Penerapan Algoritma GRAC (Graph Algorithm Clustering) untuk Graph Database Compression

Implementation of GRAC Algorithm (Graph Algorithm Clustering) in Graph Database Compression

I Gusti Bagus Ady Sutrisna[1], Kemas Rahmat Saleh W, ST., M.Eng. [2], Alfian Akbar Gozali, ST., MT.[3].

Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom
[1]igustibagus.as@gmail.com, [2]kemas@bif.telkomuniversity.ac.id, [3]alfian@tass.telkomuniversity.ac.id

Abstrak

Graph Database merupakan representasi dari pemodelan suatu koleksi data ke dalam bentuk Node dan Edge. Graph Database adalah salah satu metode implementasi dari NoSQL (Not Only SQL), yaitu sistem database yang berguna untuk penyimpanan data dalam jumlah besar dan direpresentasikan dalam bentuk graph, sehingga data memiliki aksesibilitas yang tinggi. Namun data yang disimpan dalam pemrosesan Graph Database masih belum efisien dalam hal penyimpanan data. Penyimpanan jutaan ataupun milyaran Nodes dan Edges memerlukan pengompresan dalam kebutuhan penyimpanan data.

Dalam penelitian ini, kompresi Graph Database yang akan dilakukan adalah dengan menggunakan GRAC(Graph Algorithm Clustering). Graph Database yang digunakan yaitu suatu data yang berisikan data kolaborasi antar penulis jurnal ilmiah. Dalam GRAC(Graph Algorithm Clustering), Clustering yang digunakan adalah Hierarchical Clustering. Metode Hierarchical Clustering adalah suatu metode dalam Clustering yang akan mengcluster Node menjadi bentuk Cluster Node secara hirarki. Dalam pembuatan cluster yang hirarki, strategi yang dipakai adalah Agglomerative dimana setiap Node nantinya akan digabungkan menjadi satu cluster. Untuk mendapatkan strategi Agglomerative yang efektif dan efisien maka akan dihitung jarak maximum antar cluster yang biasa disebut Complete Linkage Clustering. Setiap Node terlebih dahulu dihitung Jaccard indexnya untuk mendapatkan bobot jarak antar Node. Penggunaan Hierarchical Clustering adalah untuk membentuk Cluster Node yang memiliki kesamaan tetangga. Cluster Node nantinya akan dihubungkan dengan Cluster Edge dimana, Cluster Edge didapatkan melalui pencarian secara greedy pada setiap hubungan Cluster Node yang mengabstaksi Edge paling banyak.

Dengan menerapkan GRAC (Graph Algorithm Clustering) dengan menggunakan metode Hierarchical Clustering yang membentuk cluster yang hirarki, maka akan menghasilkan graph database yang bersifat lossless serta terkompres dengan baik.

Kata Kunci: **graph database, compression, graph algorithm clustering, Hierarchical Clustering, agglomerative, complete linkage clustering, jaccard index, greedy, scientific journal authors**

Abstract

Graph Database is a representation of data collection that formed in to Node and Edge. Graph Database is a one of implementation method in NoSQL (Not Only SQL), database system that can store big data and represent in the form of graph, that having a great performance. But, the data that store in the process of Graph Database is not efficient in that way. Storing milion of Node and Edge need compression to store the data.

In this research, Graph Database Compression using the GRAC(Graph Algorithm Clustering) method. Graph Database use the data that carrying the collaboration of scientific journal author. In the GRAC(Graph Algorithm Clustering) method, Clustering using Hierarchical Clustering. Hierarchical Clustering method is the method that cluster Node in to Cluster Node Hierarchically. The form of Hierarchical Clustering, strategy that uses is Agglomerative that every node merge into one Cluster. Complete Linkage Clustering is using in the Agglomerative strategy for count the maximum distance of Cluster. Every Node is counted the jaccard index for the distance index. Hierarchical Clustering is used to create the Cluster Node that have the same neighbourhood. Cluster Node then connect to the Cluster Edge that Cluster Edge known from greedy searching in the relasionship of Cluster Node that having the most neigbourhood.

Using the GRAC (Graph Algorithm Clustering) with Hierarchical Clustering method that create cluster Hierarchically, then create the graph database Lossless.

Kata Kunci: **graph database, compression, graph algorithm clustering, Hierarchical Clustering, agglomerative, complete linkage clustering, jaccard index, greedy, scientific journal authors**

1. Pendahuluan

Relational database merupakan sistem penyimpanan dan pengambilan data yang telah populer dan mendominasi selama lebih dari tiga dekade. Banyak aplikasi yang menggunakan relational database untuk penyimpanan data. Relational database dapat bekerja dengan baik apabila jumlah data sedikit dan memiliki data terstruktur[2]. Ketika terjadinya peningkatan jumlah data dan berbagai pemrosesan data maka relational database dengan skema yang kaku sangat tidak cocok untuk kasus data semi terstruktur bahkan tidak terstruktur[2]. Kasus data tidak terstruktur dan semi terstruktur memiliki fleksibilitas dalam hal pemrosesan data seperti halnya dengan kasus social network dengan interconnected data.

Salah satu solusi yang digunakan untuk mengatasi hal tersebut adalah menggunakan Graph Database. Graph Database adalah salah satu metode implementasi dari NoSQL (Not Only SQL) yaitu sistem database yang berguna menyimpan data dalam jumlah besar dan direpresentasikan ke dalam graph, berbentuk Node dan Edge[1]. Hal ini dilakukan karena Node dan Edge memberi peluang untuk ekstraksi informasi antar user. Kelebihan Graph Database adalah dalam hal pencarian data bisa dilakukan secara tranversal dengan setiap relasi direpresentasikan dengan suatu Edge yang menghubungkan Node-Node yang berelasi, sehingga waktu pemrosesan dapat dilakukan dengan efektif[1]. Graph Database memiliki penyimpanan yang tidak efisien. Ini dikarenakan Edge yang merepresentasikan seluruh relasi yang ada di dalam Graph Database menyebabkan ukuran yang besar dan tidak efisien. Keberadaan banyak Edge merupakan fokus penelitian agar Graph Database dapat di kompresi sehingga penyimpanan dari Graph Database menjadi lebih kecil dari yang sebelumnya dan efisien.

Dari sekian banyak algoritma Graph Database Compression yang ada, algoritma GRAC(Graph Algorithm Clustering) ini merupakan algoritma yang digunakan untuk mengkompresi Graph Database. Graph Database yang akan dilakukan adalah dengan menggunakan GRAC(Graph Algorithm Clustering). Graph Database yang digunakan yaitu suatu data yang berisikan data kolaborasi antar penulis jurnal ilmiah. Dalam GRAC(Graph Algorithm Clustering), Clustering yang digunakan adalah Hierarchical Clustering. Metode Hierarchical Clustering adalah suatu metode dalam Clustering yang akan mengcluster Node menjadi bentuk Cluster Node secara hirarki[5]. Dalam pembuatan Cluster yang hirarki, strategi yang dipakai adalah Agglomerative dimana setiap Node nantinya akan digabungkan menjadi satu Cluster. Untuk mendapatkan strategi Agglomerative yang efektif dan efisien maka akan dihitung jarak maximum antar Cluster yang biasa disebut Complete Linkage Clustering[4]. Setiap Node terlebih dahulu dihitung Jaccard indexnya untuk mendapatkan bobot jarak antar Node[6]. Penggunaan Hierarchical Clustering adalah untuk membentuk Cluster Node yang memiliki kesamaan tetangga. Cluster Node nantinya akan dihubungkan dengan Cluster Edge dimana, Cluster Edge didapatkan melalui pencarian secara greedy pada setiap hubungan Cluster Node yang mengabstaksi Edge paling banyak.

Dengan menerapkan GRAC (Graph Algorithm Clustering) dengan menggunakan metode Hierarchical Clustering yang membentuk Cluster yang hirarki, maka akan menghasilkan Graph Database yang bersifat lossless serta terkompres dengan

2. Dasar Teori / Perancangan Sistem

2.1. Graph Database

Graph Database adalah sebuah database yang menggunakan struktur graph dengan Node, Edge, dan Property untuk merepresentasikan dan menyimpan data. Graph merupakan cabang ilmu dari matematika yang dikenal mempunyai keterkaitan aplikasi dengan banyak disiplin ilmu lainnya.

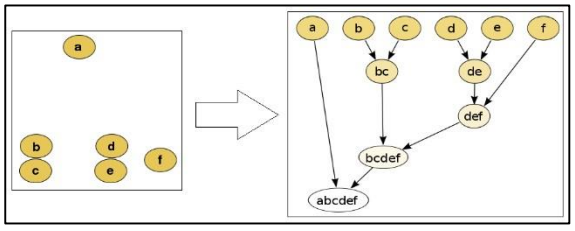
Teori graph merupakan cabang matematika dimulai oleh Euler pada awal tahun 1736. Secara konsep, sebuah graph dibentuk oleh vertex(Node atau simpul) dan Edge(lines atau sisi) yang menghubungkan vertex(Node atau simpul). Secara formal, sebuah graph adalah sepasang dari elemen G=(V,E) dimana V adalah sekumpulan simpul dan E adalah sekumpulan sisi yang dibentuk oleh sepasang simpul. Cara umum untuk menggambar graph adalah menggambar sebuah titik untuk setiap vertex dan menggabungkan kedua titik tersebut dengan sebuah garis. Tidak ada yang diaggap tidak relevan hanya bagaimana menggambar sebuah titik dan garis. Hal yang terpenting adalah informasi pasangan vertex mana yang membentuk sebuah Edge dan mana yang tidak[1].

Graph database adalah database schema-less yang menggunakan struktur data graph bersama dengan Nodes, Edges dan property tertentu untuk menggambarkan data. Node-Node menggambarkan entitas-entitas seperti people, business dan lain-lain. Properti-properti menunjukkan informasi terkait yang berhubungan dengan Node. Di sisi lain, Edge menghubungkan sebuah Node ke Node lainnya[1].
Dalam sebuah graph, derajat sebuah Node x adalah jumlah Edge yang terhubung dengan Node tersebut. Karena pada graph tak berarah sebuah Edge bisa berarah keluar dari sebuah Node atau masuk dari sebuah Node, maka derajat sebuah Node pada graph tak berarah dikalikan dengan 2. Derajat rata-rata sebuah graph merupakan jumlah Edge dalam sebuah graph dibagi dengan jumlah nodenya. Perhitungan ini dapat digunakan untuk mengetahui kepadatan suatu

graph. Pada graph tak berarah, derajat rata-ratanya dapat dihitung.

2.2. GRAC (Graph Algorithm Clustering)

GRAC(Graph Algorithm Clustering) yang di gunakan adalah hierarchical clustering. Hierarchical Clustering merupakan salah satu dari banyak algoritma clustering. Hierarchical Clustering biasa disebut juga Hierarchical Cluster Analysis atau HCA merupakan suatu metode dalam pembuatan Cluster dimana akan membentuk data hirarki dari Cluster[2]. Strategi dalam penggunaan Hierarchical Clustering yaitu Agglomerative dan Divisive. Agglomerative merupakan strategi pembentukan "bottom up", yaitu setiap data satuan membentuk Cluster itu sendiri dan bergabung menjadi satu Cluster dengan Cluster yang lainnya secara hirarki[3]. Berbeda dengan Agglomerative, Divisive merupakan kebalikannya. Divisive merupakan strategi pembentukan "top down", dimana dimulai dari satu Cluster menjadi banyak Cluster yang lainnya dan terbentuk secara hirarki. Biasanya perubahaan penggabungan dan pemisahan ini dicari menggunakan greedy. Dalam strategi agglomerative, terdapat tiga perhitungan dalam penentuan jarak antar



Gambar 1. Contoh sederhana penggunaan Agglomerative Hierarchical Clustering

Cluster, yaitu maximum biasa disebut complete linkage clustering, minimum biasa disebut single linkage clustering dan dengan rata-rata biasa disebut UPGMA. Dalam penggunaan GRAC(Graph Algortihm Clustering), Hierarchical Clustering menggunakan Strategi Agglomerative dan Complete Linkage clustering. Berikut merupakan contoh sederhana dari Agglomerative Hierarchical Clustering :

Pada contoh sederhana diatas, enam data {a}, {b}, {c}, {e} dan {f} akan dibentuk secara hirarki. Ke enam data tersebut akan digabung menjadi satu Cluster berdasarkan kedekatan jarak. Dari enam data akan menjadi satu Cluster {a,b,c,d,e,f} yang dibentuk secara hirarki.

Dari contoh sederhana diatas, maka akan di aplikasikan dalam Graph Database. Graph Database merupakan representasi database dengan bentuk graph. Dalam Graph Normal, Node digambarkan dengan lingkaran, serta garis penghubung dua Node yang disebut Edge. Di setiap Node akan memiliki ID Node. Pada Cluster Graph, terdepat entitas Cluster Node yang juga berbentuk lingkaran, di mana di dalamnya bisa terdapat Node atau Cluster Node lain, serta Cluster Edge, yang merupakan garis penghubung dua Cluster Node.

Jika diberikan graph $G = (V, E)$ dimana $V = \{v_0, \ldots, v_n\}$ adalah himpunan Node dan $E \subseteq V \times V$ adalah himpunan Edge, maka clustter graph $G' = (V', E')$ adalah sebuah graph adalah himpunan kuasa dari Cluster Nodes $V' \subseteq P(V)$ yang dihubungkan satu sama lain oleh Cluster Edge $E' \subseteq V' \times V'$. Sehingga Cluster Graph mengandung himpunan kuasa dari Node dan himpunan kuasa dari Edge graph G. Jika dua Cluster Node dihubungkan dengan sebuah Cluster Edge, maka semua Node yang ada di dalam Cluster Node pertama terhubung dengan semua Node yang ada di Cluster Node kedua.

2.3. Penerapan GRAC (Graph Algorithm Clustering)

Tujuan utama dari penerapan algoritma GRAC(Graph Algorithm Clustering) adalah untuk membuat Cluster Node dan Cluster Edge yang di dapat dari hasil Cluster dengan metode Hierarchical Clustering. Sebelum dapat menghasilkan Graph Database Compression terlebih dahulu dibentuk Cluster Graph dimana Cluster Graph berisikan Cluster Node dan Cluster Edge. Langkah pertama dalam penerapan Algoritma GRAC(Graph Algoritm Clustering) yaitu membentuk sebuah Cluster yang berikan Node. Sebuh Cluster Node yang memiliki Node-Node ini, mempunyai kesamaan tetangga. Oleh karena itu pembentukan Cluster Node didapatkan melalui proses hierarchical clustering. Tetapi sebelum dapat melakukan proses hierarchical clustering, akan dilakukan terlebih dahulu perhitungan Jaccard Index.

Untuk melakukan Hierarchical Clustering, perlu ditentukan dahulu matrix yang akan digunakan untuk menghitung jarak antar Cluster. Dalam kasus ini, jarak antar Cluster ditentukan berdasarkan kesamaan Node tetangga. Dua Node yang memiliki banyak tetangga yang sama dianggap memiliki jarak yang kecil. Sementara itu dua Node yang hanya memiliki sedikit tetangga yang sama, memiliki jarak yang besar. Perhitungan jarak antar Node ini dilakukan menggunakan Jaccard Index atau Jaccard Similarity Coefficient[6].

Cara kerja dari Jaccard Index yaitu untuk menghitung kesamaan dan perbedaan antara dua buah himpunan. Dalam kasus ini, yang dibandingkan adalah himpunan Node tetangga pada dua buah Node. Diberikan contoh Apabila ada dua buah Node a dan b, yang masing-masing memiliki himpunan tentangga A dan B, maka nilai jaccard dari Node a dan b adalah:

$$J(a,b) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Gambar 2. Metode Jaccard Index

Setelah menyusun array berisi Jaccard Index, langkah berikutnya adalah melakukan hierarchical clustering. Pada Agglomerative

Hierarchical Clustering, setiap Node singular dianggap sebagai sebuah Cluster. Kemudian pasangan Cluster yang paling kecil nilai jaccard-nya, x dan y, digabungkan menjadi sebuah Cluster xy. Karena Hierarchical Cluster pada penelitian ini menggunakan metode complete linkage clustering, maka nilai jaccard untuk Cluster xy terhadap Cluster lain, misalnya Cluster z, didapatkan dari nilai terbesar antara J(x,z) dan J(y,z).

Hal ini diulang seterusnya hingga pada akhirnya hanya terdapat satu Cluster. Bila n merupakan jumlah Node yang ada, maka jumlah pengulangan (looping) yang dilakukan untuk akhirnya menyisakan satu Cluster adalah n-1.

Setelah terhimpun Cluster Node, langkah berikutnya adalah mencari setiap kombinasi pasangan Cluster Node yang memungkinkan dari Cluster Node – Cluster Node yang ada di himpunan Cluster Node. Setiap kombinasi tersebut akan dihubungkan dengan sebuah Cluster Edge. Sebuah Cluster Node x disebut bisa berpasangan dengan Cluster Node y apabila setiap Node di dalam x memiliki Edge dengan setiap Node dalam y. Apabila x dan y berpasangan, maka akan dibentuk sebuah Cluster Edge di antara kedua Cluster Node tersebut.

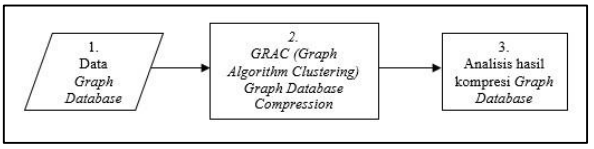
Setelah terhimpun semua Cluster Edge, langkah berikutnya adalah melakukan pencarian greedy. Disebut greedy karena pengecekan dilakukan dari Cluster Edge yang mengabstraksi Edge paling banyak terlebih dahulu, hingga ke Cluster Edge yang hanya mengabstraksi satu Edge. Pencarian secara greedy dilakukan dengan melakukan pengulangan (looping) terhadap semua Cluster Edge yang telah diurutkan sebelumnya. Untuk setiap Cluster Edge, dilakukan pengecekkan terhadap kedua Cluster Node yang dihubungkannya. Apabila semua Edge yang menghubungkan setiap Node pada Cluster Node pertama kepada setiap Node pada Cluster Node kedua belum pernah diabstraksi sebelumnya, maka Cluster Edge itu dianggap final. Apabila ada satu saja Edge antara suatu Node pada Cluster Graph pertama dengan suatu Node pada Cluster Graph kedua yang yang sebelumnya telah diabstraksi oleh sebuah Cluster Edge, maka Cluster Edge tersebut di skip, dan pengecekan dilakukan terhadap Cluster Edge berikutnya.

2.4. Perancangan Sistem

Tujuan pada tugas akhir ini adalah sistem yang mengimplementasikan teknik Graph Database Compression dalam proses penyimpanan data berdasarkan dataset dari SNAP (Stanford Network Analysis Project). Input dari sistem ini adalah data kolaborasi penulisan jurnal ilmiah. Data ini direpresentasikan dalam bentuk Node dan Edge dimana entitas-entitas digambarkan sebagai Node dan cara dimana entitas-entitas berhubungan satu sama lain digambarkan sebagai Edge. Data ini berbentuk teks yang merepresentasikan graph tak berarah. Berikut ini adalah gambaran dari sistem yang akan dibangun:

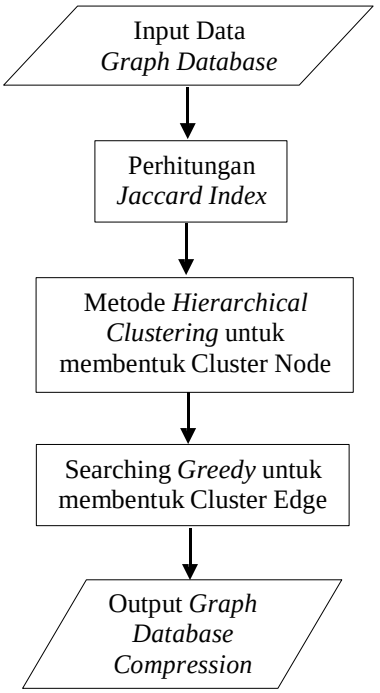
Dari gambaran umum sistem di atas maka pemrosesan yang dilakukan adalah :

1. Sistem menggunakan input dari data kolaborasi penulisan jurnal ilmiah dari SNAP(Stanford Network Analysis Project) yang kemudian disimpan dalam bentuk graph.

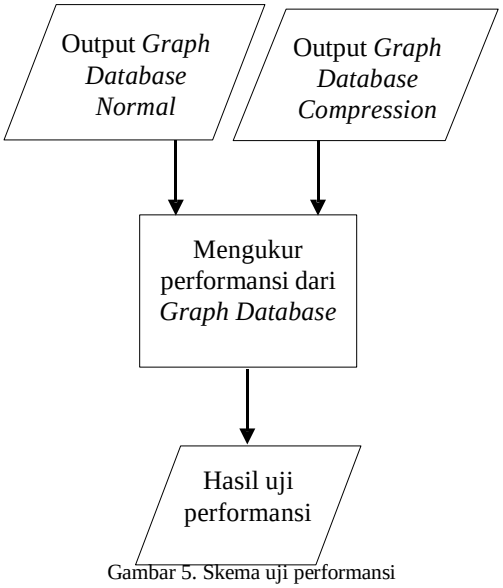


Gambar 3. Gambaran Umum Sistem

2. Sistem melakukan transformasi ke Graph Database Compression menggunakan GRAC (Graph Algorithm Clustering).
3. Sistem melakukan analisis terhadap hasil transformasi dari Graph Database ke Graph Database Compression.



Gambar 4. Skema Graph Database Compression



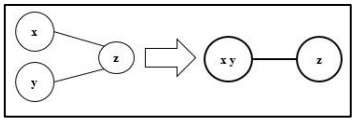
Gambar 5. Skema uji performansi

3. Pembahasan

3.1. Skenario Pengujian

Untuk pengujian ini akan dipilih lima buah dataset yang berupa graph dengan derajat rata-rata yang bervariasi. Kelima graph tersebut memiliki perbandingan jumlah Node dan Edge yang beraneka ragam. Masing-masing graph akan diproses dengan penerapan algoritma GRAC(Graph Algorithm Clustering). Akan di analisis hasil dari implementasi dengan perbandingan hasil Node dan Edge. Impementasi dilakukan untuk mengetahui hasil kompresi Graph Database. Setelah GRAC(Graph Algorithm Clustering) diimplementasikan kepada kelima graph input, dihasilkan lima buah Cluster Graph. Lalu akan dilakukan perbandingan antara Node dengan Cluster Node dan Edge dan Cluster Edge. Selain itu dilakukan juga perhitungan derajat rata-rata. Perbandingan dilakukan dan dianalisis hasilnya.

Kemudian tingkat kompresi dari Cluster Graph yang dihasilkan akan diukur dengan parameter Edge Reduction dan database Compression Rate. Penggunaan GRAC(Graph Algorithm Clustering) dalam Graph Database Compression, jumlah Edge dalam suatu graph bisa berkurang dengan cukup signifikan. Hal ini dikarenakan pengumpulan Node-Node single ke dalam hirarki Cluster Node, sehingga untuk setiap pembentukan Cluster Node, beberapa Edge bisa diabstraksi menjadi satu Cluster Edge saja. Berikut merupakan visualisasi pengurangan Edge dan pembentukan Cluster Edge.

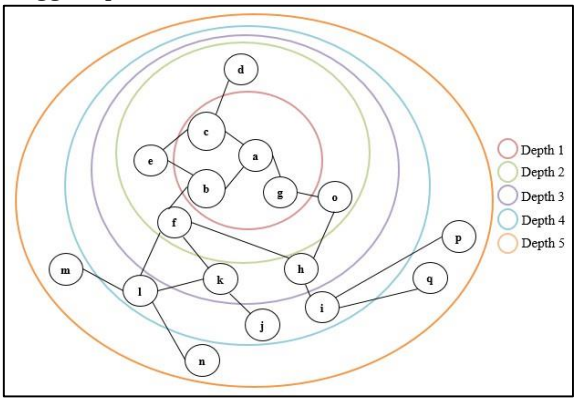


Gambar 6. Contoh pengurangan Edge

Dalam pembentukan Graph Database Compression, tingkat kompresi juga dapat dihitung dengan membandingkan ukuran Graph Database sebelum dan setelah dilakukan kompresi. Dengan menentukan ukuran file awal dan mendapatkan ukuran file hasil kompresi, maka akan dapat dianalisis hasil dari Compression Rate.

Performa database diuji dengan melakukan Query Read terhadap database awal dan database yang telah dikonversi menjadi Cluster Graph. Query Read yang dilakukan adalah pencarian Node tetangga dengan depth satu hingga lima. pengukuran untuk masing-masing depth dilakukan terhadap lima Node awal yang ditentukan secara acak. Lama waktu proses dari lima kali pencarian neighbors untuk masing-masing depth tersebut kemudian dirata-ratakan. Hasil pengukuran waktu ini kemudian digunakan untuk membandingkan performa graph. Setelah itu akan dilakukan pengujian performa terhadap Graph Normal dan Cluster Graph. Pengujian performa ini dilakukan dengan menjalankan Read Query untuk mencari tetangga suatu Node dari depth 1 hingga depth 5. Dari pengujian ini akan dilihat apakah ada perbedaan waktu yang diperlukan untuk menjalankan Query pada Graph Normal dan Cluster Graph.

Hasil lain dari Read Query pun dapat digunakan untuk mengecek apakah Cluster Graph, Graph hasil keluaran metode GRAC(Graph Algorithm Clustering) memiliki informasi yang sama dengan graph aslinya. Pengecekan Lossless Compression dilakukan dengan pengecekan jumlah tetangga dari hasil Query. Apabila keluaran dari Query terhadap graph asli dan keluaran Query terhadap Cluster Graph selalu sama, maka dapat diambil kesimpulan bahwa kompresi yang telah dilakukan adalah Lossless Compression. Berikut merupakan visualisasi Query Read dari depth 1 hingga depth 5.



Gambar 7. Contoh Read Query dari Depth 1 hingga Depth 5

3.2. Spesifikasi Data Input

Data graph yang digunakan dalam penelitian ini berasal dari website SNAP (Standford Network Analysis Project). Terdapat lima graph yang masing-masing berisi data kolaborasi antar penulis jurnal ilmiah yang terdapat dalam database Arxiv.org. Kelima graph tersebut berasal dari bidang yang berbeda yaitu Astro Physics, Condensed Matter, General Relativity, High Energy Physics, dan High Energy Physics Theory.

Tabel 1. Spesifikasi File Input dan Graph yang dibentuk dari masing-masing File

File Graph	Ukuran Asli (KB)	Ukuran Terkompresi (KB)	Compression Rate
AstroPh	5.160	1.337	74.08%
HepPh	2.949	465	84.23%
CondMat	2.363	818	65.38%
HepTh	643	263	59.09%
GrQc	344	111	67.73%

Dari tabel 1 dapat dilihat bahwa kelima graph memiliki kombinasi jumlah Node, Edge, dan derajat rata-rata yang bervariasi. Sistem ini diharapkan dapat diimplementasikan terhadap graph dengan jumlah Node dan Edge yang cukup besar.

3.3. Analisis implementasi GRAC(Graph Algorithm Clustering) dan Edge Reduction

Setelah GRAC(Graph Algorithm Clustering) diimplementasikan kepada kelima graph input, dihasilkan lima buah Cluster Graph. Setiap File AstroPh, HepPh, CondMat, HepTh dan GrQc memiliki jumlah bervariasi. Node berhasil di reduksi oleh Cluster Node, begitu pula Edge berhasil direduksi oleh Cluster Edge.

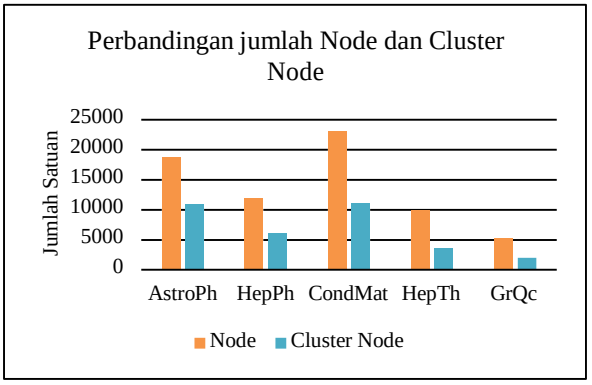
Dalam penerapan GRAC(Graph Algorithm Clustering) terlihat bahwa jumlah Edge berkurang karena membentuk Cluster Edge. Hal ini dikarenakan pengumpulan Node-Node single ke dalam hirarki Cluster Node, sehingga untuk setiap pembentukan Cluster Node, beberapa Edge bisa diabstraksi menjadi satu Cluster Edge saja.

Dari percobaan terhadap lima graph, Algoritma ini dapat dapat mengurangi jumlah Edge maksimal hingga 76,89%. Angka ini dicapai pada database HepPh yang memiliki derajat rata-rata 19,74. Edge paling sedikit tereduksi adalah pada database HepTh yaitu sebesar 42,90% tidak lebih dari 50%. Derajat rata-rata dari HepTh adalah 5,26, yang sama-sama memiliki derajat terkecil. Berikut merupakan hasil implementasi GRAC(Graph Algoritm Clustering) dan Edge Reduction dalam bentuk Tabel.

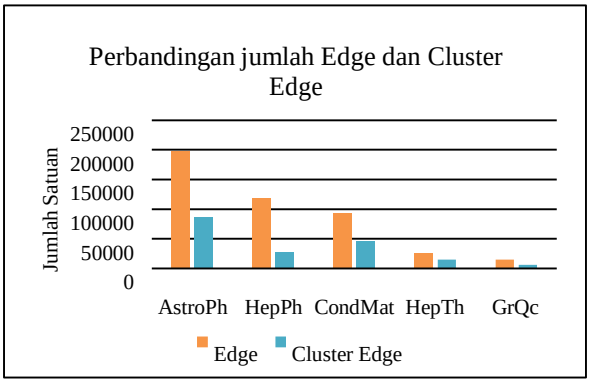
Tabel 2. Hasil analisis implementasi GRAC(Graph Algorithm Clustering) dan Edge Reduction

File Graph	Graph Normal		Cluster Graph		Edge Reduction
	Node	Edge	Cluster Node	Cluster Edge	
AstroPh	18771	198050	10955	85902	56,62%
HepPh	12006	118489	6030	27379	76,89%
CondMat	23133	93439	11154	45530	51,27%
HepTh	9875	25973	3519	14829	42,90%
GrQc	5241	14484	1988	5926	59,08%

Dalam perbandingan jumlah Node dan Cluster Node, agar terlihat secara jelas dalam bentuk visualisasi, maka akan dibuat grafik perbandingan. Grafik perbandingan ini ada dua, yaitu perbandingan jumlah Node dan Cluster Node dan perbandingan jumlah Edge dan Cluster Edge. Berikut merupakan hasilnya.



Gambar 8. Analisis perbandingan jumlah Node dan Cluster Node



Gambar 9. Analisis perbandingan jumlah Edge dan Cluster Edge

Masing-masing Cluster Graph tersebut dituliskan ke dalam file Cluster Graph. Masing-masing file Cluster Graph ini dibagi kedalam dua bagian. Bagian pertama berisi data Cluster Node. Setiap barisnya terdiri sebuah integer yang merupakan index Cluster Node, serta daftar Node dan Cluster Node yang ada di dalam Cluster Node tersebut. Bagian kedua berisi data Cluster Edge, yang menghubungkan sebuah Node atau Cluster Node dengan Node atau Cluster Node lain.

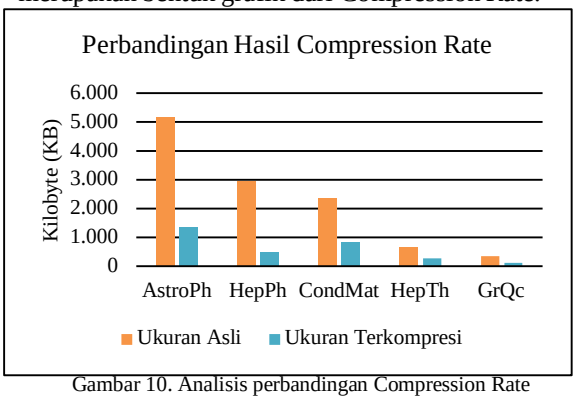
3.4. Analisis Compression Rate

Ukuran file dari kelima File memiliki hasil yang berbeda-beda. Ukuran file terbesar terdapat pada AstroPh sebesar 5.160 KB dan ukuran terkompresinya 1.337 KB memiliki ukuran Compression Rate sebanyak 74.08%. Pada HepPh sebesar 2.949 KB dan Ukuran terkompresinya 465 KB memiliki ukuran Compression Rate paling besar diantara file lainnya yaitu sebanyak 84.23%. Pada CondMat sebesar 2.3.63 KB dan ukuran terkompresinya 818 KB memiliki ukuran Compression Rate sebanyak 65.38%. Pada HepTh sebesar 643 KB dan Ukuran terkompresinya 263 KB memiliki ukuran Compression Rate paling kecil diantara file lainnya yaitu sebanyak 59.09%. Pada GrQc sebesar 344 KB dan ukuran terkompresinya 111 KB memiliki ukuran Compression Rate sebanyak 67.73%.

Tabel 3. Hasil analisis Compression Rate

File Graph	Ukuran (KB)	Jumlah Node	Jumlah Edge	Derajat Rata-rata
AstroPh	5.160	18.771	198.050	21,10
HepPh	2.949	12.006	118.489	19,74
CondMat	2.363	23.133	93.439	8,08
HepTh	643	9.875	25.973	5,26
GrQc	344	5.241	14.484	5,53

Bentuk visualisasi dari perbandingan Compression Rate akan dibuat secara grafik. Grafik menunjukkan presentase hasil kompresi dari tiap-tiap file. Berikut merupakan bentuk grafik dari Compression Rate.

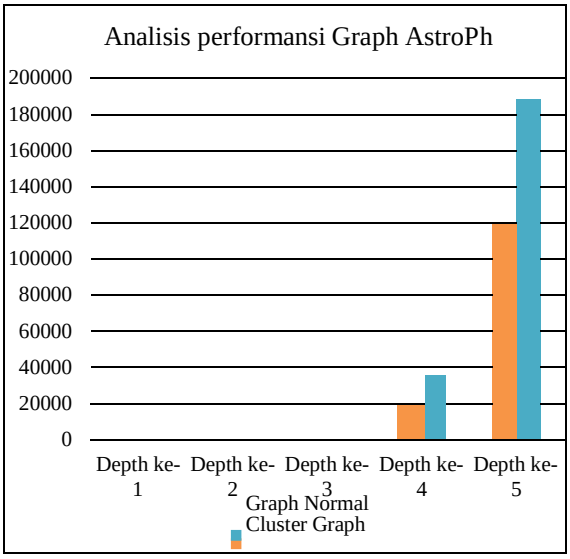


Gambar 10. Analisis perbandingan Compression Rate

3.5. Analisis Performansi Graph

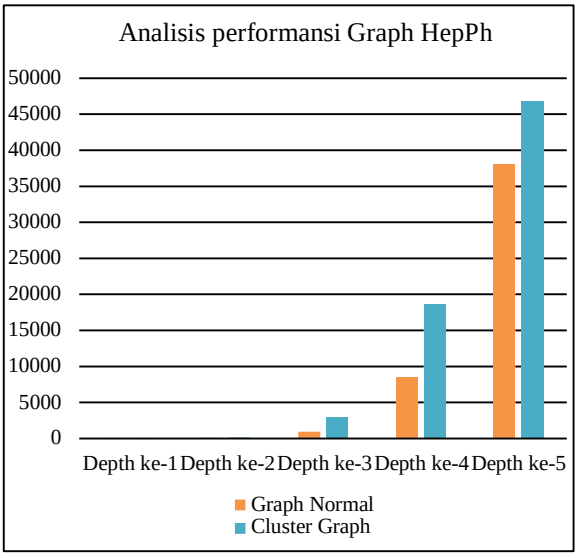
Pengujian dilakukan terhadap Graph Normal dan Cluster Graph dengan melakukan Query pencarian tetangga dengan Depth 1 hingga 5. Untuk masing-masing Depth dilakukan lima kali percobaan dengan Node berbeda yang dipilih secara acak. Lama waktu dari kelima percobaan untuk masing-masing Depth ini kemudian dirata-ratakan.

Dari gambar 11, eksekusi Query graph pada Depth 1 hingga Depth 5 masing-masing dapat ditempuh dengan waktu yang cukup. Perubahan terlihat pada Depth 3 dan 4 dimana waktu menjadi lebih lambat dalam pemrosesannya. Untuk proses awal pada Depth 1 dan 2, terjadi perubahan.



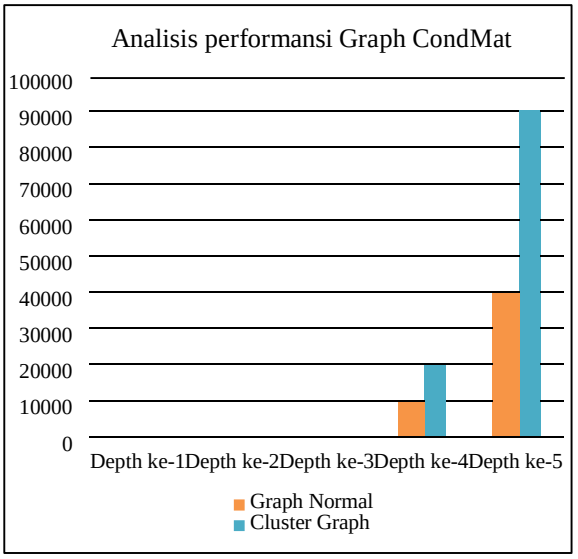
Gambar 11. Analisis performansi Graph Normal AstroPh dan Cluster Graph AstroPh

Dari gambar 12, eksekusi Query graph pada Depth 1 hinga Depth 5 yang terakhir. Pada HepPh, terlihat perubahan yang tidak konstan pada Depth 4 dan 5. Perbedaan tertinggi terdapat pada Depth 4 sebesar 53%, sedangkan Depth 5 hanya menjadi 18% lebih cepat dalam pemrosesannya. Pada Depth 5 terjadi perubahan yang signifikan sehingga, pada HepPh terlihat bahwa proses pencarian Read Query tidak selalu konsisten menjadi lambat pemrosesannya. Perubahan ini menyebabkan pencarian Read Query pada HepPh tidak konsisten. Depth 5 merupakan final dari hasil Query.



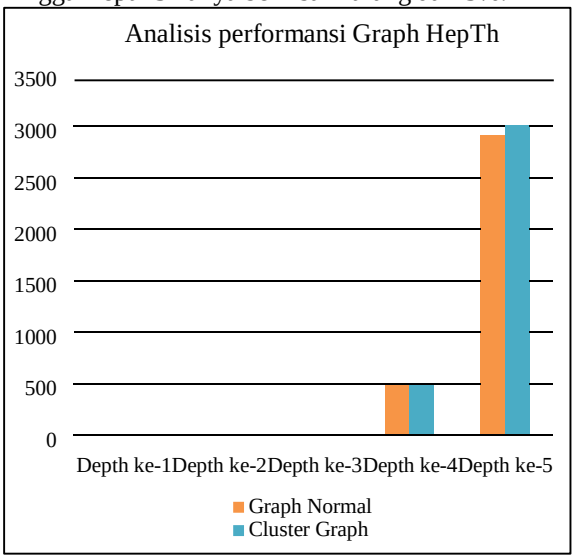
Gambar 12. Analisis performansi Graph Normal HepPh dan Cluster Graph HepPh

Dari gambar 13, eksekusi Query graph pada Depth 1 hingga Depth 5 masing-masing memiliki variasi yang berbeda. Perubahan terlihat pada Depth 4, tetapi pada Depth 5 terjadi pemrosesan pencarian yang sangat lama oleh Cluster Graph di bandingkan Graph Normal. Pada Graph CondMat terlihat lama waktu di Depth 5, perbandingannya yaitu sebesar 55%.



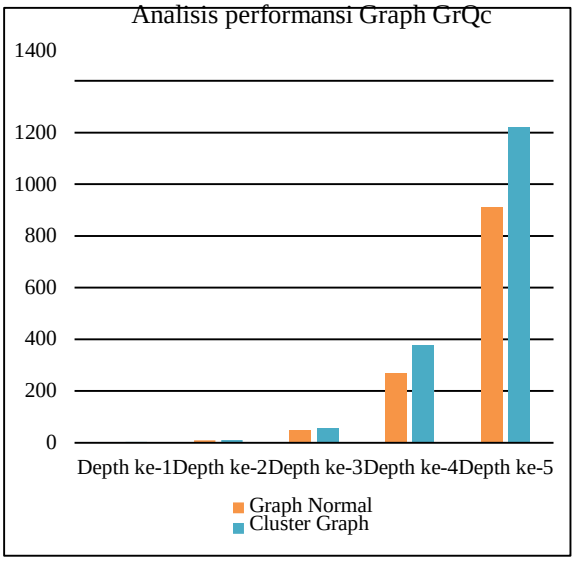
Gambar 13. Analisis performansi Graph Normal CondMat dan Cluster Graph CondMat

Dari gambar 14, eksekusi Query graph pada Depth 1 hingga Depth 5 masing-masing dapat ditempuh dengan waktu yang berbeda-beda. Tetapi perbandingan read query memiliki kisaran yang kecil. Hal ini bisa dilihat dengan perbandingan perbedaan Read Query pada setiap Depth. Kecepatan cluster graph menjadi lebih lambat pada Depth 2 3,29%, Depth 3 3,03% Depth 4 3,88%, Depth 5 4,97 sedangkan pada Depth 1 12% menjadi lebih cepat tetapi tidak signifikan. Perbandingan antara Depth 2 hingga Depth 5 hanya berkisar kurang dari 5%.



Gambar 14. Analisis performansi Graph Normal HepTh dan Cluster Graph HepTh

Dari gambar 15, eksekusi Query graph pada Depth 1 hingga Depth 5 memiliki variasi. Tetapi terlihat bahwa Cluster Graph selalu lebih lambat dibandingkan Graph Normal. Berbedaan kecepatan terlihat pada Depth 4 dan Depth 5. Karena file ini merupakan file yang memiliki ukuran terkecil, maka pemrosesannya lebih cepat dibandingkan dengan file yang lainnya.



Gambar 15. Analisis performansi Graph Normal GrQc dan Cluster Graph GrQc

3.6. Analisis Pengecekan Lossless Compression

Pengecekan bahwa Kompresi tidak menghilangkan informasi bisa dilakukan dengan Read Query. Perbedaannya, apabila pada Sub bab sebelumnya, yang dihitung adalah waktu pemrosesan Query-nya, maka di sini yang dilihat adalah hasil dari Query itu sendiri.

4. Penutup

Sebuah graph undirected yang direpresentasikan oleh Graph Database dan melakukan Graph Database Compression dari sebuah metode GRAC(Graph Algorithm Clustering) dapat menghasilkan kompresi graph yang Lossless.

Analisis pengujian menunjukkan bahwa graph dapat di kompresi maksimal sebesar 84.23% dan rata-rata hasil kompresi dari kelima file tersebut adalah 70,10%, sehingga kompresi Graph Database dapat mengurangi ukuran file dataset. Edge Reduction mengurangi Edge dan direpresentasikan dengan Cluster Edge. Dengan pengurangan Edge, maka dataset menjadi lebih efektif dalam representasi Edge. Apabila terjadi pengurangan Edge, maka akan berpengaruh juga terhadap hasil kompresi dataset. Hasil performansi rata-rata waktu Query Read Graph Database Compression lebih lambat dibandingkan Graph Database tetapi tidaklah konstan.

Daftar Pustaka

[1] I. Robinson, J. Webber, E. Eifrem, 2013. Graph Databases, O'Reilly.

[2] Vicknair et al., 2010. A Comparison of a Graph Database and a Relational Database, ACMSE.

[3] Elisa Schaeffer, 2007. Graph Clustering, Elsevier.

[4] Zhang et al., 2012, Graph Degree Linkage: Agglomerative Clustering on a Directed Graph, ECCV 2012.

[5] Daniel Mullner, 2011, Modern Hierarchical, Agglomerative Clustering Algorithm,DMS 2011.

[6] Niwattanakul, Singthongchai, Naenudorn, Wanapu, 2013, Using of Jaccard Coefficient for Keyword Similarity, IMECS 2013 Vol I, Hong Kong.

[7] J. Leskovec, "Stanford Large Network Dataset Collection," Stanford, [Online]. Available: http://snap.stanford.edu/data/. [Diakses 02 Januari 2014].

[8] Implementing Hierarchical Clustering, ELKI, [Online]. Available: http://elki.dbs.ifi.lmu.de/wiki/Tutorial/HierarchicalClustering. [Diakses 02 Januari 2014].

[9] Jaccard's Coefficient, Kardi Teknomo's Page, [Online]. Available: http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html. [Diakses 02 Januari 2014].

[10] Greedy algorithm, Princeton, [Online]. Available: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Greedy_algorithm.html. [Diakses 02 Januari 2014].