BASIS DATA PARALEL PADA SISTEM MULTIKOMPUTER MENGGUNAKAN POSTGRESQL-MPI

Rizki Arif Firdaus[1)], **Ahmad Ashari**[2)]

[1), 2)] *Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik UGM Yogyakarta*
Jl Grafika 2 Kampus UGM, Sleman, Yogyakarta 55281
Email : rizki_af@mail.ugm.ac.id[1)], ashari@ugm.ac.id[2)]

Abstrak

Basis data merupakan hal yang tidak dapat dipisahkan dalam perkembangan teknologi informasi dan komputer. Perkembangan basis data pun semakin cepat hingga mencapai ukuran yang semakin besar. Semakin besar ukuran basis data menyebabkan waktu pemrosesan query yang dibutuhkan semakin lama. Salah satu alternatif untuk mengurangi waktu pemrosesan query tersebut yaitu dengan pemrosesan paralel. Basis data yang diimplementasikan dalam platform komputasi paralel dinamakan sebagai basis data paralel.

Pada penelitian ini basis data paralel diterapkan pada sistem multikomputer yang mana digunakan delapan komputer (satu komputer sebagai master dan tujuh komputer sebagai slave/pemroses) yang saling terhubung dalam jaringan. Sistem manajemen basis data yang digunakan yaitu PostgreSQL dan memanfaatkan MPI sebagai standard pertukaran pesan antarkomputer dalam jaringan. Query paralel yang diujikan terdiri dari parallel select, parallel max, dan parallel join.

Hasil dari pengujian ini menunjukkan bahwa terjadi peningkatan performa pada query parallel select, parallel max, dan parallel join dengan kondisi sublinear speed up jika dibandingkan dengan pemrosesan query secara serial. Tingkat performa tersebut juga dapat dipertahankan dengan kondisi sublinear scale up. Selain itu, dari grafik perbandingan antara speed up atau scale up dengan jumlah pemroses dapat diperoleh fungsi polinomial dengan interpolasi Lagrange.

Kata kunci: *basis data, basis data paralel, MPI, PostgreSQL, query, scale up, sistem multikomputer, speed up.*

1. Pendahuluan

Sistem paralel berkembang dengan pesatnya seiring meningkatnya permintaan pada kemampuan dan kinerja komputasi yang tinggi. Terdapat dua tipe dasar sistem paralel, yaitu *shared memory multiprocessor system* dan *distributed memory multicomputer system*. Pada sistem *shared memory multiprocessor*, sebuah komputer memiliki beberapa prosesor yang terkoneksi dengan sebuah memori bersama. Sedangkan pada *distributed memory multicomputer system* (untuk selanjutnya disebut dengan sistem multikomputer), beberapa komputer independen dihubungkan satu sama lain melalui jaringan dan perangkat lunaknya menjadi sebuah kesatuan sistem [1].

Sistem multikomputer memerlukan sebuah jaringan komunikasi untuk menghubungkan memori antar prosesor. Setiap prosesor memiliki memori lokal masing-masing. Alamat memori pada satu prosesor tidak tergantung pada prosesor lain, sehingga tidak menggunakan konsep pengalamatan global melalui semua prosesor. Karena setiap prosesor memiliki memori lokal sendiri, operasi dari prosesor berjalan secara independen [2].

Perangkat lunak yang dapat digunakan untuk membangun sistem paralel antara lain *Parallel Virtual Machine* (PVM) yang mulai dikembangan pada tahun 1980-an dan *Message Passing Interface* (MPI) yang mulai dikembangan pada tahun 1990-an. MPI bukanlah bahasa pemrograman. MPI merupakan spesifikasi dari rutin suatu *library* yang dapat dipanggil dari program. MPI menyediakan koleksi rutin komunikasi *point-to-point*, operasi-operasi perpindahan data, komputasi global, dan sinkronisasi. Standar MPI telah berkembang hingga MPI-2 yang memiliki banyak tambahan fitur, seperti proses dinamis, dukungan klien-server, komunikasi satu sisi, I/O paralel, dan fungsi komunikasi *nonblocking* [3].

Penggunaan sistem paralel pun semakin meluas. Salah satunya berkaitan dengan basis data. Basis data merupakan hal yang tidak dapat dipisahkan dalam perkembangan teknologi informasi dan komputer. Perkembangan sistem basis data pun semakin cepat hingga mencapai ukuran yang semakin besar.

Suatu basis data berukuran 10 *terabyte* diproses menggunakan prosesor tunggal dengan kecepatan pemrosesan 1 *megabyte/second* akan memakan waktu pemrosesan selama 120 hari. Untuk mengurangi waktu pemrosesan tersebut menjadi beberapa hari atau bahkan beberapa jam saja, pemrosesan paralel merupakan alternatif jawabannya [4].

Berbagai sistem basis data dapat digunakan untuk mengelola basis data, sebagai contoh PostgreSQL. PostgreSQL merupakan salah satu *Object Relational Database Management System* (ORDBMS) yang bersifat *open source*, yang berarti bahwa *source code* dari PostgreSQL dapat digunakan secara bebas. PostgreSQL mendukung *Structured Query Language* (SQL) yang

memiliki kemampuan tran*sactions*, *subqueries*, *triggers*, dan lain-lain. Selain itu, PostgreSQL juga menyediakan *interface* untuk berbagai bahasa pemrograman seperti Python, C, C++, java, Perl, PHP, dan Tcl [5].

Berdasarkan uraian tersebut, perlu dilakukan penelitian dengan cara merancang dan mengimplementasikan basis data paralel sistem multikomputer menggunakan PostgreSQL dengan MPI-2 sebagai standar pertukaran pesan. Penelitian ini diharapkan dapat menjadi salah satu alternatif dalam pengelolaan basis data secara paralel yang mampu meningkatkan kecepatan pemrosesan dan mengurangi waktu pemrosesan pada sistem multikomputer.

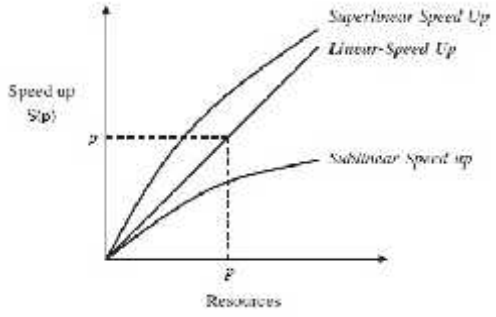
Penelitian tentang basis data paralel pernah dilakukan oleh [6]. Sistem manajemen basis data yang digunakan adalah MySQL dan lebih memfokuskan pada basis data yang memiliki *high availability* dan *load balancing transaction*. Hal ini berbeda dengan penelitian ini yang menggunakan PostgreSQL sebagai sistem manajemen basis data dan lebih memfokuskan pada peningkatan performa pemrosesan *query* yang diukur berdasarkan pada faktor *speed up* dan faktor *scale up*.

Faktor *Speed Up*

Faktor *speed up* mengacu pada peningkatan performa yang diperoleh dari penambahan elemen pemrosesan [4]. Faktor *speed up* dapat dirumuskan dengan persamaan(1).

$$S(p) = \frac{t_s}{t_p} \qquad \text{.....(1)}$$

Pada persamaan di atas, t_s merupakan waktu eksekusi pada prosesor tunggal, sedangkan t_p merupakan waktu eksekusi pada multiprosesor. *Speed up* atau *S(p)* meningkat kecepatannya dengan menggunakan multiprosesor [1]. Grafik *speed up* dapat dilihat pada Gambar 1 berikut.



Gambar 1. *Grafik Speed Up* [4]

Linear speed up mengacu pada peningkatan performa yang tumbuh secara linear dengan penambahan sumber daya (*resources*). *Sublinear speed up* yaitu ketika *speed up* kurang dari *p*. *Superlinear speed up* terjadi ketika *speed up* lebih besar dari *p* dan kondisi ini sangat jarang sekali terjadi [4].

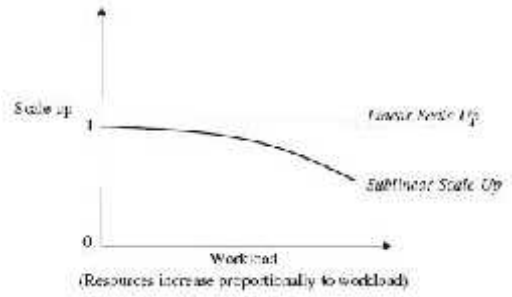
Terdapat suatu pendekatan lain untuk mengukur *speed up* yang dinamakan dengan *Amdahl's Law* (hukum Amdahl). Hukum Amdahl ini menghitung *speed up* dengan memperhatikan adanya faktor persentase komputasi secara serial. Berdasarkan hukum Amdahl, faktor *speed up* dapat dirumuskan dengan persamaan (2).

$$S(p) = \frac{t_s}{ft_s + (1-f)t_s / p} = \frac{p}{1+(p-1)f} \qquad \text{.....(2)}$$

Pada persamaan di atas, *p* menunjukkan banyaknya prosesor dan *f* merupakan persentase bagian serial yang mana semakin besar nilai *f* maka kemungkinan mencapai keadaan ideal *linear speed up* semakin kecil. Jika nilai *f* sama dengan 0%, maka keadaan *linear speed up* tercapai.

Faktor *Scale Up*

Faktor *scale up* mengacu pada kemampuan untuk melakukan proses terhadap pekerjaan yang lebih besar dalam satuan waktu yang sama dengan menyediakan lebih banyak sumber daya (atau dengan meningkatkan *derajat parallelism*). Faktor *scale up* dapat dihitung dengan membandingkan waktu proses satu prosesor pada sistem yang kecil dibandingkan dengan waktu proses multiprosesor pada sistem yang lebih besar. Grafik *scale up* dapat dilihat pada Gambar 3 berikut.



Gambar 3. *Grafik Scale Up* [4]

Jika perbandingannya sama, maka dinamakan *linear scale up* yang mengacu pada kemampuan untuk mengelola level performa yang sama ketika beban pekerjaan dan jumlah prosesor ditambah secara proporsional. Namun jika nilai *scale up* kurang dari 1, maka dinamakan *sublinear scale up.*

***Message Passing Interface* (**MPI**)**

Standar ini muncul dari keinginan memiliki sebuah antarmuka untuk model pertukaran pesan pada komputasi paralel yang dapat digunakan secara luas. Antarmuka ini diharapkan dapat membangun standar pertukaran pesan yang praktis, *portable*, efisien, dan fleksibel [7].

Beberapa operasi utama yang bisa dilakukan oleh standar MPI, yaitu komunikasi langsung antara satu proses dengan proses lainnya (*point-to-point*), melakukan operasi-operasi bersama (*collective*), pengelompokan beberapa proses dalam suatu grup tertentu, pendefinisian lingkungan dan domain komunikasi, pembentukan topologi dari proses-proses tertentu, manajemen lingkungan komunikasi, dan

permintaan data [8]. MPI dapat diimplementasikan pada arsitektur memori terdistribusi dan memori bersama. MPI menyediakan portabilitas kode sumber dari program yang menggunakan pertukaran pesan dalam bahasa pemrograman C atau Fortran pada berbagai macam arsitektur komputer [9].

Interpolasi Lagrange

Interpolasi Lagrange sangat dikenal dalam metode numerik, karena menggunakan fungsi dalam bentuk polinomial. Dari sekumpulan data berpasangan yang ada, dapat diketahui dengan pasti fungsi yang melalui titik-titik tersebut. Titik-titik yang diketahui haruslah merupakan bilangan real, dalam bidang datar. Berdasarkan fungsi ini dapat diprediksikan nilai selanjutnya untuk kasus yang ada. Interpolasi banyak digunakan untuk memprediksi nilai data berpasangan [10]. Jika fungsi yang dicari adalah *f(x)* dan cacah data *n* maka interpolasi Lagrange dapat dirumuskan seperti persamaan (3) berikut.

$$f(x) = \sum_{i=1}^{n} y_i L_i(x) \qquad \text{.....(3)}$$

$$L_i(x) = \begin{cases} 0, x \neq x_i \\ 1, x = x_i \end{cases} \quad \text{dan} \quad L_i(x) = \frac{Q_i(x)}{Q_i(x_i)}$$

dengan :

$$Q_i(x) = (x - x_0)(x - x_1)(x - x_2) \ldots\ldots (x - x_{i-1}) (x - x_{i+1}) \ldots\ldots (x - x_n)$$

2. Pembahasan

Basis data paralel yang menggunakan PostgreSQL-MPI ini ditujukan untuk menghasilkan sistem paralel yang mendukung basis data sehingga pemrosesan *query* pada basis data berjalan secara paralel. Sistem ini diharapkan mampu meningkatkan performa pemrosesan *query* tersebut.

Kebutuhan fungsional yang dimiliki sistem ini adalah sistem mampu menerima input berupa skrip pemrograman bahasa C yang di dalamnya terdapat *query* pada basis data PostgreSQL. Skrip pemrograman yang berisi *query* tersebut selanjutnya dieksekusi dengan MPI untuk dilakukan pemrosesan *query* secara paralel. Pengguna cukup melakukan sebuah *query* (instruksi) kemudian *query* tunggal tersebut dibagi-bagi ke dalam *subquery-subquery*. Masing-masing *subquery* diproses pada prosesor yang berbeda dan hasil *subquery* tersebut digabungkan untuk ditampilkan pada pengguna.

Basis data paralel ini menggunakan sistem multikomputer dengan arsitektur memori terdistribusi. Arsitektur ini secara mutlak memerlukan sebuah jaringan komunikasi untuk menghubungkan memori antar prosesor. Topologi jaringan yang digunakan adalah topologi *star* dengan sebuah *switch* sebagai penghubungnya.

Rancangan sistem memerlukan delapan buah komputer. Satu komputer berfungsi sebagai *master* yang merupakan antarmuka bagi pengguna. Pengguna dapat memasukkan input dan melakukan *monitoring* sistem melalui komputer ini, tetapi pada komputer ini tidak terjadi pemrosesan *query*. Pemrosesan *query* dilakukan secara paralel pada tujuh komputer lainnya. Komputer yang digunakan sebagai *master* diberi nama *node1*. Ketujuh komputer lainnya yang berfungsi sebagai *slave* masing-masing memiliki nama *node2*, *node3*, *node4*, *node5*, *node6*, *node7*, dan *node8*. Semua komputer tersebut memiliki spesifikasi *hardware* yang sama yaitu prosesor Intel Pentium Dual Core, memori 1 GB, media penyimpanan 40 GB, dan kartu jaringan Fast Ethernet 10/100 Mbps.

Pengguna dapat melakukan *query* untuk pengaturan tabel pada basis data yang akan digunakan untuk pengujian basis data paralel. Pengaturan tabel terdiri dari *query insert* dan *delete*. Terdapat dua tabel yang digunakan, yaitu ikkie dan ippie. Tabel ikkie digunakan untuk melakukan operasi pengaturan tabel, *select*, *max*, dan *join*. Sedangkan tabel ippie digunakan untuk melakukan operasi pengaturan tabel dan *join*.

Pengujian sistem digunakan untuk mengukur kualitas performa pemrosesan query secara paralel. Pengujian sistem ini dibagi menjadi dua bagian, yaitu pengujian sistem untuk mengukur besarnya *speed up* dan pengujian sistem untuk mengukur besarnya *scale up*. Pada setiap pengujian tersebut dijalankan masing-masing tiga macam *query* secara serial dan paralel, yaitu *query select*, *query max*, dan *query join*. Percobaan dilakukan sebanyak 25 kali pada masing-masing pengujian dan diukur dalam satuan *milliseconds* (ms).

Pengujian *Speed Up*

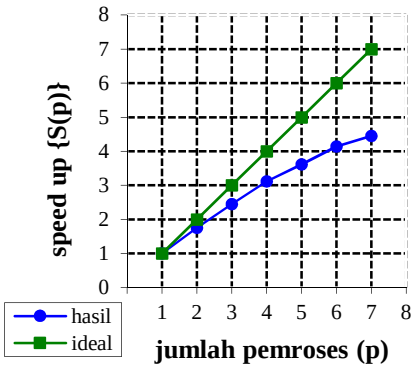
Pada pengujian ini digunakan studi kasus dengan kondisi jumlah *record* sebanyak 1.000.000 *record* yang diproses menggunakan 1 sampai dengan 7 pemroses. Performa sistem meningkat jika kondisi minimal *sublinear speed up* dengan nilai *f* kurang dari 100% terpenuhi. Jika kondisi *sublinear speed up* dicapai dengan nilai *f* sama dengan 100% maka tidak terjadi peningkatan performa sistem. Sedangkan performa sistem dikatakan menurun jika kondisi *sublinear speed up* dengan nilai *f* lebih dari 100% dicapai. Hasil pengujian *speed up* pada *parallel select* dapat dilihat pada Tabel 1 berikut.

Tabel 1. *Hasil Pengukuran Speed Up Parallel Select*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Speed Up*	*f* (%)
1000000	1	2139,52	1,0000	-
	2	1219,08	1,7550	13,96
	3	873,04	2,4507	11,21
	4	686,68	3,1157	9,46
	5	592,72	3,6097	9,63
	6	516,68	4,1409	8,98
	7	480,36	4,4540	9,53

Persentase nilai *f* cenderung sama dengan rata-rata 10,46%. Nilai *f* rata-rata ini menunjukkan bahwa sistem mengalami peningkatan performa dengan kondisi

sublinear speed up. Grafik perbandingan antara faktor *speed up* {*S(p)*} dengan jumlah pemroses (*p*) dapat dilihat pada Gambar 4 berikut.



Gambar 4. *Grafik Speed Up pada Parallel Select*

Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,1.755), (3, 2.4507), (4, 3.1157), (5, 3.6097), (6, 4.1409), dan (7, 4.454). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (4) berikut.

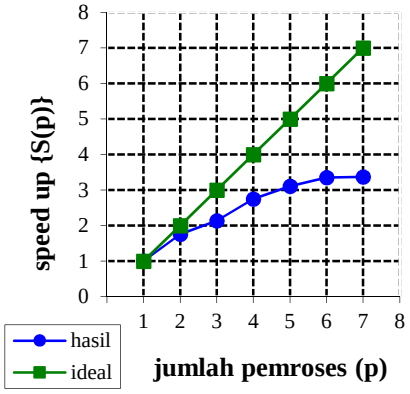
$$S(p) = -\frac{1.33}{720}p^6 + \frac{10.34}{240}p^5 - \frac{56.86}{144}p^4 + \frac{10.80}{144}p^3 - \frac{1.08}{120}p^2 + \frac{3.45}{60}p - 1.86 \quad \text{.....(4)}$$

Hasil pengujian *speed up* pada *parallel max* dapat dilihat pada Tabel 2 berikut.

Tabel 2. Hasil *Pengukuran Speed Up Parallel Max*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Speed Up*	*f* (%)
1000000	1	394,92	1,0000	-
	2	224,96	1,7555	13,93
	3	184,72	2,1379	20,16
	4	143,60	2,7501	15,15
	5	127,00	3,1096	15,20
	6	117,76	3,3536	15,78
	7	117,16	3,3708	17,95

Persentase nilai *f* cenderung sama dengan rata-rata 16,36%. Grafik perbandingan antara *speed up* dengan jumlah pemroses dapat dilihat pada Gambar 5 berikut.



Gambar 5. *Grafik Speed Up pada Parallel Max*

Nilai *f* rata-rata dan grafik tersebut menunjukkan bahwa sistem mengalami peningkatan performa dengan kondisi *sublinear speed up*. Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,1.7555), (3, 2.1379), (4, 2.7501), (5, 3.1096), (6, 3.3536), dan (7, 3.3708). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (5) berikut.

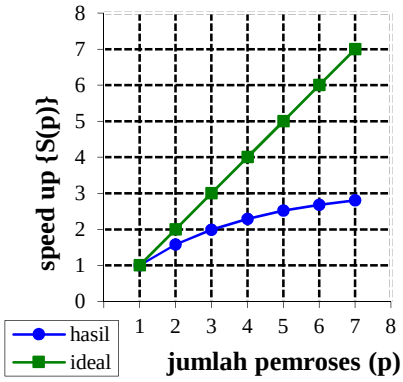
$$S(p) = -\frac{2.57}{720}p^6 + \frac{21.42}{240}p^5 - \frac{127.27}{144}p^4 + \frac{1.56}{144}p^3 - \frac{4.07}{120}p^2 + \frac{12.32}{60}p - 6.09 \quad \text{.....(5)}$$

Hasil pengujian *speed up* pada *parallel join* dapat dilihat pada Tabel 3 berikut.

Tabel 3. *Hasil Pengukuran Speed Up Parallel Join*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Speed Up*	*f* (%)
1000000	1	4670,84	1,0000	-
	2	2956,08	1,5801	26,58
	3	2352,16	1,9858	25,54
	4	2039,48	2,2902	24,89
	5	1851,80	2,5223	24,56
	6	1741,48	2,6821	24,74
	7	1666,84	2,8022	24,97

Persentase nilai *f* cenderung sama dengan rata-rata 25,21%. Nilai *f* rata-rata ini menunjukkan bahwa sistem mengalami peningkatan performa dengan kondisi *sublinear speed up*. Grafik perbandingan antara faktor *speed up* {*S(p)*} dengan jumlah pemroses (*p*) dapat dilihat pada Gambar 6 berikut.



Gambar 6. *Grafik Speed Up pada Parallel Join*

Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,1.5801), (3, 1.9858), (4, 2.2902), (5, 2.5223), (6, 2.6821), dan (7, 2.8022). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (6) berikut.

$$S(p) = \frac{0.05}{720}p^6 - \frac{0.29}{240}p^5 + \frac{1.09}{144}p^4 - \frac{0.90}{144}p^3 - \frac{17.77}{120}p^2 + \frac{59.29}{60}p + 0.16 \quad \text{.....(6)}$$

Berdasarkan data-data pengujian dapat disimpulkan bahwa performa sistem meningkat dengan kondisi *sublinear speed up* pada *query select*, *max*, dan *join*.

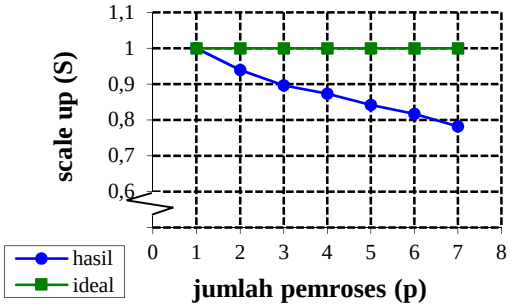
Pengujian *Scale Up*

Pada pengujian ini digunakan studi kasus dengan kondisi beban pekerjaan antara 1.000.000 sampai dengan 7.000.000 *record* yang diproses menggunakan 1 sampai dengan 7 pemroses. Pengujian ini mencapai kondisi ideal jika nilai faktor *scale up* sama dengan 1. Hasil pengujian *scale up* pada *parallel select* dapat dilihat pada Tabel 4.

Tabel 4. *Hasil Pengukuran Scale Up Parallel Select*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Scale Up*
1000000	1	2139,52	1,0000
2000000	2	2278,56	0,9390
3000000	3	2387,80	0,8960
4000000	4	2450,64	0,8730
5000000	5	2542,24	0,8416
6000000	6	2618,32	0,8171
7000000	7	2736,64	0,7818

Nilai terbesar diperoleh pada kondisi 2.000.000 *record* dan 2 pemroses dengan besaran nilai yaitu 0,939, sedangkan nilai terkecil diperoleh pada kondisi 7.000.000 *record* dan 7 pemroses dengan besaran nilai yaitu 0,7818. Grafik perbandingan antara nilai faktor *scale up* dengan jumlah pemroses dapat dilihat pada Gambar 7 berikut.



Gambar 7. *Grafik Scale Up pada Parallel Select*

Gambar 7 menunjukkan bahwa faktor *scale up* pada *parallel select* berada pada kondisi *sublinear scale up*. Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,0.939), (3, 0.896), (4, 0.873), (5, 0.8416), (6, 0.8171), dan (7, 0.7818). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (7) berikut.

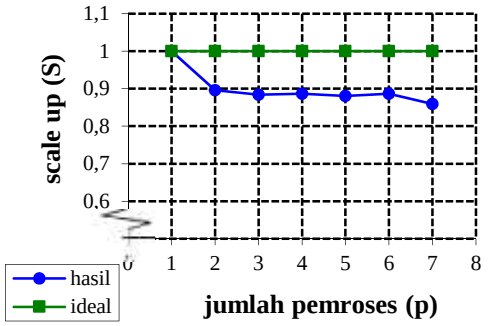
$$S(p) = -\frac{0.15}{720}p^6 + \frac{1.20}{240}p^5 - \frac{6.79}{144}p^4 + \frac{31.60}{144}p^3 - \frac{61.97}{120}p^2 + \frac{31.06}{60}p + 0.82 \quad \text{.....(7)}$$

Hasil pengujian *scale up* pada *parallel max* dapat dilihat pada Tabel 5 berikut.

Tabel 5. *Hasil Pengukuran Scale Up Parallel Max*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Scale Up*
1000000	1	394,92	1,0000
2000000	2	440,72	0,8961
3000000	3	446,68	0,8841
4000000	4	445,56	0,8863
5000000	5	448.72	0,8801
6000000	6	445,56	0,8863
7000000	7	459,76	0,8590

Nilai terbesar diperoleh pada kondisi 2.000.000 *record* dan 2 pemroses dengan besaran nilai yaitu 0,8961, sedangkan nilai terkecil diperoleh pada kondisi 7.000.000 *record* dan 7 pemroses dengan besaran nilai yaitu 0,8590. Grafik perbandingan antara nilai faktor *scale up* dengan jumlah pemroses dapat dilihat pada Gambar 8 berikut.



Gambar 8. *Grafik Scale Up pada Parallel Max*

Gambar 8 menunjukkan bahwa faktor *scale up* pada *parallel max* berada pada kondisi *sublinear scale up*. Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,0.8961), (3, 8841), (4, 8863), (5, 0.8801), (6, 0.8863), dan (7, 0.859). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (8) berikut.

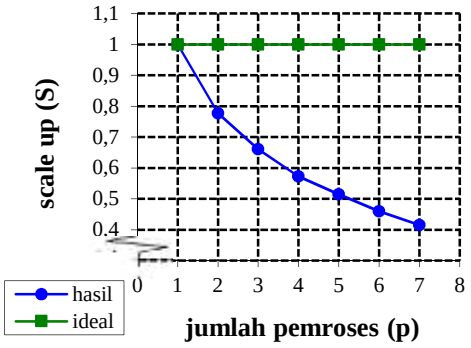
$$S(p) = -\frac{0.10}{720}p^6 + \frac{0.67}{240}p^5 - \frac{2.90}{144}p^4 + \frac{8.10}{144}p^3 + \frac{0.48}{120}p^2 - \frac{17.08}{60}p + 1.24 \quad \text{.....(8)}$$

Hasil pengujian *scale up* pada *parallel join* dapat dilihat pada Tabel 6 berikut.

Tabel 6. *Hasil Pengukuran Scale Up Parallel Join*

Jumlah *Record*	Jumlah *Node*	Waktu Rata-rata (*ms*)	*Scale Up*
1000000	1	4670,84	1,0000
2000000	2	6016,88	0,7763
3000000	3	7068,68	0,6608
4000000	4	8153,28	0,5729
5000000	5	9078,52	0,5145
6000000	6	10159,36	0,4598
7000000	7	11236,60	0,4157

Nilai terbesar diperoleh pada kondisi 2.000.000 *record* dan 2 pemroses dengan besaran nilai yaitu 0,939, sedangkan nilai terkecil diperoleh pada kondisi 7.000.000 *record* dan 7 pemroses dengan besaran nilai yaitu 0,7818. Grafik perbandingan antara nilai faktor *scale up* dengan jumlah pemroses dapat dilihat pada Gambar 9 berikut.



Gambar 9. *Grafik Scale Up pada Parallel Join*

Gambar 9 menunjukkan bahwa faktor *scale up* pada *parallel join* berada pada kondisi *sublinear scale up.* Pada grafik tersebut cacah datanya berjumlah tujuh titik, yaitu (1, 1), (2,0.7763), (3, 0.6608), (4, 0.5729), (5, 0.5145), (6, 0.4598), dan (7, 0.4157). Dari cacah tersebut digunakan interpolasi Lagrange untuk memperoleh suatu fungsi *S(p)*. Hasil perhitungan interpolasi Lagrange dapat dilihat pada persamaan (9) berikut.

$$S(p) = \frac{0.17}{720}p^6 - \frac{1.41}{240}p^5 + \frac{8.45}{144}p^4 - \frac{43.20}{144}p^3 + \frac{101.57}{120}p^2 - \frac{82.51}{60}p + 1.78 \quad \text{.....(9)}$$

Berdasarkan data-data pengujian dapat disimpulkan bahwa sistem mampu mempertahankan tingkat performanya dengan kondisi *sublinear scale up* pada *query select*, *max*, dan *join.*

3. Kesimpulan

Basis data paralel telah berhasil diujikan pada sistem multikomputer menggunakan skrip bahasa C dan dapat diimplementasikan dengan baik pada PostgreSQL menggunakan antarmuka pertukaran pesan MPI-2. Hasil pengujian *speed up* menunjukkan bahwa basis data paralel mampu meningkatkan performanya dengan kondisi *sublinear speed up* pada semua *query parallel* yang diujikan. Selain itu, hasil pengujian *scale up* menunjukkan bahwa basis data paralel mampu mengelola tingkat performanya dengan kondisi *sublinear scale up* pada semua *query parallel* yang diujikan.

Berdasarkan hasil penelitian tersebut, perlu dilakukan pengembangan ke arah pembangunan sistem basis data paralel yang sepenuhnya yang berarti bahwa semua pemrosesan query dalam sistem tersebut dilakukan secara paralel. Di samping itu, perlu juga dilakukan implementasi pada basis data yang sesungguhnya dengan ukuran dan tingkat kompleksitas (struktur tabel maupun *query*) yang lebih tinggi.

Daftar Pustaka

[1] B. Wilkinson and M. Allen, *Parallel Programming Techniques & Applications Using Networked Workstations and Parallel Computers*, 2nd edition, Pearson Education, Inc., 2004.
[2] T. Maryanto, "Aplikasi *High Performance Computing* (HPC) Menggunakan *Parallel Processing* untuk *Computational Fluid Dynamics*," Surabaya : Institut Teknologi Surabaya, 2008.
[3] H. El-Rewini and M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, New Jersey : John Wiley & Sons, Inc., 2005.
[4] D. Taniar, C. H. C. Leung, W. Rahayu, S. Goel, *High-Performance Parallel Database Processing and Grid Databases*, John Wiley & Sons, Inc., 2008.
[5] E. Utami and S. Raharjo, "Koneksi Database PostgreSQL dengan Bahasa C menggunakan Embedded SQL," 2003.
[6] D. Setiadi, "*Parallel Database* dengan menggunakan MySQL *Cluster* sebagai *High Availability Database* dan *Load Balancing Transaction*," Yogyakarta : Universitas Gadjah Mada, 2007.
[7] MPI Forum, *MPI : A Message Passing Interface Standard Version 2.1*, Tennessee : University of Tennessee, 2008.
[8] A. Bustamam, H. Suhartanto, T. Basaruddin, "Implementasi Metode Iteratif Paralel Implisit Multistep Runge-Kutta pada Sistem Paralel MPI-Linux," *in Proc. Lokakarya Komputasi dalam Sains dan Teknologi NuklirXIII BATAN*, July 3-4, 2002.
[9] N. MacDonald, E. Minty, J. Malard, T. Harding, S. Brown, M. Antonioletti, *Writing Message Passing Parallel Programs with MPI*, Edinburgh Parallel Computing Centre, Edinburgh : The University of Edinburgh, 1995.
[10] Krisnawati, "Implementasi Interpolasi Lagrange untuk Prediksi Nilai Data Berpasangan dengan Menggunakan Matlab," *in Proc. Seminar Nasional Teknologi*, November 24, 2007.

Biodata Penulis

Rizki Arif Firdaus, memperoleh gelar Sarjana Komputer (S.Kom), Program Studi Ilmu Komputer Universitas Gadjah Mada Yogyakarta, lulus tahun 2010. Saat ini menjadi Mahasiswa di Program Pasca Sarjana Magister Teknik Informatika Universitas Gadjah Mada Yogyakarta.

Ahmad Ashari, memperoleh gelar Sarjana Sains (S.Si) Program Studi Elektronika dan Instrumentasi Jurusan Fisika Universitas Gadjah Mada Yogyakarta, lulus tahun 1988. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Ilmu Komputer Universitas Indonesia, lulus tahun 1992. Memperoleh gelar Doktor (Dr. techn) Bidang Informatics, Vienna University of Technology, Austria. Saat ini menjadi Dosen di Program Pascasarjana Magister Teknik Informatika Universitas Gadjah Mada Yogyakarta.