Kehandalan Data Reading Cache Engine MyBatis Framework Terhadap Algoritma LRU, FIFO, SOFT dan WEAK dalam Lingkungan Java dan MySQL

Arief Ginanjar[1], Benie Ilman[2]

[1,2] Fakultas Teknik, Program Studi Informatika, Universitas Langlangbuana

[1] arief.ginanjar@unla.ac.id, [2] benie.ilman@gmail.com

Abstrak

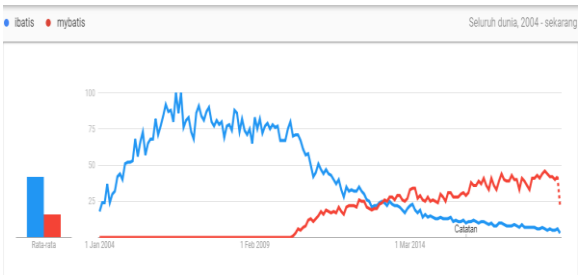
Terdapat beberapa framework yang dapat digunakan dalam membuat aplikasi dalam lingkungan pemrograman java, baik dalam aplikasi web atau aplikasi desktop. Ketika peneliti fokus terhadap java web framework terdapat satu framework yang sudah populer sejak tahun 2004 yaitu Spring Framework, dengan kemampuan kemudahan dalam penggabungan dengan berbagai framework seperti Hibernate Framework, Ibatis atau MyBatis Framework serta beberapa framework lain. Penelitian ini dilakukan untuk mengetahui perbandingan kemampuan membaca data dari aplikasi web service yang dibangun menggunakan bahasa pemrograman java dengan Spring Framework dan dikombinasikan dengan MyBatis Framework. Dengan memaksimalkan manfaat dari Cache Engine dalam MyBatis Framework, penulis ingin mengetahui lebih dalam kehandalan masing-masing algoritma yang terdapat dalam Cache Enine MyBatis Framework yaitu LRU, FIFO, SOFT dan WEAK. Dengan menggunakan Evolutionary Prototyping penelti mencoba menyusun rencana kerja analisis dan pengujian terhadap perangkat lunak yang dibangun, serta pengujian kinerja framework yang terintegrasi dengan web service kemudian diakses oleh aplikasi pengujian third party code yang dilakukan secara berulang-ulang dengan rentang waktu tertentu

Kata kunci : *di Spring Framework, Ibatis Framework, MyBatis Framework, Algoritma Cache Engine, LRU, FIFO, SOFT, WEAK*

1. Pendahuluan

A. Latar Belakang

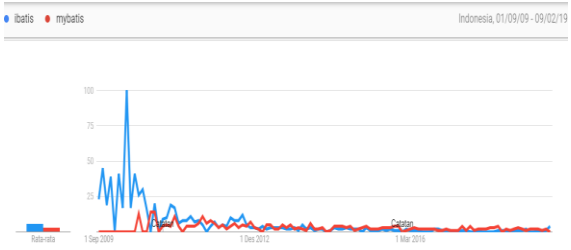
MyBatis Framework merupakan *forking* dari generasi pertama *Database Bridging Layer* berjenis *Query Expose* yang berkembang dan mulai digunakan dalam implementasi pembuatan aplikasi *web* menggunakan *java platform* pada tahun 2010. Seiring waktu perkembangan *MyBatis Framework* semakin diminati oleh para programmer *java* hingga tahun 2019, namun masih kalah populer jika di bandingkan dengan tingkat popularitas *framework* generasi pertama yaitu *iBatis*. Hal tersebut dapat diketahui dari infografis berikut yang menggambarkan tingkat peminatan para programmer *java* terhadap *iBatis* dan *MyBatis* di dunia antara tahun 2004 hingga 2019.



Gambar 1. Popularitas *iBatis dan MyBatis Framework* di dunia dari 2004 hingga 2019 (Sumber: Google Trend).

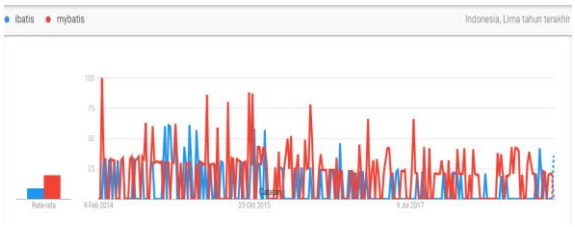
Dari infografik Gambar 1 terlihat bahwa *IBatis Framework* lebih populer antara tahun 2004 hingga 2015, tetapi pada tahun 2010 terlihat *MyBatis Framework* mulai dikenal oleh para programmer untuk diimplementasikan dalam pekerjaan mereka.

Gambar 2. Popularitas *IBatis* dan *MyBatis framework* di dunia dari 2009 hingga 2019 (Sumber: Google Trend).



Gambar 3. Popularitas *IBatis* dan *MyBatis Framework* di Indonesia dari 2009 hingga 2019 (Sumber: Google Trend).

Pada gambar 3 terlihat peminatan *IBatis* dan MyBatis *Framework* oleh para programmer *java* di Indonesia antara periode 2009 hingga 2012 masih fluktuatif dan cenderung lebih banyak peminat *IBatis* pada awal periode, namun pada tahun 2012 hingga sekarang peminatan *MyBatis* dan *IBatis* cenderung menurun.



Gambar 4. Popularitas *IBatis* dan *MyBatis Framework* di Indonesia dari 2014 hingga 2019 (Sumber: Google Trend).

Implementasi penggunaan *IBatis* dan *MyBatis Framework* dalam pengembangan aplikasi berbasis *java*, seringkali dikombinasikan dengan *framework* lain diantaranya adalah; *Spring Framework* dan *Struts Framework* yang menjadi lapisan terluar dan menjadi penghubung *Database Bridging Layer* dengan pengguna sistem yang dibangun menggunakan pemrograman *java*..

B. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian ini adalah mengetahui nilai kehandalan masing-masing spesimen seperti yang tercantum dalam Tabel 1 yang akan diuji dengan serangkaian *scenario* pengujian menggunakan sejumlah variatif jumlah data serta uji tekanan dengan cara melakukan proses *load* dengan beberapa macam jumlah hit *url load*.

Dengan serangkaian pengujian tersebut diharapkan diketahui kehandalan masing-masing algoritma dalam *Cache Engine MyBatis Framework*. Sehingga diharapkan dapat menjadi sumber referensi dalam pengambilan keputusan ketika harus memilih jenis algoritma *Cache MyBatis Framework* yang diterapkan dalam pengembangan aplikasi berbasis java.

TABEL I.
SPESIMEN *FRAMEWORK* YANG AKAN DIUJI

Spesimen	Framework Utama	Cache Framework	Algoritma
1	Spring + MyBatis	Ya	LRU
2	Spring + MyBatis	Ya	FIFO
3	Spring + MyBatis	Ya	SOFT
4	Spring + MyBatis	Ya	WEAK

C. Batasan Masalah

Penelitian ini hanya terfokus terhadap kecepatan *data loading url* dan tidak melakukan pengujian terhadap operasi *insert, update dan delete* dengan spesifikasi teknis sebagai berikut:

- Menggunakan sistem basis data *MariaDB* versi 10.2.31 yang diletakkan terpisah Sistem Operasi *Microsoft Windows* 7 *Genuine* 64 *bit* didalam *Oracle VM Virtual Box* versi 5.2.6.
- Menggunakan koneksi jaringan *local host - guest virtualbox connection*.
- Menggunakan Sistem Operasi *Microsoft Windows* 7 *Genuine* 64 *bit* untuk *host environment*.
- Menggunakan *Netbean* 8.2 sebagai *Integrated Development Editor*.
- Menggunakan *Spring Framework* versi 4.0.1 sebagai *MVC Framework*.
- Menggunakan *Spring-Json taglib* sebagai *library* untuk *output json*.
- Menggunakan *application container Apache Tomcat* versi 9.0.12 sebagai *media deployment application*.
- Menggunakan *Java Virtual Machine* versi 1.8.0.241 64 *bit* dengan *setting core JVM default* tanpa merubah konfigurasi *default* sebagai *platform environment*.

2. Metode Penelitian

A. Pelaksanaan Penelitian

Metodologi yang digunakan pada pembuatan laporan penelitian dengan menggunakan metode penelitian kuantitatif eksplorasi dan *evolutionary prototyping* serta ditambah proses uji kemampuan performa *framework* dengan perulangan proses serta pendekatan sistem terhadap elemen dan komponen [2][6][7], adapun urutan proses dilakukan sebagai berikut:

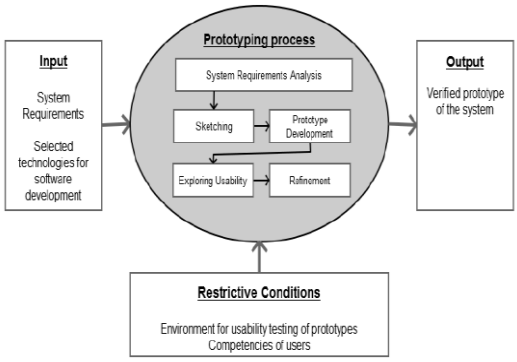
- Studi Literatur, Mempelajari sumber-sumber pustaka yang dapat dijadikan referensi. Sumber-sumber pustaka dapat berupa buku, *paper* atau halaman *web* yang membahas tentang *Spring Framework*, *Mybatis Framework* dan *Cache Engine*.
- Analisis, Menjelaskan bagaimana melakukan analisis terhadap arsitektur serta teknik pemrograman terhadap setiap *framework* serta bagaimana menggabungkan atau mengkombinasikan setiap *framework* tersebut.
- Perancangan Perangkat Lunak, Dalam hal ini peneliti melakukan perancangan terhadap perangkat lunak mengkombinasikan *framework* yang akan dibangun berdasarkan hasil yang diperoleh dari analisis. Serta melakukan analisis perancangan *scenario* untuk memenuhi syarat pengujian yang akan dilakukan.
- Implementasi Perangkat Lunak, yang akan

dikembangkan berdasarkan hasil yang diperoleh dari perancangan dan analisis. Implementasi menghasilkan produk perangkat lunak yang berisi kombinasi *framework* seperti yang telah ditentukan dalam *scenario* uji.

- Pengujian dan Evaluasi, terhadap produk perangkat lunak yang telah dibangun dan kemudian melakukan evaluasi dari setiap *scenario* pengujian yang dilakukan.
- Iteratif merupakan proses berulang terhadap pengembangan perangkat lunak yang sedang dikembangkan ketika proses bisnis belum memenuhi spesifikasi maka lakukan perancangan dan implementasi ulang terhadap perangkat lunak hingga memenuhi syarat yang diinginkan.
- Pengujian Kinerja Aplikasi, proses pengujian aplikasi terhadap tekanan dengan cara *loop loading url* yang dilakukan berkali-kali terhadap spesimen *framework* yang telah dibangun menjadi aplikasi menggunakan metode *url looping script*.

B. Model Evolutionary Prototyping

Proses yang dilakukan peneliti merujuk kepada tahapan *model* evolusioner *prototyping* maka proses tersebut dapat diilustrasikan seperti yang terlihat pada Gambar 5. Proses evolusioner *prototyping* yang terdiri dari proses berikut *input*, *prototyping process*, dan *output*, tetapi selama pelaksanaan proses harus dibatasi dengan syarat; setiap fungsi yang dibangun harus memenuhi syarat yang sesuai dengan *system requirement*, dengan kemampuan user yang terlibat dalam *prototyping* harus memenuhi syarat *system requirement* [8].



Gambar 5 Metodologi *Evolutionary Prototyping* [8].

C. Arsitektur Spesimen

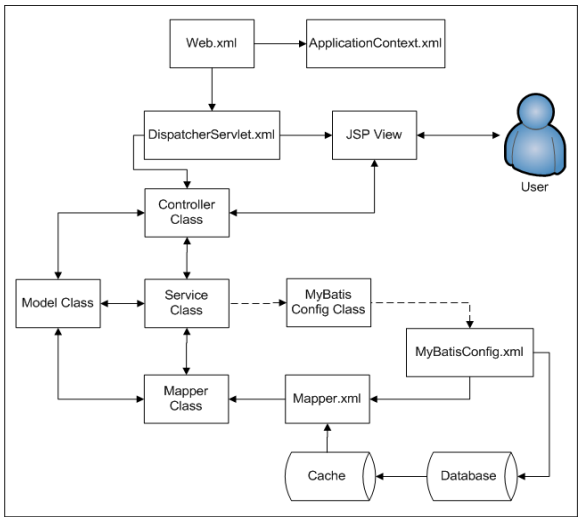
Dengan pendekatan prinsip *Object Oriented Programming* serta pendekatan *Model, View dan Controller* MVC, serta abstraksi data yang memungkinkan *programmer* untuk mengembangkan pemikiran yang rumit tanpa harus fokus terhadap komponen, sedangkan enkapsulasi memungkinkan kita untuk fokus terhadap kemampuan *software* tanpa harus berpikir detil terhadap kerumitan proses yang terjadi. [4][5].

Penggabungan arsitektur *Spring Framework* dan *MyBatis Framework* menjadi sebuah sistem yang saling melengkapi, diharapkan dengan arsitektur yang dibangun dan dengan pengujian yang dilakukan dapat ditemukan titik optimal dalam proses penggabungan tersebut [1][3].

3. Proses Penelitian

A. Perancangan Spesimen

Untuk membangun perangkat lunak sebagai media pengujian spesimen, dibutuhkan penggabungan *framework Spring* dan *MyBatis*, maka peneliti perlu merancang sebuah arsitektur aplikasi sederhana untuk dijadikan sebagai media pengujian performa aplikasi dengan perancangan sebagai berikut:



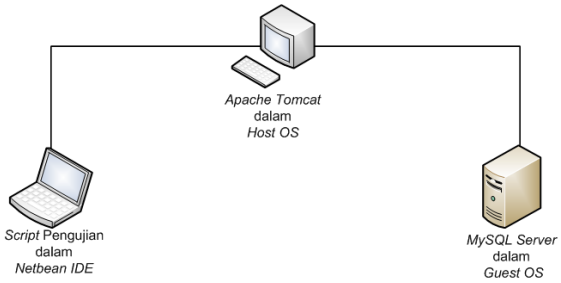
Gambar 6. Arsitektur Aplikasi Menggunakan *Cache* [3]

B. Konfigurasi Spesimen

Alur pengujian setiap spesimen yang disimpan dalam *server oracle virtual box* yang terinstall didalam *guest OS*, kemudian Apache Tomcat disimpan dalam *guest OS*, selanjutnya script pengujian dijalankan menggunakan Netbean IDE versi 8.2 dengan ilustrasi dapat terlihat pada Gambar 8.

Setiap *framework* yang terlihat dalam Tabel 1 membutuhkan juga beberapa *library* lain sebagai *support* terhadap *framework* utama dalam spesimen penelitian, dengan perincian *library* dapat dilihat dalam Tabel 2.

Data yang terdapat dalam Tabel 2 merupakan spesimen pengujian untuk *Spring Framework* dengan versi 4.0.1 dikombinasikan dengan *Java Database Connection* (JDBC) membutuhkan *library* lain yaitu *JSP Standard Tag Library* (JSTL) yang mempunyai fungsi untuk menterjemahkan data yang berasal dari lingkungan *java class* terhadap lingkungan *java web*. [3]



Gambar 7. Alur Pengujian Kinerja *MyBatis Framework* terhadap beberapa *algoritma Cache* [3].

TABEL II.
LIBRARY PELENGKAP DARI SETIAP SPESIMEN

Library	**Spesimen**			
	1	**2**	**3**	**4**
JSTL	√	√	√	√
MySQL JDBC	√	√	√	√
JSON Taglib	√	√	√	√
Ehcache	√	√	√	√

Ehcache merupakan teknologi yang berbasis *hardware* atau *software* yang digunakan sebagai penyimpanan data sementara yang berfungsi untuk membantu tingkat performa dari aplikasi atau sistem. Penerapan teknologi *cache* sudah cukup banyak digunakan oleh para pengembang aplikasi untuk meningkatkan kecepatan akses terhadap *view* data aplikasi, setiap *framework* mempunyai arsitektur berbeda dalam implementasi *cache* terhadap data yang dikelolanya, hal tersebut cukup menarik untuk dijadikan bahan penelitian selanjutnya.

C. Pengumpulan Data

Pengumpulan data yang dilakukan yaitu berupa menjalankan script aplikasi pengujian terhadap spesimen data yang terbangun dari komposisi *framework* seperti yang terlihat dalam Tabel 1 dan terinstall dalam lingkungan *application container* dengan nama produk *Apache Tomcat* versi 9.0.12 kemudian dijalankan menggunakan *test script* dalam lingkungan *Netbean* 8.2 seperti yang terlihat pada Gambar 8.

```
public static void main(String[] args) throws IOException, JSONException {
    // TODO code application logic here
    long secondStart = System.currentTimeMillis();
    JSONArray jsonArray = readJsonArrayFromUrl("http://localhost:8080/Research-Spring-Jdbc/json.htm");
    for (int i = 0; i < jsonArray.length(); i++) {
        //get the JSON Object
        JSONObject obj = jsonArray.getJSONObject(i);
        Integer id = obj.getInt("id");
        String nama = obj.getString("nama");
        Integer anak = obj.getInt("anak");
        System.out.println("id = "+id+", nama = "+nama+", anak = "+anak);
    }
    long secondFinish = System.currentTimeMillis();
    long elapseTime = secondFinish - secondStart;
    System.out.println(" second "+elapseTime);
}
```

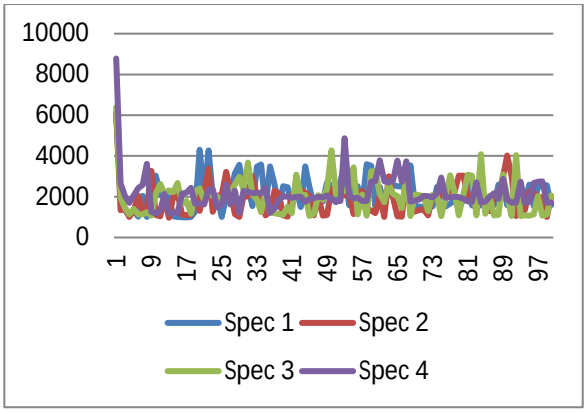
Gambar 8. Salah satu script baris perintah untuk pengujian *performance spring framework*. [3]

D. Hasil Penelitian.

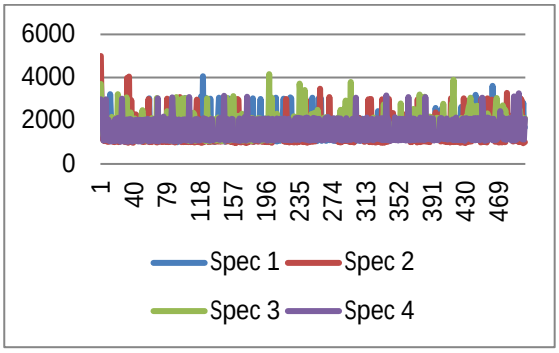
Selama pelaksanaan pengujian kinerja serta perbanding kehandalan algoritma *cache engine MyBatis framework* dengan menggunakan *Spring Framework* sebagai *frontend* ditemukan bahwa ada kecenderungan positif yang terjadi ketika melakukan pengujian terhadap spesimen dua yang menggunakan algoritma *FIFO* dan spesimen empat yang menggunakan algoritma *WEAK*, sedangkan algoritma LRU dan *SOFT* berada dalam posisi moderat dimana tidak menunjukan gejala *low performance* atau *high performance.*

Dari hasil pengujian berdasarkan *scenario* yang telah ditetapkan, ditemukan kondisi bahwa spesimen tiga dan empat tidak semata-mata murni menggunakan algoritma *ehcache* yang berada didalam *MyBatis framework* tetapi juga menggunakan algoritma *SOFT* dan *WEAK* bawaan dari *Java Virtual Machine* (*JVM*).

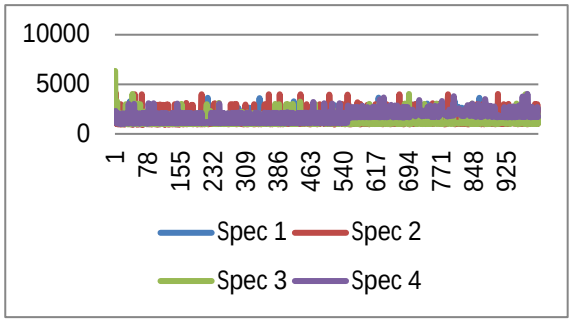
Kondisi tersebut memungkinkan untuk melakukan pengujian empat buah spesimen dengan fokus terhadap data hasil *hit* 100, *hit* 500, *hit* 1000 dan *hit* 2000 untuk memberi keseimbangan hasil data pengamatan untuk seluruh spesimen dan *scenario*. Dalam pembahasan selanjutnya representasi data yang ditampilkan sudah di konversi menjadi diagram garis seperti yang terlihat pada Gambar 9, 10, 11 dan Gambar 12.



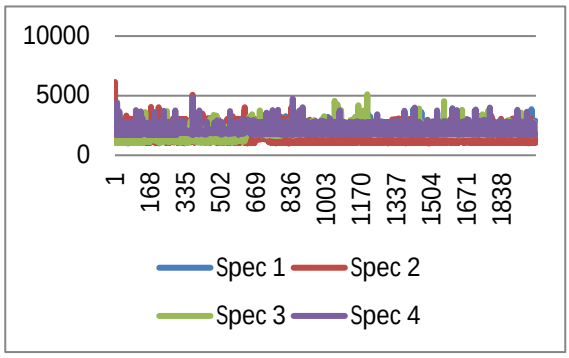
Gambar 9. Hasil Pengujian Scenario *hit* 100.



Gambar 10. Hasil Pengujian Scenario *hit* 500.



Gambar 11. Hasil Pengujian Scenario *hit* 1000



Gambar 12. Hasil Pengujian Scenario *hit* 2000

Dari data-data gambar yang diwakili diagram garis seperti yang terlihat diatas peneliti akan mengambil data pokok yaitu; kecepatan load rata - rata spesimen, kecepatan *load* paling rendah spesimen, kecepatan *load* paling tinggi spesimen, kecepatan *load* bernilai sama yang paling sering muncul di spesimen, serta kecepatan load nilai tengah di spesimen. Seluruh data tersebut tersajikan dalam Tabel 3 berikut.

TABEL III.
HASIL UJI AKSES DARI SETIAP SPESIMEN

Scenario vs Spesimen		Kecepatan *Load* (mili detik)				
		Average	*Slow*	*Fast*	*Modus*	*Median*
Hit	1	2,156.04	6,056.00	986.00	1,580.00	1,996.00
	2	1,888.44	6,149.00	970.00	**2,036.00**	2,016.00
	3	2,012.33	6,393.00	1,046.00	1,127.00	**2,034.50**
	4	**2,177.22**	**8,780.00**	1,177.00	1,614.00	1,992.00
Hit 500	1	1,662.75	4,149.00	1,004.00	1,072.00	1,637.50
	2	1,481.30	5,000.00	958.00	1,021.00	1,119.50
	3	1,811.96	4,160.00	1,040.00	1,741.00	1,816.50
	4	1,473.18	**3,273.00**	1,027.00	1,111.00	1,143.50
Hit 1000	1	1,794.58	3,654.00	1,043.00	1,625.00	1,658.00
	2	1,520.94	4,182.00	**932.00**	**976.00**	1,099.50
	3	**1,420.60**	6,352.00	962.00	1,039.00	**1,070.00**
	4	1,716.71	4,036.00	1,016.00	1,123.00	1,730.50
Hit 2000	1	1,825.14	4,026.00	1,087.00	1,640.00	1,672.00
	2	1,545.26	6,145.00	990.00	1,063.00	1,161.50
	3	1,874.49	5,117.00	997.00	1,763.00	1,786.00
	4	2,039.34	4,857.00	**1,680.00**	1,770.00	1,841.50
Nilai Max		2,177.22	8,780.00	1,680.00	2,036.00	2,034.50
Nilai Min		1,420.60	3,273.00	932.00	976.00	1,070.00

D. Pembahasan Data Hasil Pengujian

Kecepatan data akses terhadap masing - masing spesimen dibandingkan dengan masing-masing *scenario* berdasarkan data dari tabel 3, diketahui bahwa:

- Spesimen satu, dua dan tiga dari setiap pengujian mempunyai kecepatan akses yang cukup signifikan tinggi dalam kecepatan rata-rata setiap *access loop*.
- Spesimen satu dan dua dalam jumlah *hit* 100 dapat mencapai kecepatan *access loop* yang cukup tinggi, tetapi kecepatan menurun ketika berada di *hit* 500, *hit* 1000 dan *hit* 2000.
- Spesimen dua mempunyai kecepatan paling tinggi untuk pengujian *hit* 500 dan kebalikannya untuk kecepatan paling rendah adalah spesimen empat untuk pengujian *hit* 100.
- Untuk rata-rata kecepatan tertinggi dapat dilihat pada spesimen tiga *hit* 1000 dengan kecepatan 1.420,60 mili detik.
- Sedangkan untuk kecepatan tertinggi akses dapat dilihat pada spesimen dua *hit* 1000 dengan kecepatan 932 mili detik.
- Untuk rata - rata kecepatan akses paling rendah dapat dilihat pada spesimen empat *hit* 500 dengan kecepatan 3.273,00 mili detik.

- Sedangkan untuk kecepatan akses paling rendah yaitu spesimen 4 lima *hit* 100 dengan kecepatan 8.780 mili detik..

Dari tujuh pembahasan hasil kesimpulan sementara yang berasal dari Tabel 3, peneliti menarik kesimpulan bahwa spesimen dua unggul dalam kecepatan akses, , diikuti dengan spesimen satu, kemudian diikuti spesimen tiga dan serta terakhir yaitu spesimen empat.

4. Kesimpulan

Penelitian yang telah dilakukan tentang investigasi pengujian performa Algoritma *Cache Engine* dalam *MyBatis Framework* dengan fokus hanya terhadap *data loading url* yang telah dilakukan maka terdapat beberapa hasil yang dapat diketahui yaitu;

- Algoritma *LRU* sangat cocok untuk jenis data loading yang jumlah hit nya sedikit karena mempunyai kecepatan *loading* tinggi tanpa harus ada proses *buffering*.
- Algoritma *FIFO* sejenis dengan *LRU* namun sangat cocok untuk jumlah data yang minimal sehingga tidak membutuhkan *effort buffering*.
- Algoritma *SOFT* dan *WEAK* karena core algoritma berada di sisi *JVM* maka perlu waktu extra untuk proses *loading* algoritma antara *JVM* dengan *cache engine*, maka secara performa lebih rendah dari algoritma LRU dan FIFO..

DAFTAR PUSTAKA

[1] Ginanjar, A., Hendayun, M., 2019, Spring Framework Reliability Investigation Against Database Bridging Layer Using Java Platform, Procedia Computer Science, Elsevier.

[2] Jogiyanto, H.M., 2008, Analisis dan Perancangan Sistem: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, Andi Offset., Yogyakarta.

[3] Ilman, B., Ginanjar, A., 2019, Rancang Bangun Web Service JSON Menggunakan Kombinasi Spring dan MyBatis Framework dalam Lingkungan Java Platform, Jurnal Teknologika., STT Wastukancana, Purwakarta.

[4] Kalelkar, Medha., Churi, Prathamesh., Kalelkar, Deepa., 2014, Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture., International Journal of Computer Application., New York, USA.

[5] Kendal, Simon., 2009, Object Oriented Programming using Java., Ventus Publishing Aps, Frederiksberg, Denmark.

[6] Sugiyono., 2012, Metode Penelitian Kuantitatif Kualitatif dan R&D., Alfabeta, Bandung.

[7] Roger, S. Pressman, Ph.D., 2012, Rekayasa Perangkat Lunak, Edisi 7 : Buku 1 Andi Offset, Yogyakarta.

[8] Nacheva, Radka, 2017, Prototyping Approach in User Interface Development, Conference on Innovative Teaching Methods, University of Economics Varna, Bulgaria.