Available at https://jtsiskom.undip.ac.id (3 April 2022)
DOI:10.14710/jtsiskom.2022.xxxxx

Jurnal Teknologi dan Sistem Komputer, 10(2), 2022, 1-10

Spatial Skyline Query Based on Surrounding Environment Untuk *Data Streaming* Menggunakan *Apache-Spark*

Raden Muhamad Firzatullah[1)], Taufik Djatna [2)], Annisa [3)], Andrianingsih [1)]

[1)]*Program Studi Informatika, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional*
Jl. Sawo Manila No.61, Pejaten Barat, Daerah Khusus Ibukota Jakarta, Indonesia 12520

[2)]*Departemen Teknologi Industri Pertanian, Fakultas Teknologi Pertanian, Institut Pertanian Bogor*
Jl. Raya Dramaga, Babakan, Kec. Dramaga, Kota Bogor, Indonesia 16680

[3)]*Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor*
Jl. Raya Dramaga, Babakan, Kec. Dramaga, Kota Bogor, Indonesia 16680

Abstract - Previous research on Spatial Skyline Query Based on Surrounding Environment left a challenge in finding skyline objects that support the use of mobile devices. This study introduces a method that allows users to search for spatial objects in real-time on streaming data using Apache-Spark. This study aims to test the effectiveness and efficiency of Apache-Spark in providing recommendations for Spatial Skyline Query Based on Surrounding Environment in processing streaming data on mobile devices. Measurements on each parameter show that the computational time of the proposed algorithm is superior to the previous algorithm in cluster computing and is able to process streaming data in real-time so that it can be implemented on mobile devices.

Keywords *- Skyline Query; Data Streaming; Spatial Object; Apache-Spark; Cluster Computing*

Abstrak - Penelitian sebelumnya mengenai Spatial Skyline Query Based on Surrounding Environment meninggalkan tantangan dalam pencarian objek skyline yang mendukung penggunaan perangkat seluler. Penelitian ini memperkenalkan metode yang memungkinkan pengguna untuk mencari objek spasial secara real-time pada data streaming menggunakan Apache-Spark. Penelitian ini bertujuan mengembangkan algoritma yang dapat memberikan rekomendasi objek skyline yang lebih baik bagi pengguna pada perangkat bergerak. Hasil pengujian pada masing-masing parameter menunjukkan bahwa waktu komputasi algoritma yang diusulkan pada komputasi klaster lebih unggul dibandingkan dengan algoritma terdahulu serta mampu memproses data streaming secara real-time sehingga dapat diimplementasikan pada perangkat bergerak.

Kata Kunci *– Skyline Query; Data Streaming; Spatial Object; Apache-Spark; Cluster Computing*

I. PENDAHULUAN

Spatial Skyline Base on Surrounding Environment dijelaskan oleh [1] sebagai metode untuk menentukan objek spasial *skyline* dari sekumpulan objek spasial dengan mempertimbangkan objek sekitarnya. Metode sebelumnya yang dilakukan pada pencarian objek spasial *skyline* hanya mempertimbangkan kondisi ketika pengguna berada pada posisi diam seperti pada [2] dan [3] dalam mencari objek skyine pada *spatial big data*, [4] dan [5] dalam menentukan objek *skyline* untuk *smart cities*, [6], [7], [8], [9] dalam menentukan objek *skyline* menggunakan data *Google Places API* dan [10], [11], [12], [13] dalam menentukan objek *skyline* berdasarkan *surrounding object*. Berdasarkan kekurangan dari penelitian terdahulu, penelitian ini memperkenalkan teknik *Spatial Skyline Base on Surrounding Environment* sebagai ide utama tetapi mendukung pemrosesan pada perangkat bergerak.

Algoritma *Spatial Skyline Based on Surrounding Environment* memerlukan sumber data lokasi berupa objek spasial dan atribut non spasial [14]. Penggunaan data statis pada perangkat bergerak tidak mungkin diterapkan, dikarenakan lokasi pengguna pada waktu tertentu dapat berubah, sehingga keberadaan objek spasial terhadap pengguna pun akan berubah [15]. Terkait dengan masah tersebut diperlukan suatu sumber data dinamis yang mampu menyediakan update data secara simultan mengikuti pergerakan pengguna. *Data streaming* dijelaskan oleh [16] merupakan data yang dapat dipanggil secara terus menerus oleh satu atau lebih sumber, yang bersifat kontinyu, berubah dengan cepat, dan memiliki ukuran yang besar. Sehingga pada penelitian ini, data streaming digunakan sebagai penyedia data spasial. *Data streaming* yang digunakan dalam penelitian ini berupa lokasi *Point of Interest* (POI) beserta atribut spasial dan non spasial seperti harga, *rating* dan lain sebagainya. POI dijelaskan oleh [17] sebagai lokasi titik tertentu yang memiliki makna atau menarik bagi seseorang. Dalam beraktivitas, pengguna cenderung mempertimbangkan biaya yang lebih rendah dan tingkat layanan yang lebih tinggi. Pada penelitian [18], *data streaming* digunakan sebagai sumber data dalam penerapan algoritma *skyline* dan pencarian objek spasial.

Perspektif *data streaming* sangat berbeda dengan data statis. Data statis tidak memiliki hubungan antara waktu pemrosesan awal dan pemrosesan selanjutnya. [19] menyatakan bahwa *data streaming* memerlukan pengolahan data yang cepat karena data mengalir secara

*) Raden Muhamad Firzatullah
Email: firzatullah@civitas.unas.ac.id

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

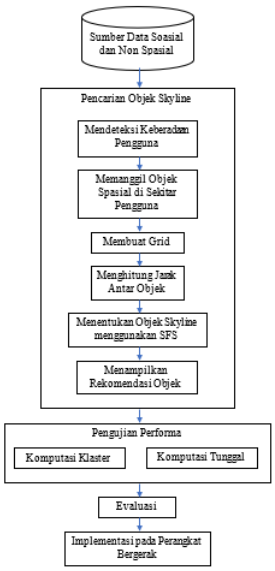
terus menerus dan berubah dalam jendela waktu tertentu [20]. Lambatnya pengolahan *data streaming* mempengaruhi relevansi hasil yang diperoleh [21]. Dalam penelitian [1], penambahan jumlah parameter mengakibatkan peningkatan waktu pemrosesan. Apache-Spark dijelaskan oleh [19] sebagai kerangka kerja untuk pemrosesan *data streaming* yang menerapkan komputasi klaster dimana penanganan data terdistribusi toleran terhadap kesalahan. Komputasi klaster [22] adalah sekelompok komputer yang saling berhubungan dan bekerja sama dalam melakukan proses komputasi sebagai satu sistem [23]. Penelitian ini menggunakan *Apache-Spark* sebagai kerangka kerja manajemen untuk pengoperasian komputasi terdistribusi, termasuk komputasi paralel dan komputasi klaster dalam menangani pemrosesan data streaming real-time. Penelitian sebelumnya oleh [24] menunjukkan bahwa *Apache-Spark* memiliki keunggulan dalam proses komputasi dan akurasi dibandingkan dengan OpenMP dan Beowulf dalam pemrosesan *machcine learning*.

Berdasarkan masalah yang telah dipaparkan sebelumnya, belum ada algoritma yang dapat memberikan rekomendasi objek spasial *skyline* yang mempertimbangkan preferensi objek spasial disekitarnya secara *real-time* dan dapat diimplementasikan pada perangkat bergerak. Sehingga, penelitian ini bertujuan dalam mengembangkan metode baru yang mampu memproses objek spasial *skyline* berdasarkan pertimbangan prefrensi objek disekitar secara *real-time* dengan memanfaatkan *Apache-Spark* dan *data streaming*. Selanjutnya metode ini, diimplementasikan pada perangkat bergerak untuk memberikan akses lokasi yang lebih baik bagi pengguna.

.

II. METODE PENELITIAN

A. Kerangka dan Tahapan Penelitian



Gambar 1. Kerangka Penelitian

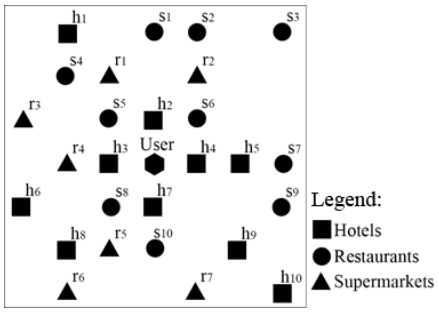
Berdasarkan kerangka penelitian yang tertera pada Gambar 1, terdapat beberapa tahapan utama pada penelitian ini diantaranya pencarian objek skyline, pengujian performa algoritma, evaluasi dan implementasi algoritma pada aplikasi perangkat bergerak. Tahapan awal dalam pencarian objek skyline pada data streaming diawali dengan proses buffer data yang akan dijabarkan pada sub bab selanjutnya, dilanjutkan dengan mengukuran jarak dengan memanfaatkan *grid* pada setiap lokasi objek spasial dan tahapan akhir menentukan objek *skyline* berdasarkan objek spasial dan non-spasial menggunakan SFS *Skyline*. Pada tahapan pengujian performa, penelitian ini membandingkan performa algoritma yang diusulkan dengan algoritma terdahulu, dimana pengujian dilakukan pada dua lingkungan komputasi diantaranya komputasi klaster dan komputasi tunggal dengan memanfaatkan kombinasi indikator konfigurasi. Pada tahapan selanjutnya, dilakukan proses evaluasi dari hasil pengujian performa algoritma pada masing-masing indikator evaluasi dan lingkungan komputasi, sehingga dapat diketahui apakah algoritma yang diusulkan cocok untuk diimplementasikan pada perangkat bergerak. Tahap akhir, algoritma diimplementasikan pada perangkat bergerak.

B. Metode Pencarian Spatial Skyline Object pada Data Streaming

Pada bagian ini, akan dijelaskan metode yang diusung berdasarkan contoh kasus pencarian rekomendasi hotel yang dekat dengan restoran dan hotel berdasarkan preferensi penggun. *Sliding window* [23][24] yang digunakan dalam memproses *streaming data* pada penelitian ini dijabarkan dalam contoh sebagai berikut. Pengguna ingin mencari hotel murah yang dekat dengan supermarket dan restoran dengan mempertimbangkan kualitas pelayanan terbaik, beriringan dengan mencari supermarket dan restoran

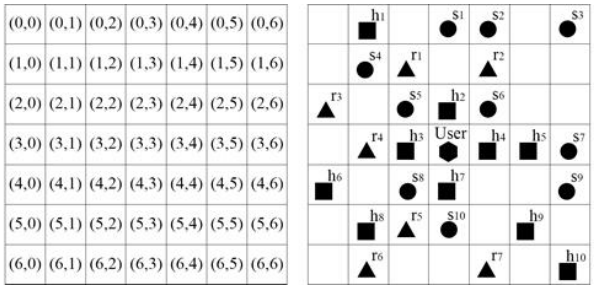
Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

yang memiliki harga yang murah. Gambar 2 menunjukkan distribusi keberadaan objek spsial.



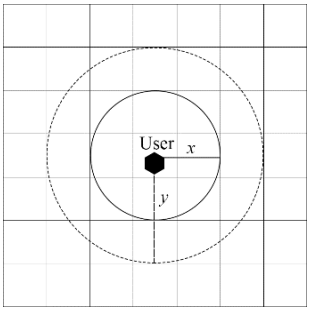
Gambar 2. Distribusi objek spsial

Hotel merupakan fasilitas utama yang digambarkan dengan simbol persegi, H = {h_1, h_2, h_3, …, h_{10}}. Restoran dan Supermarket merupakan fasilitas pendukung yang berada disekitar fasilitas utama digambarkan dengan simbol lingkaran R = {r_1, r_2, r_3, …, r_7} untuk Restoran dan simbol segitiga S = {s_1, s_2, s_3, …, s_{10}} untuk supermarket, sedangkan lokasi pengguna digambarkan dengan simbol segi enam. Setelah memberikan simbol pada masing-masing objek, langkah selanjutnya adalah menambahkan *grid* dengan ukuran *n*×*n* pada lokasi pencarian dan memberikan label id pada masing-masing *grid* seperti yang ditunjukkan pada Gambar 3. Setelah itu, atur radius area streaming dan radius area pencarian fasilitas utama yang bertitik pusat pada lokasi pengguna.



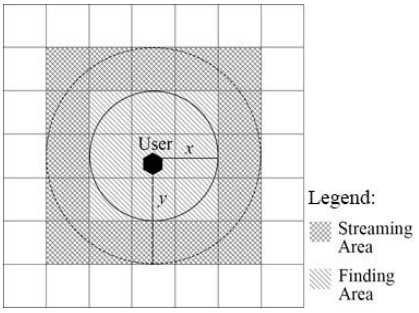
Gambar 3. Objek spasial pada *grid*

Streaming area merupakan area yang mencakup fasilitas utama dan fasilitas pendukung disekitarnya dalam suatu *sliding window* tertentu, sedangkan area pencarian adalah area yang mancakup fasilitas utama sebagai calon objek *skyline* pada *sliding window* yang telah ditentukan. Pada Gambar 4, radius *area streaming* digambarkan dengan lingkaran yang memiki jari-jari *x*, dimana *x* mewakili radius dari *area streaming* fasilitas utama sebagai calon objek *skyline*, sedangkan jari-jari *y* mewakili radius dari *area streaming* fasilitas pendukung yang berada disekitar kandidat objek *skyline* seperti yang digambarkan pada Gambar 5.



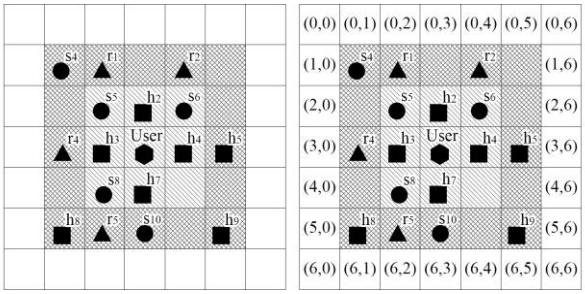
Gambar 4. Radius *area streaming*

Hotel yang berada di *area streaming* tetapi di luar area pencarian fasilitas utama seperti h_5, h_8 dan h_9, tidak termasuk dalam kandidat *skyline*. Namun fasilitas di sekitar *W* seperti supermarket s_4, s_{10} restoran r_1, r_2, r_4, r_5 dianggap sebagai fasilitas pendukung di sekitar yang dipertimbangkan dalam pemilihan hotel terbaik.



Gambar 5. Radius *streaming area* dan *finding area*

Pada Gambar 6(a) hotel yang dianggap sebagai kandidat objek *skyline* adalah hotel yang berada di dalam radius *finding area* seperti h_2, h_3, h_4, dan h_7. Sedangkan supermarket dan restoran yang berada di dalam radius pencarian juga *area streaming* diantaranya supermarket s_4, s_5, s_6, s_8, s_{10} dan restaurant r_1, r_2, r_4, r_5 digunakan dalam menentukan objek *skyline*.

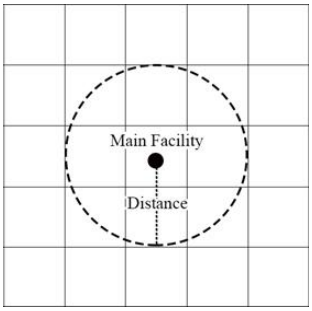


Gambar 6. Objek spasial pada *grid*

Supermarket dan Restoran yang berada di dalam *streaming area* menjadi pertimbangan dalam menentukan objek skyline dari hotel, hal tersebut dikarena kemungkinan jumlah Restoran dan Supermarket yang berada didalam *area streaming* lebih banyak dibandingkan dengan area pencarian. Akan tetapi restoran dan supermarket yang berada di luar *streaming area* seperti yang berada di *grid* {(0,0), (0,1), (0,3), ... (6,6)} tidak dimasukan ke dalam *sliding window*. Hal tersebut dikarenakan restoran dan

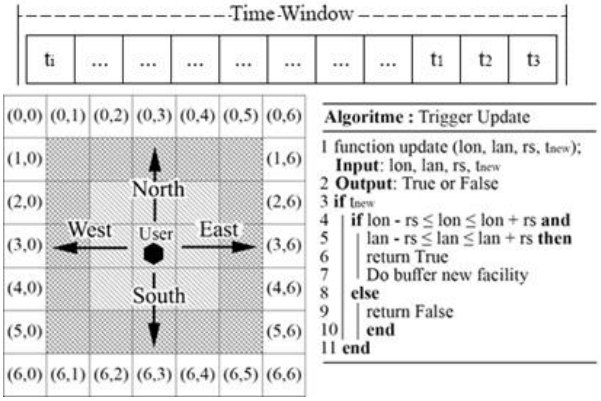
Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

supermarket tersebut memiliki jarak yang lebih jauh dibandingkan supermarket dan restoran yang berada di area pencarian dan *area streaming*, seperti yang ditunjukan pada Gambar 6(a). Proses *buffer* pada satu *sliding window* baru diproses apabila pengguna meninggalkan area pencarian.



Gambar 7. Jarak antara *main facility* dengan *surrounding facility*

Penghitungan jarak antara hotel dengan lokasi supermarket juga restoran didasarkan pada jarak hotel dengan garis pembatas pada *grid* yang menampung lokasi supermarket atau restoran. Semisalnya pada Gambar 7 jarak antara hotel (*main facility*) dengan *grid* yang menampung supermarket atau restoran berjarak 1,5 *grid*. Dengan demikian, didapatkan jarak minimum antara hotel dengan supermarket atau restoran disekitarnya adalah 1,5 dari sisi *grid*. Sebagai contoh, jika sebuah *grid* berukuran 4 meter, maka jarak antara hotel menuju supermarket atau restoran dapat dihitung dengan 1,5 × 4 meter, yaitu 6 meter.



Gambar 8. Ilustrasi dan *pseudocode* proses *buffer data*

Gambar 8 menunjukan bahwa keberadaan pengguna akan diperiksa dalam rentan waktu tertentu *Time Window* (t_i). Dimana bila pengguna meninggalkan area pencarian baik ke barat, timur, utara atau selatan, proses *buffer* fasilitas di area pencarian, baik hotel, supermarket juga restoran akan berubah menyesuaikan keberadaan pengguna saat ini. Sementara itu, pergerakan pengguna akan diidentifikasi pada periode tertentu sebagai pemicu *buffer data*. Gambar 8 menunjukkan bagaimana pemicu memperbarui kondisi *sliding windows*. Setelah ditentukan kandidat objek *skyline* dan fasilitas pendukung disekitarnya serta penghitungan atribut juga jarak dilakukan maka selanjutnya proses penentuan objek *skyline* ditentukan menggunakan algoritma fundamental SFS [25].

C. Metode Pengukuran Jarak Antar Objek

Haversine Formula adalah persamaan dalam navigasi, memberikan panjang jarak antara dua titik pada bola dunia yang memperimbangkan garis bujur dan garis lintang. *Haversine Formula* menghitung jarak antara lokasi berdasarkan garis lintang (φ) dan garis bujur (λ) yang dimiliki lokasi tersebut. Sedangkan jari-jari bumi (r) memiliki nilai tetap yakni (jari-jari = 6.371km). Dimana sumber data dari objek spasial bersumber dari *Google Places API* yang memiliki atribut berupa garis bujur dan garis linting. *Google Places API* sendiri menggunakan sistem koordinat WGS 84/*Pseudo-Mercator* [26]. Berdasarkan hal terebut, penghitungan jarak menerapkan *haversine formula* dalam menentukan jarak antar objek. Sedangkan rumus *haversine formula* dijabarkan pada Persamaan 1.

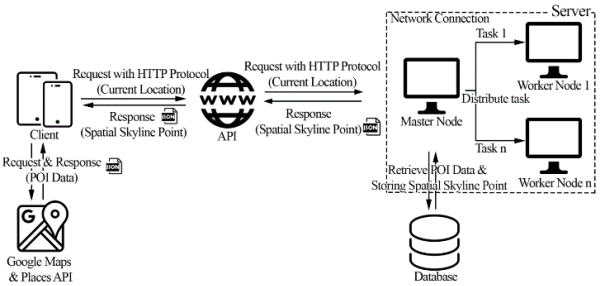
$$d = 2r \cdot archsin\,(h)$$
$$h = \left(\sqrt{sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \quad (1)$$

Semisalnya pada contoh kasus yang telah kami jabarkan, diketahui lokasi keberadaan pengguna berada di garis lintang -6.3928 dan garis bujur 106.7617. Sedangkan lokasi hotel terdekat berada di *grid* A. Garis sisi *grid* A yang paling dekat dengan lokasi pengguna berada pada garis lintang -6.698962 dan gartis bujur 106.950897. Sehingga dapat dihitung jarak antara pengguna dengan hotel terdekat sebagai kandidat objek skyline speerti pada Persamaan 2, sehingga didapatkan jarak sejauh 23 km.

$$D = 2 \cdot \sin^{-1}\left(\sqrt{\sin\left(\frac{106.95 - 106.761}{2}\right)^2 + \sin\left(\frac{-6.6989 - -6.3928}{2}\right)^2 \cdot \cos(106.761) \cdot \cos(106.95)}\right) \cdot \mu_E \quad (2)$$

Setelah jarak keberadaan pengguna dengan objek spasial utama dan objek spasial pendukung dihitung. Selanjutnya dilakukan proses penentuan objek *skyline* menggunakan SFS *skyline*.

D. Arsitektur Sistem Informasi



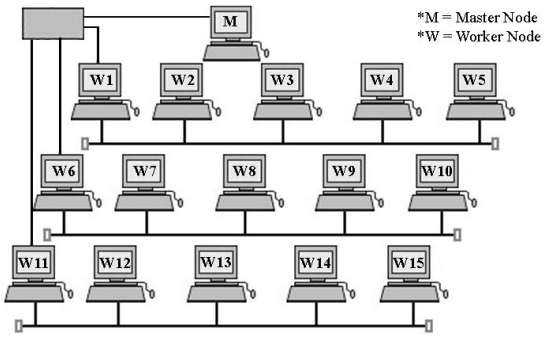
Gambar 9. Arsitektur Sistem Informasi

Arsitektur sistem dikembangkan dengan menggabungkan dua *platform* yang berbeda. *Platform server* yang menjalankan algoritme bertindak sebagai

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

mesin komputasi yang mengimplementasikan komputasi klaster *Apache-Spark* sedangkan *platform* klien pada perangkat bergerak bertugas dalam menampilkan antarmuka pengguna yang ditunjukan pada Gambar 9. Komunikasi antara perangkat bergerak dengan perangkat *server* memanfaatkan teknologi *Restful API* sebagai penghubung antara dua *platform* berbeda. Sedangkan *JSON (JavaScript Notation)* digunakan sebagai format trasfer data baik untuk pada komunikasi internal sistem maupun eksternal sistem seperti komunikasi dengan *Google Places API*.

E. Topology



Gambar 10. Topologi jaringan pada lingkungan komputasi klaster

Gambar 10 menunjukkan topologi yang digunakan pada lingkungan komputas klaster dan merupakan topologi *hybrid* yang menggabungkan topologi bus dengan topologi star, dimana terdapat satu buah *switch* yang menghubungkan 15 simpul pekerj satu sama lain melalui kabel UTP. Sedangkan simpul master yang bertugas sebagai kontrol dihubungkan langsung ke perangkat *switch*. Pengujian konektivitas jaringan dilakukan dengan mengirimkan perintah *ping* dari *node master* ke *node worker* dimana hasil pengujian menunjukan waktu respon sekitar 5-20 ms.

F. Indikator Evaluasi

Langkah terakhir adalah evaluasi, pada langkah ini kinerja dari algoritma dan komputasi pemrosesan data dievaluasi. Evaluasi dilakukan dengan mencoba berbagai kombinasi indikator pemrosesan data, baik pada bagian *dataset*, perngkat komputasi dan juga algortima. Evaluasi dilakukan dengan mengukur *Speedup Factor S(p)* [23] yang berjalan pada masing-masing kombinasi indikator pemrosesan data. Pengukuran *S(p)* dilakukan dengan menghitung waktu komputasi pada prosesor tunggal (t_s) dan waktu komputasi pada multiprosesor (t_s). Persamaan 2 menunjukan formula dalam menghitung *speed up factor* dan waktu komputasi multiprosesor.

$$S(p) = \frac{t_s}{t_p} \qquad (2)$$

$$t_p = t_{comm} + t_{comp}$$

Message-passing computing (t_p) adalah waktu komputasi multiprosesor, dimana waktu komputasi multiprosesor terdiri dari waktu komunikasi antar perangkat komputasi (t_{comm}) dan waktu komputasi data pada masing-masing perangkat komputasi (t_{comp}).

Pengujian dilakukan dengan mempertimbangkan kombinasi idikator evaluasi pada pemrosesan data. Konfigurasi indikator dan parameter dalam pengujian ditunjukan pada Tabel 1. Parameter yang digunakan dalam pengujian merupakan parameter yang digunakan pada penelitian [24], dimana konfigurasi dari parameter dapat menguji efektifitas komputasi dalam mencari objek *skyline*.

Tabel 1. Indikator evaluasi

Parameter	Value	Default
Dimensi atribut data	1, 2, 4, 8	2
Jumlah data	500, 1k, 4k, 8k	1k
Objek spasial pendukung	2, 4, 8, 16	4
Objek kandidat *skyline*	1, 5, 10, 20	5

G. Skenario Komputasi

Dalam menguji kinerja algoritma akan dibandingkan waktu pemrosesan algoritma *skyline based on surrounding environment* konvensional dan algoritma *skyline* yang diusulkan, dimana pengujian dilakukan pada dua lingkungan komputasi yang berbeda, diantaranya komputasi dengan pemrosesan tunggal dan komputasi menggunakan komputasi klaster. Komputasi klaster menggunakan kerangka kerja *Apache-Spark*, dimana lingkungan komputasi klaster memiliki 20 unit simpul pekerja dan 1 unit simpul master yang saling terhubungan pada jaringan LAN. Total unit prosesing pada komputasi klaster berjumlah 40 inti *processor* dan 80 GB RAM. Dataset yang digunakan berupa data spasial dan non-spasial yang bersumber dari *Google Place API*, pemanggilan *dataset* dilakukan berdasarkan *sliding window* yang telah ditentukan. Masing-masing pengujian kombinasi indikator dilakukan sebanyak 50 iterasi, dimana waktu rata-rata dari ke-50 iterasi tersebut digunakan sebagai waktu komputasi pada masing-masing kombinasi indikator.

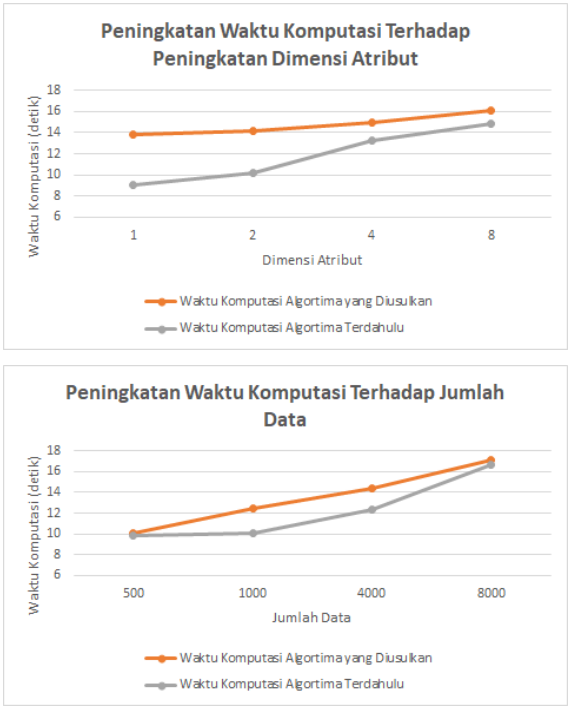
III. Hasil dan Analisis

A. Komputasi Tunggal

Pengujian dilakukan dengan membandingkan waktu komputasi algoritma yang diusulkan dengan algoritma terdahulu pada lingkungan komputasi tunggal. Dimana indikator evaluasi pada Tabel 1 divariasikan dalam menguji kinerja komputasi. Berdasarkan Gambar 11, waktu komputasi algortima terdahulu memiliki keunggulan dibandingakan dengan waktu komputasi algortima yang diusulkan, hal tersebut dikarenakan pada algoritma yang diusulakan tahapan dalam pencarian objek skyline lebih banyak dibandingkan dengan komputasi algoritma terdahulu [1], tahapan yang menambah waktu komputasi semakin lama diantaranya tahap pembuatan *grid*, penghitungan jarak antara grid dan pengguna, serta penghitungan jarak menggunakan

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

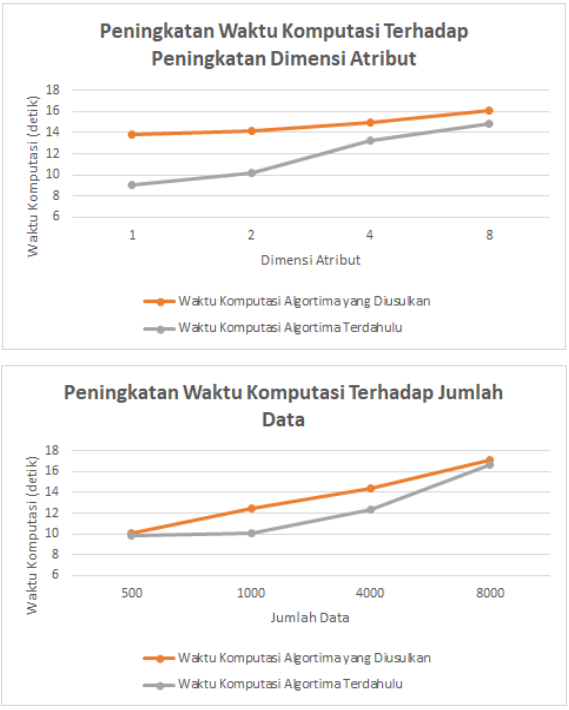
haversine [8]. Namun peningkatan waktu komputasi terhadap penginkatan parameter pengujian pada algoritma yang diusulkan tidak sepesat algoritma terdahulu. Hal tersebut dikarenakan jumlah grid yang dibuat pada algoritma yang diusulkan akan selalu tetap meskipun jumlah parameter ditingkatkan, sehingga peningkatan waktu komputasi cenderung lebih stabil. Hal tersebut selaras dengan penjelasan pada [8] yang menyatakan bahwa peningkatan atribut, jumlah *grid* dan jumlah data akan meningkatkan waktu komputasi.



Gambar 11. Perbandingan waktu komputasi algoritma pada unit komputasi tunggal

B. Komputasi Klaster

Pengujian kedua membandingkan waktu komputasi algoritma yang diusulkan dengan algoritma terdahulu pada lingkungan komputasi klaster. Dimana pada pengujian kedua, seluruh simpul pekerja digunakan dalam memproses algoritma.



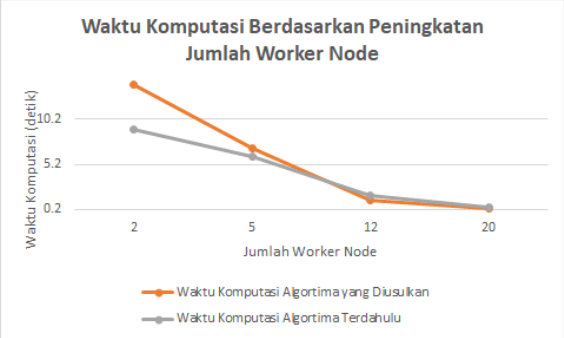
Gambar 12. Perbandingan waktu komputasi algoritma pada unit komputasi klaster

Berdasarkan pengujian menggunakan parameter peningkatan jumlah objek pendukung serta peningkatan jumlah objek kandidat *skyline*, waktu komputasi algoritma yang diusulkan lebih unggul dibandingkan waktu komputasi algoritma terdahulu seperti pada Gambar 12, hal tersebut dikarenakan pada algoritma yang diusukan jumlah objek spasial yang dipanggil, baik objek kandidat *skyline* maupun objek pedukung lebih sedikit dibandingkan dengan algoritma terdahulu dikarenakan pada algoritma yang diusulkan objek spasial yang berada diluar area *streaming* tidak akan dipertimbangkan sebagai objek pendukung serta objek spasial yang berada diluar area pencarian tidak akan dipertimbangkan sebagai objek kandidat *skyline*. Seperti pada [27] dan [28], jumlah data yang dipanggil pada sumber data spasial dapat dibatasi dengan berbagai cara diantaranya penentuan radius pemanggilan serta penentuan langsung jumlah objek yang dipanggil. Jumlah objek yang dipanggil akan berdampak pada beban *bandwidth* dan sumber daya komputasi yang diperlukan.

C. Pengaruh Jumlah Simpul Pekerja Terhadap Waktu Komputasi Klaster

Pengujian ketiga membandingkan waktu komputasi algoritma berdasarkan jumlah simpul pekerja yang digunakan pada komputasi klaster. Pengujian menggunakan nilai *default* pada masing-masing parameter seperti pada Tabel 1.

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

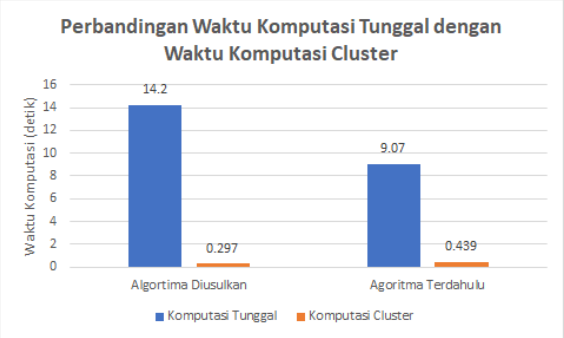


Gambar 13. Pengaruh jumlah simpul pekerja terhadap waktu komputasi klaster

Pengujian algoritma pada masing-masing jumlah simpul pekerja di lingkungan komputasi klaster, menunjukan peningkatan jumlah simpul pekerja sebanding dengan peningkatan waktu komputasi baik pada algoritma yang diusulkan, maupun algoritma tedahulu seperti pada Gambar 13. Selaras dengan yang tertera pada [29] dan [30] bahwa peningkatan jumlah simpul pekerja, sebanding dengan peningkatan *resource* komputasi baik pada sisi memori (RAM) ataupun pada sisi jumlah inti komputasi. Sehingga sumber daya komputasi yang meningkat seiring dengan peningkatan performa komputasi.

D. Perbandingan Waktu Komputasi Tunggal dengan Komputasi Klaster

Selanjutnya kami meninjau waktu komputasi algoritma yang diusulkan dan algoritma terdahulu pada lingungan komputasi tunggal dan komputasi klaster dan membandingkan waktu komputasi pada masing-masing lingkungan komputasi. Sedangkan parameter yang digunakan bernilai *default* seperti pada Tabel 1.

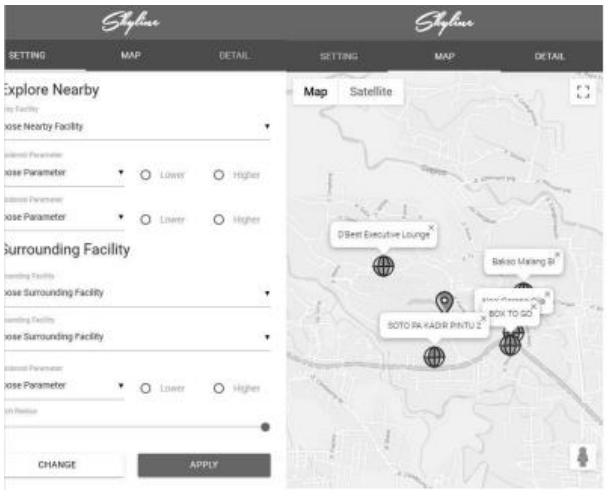


Gambar 14. Perbandingan komputasi tunggal dengan komputasi klaster

Gambar 14 menunjukan komputasi klaster lebih unggul dibandingkan dengan komputasi tunggal, hal tersebut dikarenakan jumlah sumer daya inti pemroses serta memory (RAM) yang dimiliki komputasi klaster lebih unggal dibandingkan dengan komputasi tunggal. Pada [31] dan [32] dijelaskan bahwa kelemahan komputasi klaster pada sisi waktu komunikasi data antar simpul pekerja tidak terlalu mempengaruhi waktu komputasi secara signifikan dikarenakan lingkungan komputasi klaster dibangun pada LAN dengan media transmisi berupa kabel UTP.

E. Implementasi Algoritma pada Aplikasi Perangkat Bergerak

Tahap akhir, algoritma yang diusulkan diimplementasikan pada aplikasi perangkat bergerak, dimana aplikasi perangkat bergerak menampilkan *user* antarmuka pengguna dalam menentukan preferensi pencarian objek *skyline* seperti yang ditunjukan pada Gambar 15, sedangkan komputasi dilakukan pada lingkungan *cloud*, dimana lingkungan *cloud* menerapkan komputasi klaster.



Gambar 15. *User interface* aplikasi perangkat bergerak

Pengujian aplikasi perangkat bergerak dilakukan menggunakan metode kotak hitam yang dilakukan oleh lima orang pengguna, selanjutnya pengguna menguji fungsionalitas, kegunaan dan performa dari aplikasi. Dari hasil pengujian didapatkan nilai fungsionalitas 4,97/5, nilai kegunaan 4,61/5 dan performa 4,79/5. Merujuk pada penelitian [32] bahwa pengujian sistem yang memiliki nilai 80% dari keseluruhan skor menyeluruh, dianggap layak untuk digunakan.

IV. KESIMPULAN

Dari penelitian yang dilakukan, disimpulkan bahwa algoritma yang diusulkan dapat memproses data spasial *skyline* secara streaming secara *real-time*. Serta pemanfaatan komputasi klaster berbasis *cloud* dalam memproses data spasial secara *real-time*, memungkinkan pengimplementasian algoritma yang diusulkan pada aplikasi perangkat bergerak.

Berdasarkan kesimpulan tersebut, pekerjaan menantang berikutnya adalah memanfaatkan sumber data yang lebih lengkap dibandingkan dengan *Google Place API*, menguji komputasi *klaster* berbasis *cloud* seperti yang disedikan pada *Google Cloud Performance (GPC)* ataupun *Amazon Web Service (AWS)*.

DAFTAR PUSTAKA

[1] C. Zhiming, M. S. Arefin, and Y. Morimoto, “Skyline

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

queries for spatial objects: A method for selecting spatial objects based on surrounding environments," in *2012 Third International Conference on Networking and Computing*, 2012, pp. 215–220.

[2] C. Kalyvas and M. Maragoudakis, "Skyline and reverse skyline query processing in spatialhadoop," *Data \& Knowl. Eng.*, vol. 122, pp. 55–80, 2019.

[3] C. Li, A. Zaman, Y. Morimoto, and others, "MapReduce-based computation of area skyline query for selecting good locations in a map," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 4779–4782.

[4] S. Elmi and J.-K. Min, "Spatial skyline queries over incomplete data for smart cities," *J. Syst. Archit.*, vol. 90, pp. 1–14, 2018.

[5] B. Jiang and X. Du, "Personalized travel route recommendation with skyline query," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2018, pp. 549–554.

[6] X. Zhu, J. Wu, W. Chang, G. Wang, and Q. Liu, "Authentication of skyline query over road networks," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, 2018, pp. 72–83.

[7] A. Annisa and L. Angraeni, "Location Selection Query in Google Maps using Voronoi-based Spatial Skyline (VS2) Algorithm," *J. Online Inform.*, vol. 6, no. 1, pp. 25–32, 2021.

[8] A. Zaman, Y. Morimoto, and others, "Area skyline query for selecting good locations in a map," *J. Inf. Process.*, vol. 24, no. 6, pp. 946–955, 2016.

[9] M. Fort, J. A. Sellarès, and N. Valladares, "Nearest and farthest spatial skyline queries under multiplicative weighted Euclidean distances," *Knowledge-Based Syst.*, vol. 192, p. 105299, 2020.

[10] B. Shen, M. S. Islam, D. Taniar, and J. Wang, "Direction-based spatial skyline for retrieving surrounding objects," *World Wide Web*, vol. 23, no. 1, pp. 207–239, 2020.

[11] R. Amin, T. Djatna, I. S. Sitanggang, and others, "Recommendation System based on Skyline Query: Current and Future Research," in *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*, 2020, pp. 1–7.

[12] R. Tan and W. Si, "Privacy-Preserved Spatial Skyline Queries in Location-Based Services," in *Transactions on Computational Science XXX*, Springer, 2017, pp. 50–72.

[13] B. Shen, S. Islam, and D. Taniar, "Direction-based spatial skyline for retrieving arbitrary-shaped surrounding objects," *Comput. J.*, vol. 63, no. 11, pp. 1668–1688, 2020.

[14] T. Djatna, F. H. Putra, and others, "An Implementation of Area Skyline Query to Select Facilities Location Based on User's Preferred Surrounding Facilities," in *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2020, pp. 15–20.

[15] D. Namiot, "On big data stream processing," *Int. J. Open Inf. Technol.*, vol. 3, no. 8, 2015.

[16] X. Li, Y. Wang, X. Li, and Y. Wang, "Parallel skyline queries over uncertain data streams in cloud computing environments," *Int. J. Web Grid Serv.*, vol. 10, no. 1, pp. 24–53, 2014.

[17] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M.- Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 363–372.

[18] Y. Fu, J. Karstensen, and P. Brandt, "On the meridional ageostrophic transport in the tropical Atlantic," *Ocean Sci.*, vol. 13, no. 4, pp. 531–549, 2017.

[19] M. Armbrust *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 1383–1394.

[20] T. De Matteis and G. Mencagli, "Parallel patterns for window-based stateful operators on data streams: an algorithmic skeleton approach," *Int. J. Parallel Program.*, vol. 45, no. 2, pp. 382–401, 2017.

[21] N. Sarkas, G. Das, N. Koudas, and A. K. H. Tung, "Categorical skylines for streaming data," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 239–250.

[22] G. Ruan, E. Wernert, T. Gniady, E. Tuna, and W. Sherman, "High performance photogrammetry for academic research," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018, pp. 1–8.

[23] M. Zaharia *et al.*, "Apache spark: a unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[24] S. Allam, "Usage of Hadoop and Microsoft Cloud in Big Data Analytics: An Exploratory Study," *Sudhir Allam.(2018). USAGE HADOOP MICROSOFT CLOUD BIG DATA Anal. AN Explor. STUDY. Int. J. Innov. Eng. Res. Technol.*, vol. 5, no. 10, pp. 27–32, 2018.

[25] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, 2003, vol. 3, pp. 717–719.

[26] L. Lapon, K. Ooms, and P. De Maeyer, "The influence of the Web Mercator projection on the global-scale cognitive map of web map users," in *International Conference on Spatial Information Theory*, 2017, pp. 79–82.

[27] X. Wang, W. Zhang, Y. Zhang, X. Lin, and Z. Huang, "Top-k spatial-keyword publish/subscribe over sliding window," *VLDB J.*, vol. 26, no. 3, pp. 301–326, 2017.

[28] L. Zhang, J. Zhao, and W. Li, "Online and unsupervised anomaly detection for streaming data using an array of sliding windows and PDDs," *IEEE Trans. Cybern.*, 2019.

[29] U. Sopaoglu and O. Abul, "A top-down k-anonymization implementation for apache spark," in *2017 IEEE international conference on big data (big data)*, 2017, pp. 4513–4521.

[30] O. A. Sarumi, C. K. Leung, and A. O. Adetunmbi, "Spark-based data analytics of sequence motifs in large omics data," *Procedia Comput. Sci.*, vol. 126, pp. 596–605, 2018.

[31] I. Mavridis and H. Karatza, "Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark," *J. Syst. Softw.*, vol. 125, pp. 133–151, 2017.

[32] L. Shi, X. Meng, E. Tseng, M. Mascagni, and Z. Wang, "SpaRC: scalable sequence clustering using Apache Spark," *Bioinformatics*, vol. 35, no. 5, pp. 760–768, 2019.

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021