Perancangan Improvisasi Arsitektur *Web Crawler* Berbasis *Multi-Threading* dan *Multi-Processing* Dengan Menggunakan Bahasa Pemrograman *Rust*

Muhammad Daffa Haryadi Putra, Muhammad Eka Suryana, Med Irzal
Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Jakarta
Jakarta Timur, Indonesia
daffahr15@protonmail.com, eka-suryana@unj.ac.id, medirzal@unj.ac.id

***Abstrak*—Mesin pencari atau *search engine* merupkan *software* yang digunakan untuk melakukan pencarian terhadap informasi tertentu. Untuk menjalankan proses pencarian diperlukan jumlah data yang banyak yang terkumpul dan dapat diakses dengan mudah, proses pengumpulan data ini lah yang disebut *crawling*. Penelitian ini mencoba untuk memperbaiki kekurangan-kekurangan dari *crawler* versi lazuardy dengan penekanan dalam efisiensi peforma dan penggunaan *computing resource*. Penelitian ini menggunakan metode *multi-threading* dan *multi-processing* untuk membagi beban tugas kerja dari *crawler* menjadi dua modul yaitu, *scouter* dan *parser*, selain itu algoritma *breadth-first search* yang digunakan dalam *crawler* dimodifikasi untuk membatasi halaman web apa yang dapat di jelajahi oleh *crawler*. Hasil akhir dari penelitian ini menunjukkan bahwa terdapat improvisasi dengan metode baru ini sebesar 17x dibandingkan dengan *crawler* orisinil, dengan catatan penyeratan halaman terunduh antar *domain* belum berhasil.**

***Kata Kunci*—Search Engine, Web Crawler, Rust Programming Language, Multi-Thread, Multi-Process**

I. Pendahuluan

Search engine merupakan sebuah program yang digunakan untuk menjelajahi dan mencari informasi dari web [1]. Terdapat beberapa komponen yang membangun arsitektur *Search engine* seperti *Web crawler*, *Page rank*, dan *indexer* [2]. Dalam proses pencarian web yang dilakukan *Search engine* tahap pertama yang di lakukan adalah *Web crawler* menjelajahi dan mengekstraksi data-data dari list *url* lalu menyimpan data tersebut dan data lain yang terkait ke dalam database [2]. Data yang disimpan akan di-index, diberikan skor dan di urutkan melalui algoritma *pagerank* [2] *Web Crawler* merupakan komponen penting dalam pembuatan arsitektur *Search engine* secara keseluruhan. Penelitian sebelumnya yang telah dilakukan oleh *Lazuardy Khatulisitwa* telah berhasil mengimplementasikan *Web crawler* kedalam arsitektur *Search engine* yang berjalan [3]. *Web crawler* tersebut mengimplentasikan algoritma *Breadth First Search* dengan modifikasi algoritma *similiarity based* untuk meningkatkan akurasi dari proses *crawling* dan pengambilan data dari suatu halaman [3]. Algoritma *Modified Similarity-Based* yang digunakan oleh *Fathan* untuk memperbaiki akurasi dari *Breadth First Search* memanfaatkan konsep penyimpanan *queue* dalam melakukan proses *crawling* [4]. Dalam proses tersebut *crawler* akan menyimpan 2 jenis *queue* yaitu, *hot queue* untuk menyimpan *url* yang mengandung kata *anchor* sedangkan *url queue* digunakan untuk menyimpan *url* lain [5]. Proses ini dapat membantu *crawler* untuk mengunjungi dan melakukan *crawling* ke dalam *page* yang terdapat di *hot queue* terlebih dahulu bila *page* yang berkaitan dengan kata *anchor* di kunjungi terlebih dahulu maka *child page*-nya kemungkinan besar akan memiliki konten yang berkaitan dengan kata *anchor* tersebut [5].

Arsitektur dari *crawler* yang di kembangkan oleh Lazuardi, menggunakan *python* sebagai bahasa pemograman dan *library* pendukung yang digunakan adalah *beautifulsoup4* untuk melakukan *parsing* dari halaman *website*, *request* untuk mengirimkan request kepada halaman *website* yang ingin di ambil data-nya, dan *regex* untuk melakukan pencocokan kata - kata yang telah di dapat dengan *keyword* yang sudah di tentukan [3]. Dari hasil penelitian *lazuardi* terdapat beberapa saran peningkatan yang tercatat, dimana salah satunya terkait dengan meningkatkan kinerja dan peforma dari *web crawler* agar memiliki penggunaan *RAM* yang lebih kecil dan mencapai kinerja yang maksimal [3].

Salah satu metode untuk mempercepat jalannya *search engine* adalah *Multi-threading* [6]. Metode ini sudah pernah digunakan dalam *search engine* sebelumnya, tetapi *search engine* ini mencari data bukan ke *web* tetapi pada kumpulan data teks atau dapat disebut dengan nama *text search* [6]. Dari hasil penelitian tersebut ditemukan metode *multi-threading* yang digunakan berhasil mencapai improvisasi yang sebelumnya membutuhkan waktu 16 menit dalam menjelajahi seluruh data teks menjadi 4 menit, yang berarti berhasil mencapai improvi-

asi waktu eksekusi program sebesar 4x [6]. Dalam penelitian tersebut metode *multi-threading* digunakan untuk memecah proses pengambilan data dari sumber data dan proses parsing dari data teks yang sudah di ambil [6].

Dalam konteks *search engine* untuk pencarian web penelitian yang dilakukan oleh *Pramudita, Y.D et all* telah menunjukkan bahwa mekanisme *multi-threading* dapat diimplementasi dengan benar [7]. Dalam penelitian tersebut tiap-tiap *thread* menjalankan satu *instance* dari *crawler* nya itu sendiri, dan penelitian tersebut berhasil mencapai percepatan waktu *crawling* selama 123 detik [7].

Selanjutnya penelitian hanya akan melakukan improvisasi terhadap komponen *web crawler* saja untuk membatasi area penelitian. Penelitian ini akan berusaha untuk meningkatkan performa, yang dimana merupakan jumlah halaman yang terkumpul pada waktu yang sudah definisikan. Berdasarkan hasil penelitian *Pramudita, Y.D et all*, yang dimana menjalankan keseluruhan proses *crawler* dalam satu thread [7], maka penelitian ini akan berusaha untuk meningkatkan performa dengan memisahkan proses *parsing* dalam *crawler* dalam proses yang berbeda atau yang dapat disebut dengan metode *multi-processing*. Selain itu penelitian ini juga akan berusaha untuk meningkatkan akurasi hasil proses *crawling* dengan menggunakan algoritma *breadht-first search* yang dimodifikasi dengan tujuan agar *crawler* hanya menjelajahi domain yang telah ditentukan saja, sehingga diharapkan hasil proses *crawling* hanya akan berisi halaman web yang diinginkan. Perbaikan lain yang akan dilakukan adalah dengan menggunakan bahasa pemograman dengan waktu eksekusi yang lebih cepat, yaitu *rust* [8]. Keputusan ini didasari dari hasil pengujian bahasa pemograman *rust* dalam proses dengan intensitas tinggi dan konteks *low-level* [8].

II. Kajian Pustaka

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis.

$$a + b = \gamma \tag{1}$$

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

III. Desain Model

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

IV. Hasil dan Pembahasan

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opine-

mur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

References

[1] T. Seymour, D. Frantsvog, and S. Kumar, "History of search engines," *International Journal of Management & Information Systems (IJMIS)*, vol. 15, no. 4, pp. 47–58, 2011.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," 1998.

[3] L. Khatulistiwa, "PERANCANGAN ARSITEKTUR SEARCH ENGINE DENGAN MENGINTEGRASIKAN WEB CRAWLER, ALGORITMA PAGE RANKING, DAN DOCUMENT RANKING," 2023.

[4] M. F. Qorriba, "PERANCANGAN CRAWLER SEBAGAI PENDUKUNG PADA SEARCH ENGINE," 2021.

[5] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," 1998.

[6] G. Sun, H. Xiang, and S. Li, "On Multi-Thread Crawler Optimization for Scalable Text Searching," 2019.

[7] Y. D. Pramudita, D. R. Anamisa, S. S. Putro, and M. A. Rahmawanto, "Extraction System Web Content Sports New Based On Web Crawler Multi Thread," *Journal of Physics: Conference Series*, vol. 1569, no. 2, p. 22077–22078, Jul. 2020.

[8] Y. Lin, S. M. Blackburn, A. L. Hosking, and M. Norrish, "Rust as a language for high performance GC implementation," *SIGPLAN Not.*, vol. 51, no. 11, pp. 89–98, Jun. 2016, doi: 10.1145/3241624.2926707.