improves the effectiveness of Government*). Secara konseptual, OGI merupakan inisiatif lebih lanjut dari pemerintah untuk bergerak proaktif mencapai tujuan keterbukaan informasi yang diinginkan oleh UU Keterbukaan Informasi Publik. *Open Government Indonesia* (OGI) adalah bagian dari gerakan *Global Open Government Partnership* (OGP) yang saat ini memiliki 60 negara anggota (dan terus bertambah), di mana pada tahun 2013 Indonesia menjadi *Lead Chair OGP* di laman http://opengovindonesia.org/

Skema ini merupakan suatu kebijakan mendasar dari Pemerintah untuk memindahkan berbagai data dasar ke wilayah publik dan sekarang *open data* di Indonesia sudah dapat diakses di halaman web pemerintah dengan laman http://data.go.id/. Informasi yang

[1]http://opengovindonesia.org

disajikan di dalamnya belum lengkap dan tidak *update* secara *realtime*. Agar program *Open Government Indonesia* berjalan sesuai dengan tujuan awal dan untuk menuju standar *World Class Government* terdapat beberapa standar penyajian data perlu diperbaiki, di mana perlu ratusan ribu ataupun jutaan data yang harus diupload ke *server* dan pada kajian ilmiah ini akan dibahas bagaimana *design* infrastruktur untuk mendukung aplikasi secara optimal, di mana data tersebut dapat diakses oleh semua orang dalam waktu yang *real time*. Maka dibutuhkan *design* perangkat keras *(server)* dan jaringan yang mempunyai performa komputasi tinggi.

Salah satu permasalahan untuk menerapkan *Open Government* adalah *High Performance Computing* di mana dapat dipastikan yang melakukan *request data* pada sistem akan sangat banyak dengan jumlah penduduk Indonesia adalah sekitar 155.816.970 jiwa berada pada rentangan usia produktif pada umur 15-64 tahun dan sebanyak 88,1 juta dari populasi total menggunakan internet di Indonesia dengan 78,5% tinggal di Indonesia bagian barat. Ibukota DKI Jakarta menjadi wilayah dengan penetrasi paling tinggi dengan 65% pengguna internet. Disusul oleh DI Yogyakarta dengan 63% pengguna internet. Tercatat sekitar 53 juta pengguna internet terkonsentrasi di pulau Jawa dan Bali. Posisi terendah ditempati oleh Papua dengan 20% pengguna internet dari total populasi penduduknya[2]. Pemanfaatan teknologi HPC tentunya akan sangat membantu pelaksanaan *Open Government*, karena dengan kemampuan yang telah ada saat ini, *HPC* tersebut dapat mendukung *request* sampai dengan *One Billion per Second* dengan memanfaatkan *In Memory Computing* (GridGain, 2015). Salah satu permasalahan pada *HPC* adalah terjadinya *bottleneck*[3]. *Bottleneck* tersebut dapat terjadi antara *CPU*, *Memory* dan Media Penyimpanan. *Bottleneck* tersebut terjadi akibat adanya ketidaksesuaian kemampuan melakukan transfer data antara sumber dengan tujuan dan sebaliknya, sehingga menyebabkan terjadi antrian data dan pada akhirnya menyebabkan berbagai masalah, salah satu yang paling sering terjadi adalah lamanya waktu pemrosesan.

Untuk mengatasi permasalahan tersebut, kajian ilmiah yang telah dilakukan (Ammons, 2004), menggunakan algoritma untuk mendeteksi *bottleneck* yang terjadi pada sebuah perangkat lunak. Aturan dalam perangkat lunak harus memenuhi 80/20: optimasi secara agresif pada beberapa bagian vital eksekusi mengakibatkan peningkatan performa secara maksimal. Akan tetapi, menemukan bagian yang dibutuhkan bisa sangat sulit, terutama pada sistem skala besar seperti aplikasi web. Setiap menjalankan analisis tersebut, perangkat lunak tersebut membuat profil dari setiap hasil eksekusi. Pada waktu yang akan datang akan diketahui bagian mana saja dari aplikasi tersebut yang mengalami *bottleneck*.

Selain itu, salah satu hal lainnya yang menarik perhatian adalah *bottleneck* dapat terjadi pada transfer data dari media penyimpanan sendiri (Vasudeva, 2012) maupun melalui media jaringan (Marazakis, 2007).

Vasudeva (2012:10), menunjukkan bahwa performa dari prosesor *server* dengan *disk I/O* memiliki *gap* atau jarak yang sangat jauh. Untuk setiap waktu pemrosesan yang sama, pada waktu pemrosesan di dalam *disk I/O*, dapat dihasilkan jutaan operasi pemrosesan data di dalam *CPU* dan ratusan ribu operasi pemrosesan data di dalam memori yang dapat dilakukan. Pada saat terjadi operasi pemrosesan di dalam *CPU*, waktu yang dapat dihasilkan pada rentang 0,5*ns* sampai dengan 100*ns*, sedangkan untuk waktu pemrosesan di dalam disk, berada pada rentang waktu 500.000*ns* atau 500*ms* sampai dengan 20.000.000*ns* atau 20*s*.

Marazakis menggunakan teknologi 10GbE dengan *NIC* yang mendukung, untuk mengetahui kemampuan transfer maksimal dari protokol yang diusulkan. Akan tetapi, karena keterbatasan pada kemampuan transfer pada *PCI Express*, *peak throughput*, *memory to memory transfer*, dengan mengirimkan 4KB pesan

[2]http://tekno.liputan6.com/read/2197413/jumlah-pengguna-internet-indonesia-capai-881-juta
[3]Diambil dari halaman http://www.oxforddictionaries.com/definition/english/bottleneck

secara *peer to peer*, untuk melakukan transfer satu arah, menghasilkan kecepatan sebesar 626MBps atau menggunakan kemampuan transfer sebesar 6,26% dari kemampuan secara teoretis dari 10GbE tersebut.

Hasil dari perbaikan protokol yang dilakukan oleh Marazakis, membuat terjadinya peningkatan dari 200 menjadi 290 MBps atau sekitar 45% pada saat menggunakan 8 Disk SATA RAID 0 dan dengan menggunakan *Ramdisk*, performa puncak meningkat dari 320 menjadi 470MBps atau meningkat sebesar 48%.

Selain itu, perkembangan prosesor sebagai pusat terjadinya berbagai komputasi, mengalami percepatan yang jauh melampaui kemampuan memori komputer (Carvalho, 2002). Masalah utama dalam perbedaan perkembangan tersebut karena industri semikonduktor dibagi menjadi dua, industri mikroprosesor dan industri memori. Akibatnya, industri mikroprosesor fokus kepada peningkatan kecepatan dan industri memori fokus kepada peningkatan kapasitas, sehingga peningkatan performa mikroprosesor mencapai 60% per tahun, sedangkan peningkatan kapasitas memori mengalami peningkatan kurang dari 10% per tahun (Carvalho, 2002:Fig 1).

Pada penelitian sebelumnya, telah dilakukan oleh Ousterhout (2010) dengan mengimplementasikan teknologi *"In Memory Computing"* ke dalam *cloud server* yang berbasis *memory* DRAM yang menghasilkan *throughput* 100-1000 kali dibandingkan dengan sistem berbasis *disk* dan memiliki *latency* 100-1000 kali lebih rendah. Selain itu, dijelaskan juga penggunaan utama dari *RAMCloud* dan apa saja tantangan dan hambatan beserta keuntungan menggunakan *RAMCloud*. Kapasitas dari RAMCloud juga telah mencapai ratusan Tera Byte, bergantung kepada slot memori tiap komputer server yang digunakan.

Selain itu, (Ansori, 2015) memanfaatkan *notebook* untuk mengatasi permasalahan *bottleneck* pada sistem *storage disk* tersebut, dengan melakukan proses memindahkan seluruh *root file system* dari sistem operasi yang berada didalam *storage device* ke memori komputer dengan memanfaatkan teknologi *tmpfs* dengan spesifikasi *notebook* ditunjukkan pada Gambar 2.

Proses untuk memindahkan keseluruhan *root file system* tersebut ke dalam *memory*, dilakukan langkah-langkah sebagai berikut:

1. Memilih distro linux yang mempunyai dukungan untuk *booting* dari *initramfs.*
2. Melakukan instalasi ke dalam *disk*, dan dipecah tidak menjadi satu *root file system*.
3. *Boot* sistem hasil instalasi, *update system*, dan berisi aplikasi paling minimum. Setiap aplikasi akan di-*load* di dalam *memory.*
4. Melakukan modifikasi pada bagian *mounting system* pada *fstab.*
5. Melakukan perubahan pada *script initramfs* dan melakukan *build ulang initramfs* agar dapat *booting* dari *memory* komputer.
6. Mengubah berkas *grub* agar dapat menjalankan *script initramfs* yang melakukan *booting* menggunakan *tmpfs* di dalam *memory* komputer.

Gambar 2. Spesifikasi *Notebook*
(Ansori 2015: 48)

Hasil pengujian yang telah dilakukan oleh Ansori, menunjukkan bahwa dari hasil *benchmarking*, didapatkan data sebagaimana pada tabel 1.

Tabel 1. Hasil *Benchmarking* dengan *Phoronix Test Suites*

No.	Keterangan	Hard Disk Drive	Memory
1	*AIO-Stress Test*	3076.56MB/s	5450.16MB/s
2	*FS Mark*	23.23 Files/s (1000 Files 1MB sizes)	4 8 7 1 . 3 3 Files/s (4000 Files, 32 Sub Dirs, 1MB Size)
3	*DBench*	97.14MB/s (128 Clients)	7176.82MB/s (256 Clients)
4	*Iozone – 8GB Write*	82.65 MB/s	5299.84 MB/s
5	*PostMark*	5396 TPS	8429 TPS
6	*Apache Benchmark*	39242.25 RPS	41255.76 RPS

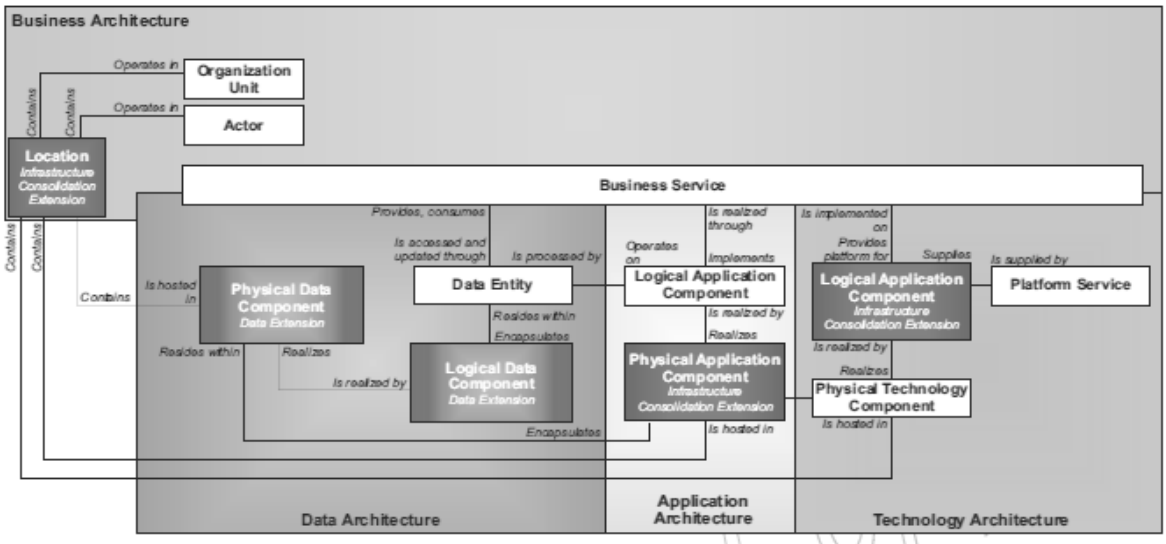
(Ansori 2015:49-56)

Hasil dari pengujian untuk penelitian yang dilakukan Ansori (2015), membuktikan bahwa dengan menggunakan *notebook* dengan cara memindahkan seluruh proses ke dalam memori komputer, didapatkan peningkatan kinerja yang sangat signifikan dan cocok untuk digunakan pada lingkungan yang membutuhkan intensitas komputasi dan transfer data yang sangat tinggi.

HASIL DAN PEMBAHASAN

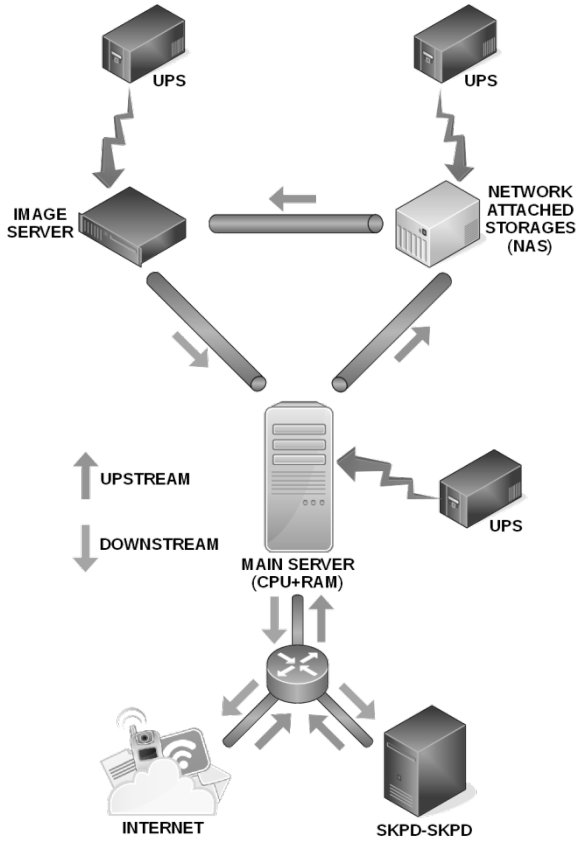
Pada kajian ilmiah ini, difokuskan pada proses desain awal perangkat keras berupa server, perangkat pendukung dan jaringan komputer agar dapat digunakan pada lingkungan yang membutuhkan intensitas komputasi dan transfer data yang sangat tinggi seperti pada *open government data* dengan mengkaji dari data penelitian sebelumnya, sehingga diharapkan *bottleneck* yang terjadi dapat diminimalkan dan mendapatkan kinerja komputasi yang maksimal.

Desain awal spesifikasi perangkat keras dan perangkat jaringan untuk mendukung *Open Government* ini menggunakan TOGAF 9.1 sebagai acuan yang ditunjukkan pada Gambar 3. Pada standardisasi TOGAF 9.1 tersebut, dibagi menjadi 4 bagian besar, antara lain *Business Architecture, Data Architecture, Application Architecture dan Technology Architecture.*



Gambar 3. Standar TOGAF 9.1

Penelitian ini, hanya fokus pada *Technology Architecture* yang harus dikembangkan agar data dapat diakses oleh lebih dari puluhan ribu user dalam satu waktu, desain awal meliputi bagian pada *Physical Technology Component pada bagian* desain pada perangkat keras yaitu berupa server beserta pendukungnya dan desain pada jaringan atau *Logical Aplication Component Infrastructure* yang ditunjukkan pada Gambar 4.

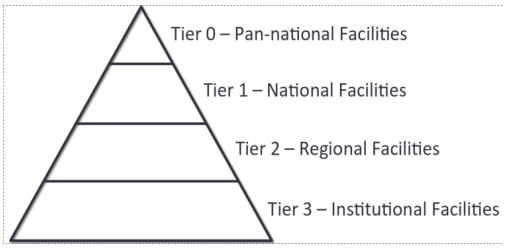


Gambar 4. Konsep Desain Awal

Selain itu, posisi dan kebutuhan untuk menggunakan HPC dalam tiap *Tier* yang dibutuhkan bergantung kepada level dari *tier* tersebut. Diurutkan dari bawah piramida pada Gambar 5, *Tier* 3 digunakan untuk fasilitas yang dibutuhkan dalam skala institusi atau universitas. *Tier* 2 digunakan untuk fasilitas daerah (kabupaten atau kota) sampai dengan provinsi. *Tier* 1 digunakan untuk fasilitas dalam skala nasional atau satu negara, sedangkan *Tier* 0 digunakan untuk fasilitas antarnegara yang terhubung satu dengan yang lainnya.

Kompleksitas antar *Tier* dari *Tier* 3 sampai ke *Tier* 0 tentunya memiliki perbedaan yang sangat signifikan di bagian infrastruktur, perangkat keras dan perangkat lunak yang digunakan. Hal tersebut terjadi karena kebutuhan dalam setiap *Tier* juga berbeda.

Tier yang berada di atas dan memiliki *tier* di bawahnya, harus memiliki kemampuan untuk dapat menghubungkan dan mengakomodasi kebutuhan dari *Tier* di bawahnya karena *Tier* yang berada di bawahnya dapat berjumlah lebih dari satu bagian, sehingga *Tier* 3 adalah *Tier* yang relatif tidak kompleks dalam kebutuhan infrastruktur, perangkat keras dan perangkat lunak. Akan tetapi, *Tier* 2 harus mampu untuk mengakomodasi kebutuhan dari *Tier* 3 dan kebutuhan terhadap *Tier* 2 sendiri. Begitu juga untuk *Tier* 1 dan Tier 0 yang berada di atasnya.



Gambar 5. Piramida Kedudukan HPC dalam Beberapa *Tier*

Berdasarkan kebutuhan dari tiap *Tier* tersebut, pada kajian ilmiah ini kami mencoba untuk melakukan improvisasi dalam desain infrastruktur dan perangkat jaringan yang ada agar dapat menghindari terjadinya *bottleneck* dan dapat menyelesaikan permasalahan yang terjadi akibat adanya *bottleneck* tersebut terhadap ketersediaan akses informasi.

Pada desain perangkat keras berupa server, beberapa hal penting yang harus diperhatikan, antara lain:

1. Processor

Kemampuan prosesor dalam menangani sebuah proses komputasi akan berbeda antara satu tipe dengan tipe lainnya apalagi pada saat menggunakan satu atau lebih keluarga prosesor yang berbeda dalam satu sistem, misalkan menggunakan prosesor Intel atau AMD. Untuk mendapatkan perkiraan kemampuan dari sebuah prosesor dalam melakukan pemrosesan sebuah data atau informasi, dapat kita lakukan dengan memperhatikan masing-masing nilai dari parameter[4] *instruction count (IC), clocks per instruction (CPI), and clock time (CT)*.

[4]http://www.d.umn.edu/~gshute/arch/performance-equation.xhtml

Instruction Count adalah metode pengukuran dinamis untuk jumlah instruksi tereksekusi pada sebuah program, biasanya ditandai dengan *loop* dan *recursions*. Untuk tujuan perbandingan antara dua mesin dengan *instruction set* berbeda, dapat dilakukan perbandingan dengan cara melakukan kompilasi dari kode bahasa pemrograman tingkat tinggi yang sama pada sebuah kedua mesin tersebut.

Clock Per Instruction adalah efektivitas rerata terhadap keseluruhan eksekusi instruksi di dalam sebuah program, termasuk di dalamnya adalah kasus untuk instruksi kategori pada cabang, beban dan penyimpanan. *CPI* dipengaruhi oleh paralelisme level instruksi dan kompleksitas instruksi. Tanpa adanya paralelisme level instruksi, instruksi sederhana membutuhkan 4 atau lebih *cycles* sebelum tereksekusi.

Clock Time adalah periode dari sebuah *clock* yang melakukan sinkronisasi jalur sirkuit di dalam prosesor. Contohnya adalah, prosesor berkecepatan 1GHz memiliki *latency* sebesar 1ns dan prosesor berkecepatan 4GHz memiliki *latency* sebesar 0,25ns. Jadi, secara ideal, untuk menyelesaikan satu set instruksi di dalam prosesor, *latency* yang terjadi bergantung kepada kecepatan dari prosesor tersebut.

Clock time dipengaruhi oleh teknologi sirkuit dan kompleksitas dari sebuah pekerjaan diselesaikan dalam satu *clock*. Gerbang logika tidak bekerja secara instan. Setiap gerbang memiliki *delay* propagasi yang bergantung pada jumlah input dari sebuah *gate* dan jumlah input yang terhubung pada gerbang output. Menambahkan jumlah input dan output akan memperlambat waktu propagasi.

Selain itu, untuk dapat menghitung besaran *bandwidth* yang dibutuhkan pada sebuah prosesor, dapat digunakan persamaan 1.

$$MB=PsxMbxMch \quad (1)$$

Keterangan:
MB = *Max processor bandwidth*
Ps = *Processor speed*
Mb = *Memory bus width*
Mch =*Max of Memory Channels Processor*

Sebagai contohnya untuk perhitungan *bandwidth* adalah dengan menggunakan simulasi[5], didapatkan hasil perhitungan sebesar 1.6GHz x 64bits x 2 sama dengan 25.6 GB/s.

2. Memory

Secara teoretis, *maximum memory bandwidth* disebut juga sebagai *"burst rate"* adalah *bandwidth* yang dapat dicapai maksimal oleh memori, ditunjukkan pada persamaan 2.

$$Br=BcxNdxMbxNif \quad (2)$$

Keterangan:
Br = *Burst Rate*
Bc = *Base Clock Frequency*
Nd = *Number data transfer per clock*
Mb = *Memory bus width*
Nif = *Number of Interface*

Sebagai contoh, komputer dengan memori *dual-channel* dan sebuah memori modul DDR3-1600 untuk tiap *channel* yang berjalan pada kecepatan 800 MHz akan memiliki *burst rate* sebesar 800,000,000 *clocks* per detik x 2 jalur per *clock* x 64 bit per jalur × 2 *interfaces* adalah sebesar 204.800.000.000 (204,8 *billion*) *bit persecond* (dalam satuan *bytes* menjadi 25.600 MB/s or 25,6 GB/s).

3. Tipe Server

Pada bagian tipe server, ada tiga poin utama yang harus diperhatikan untuk memaksimalkan penggunaannya, yaitu:

1) Server dengan jumlah prosesor sangat banyak
2) Server dengan jumlah memori RAM sangat besar
3) Server dengan jumlah media penyimpanan seperti HDD dan SSD sangat besar
4) Server dengan dua atau tiga kombinasi dari ketiga poin sebelumnya.

Dari keempat poin konfigurasi server tersebut, masing-masing konfigurasi pada keempat poin tersebut memiliki fungsi dan tujuan yang berbeda-beda.

[5]http://ark.intel.com/products/67356/Intel-Core-i7-3612QM-Processor-6M-Cache-up-to-3_10-GHz-rPGA

4. *Multi Rack* Server

Server yang digunakan untuk media pemrosesan utama, ada kemungkinan tidak hanya menggunakan satu buah server, tetapi menggunakan lebih dari satu server untuk pemrosesan data yang dilakukan dalam satu rak tersebut.

Untuk menghubungkan antarkomputer di dalam rak tersebut maupun menghubungkan dari komputer ke jaringan utama, pada saat menggunakan teknologi ini, akan membutuhkan *Network Interface Card (NIC)* dengan kemampuan sama atau sebanding dengan kecepatan *memory bandwidth* yang ada di dalam *memory* yang digunakan. Apabila tidak sebanding, maka transfer rate tersebut akan mengalami *bottleneck* di NIC sebesar persamaan 3.

$$\%BNIC = \frac{Br - Tr}{Br} \times 100\% \quad (3)$$

Keterangan:

% BNIC = Prosentase *Bottleneck* di NIC

Tr = Kapasitas transfer NIC

Sebagai contoh, apabila *burst rate* adalah 25,6GBps, sedangkan kemampuan transfer jaringan menggunakan 10Gbps atau sekitar 1,25GBps, maka akan mengalami persentase *bottleneck* sebesar 95,11%. Hal ini akan berbeda ketika kita menggunakan teknologi yang mampu melakukan kecepatan transfer hingga kapasitas 100Gbps atau sekitar 12,5GBps, sehingga akan mengalami persentase *bottleneck* sebesar 51,17%.

5. Backup Server

Backup server sangat penting untuk disediakan karena teknologi ini diletakkan didalam memori yang memiliki sifat sementara dan dapat terjadi kehilangan data akibat kehilangan daya secara tiba-tiba.Oleh karena itu, kebutuhan kapasitas penyimpanan pada *backup server* adalah minimal 2x dari kapasitas server utama.

Selain itu, karena berfungsi sebagai *backup*, maka tidak memerlukan prosesor dengan spesifikasi yang sangat tinggi, sedangkan untuk media *LAN* yang dibutuhkan minimal adalah berkecepatan *gigabit ethernet* untuk mengurangi *bottleneck* dan lamanya waktu *backup* yang dapat terjadi, karena perbedaan kecepatan *transfer data*.

Jumlah dan kecepatan *network transfer* dari *backup server* bergantung pada *file system layout*, *system CPU load* dan *memory usage* (Symantec, 2011). Selain itu, pemilihan jenis backup server juga bergantung terhadap seberapa penting data dan besaran budget yang dimiliki yang ditampilkan pada Tabel 2.

Pertimbangan lain yang harus diperhatikan adalah kemampuan dari *network* data transfer yang berkaitan erat dengan kemampuan dari *network technology* yaitu kebutuhan dan kemampuan transfer data dengan menggunakan teknologi *Fast Ethernet, Gigabit Ethernet dan Ten Gigabit Ethernet* seperti yang ditampilkan pada Tabel 3.

Dari Tabel 2 dan Tabel 3, dapat diambil kesimpulan bahwa dengan menggunakan teknologi jaringan yang sesuai dengan kebutuhan dan kapasitas yang dimiliki untuk menentukan berapa besar data yang mampu ditransfer ke dalam *server backup* setiap jam.

Selain itu, untuk mempermudah penyimpanan, data yang berada pada server *backup* tersebut dijadikan *file image* berekstensi *ISO* atau *file image* yang terkompresi seperti *zip* atau *gzip*.Alasan mengapa menggunakan *ISO, zip* atau *gzip* adalah kemudahan dalam melakukan proses penyimpanan dan dapat digunakan secara *on-the-fly* untuk dilakukan *booting* dari *file image* tersebut sesuai dengan *image* yang dibutuhkan di kemudian hari.

File image yang telah disebutkan sebelumnya, dapat dengan mudah dimanipulasi untuk dijadikan *file image booting* melalui jaringan dan langsung terpasang di dalam *memory server*, sehingga dapat dengan mudah mengganti peran dan fungsi dari server utama secara relatif cepat dan mudah.

Tabel 2. *Drive Controller Data Transfer Rate*

Drive Controller	*Theoretical megabytes per second*	*Theoretical gigabytes per hour*
ATA-5 (ATA/ATAPI-5)	66	237.6
Wide Ultra 2 SCSI	80	288
iSCSI	100	360
1 gigabit Fibre Channel	100	360
SATA/150	150	540
Ultra-3 SCSI	160	576
2 gigabit Fibre Channel	200	720
SATA/300	300	1080
Ultra320 SCSI	320	1152
4 gigabit Fibre Channel	400	1440

(Symantec, 2011:28)

Tabel 3. *Network Data Transfer*

Network Technology	*Theoretical gigabytes per hour*	*Typical gigabytes per hour*
100BaseT	36	25
1000BaseT	360	250
10000BaseT	3600	2500

(Symantec, 2011:29)

6. Backup Power Supply

Pada perangkat *backup power supply*, dalam menentukan besaran daya yang diperlukan untuk dapat menangani kebutuhan setiap server dan untuk mengetahui berapa lama *backup power supply* tersebut dapat bertahan dalam melakukan *supply* daya, dapat dilakukan perhitungan dengan cara daya dikalikan satuan waktu tertentu untuk setiap server untuk mendukung proses *backup* yang terjadi.

$$Tt = \frac{D}{KTK} \qquad (4)$$

$$\sum P = \frac{N \times Ps \times D}{KTK} \qquad (5)$$

Keterangan :

Tt = Waktu transfer (detik)

$\sum P$ =Total daya listrik yang dibutuhkan selama melakukan *backup* (kWH)

N = Jumlah server yang dimiliki

Ps = Daya listrik tiap server (Watt)

D = Besar data yang dimiliki (GB)

KTK = Kemampuan Transfer Terkecil (MBps)

Mengacu kepada desain yang telah dibuat, apabila untuk masing-masing server diasumsikan membutuhkan daya sebesar 500W dengan jumlah data 150GB dan ditransfer dengan jaringan 1Gbps, maka total daya yang dibutuhkan adalah 0,5kWH dengan waktu transfer sebesar 1200 detik.

Jadi, untuk mengetahui kapasitas *backup supply daya* secara maksimal adalah dengan menggunakan D sebesar kapasitas data yang dimiliki dan KTK terkecil yang dimiliki oleh komponen yang berada di dalam server, perangkat jaringan dan perangkat pendukung jaringan.

7. Jaringan Data

Untuk desain pada perangkat jaringan, hal yang harus diperhatikan adalah media transmisi dan perangkat yang digunakan pada kedua bagian *backup* server, jaringan Internet dan jaringan lokal untuk SKPD yang ada. Pemilihan *bandwidth* dan media transmisi sangat berpengaruh pada jumlah pengguna yang dapat terkoneksi pada satu waktu yang bersamaan. Berdasarkan data kecepatan internet yang ada di Indonesia saat ini (Akamai, 2014), rata-rata kecepatan pengguna internet di Indonesia adalah 2,4Mbps.

Apabila dengan perencanaan jaringan seperti pada Gambar 3, maka kita dapat melakukan estimasi (secara teoretis) untuk jumlah pengguna yang dapat terkoneksi pada satu waktu secara bersama-sama dari jaringan internet ditunjukkan pada persamaan 6.

$$\sum user = \frac{Bandwidthjaringan}{Kecepatanrerata} \qquad (6)$$

Sebagai contoh, apabila menggunakan jaringan berkapasitas 1Gbps dengan rerata kecepatan tiap koneksi adalah 2.4Mbps, maka jumlah *user* yang dapat terkoneksi pada satu waktu secara bersamaan sekitar 416 user. Besaran *bandwidth* jaringan yang dapat digunakan secara maksimal adalah tidak melebihi dari *bandwidth* memori komputer yang dimiliki. Jika melebihi, maka *bottleneck* akan terjadi pada memori akibat ketidakseimbangan *bandwidth*.

Untuk saat ini, teknologi untuk melakukan transfer pada jaringan, telah mencapai kecepatan 300Gbit/s atau sekitar 37,5GB/s yang dimiliki oleh device *Infiniband EDR 12×*. Oleh karena itu, diperlukan juga *switch* atau *router* yang memiliki kemampuan sesuai dengan spesifikasi yang dibutuhkan dan dapat melakukan proses manajemen lalu lintas data dan *bandwidth* yang dimiliki.

Manajemen *bandwidth* harus benar-benar tertata dengan rapih dan sesuai dengan kebutuhan masing-masing bagian. Manajemen tersebut dapat dilakukan dengan cara memprioritaskan antara kebutuhan untuk proses *upstream* dan kebutuhan untuk proses *downstream*.

Apabila kebutuhan *upstream* lebih besar, maka *bandwidth* yang tersedia dialokasikan lebih daripada *downstream*, begitu juga sebaliknya. Apabila dimungkinkan, NIC yang digunakan untuk *upstream* dan *downstream* juga dipisahkan agar mudah untuk melakukan manajemen jaringan.

8. *Bottleneck*

Pada persentasinya (Vasudeva 2012:11), menunjukkan bagian-bagian yang akan terjadi *bottleneck* pada sebuah data center. *Bottleneck* tersebut dibagi menjadi 6 bagian, antara lain:

1) Application

Bottleneck pada bagian ini sering terjadi pada bagian *Excessive Locking, Data Contention dan I/O Delays/Error*.

2) Network I/O

Pada bagian ini, *bottleneck* yang sering terjadi berada pada bagian *Network Congestion, Dropped Packet, Data Retransmissions, Timeouts* dan *Component Failures*.

3) Storage I/O Connect

Pada bagian ini, *bottleneck* yang sering terjadi adalah pada komponen *Lack of Bandwidth, Overloaded PCIe Connect* dan *Storage Device Contention*.

4) User Bottleneck

Pada bagian ini, *bottleneck* yang sering terjadi berada pada bagian *Connectivity Timeouts* dan *Workload Surges*.

5) Server Bottleneck

Pada bagian ini, sering terjadi *bottleneck* akibat dari *Lack of Server Power, IO Wait and Queuing CPU* dan *Overhead I/O Timeouts*.

6) Device Bottleneck

Pada bagian ini, parameter yang sering membuat terjadi *bottleneck* adalah *Device I/O Hotspots, Cache Flush* dan *Lack of Storage Capacity*.

Selain dari besaran *bandwidth*, ada satu hal lain yang perlu diperhatikan, yaitu *latency* dari

setiap media, mulai dari media penyimpanan, memori sampai dengan media transmisi di jaringan (Larsen, 2009).

Model *latency* yang digunakan sebagai analisis antara lain:

1) *Synchronization latency*

   Latency ini terjadi akibat proses sinkronisasi pesan atau *flags* pada sebuah aplikasi yang berjalan pada sistem terdistribusi di berbagai lokasi komputer. Contoh yang paling sering terjadi adalah pada bagian komputasi paralel.

2) *Distributed memory latency*

   Latency ini terjadi akibat proses sinkronisasi thread yang berjalan pada satu komputer dengan melakukan akses memori dari satu komputer ke komputer lainnya di dalam sebuah *cluster*. Contohnya terjadi pada *High Performance Computer (HPC)*. *Latency* yang terjadi dipengaruhi oleh *latency* dari media transmisi dan *latency* dari memori tersebut.

3) *Storage media latency*

   Latency ini biasanya terjadi pada saat kita melakukan akses pada *storage media* seperti *HDD* dan *SSD*, biasanya dalam orde milidetik.

Berdasarkan *latency* yang telah disebutkan sebelumnya, dengan menggunakan media penyimpanan di dalam memori, maka *latency* pada *storage media* telah memasuki orde satuan mikrodetik. Jauh lebih cepat dan melampaui *storage media* konvensional. Maka, *latency* yang terjadi pada bagian nomor satu dan dua yang harus menjadi perhatian selanjutnya dalam melakukan desain awal ini.

Dari hasil kajian yang telah dilakukan pada berbagai komponen sebelumnya, didapatkan prakiraan atau estimasi dari perangkat utama dan perangkat pendukung yang dapat digunakan untuk melakukan perencanaan awal seperti pada gambar desain awal yang diusulkan pada saat akan membangun suatu sistem yang akan dimanfaatkan untuk *Open Government*.

PENUTUP

Dari hasil desain awal yang dilakukan, secara teoretis, berdasarkan kajian yang dilakukan, telah berhasil meminimalkan secara maksimal agar tidak terjadi *bottleneck* pada media penyimpanan dan *transfer* data di dalam jaringan, nantinya akan mendukung *High Performance Computing* yang dibutuhkan pada implementasi *Open Government* dengan ketentuan apabila mengikuti aturan yang telah diberikan pada bagian sebelumnya secara menyeluruh.

Akan tetapi, apabila ada hal-hal yang harus diubah sesuai dengan kondisi yang dibutuhkan telah diberikan juga beberapa parameter melalui persamaan-persamaan yang dapat digunakan sebagai acuan dalam melakukan desain dari komponen *High Performance Computer* di dalam server dan jaringan, karena pada desain spesifikasi perangkat keras dan perangkat jaringan atau infrastruktur *High Performance Computing* pada kajian kali ini belum mempertimbangkan *load memory* dan *bandwidth* dari aplikasi (perangkat lunak) yang harus dibangun atau yang telah dimiliki agar dapat menyesuaikan pada kebutuhan di bidang *Open Government*.

Pada kajian ilmiah yang akan datang, akan dilakukan uji validitas desain awal dengan mempertimbangkan *Load* dari Aplikasi yang harus dibangun untuk mengembangkan *Open Government* di Indonesia dengan memperhatikan siapa saja *stakeholder* yang berhak mengakses sistem tersebut dan pengaruhnya terhadap apa saja yang telah dilakukan untuk setiap bagian agar dapat diterapkan pada kondisi sebenarnya.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih atas acara GNOME ASIA SUMMIT 2015 yang terselenggara di Depok pada 7-9 Mei 2015, yang menjadi awal pertama kali penulis melakukan penelitian ilmiah tentang tema "*In Memory Computing"*.

Selain itu, penulis juga mengucapkan terima kasih kepada penulis kedua sebagai partner pertama antarkampus yang bersedia mendukung dan ikut serta dalam penulisan karya ilmiah ini.

DAFTAR PUSTAKA

Akamai. (2014). *The state of internet report. Asia-Pacific highlights-first quarter*.

Ammons, Glenn, Jong-Deok Choi, Manish Gupta, & Nikhil Swamy. (2004). *Finding and removing performance bottlenecks in large systems. In ECOOP 2004–Object-oriented programming* (pp. 172-196). Springer Berlin Heidelberg.

Carvalho, Carlos. (2002). *The gap between processor and memory speeds. In Proc. of IEEE International Conference on Control and Automation*.

OGI. (2013). *Laporan pelaksanaan open government Indonesia Tahun 2012. Open government Indonesia: Era baru keterbukaan pemerintah.*

Ousterhout, John, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra et al. (2010). *The case for RAMClouds: scalable high-performance storage entirely in DRAM. ACM SIGOPS Operating Systems Review* 43, no. 4 (pp. 92-105).

Symantec. (2011). *Symantec netbackup TM backup planning and performance tuning guide UNIX, Windows and Linux Release 7.0 through 7.1.*

Undang-Undang Republik Indonesia Nomor 14 Tahun 2008 Tentang Keterbukaan Informasi Publik. (2008).

Jurnal dan Prosiding

Larsen, Steen, Parthasarathy Sarangam, Ram Huggahalli, & Siddharth Kulkarni. (2009). *Architectural breakdown of end-to-end latency in a TCP/IP network. International journal of parallel programming* 37, no. 6 (pp. 556-571).

Marazakis, Manolis, Vassilis Papaefstathiou, & Angelos Bilas. (2007). *Optimization and bottleneck analysis of network block I/O in commodity storage systems.*In *Proceedings of the 21st annual international conference on Supercomputing*, (pp. 33-42).ACM.

Internet

Ansori, Anton Siswo Raharjo. (2015). *An overview, maximize the ability of the GNU/ Linux Operating System using "in memory computation" for Academic, Business and Government*. GNOME ASIA 2015. Depok. Diperoleh tanggal 15 Juli 2015 dari http://www.slideshare.net/AntonSiswo/final-presentasi-gnome-asia-47979920.

GridGain White Paper. Gridgain in memory data fabric: Meeting the Extreme Demands of Financial Services Computing. Diperoleh tanggal 09 November 2015 dari http://www.gridgain.com/wp-content/uploads/2015/09/GridGain_Brief_Financial_Services.pdf.

HPC Architerctures, Types of Currently in Use. Diperoleh tanggal 09 November 2015 dari http://www.archer.ac.uk/training/course-material/2014/04/IntroHPC_Edi/Slides/L07_HPCArch.pdf.

Open Government Partnership UK National Action Plan 2013 to 2015. Diperoleh tanggal 15 Juli 2015 dari https://www.gov.uk/government/consultations/open-government-partnership-uk-national-action-plan-2013

Vasudeva, Anil. (2012). *Solid State Storage: Key to NextGen Enterprise & Cloud Storage.* (2012). Diperoleh tanggal 15 Juli 2015 dari http://www.snia.org/sites/default/files/AnilVasudeva_Solid_State_Storage_Key_NextGen.pdf