"Tema: 4 (teknik dan energi baru dan terbarukan)"

PERANCANGAN DAN IMPLEMENTASI *SYSTEM OPTIMIZER* PADA SISTEM BASIS DATA TERDISTRIBUSI MENGGUNAKAN METODE *TRIAL AND ERROR*

Oleh

Eddy Maryanto, Teguh Cahyono
Fakultas Teknik Universitas Jenderal Soedirman
eddy_maryanto@yahoo.com, teguh_sokaraja@yahoo.com

ABSTRAK

Basis data terdistribusi adalah sekumpulan basis data yang berinterrelasi secara logika dan terdistribusi pada jaringan 49ystem49r. Basis data terdistribusi mempunyai beberapa keunggulan antara lain reliabilitas yang tinggi dengan adanya basis data dan transaksi yang terdistribusi, performa yang lebih baik dengan adanya fragmentasi basis data yang memungkinkan eksekusi *query* secara *parallel*, serta kemudahan dalam pengekspansian 49 ystem. Pada penelitian ini akan dilakukan perancangan dan implementasi *system optimizer*, yang mana komponen ini berfungsi untuk mengoptimalkan kinerja dari sebuah 49ystem basis data terdistibusi. Metode yang digunakan dalam *system optimizer* yang dibuat adalah trial and error.

Kata Kunci: Basis Data Terdistribusi, *System Optimizer*, *Trial and Error*

ABSTRACT

Distributed database is a set of databases that is interrelated logically and distributed over a computer network. Distributed database has several advantages such as high reliability, better performance, and the ease in system expansion. In this research, system optimizer for distributed database is developed. This component has a function to optimize system performance. The method used in this system optimizer is trial and error method.

Key words: Distributed Database Systems, System Optimizer, Trial and Error

PENDAHULUAN

Teknologi basis data telah merubah paradigma pemrosesan data. Pada paradigma awal, setiap aplikasi mendefinisikan, memelihara, dan memiliki datanya sendiri-sendiri. Setelah dikembangkannya teknologi basis data, pendefinisisan dan pemeliharaan data dilakukan secara terpusat dan aplikasi-aplikasi berbagi data yang ada (*data sharing*). Dengan adanya teknologi basis data juga berdampak pada independensi data, dimana aplikasi-aplikasi tidak akan terpengaruh oleh adanya perubahan pisikal maupun 49ystem49 pada struktur data yang ada dan sebaliknya.

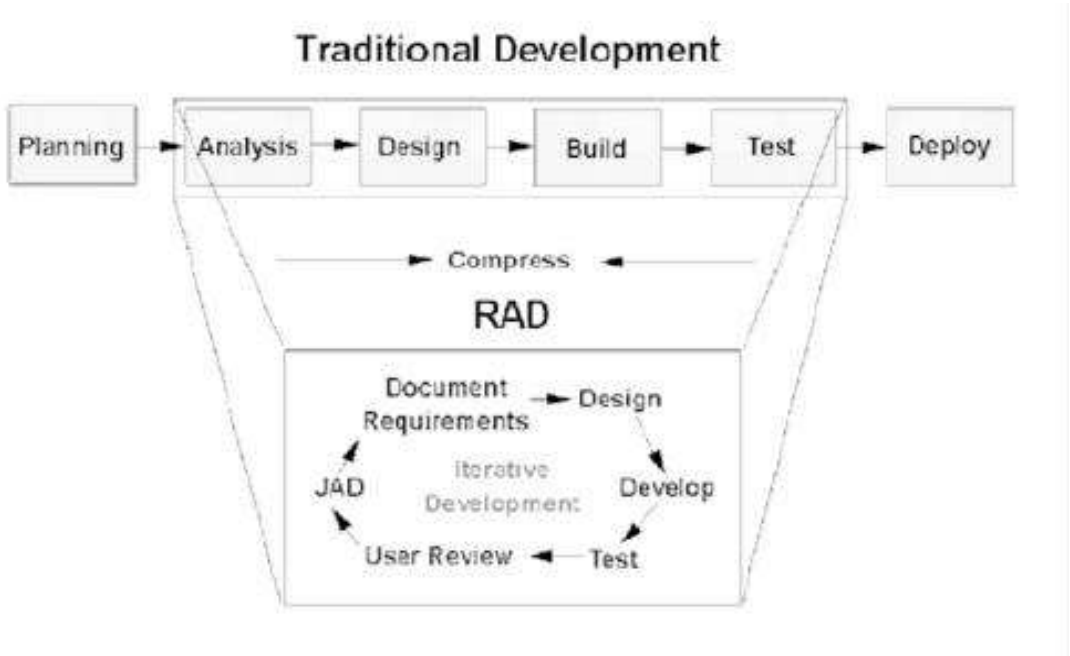
Sejalan dengan adanya kemajuan dalam teknologi jaringan 50ystem50r, teknologi 50ystem basis data telah dikembangkan 50ystem50 teknologi 50ystem basis data terdistribusi atau *distributed database system* (DDBS) yang mana teknologi ini merupakan perpaduan dari dua teknologi yaitu teknologi basis data dan teknologi jaringan 50ystem50r. Dengan adanya teknologi 50ystem basis data terdistribusi, paradigma pengelolaan data telah mengalami pergeseran dari pengelolaan yang terpusat menjadi pengelolaan data yang terintegrasi (*integrated data management*) (Özsu et.al, 2010).

Sistem basis data terdistribusi menjanjikan kinerja yang lebih baik dibandingkan 50ystem basis data terpusat. Peningkatan kinerja meliputi reliabilitas 50ystem yang lebih tinggi, eksekusi *query* secara 50ystem50r, dan kemudahan dalam ekspansi 50ystem. Disamping menjanjikan kinerja yang lebih baik, basis data terdistribusi memunculkan beberapa masalah yang sampai saat ini belum mempunyai solusi yang memuaskan dan perlu pengkajian lebih lanjut. Permasalahan-permasalahan tersebut antara lain proses fragmentasi, replikasi, dan alokasi pada suatu infrastruktur jaringan 50ystem50r dengan topologi dan karakteristik tertentu (Ryeng, 2011).

Permasalahan lainnya yang juga perlu dikaji adalah masalah yang terkait dengan impelementasi dari konsep atau model yang terkait dengan 50ystem basis data terdistribusi seperti mengimplementasi server data *query*, pemrosesan *joint-query*, dan *caching* hasil *query* intermediate maupun final yang efisien. Pada penelitian ini akan dikaji tentang permasalahan bagaimana membuat implementasi system optimizer yang efisien.

METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah studi literatur dan eksperimen. Pertama-tama ditentukan spesifikasi untuk *system optimizer* yang akan dibuat. Berdasarkan spesifikasi yang sudah ditentukan, dibuatlah rancangan untuk *system optimizer* tersebut. Hasil perancangan disajikan dalam bentuk model blok fungsional. Pada tahapan selanjutnya, dilakukan implementasi *system optimizer* berdasarkan hasil perancangan yang sudah dibuat pada tahapan sebelumnya dan dilanjutkan dengan pengujian untuk aspek fungsionalitas, efisiensi dan kehandalan dari *system optimizer* hasil implementasi untuk mengetahui apakah sudah efisien, handal, dan sudah dapat berfungsi sebagaimana yang diharapkan dan sesuai dengan spesikasi yang sudah ditentukan. Metode pengembangan 50ystem yang digunakan adalah metode Rapid Application Development (RAD) dengan tahapan pengembangan sebagaimana disajikan pada Gambar 1. Jenis pengujian yang digunakan *blackbox testing*, yang didahului dengan proses verifikasi dan validasi terhadap 50ystem yang sudah dibuat.



Gambar 1. Rapid Application Development (RAD)

HASIL DAN PEMBAHASAN

A. Optimalitas Sistem

Diasumsikan terdapat sejumlah fragmen $F = \{f_1, f_2, f_3, \ldots, f_l\}$ dan system berdistribusi terdiri dari situs-situs $S = \{s_1, s_2, s_3, \ldots, s_m\}$ dimana sejumlah $Q = \{q_1, q_2, q_3, \ldots, q_n\}$ aplikasi berjalan. Masalah alokasi adalah menemukan distribusi F yang optimal pada S.

Optimalitas sistem dapat didefinisikan terhadap dua ukuran (Dowdy dan Foster dalam Özsu, 2010):

1. Biaya minimal (*minimal cost*). Fungsi biaya terdiri dari biaya penyimpanan f_i pada setiap situs s_i, biaya *query* f_i pada setiap situs s_i, biaya update f_i pada setiap situs dimana f_i disimpan, dan biaya komunikasi data. Masalah alokasi di sini adalah bagaimana skema alokasi untuk meminimalkan fungsi biaya.

2. Performa (*performance*). Strategi alokasi dirancang untuk memelihara ukuran performa. Dua diantara ukuran performa yang ada adalah: (1) waktu tanggap (*response time*) yang mana waktu tanggap ini harus diminimalkan, dan (2) laju transfer data (*system throughput*) yang mana laju transfer data harus dimaksimalkan.

B. Informasi yang Dibutuhkan pada Tahap Alokasi

Pada tahap alokasi kita membutuhkan data kuantitatif tentang basis data, aplikasi yang berjalan pada basis data, jaringan komunikasi, kemampuan pemrosesan, dan keterbatasan penyimpanan setiap situs yang ada.

Informasi Basis Data

Informasi basis data yang diperlukan pada proses alokasi adalah selektivitas f_i oleh *query* q_j dan ukuran fragmen f_i, $i = 1,2,3,...,$ l dan $j = 1,2,3,...,$ n. Selektivitas f_i oleh *query* q_j Özsu (2010) dinyatakan dengan $sel_j(f_i)$, sedangkan ukuran f_i dari fragmen dinyatakan dengan $size(f_i)$ dan dihitung dengan menggunakan formula $size(f_i)=card(f_i)*length(f_i)$, dimana $length(f_i)$ adalah panjang tupel fragmen f_i.

Informasi Aplikasi

Dua ukuran penting yang berkaitan dengan aplikasi yang diperlukan pada tahap alokasi adalah akses baca oleh *query* q_i terhadap fragmen f_i saat *query* tersebut dieksekusi yang mana dalam Özsu (2010) dinotasikan dengan RR_{ij}, dan terhadap akses *update* oleh *query* q_i terhadap fragmen f_i saat *query* tersebut dieksekusi yang dinotasikan dengan UR_{ij}. Dalam hal ini perlu didefinisikan dua buah matriks *RM* dan *UM* dengan elemen masing-masing berturut-turut r_{ij} dan u_{ij} sebagai berikut:

$$r_{ij} = \begin{cases} 1, \text{ jika query } q_i \text{ membaca fragmen } f_j \\ 0, \text{ jika query } q_i \text{ tidak membaca fragmen } f_j \end{cases}$$

$$u_{ij} = \begin{cases} 1, \text{ jika query } q_i \text{ meng}\textit{update}\text{ fragmen } f_j \\ 0, \text{ jika query } q_i \text{ tidak meng}\textit{update}\text{ fragmen } f_j \end{cases}$$

Selain itu didefinisikan juga vektor $\boldsymbol{O}$ dari nilai-nilai o_i yang menspesifikasikan situs asal *query* i, dan kendala *response-time*, yaitu *response-time* maksimum yang diperkenankan untuk setiap aplikasi.

Informasi Situs

Kapasitas penyimpanan dan pemrosesan data dari setiap situs komputer perlu diketahui. Kapasitas penyimpanan dan pemrosesan dari setiap komputer situs ini dapat dengan mudah ditentukan berdasarkan estimasi. Biaya satuan untuk penyimpanan data pada sebuah komputer situs s_i dalam Özsu (2010) dinyatakan dengan USC_i. Biaya pemrosesan per satu unit pekerjaan pada situs komputer s_i juga perlu ditentukan. Biaya pemrosesan per satu unit pekerjaan pada situs komputer s_i dinyatakan dengan LPC_i. Ukuran untuk pekerjaan yang dimaksud di sini harus sama dengan satuan yang digunakan untuk *RR* dan *UR*.

Informasi Jaringan

Salah satu informasi yang terkait dengan jaringan yang dibutuhkan pada tahap alokasi adalah biaya komunikasi data. Apabila diasumsikan bahwa sistem terdistribusi yang kita kembangkan menggunakan jaringan yang sederhana, maka biaya komunikasi data dapat didefinisikan sebagai fungsi dari kerangka data (*data frame*). Özsu (2010) menggunakan notasi g_{ij} untuk menyatakan biaya komunikasi data per *frame* data antara situs s_i dan s_j.

C. Model Alokasi

Model alokasi yang digunakan pada penelitian ini adalah model alokasi yang disusun dengan tujuan untuk meminimalkan biaya total pemrosesan dan penyimpanan data dan pada saat yang bersamaan harus dipenuhi pula batasan waktu tanggap yang ada dalam Özsu (2010). Model yang digunakan dalam penelitian ini memiliki rumusan sebagai berikut:

minimize*(Total Cost)*

dengan batasan:

response-time *constraint*
storage *constraint*
processing *constraint*

Variabel keputusan dari model tersebut adalah x_{ij} yang didefinisikan sebagai berikut:

$$x_{ij} = \begin{cases} 1, \text{jika fragment } f_i \text{ disimpan di situs } j_j \\ 0, \text{lainnya} \end{cases}$$

Biaya total (TC) dihitung menggunakan rumus sebagai berikut:

$$\text{TC} = \sum_{\forall s_j \in S} \sum_{\forall f_k \in F} STC_{jk} + \sum_{\forall q_i \in Q} QPC_i$$

STC_{jk} adalah biaya penyimpanan fragment f_k pada situs s_j dan QPC_i adalah biaya pemrosesan *query* oleh aplikasi q_i. Biaya penyimpanan fragment f_k pada situs s_j ditentukan dengan menggunakan formula

$$STC_{jk} = USC_j * size(f_k) * x_{jk}$$

Dengan sumasi ganda akan diperoleh biaya penyimpanan total dari semua fragmen pada semua situs. Özsu (2010) menspesifikasikan biaya pemrosesan *query* QPC_i menjadi dua komponen yaitu biaya pemrosesan *PC* dan biaya transmisi *TC*. Sehingga biaya pemrosesan *query* QPC_i oleh aplikasi q_i adalah

$$QPC_i = PC_i + TC_i$$

Biaya pemrosesan *PC* terdiri dari biaya akses *AC*, biaya pemeliharaan integritas data *IC*, dan biaya pemeliharaan konsistensi data *CC*, sehingga *PC* dapat dituliskan sebagai:

$$PC_i = AC_i + IC_i + CC_i$$

Spesifikasi rinci dari setiap faktor biaya tergantung pada algoritma yang digunakan. Apabila biaya pemrosesan lokal dianggap sama untuk proses akses maupun *update*, maka biaya akses AC mempunyai bentuk:

$$AC_i = \sum_{\forall s_j \in S} \sum_{\forall f_k \in F} (u_{ik} * UR_{ik} + r_{ik} * RR_{ik}) * x_{jk} * LPC_j$$

Suku (UR_{ij} + RR_{ij}) merupakan total jumlah akses dan update oleh query q_i terhadap fragment F_k. Sumasi ganda menghasilkan jumlah akses terhadap semua *fragment* oleh query q_i. Pengalian dengan LPC_k menghasilkan biaya akses pada situs S_j. Sedangkan x_{jk} digunakan untuk memilih hanya nilai biaya untuk situs dimana *fragment* disimpan. Satu hal penting yang perlu diperhatikan, untuk perhitungan biaya akses *AC* di atas, masih bersifat umum. Untuk memperoleh biaya akses yang lebih realistik seharusnya mempertimbangkan bahwa pada proses akses, sebuah *query* yang kompleks biasanya didekomposisi menjadi beberapa *subquery*. Selanjutnya hasil dari masing-masing *subquery* akan digabungkan dengan menggunakan operasi *join* untuk menghasilkan hasil *query* yang selengkapnya, sehingga biaya *join* juga harus dipertimbangkan pada perhitungan biaya akses *AC*. Komponen biaya integritas *IC* dan konsistensi *CC* juga harus dibuat lebih terperinci sesuai dengan relita yang ada.

Biaya transmisi diformulasikan dengan pemikiran bahwa pada proses *update* akan ada dua proses yaitu proses transmisi *update* dari situs asal ke situs-situs dimana *fragment* perlu di*update*, dan proses transmisi konfirmasi dari situs-situs dimana *fragment* perlu di*update* ke situs asal. Sementara itu, pada proses *retrieve* akan ada dua proses yaitu transmisi *query* atau *subquery* dari situs asal ke situs-situs tujuan dan juga akan ada proses transmisi hasil *query* dari situs-situs ke situs asal. Sehingga biaya transmisi pada proses *update* dan proses *retrieve* masing-masing berturut-turut dapat diformulasikan sebagai berikut:

$$TCU_i = \sum_{\forall s_j \in S} \sum_{\forall f_k \in F} u_{ik} * x_{jk} * g_{o(i),j} + \sum_{\forall s_j \in S} \sum_{\forall f_k \in F} u_{ik} * x_{jk} * g_{k,o(i)}$$

$$TCR_i = \sum_{\forall f_k \in F} \min_{s_j \in S} (r_{ij} * x_{jk} * g_{o(i),j} + u_{jk} * x_{jk} * \frac{Sel_i(F_k) * length(f_k)}{fsize} * g_{j,o(i)}$$

Berdasarkan formula TCU_i dan TCR_i, dapat dihitung nilai $TC_i = TCU_i + TCR_i$.

D. Pembatas

Secara umum pembatas waktu tanggap dinyatakan sebagai

$$ET_i \leq RT_i, \forall q_i \in Q$$

yang mana ET_i adalah waktu eksekusi *query i*, dan RT_i adalah waktu tanggap dari aplikasi q_i. Selanjutnya, pembatas penyimpanan secara umum dinyatakan sebagai

$$\sum_{\forall f_k \in F} STC_{jk} \leq SSC_j, \forall s_j \in S$$

yang mana STC_{jk} adalah ukuran fragmen f_k, dan SSC_j adalah total kapasitas penyimpanan situs s_j. Sumasi pada STC_{jk} menghasilkan total ukuran fragmen yang disimpan pada situs s_j. Pembatas yang ketiga adalah pembatas pemrosesan, yang secara umum dinyatakan sebagai

$$\sum_{\forall q_i \in Q} PL_{ik} \leq PC_j, \forall s_j \in S$$

yang mana PL_{jk} adalah beban pemrosesan *query* q_i pada situs s_k, dan PC_j adalah kapasitas pemrosesan situs s_j. Sumasi pada PL_{ik} menghasilkan total beban pemrosesan situs s_j.

E. Metode Penyelesaian

Ada beberapa alternatif metode yang dapat digunakan untuk menentukan solusi optimal dari masalah alokasi ini antara lain metode pemrograman linier dan metode *trial and error*. Pada penelitian ini metode yang digunakan adalah metode trial and error karena metode ini karena kesederhanaannya. Kelemahan dari metode ini adalah tidak ada langkah yang pasti untuk mencapai solusi yang optimal dan tidak ada petunjuk untuk mengetahui apakah solusi yang didapat saat ini sudah optimal atau belum. Jadi solusi yang didapat saat ini merupakan solusi yang optimal relatif.

KESIMPULAN

Metode *trial and error* dapat diterapkan pada *system optimizer* pada basis data terdistribusi. Dengan metode ini, pola alokasi awal fragmen ditentukan berdasarkan salah satu kriteria misalkan biaya komunikasi data atau biaya penyimpanan yang minimal. Selanjutnya *system optimizer* akan

mencoba pola alokasi lainnya untuk memaksimalkan kriteria lainnya misalnya *system throughput*, sehingga biaya secara keseluruhan menjadi minimal.

UCAPAN TERIMA KASIH

Pada kesempatan ini pula, tim peneliti mengucapkan terimakasih yang sebesar- besarnya kepada Pimpinan UNSOED melalui LPPM yang telah memberikan dukungan dana, sehingga penelitian ini dapat terlaksana.

DAFTAR PUSTAKA

Graba, J. 2007. *An Introduction to Network Programming with Java*. Revised Edition. Springer. New York, USA.

Gunadi, K., dan Julistiono, I.K. 2001. Penerapan Arsitektur Multi-Tier dengan DCOM dalam Suatu Sistem Informasi. *Jurnal Informatika* 2(2**)**: 62 – 67

Özsu, M.T. and Valduriez, P. 2010. *Principles of Distributed Database Systems*. Third Edition. Springer. New York, USA.

Ryeng, N.H. 2011. Improving Query Processing Performance in Large Distributed Database Management Systems. *Thesis for the degree of Philosophiae Doctor*. Norwegian University of Science and Technology. Trondheim, Norwegia.

Taylor, R. 2010. *Query Optimization for Distributed Database Systems*. University of Oxford. United Kingdom.