PERBANDINGAN KINERJA HBASE DAN MONGODB SEBAGAI *BACKEND* IOT DATA *STORAGE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Niki Yuniar Wicaksono
NIM: 135150200111082

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PERBANDINGAN KINERJA HBASE DAN MONGODB SEBAGAI *BACKEND* IOT DATA *STORAGE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Niki Yuniar Wicaksono
NIM: 135150200111082

Skripsi ini telah diuji dan dinyatakan lulus pada
01 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Sakti Pramukantoro, S.Kom., M.Kom	Widhi Yahya, S.Kom., M.Sc
NIK: 201102 860805 1 001	NIK: 201607 891121 1 00 1

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 01 Agustus 2018

Niki Yuniar Wicaksono

NIM: 135150200111082

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga laporan skripsi yang berjudul "Perbandingan Kinerja Hbase dan MongoDB sebagai *Backend IoT Data Storage*" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak yang telah memberikan bantuan baik lahir maupun batin selama penulisan skripsi ini. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Eko Sakti P.,S.Kom.,M.Kom dan bapak Widhi Yahya, S.Kom., M.Sc selaku dosen pembimbing skripsi penulis yang dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Informatika
3. Bapak Agus Wahyu Widodo, S.T, M.Cs, selaku Ketua Program Studi Teknik Informatika
4. Kedua orang tua dan seluruh keluarga besar atas segala nasehat, kasih sayang dan kesabaran dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
5. Teman satu topic, adam dan maxi yang selalu menjadi support system pengerjaan skripsi.
6. Teman-teman terdekat Gabriel Arganata, Hudan Abdur Rohman, Muhammad Fauzi, Ferdy Wahyu Rianto, Abyan Safitra, Novelasari, Putri Nur Fadilla, Rizky Kharisma.
7. Teman, sekaligus sahabat "Tetap di Hati" Ega, Abi, Bayu, Zam, yang selalu menjadi penyemangat selama menjalani perkuliahan dan penyelesaian skripsi.
8. Teman-teman "Topic Data IoT" yang selalu Bersama dan memberi dukungan selama pengerjaan skripsi.
9. Keluarga BEM Bersatu 2 dan Bersatu 3 yang sudah memberikan banyak pengalaman selama masa perkuliahan.
10. Keluarga PSDM Bersatu 2, dan "FUNFAMS" yang sudah memberikan banyak pengalaman, cerita selama masa perkuliahan.
11. Teman-teman Teknik Informatika angkatan 2013 yang selalu mendukung dan berbagi ilmu dari awal perkuliahan sampai tahap akhir penyelesaian skripsi.
12. Teman -Teman "KKN ASYIK" Hudan, Adi, Ivan, Bayu, dan Christy sebagai teman seperjuangan magang dari awal hingga akhir pengerjaan skripsi.
13. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya

yang telah banyak memberi bantuan dan dukungan selama penyelesaian skripsi ini.
14. Semua pihak yang telah membantu dan berbagi ilmu dalam penyelesaian skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 01 Agustus 2018

Niki Yuniar Wicaksono
Nikiyw05@gmail.com

ABSTRAK

Niki Yuniar Wicaksono, Perbandingan Kinerja Hbase dan MongoDB Sebagai *Backend IoT Data Storage*

Dosen Pembimbing: Eko Sakti Pramukantoro., S.Kom, M.Kom dan Widhi Yahya., S.Kom., M.Sc.

IoT *(Internet Of Things)* saat ini sedang berkembang dibuktikan dengan banyaknya penelitian yang membahas IoT. IoT terdiri dari dua komponen utama yaitu Internet dan *Things* yang saling terhubung, untuk mengumpulkan data dan informasi kedalam media penyimpanan data yang bisa diolah sesuai kepentingan. Namun terdapat tantangan dalam membangun media penyimpanan data IoT, yaitu volume data yang besar, bentuk dan format data yang beragam. Pada penelitian sebelumnya telah dikembangkan sebuah *Framework* media penyimpanan data IoT dari node sensor. *Framework* tersebut menggunakan *NoSQL mongoDB* sebagi media penyimpanan datanya. Pada penelitian ini, diusulkan sebuah *IoT data storage* NoSQL Hbase dengan menggunakan *Framework MongoDB* karena *less scema* dan *writing speed* yang bagus dalam pengujian *performa random writing test*. Adapun parameter uji kinerjanya menggunakan *Runtime*, Throughtput, *Memory Usage*, *CPU Usage* dan *Disk I/O* dari *Database* server ketika melakukan operasi *insert text* dan gambar. Sedangkan untuk pengujian *Framework* menggunakan pengujian penyimpanan data. Untuk pengujian kinerja MongoDB unggul atas uji *text* pada parameter runtime sebesar 33s, throughput 1559 ops/s, CPU Usage 48%, Memory Usage 67%, dan Disk I/O 4354kb. Saat uji gambar hasil perbedaan berbeda sedikit keunggulan mongoDB pada parameter Runtime 64s, Throughput 90 ops/s, CPU Usage 46%, Memory Usage 84%, sedangkan saat uji gambar Hbase unggul pada parameter *Disk I/O* sebesar 109.462kb.

Kata kunci: IoT *(Internet Of Things), Hbase, MongoDB, Kinerja, Data Storage*

ABSTRACT

Niki Yuniar Wicaksono, Perbandingan Kinerja Hbase dan MongoDB Sebagai *Backend IoT Data Storage*

Dosen Pembimbing: Eko Sakti Pramukantoro., S.Kom, M.Kom dan Widhi Yahya., S.Kom., M.Sc.

IoT (Internet Of Things) is currently evolving as evidenced with many studies that discuss the IoT. IoT consists of two main components of the Internet and interconnected Things, to collect data and information into data storage media that can be processed as needed. However, there are challenges in building IoT data storage media,like a large data volumes, diverse forms and data formats. In previous research has developed a Framework of IoT data storage media from the sensor node. The Framework uses NoSQL mongoDB as its data storage. In this study, proposed an IoT NoSQL Hbase data storage using the MongoDB Framework because it free scema and good writing speed in the performance testing of random writing tests. The parameters for comparing performance are Runtime, Throughtput in performing insert operations, as well as Memory Usage, CPU Usage and Disk /O from the Database server when performing insert operations. While for testing the Framework of data storage media with Hbase using storage testing For testing the performance of MongoDB, better result test on runtime parameters of 33s, throughput of 1559 ops/s, 48% Usage CPU, 67% Memory Usage, and 4354kb Disk I/O. When testing images the results of differences in slightly different better of mongoDB on Runtime parameters 64s, throughput 90 ops/s, CPU Usage 46%, Memory Usage 84%, while when testing Hbase images superior in Disk I/O parameters of 109.462kb.

Keywords: IoT *(Internet Of Things), Hbase, MongoDB, Performance, Data Storage*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

IoT (I*nternet of Things*) saat ini sedang berkembang dibuktikan dengan berbagai penetian yang membahas IoT. IoT pada dasarnya terdiri dari dua komponen utama yaitu internet dan *Things* yang saling terhubung. IoT dapat di *representasikan* sebagai satu set *Things* yang saling terkoneksi melalui internet. *Things* dapat berupa seperti sensor, peralatan rumah tangga, *tags*, dan lain sebagainya. IoT dapat mengumpulkan data dan informasi dari lingkungan fisik(*environment*), yang selanjutnya data tersebut akan diproses sesuai dengan kepentingan dan kebutuhan di setiap bidangnya. Konsep IoT mengacu pada tiga elemen utama, yaitu barang fisik yang dilengkapi modul IoT, perangkat koneksi ke internet seperti modem dan router, dan *Cloud data center* tempat menyimpan aplikasi beserta *data storage*. Jika membahas mengenai data maka tidak lepas dari media penyimpanan data tersebut. Dalam hal ini kita menyebutnya sebagai database. (Pramukantoro. Dkk.,,2017)

Terdapat tantangan dalam membangun media penyimpanan data IoT, yaitu volume data yang besar, serta bentuk dan format data yang beragam. Pada penelitian sebelumnya (Pramukantoro.Dkk., 2017) telah dikembangkan sebuah *Framework* media penyimpanan data IoT dari node sensor. *Framework* tersebut menggunakan NoSQL mongoDB sebagi media penyimpanan datanya. Skenario pengujian yang digunakan untuk menguji *Framework* tersebut meliputi *functional testing, scalability testing dan respon time testing* dengan *IoT Apps*. Hasil yang didapat dari functional testing adalah *Framework* yang dikembangkan dapat mengirim dan menyimpan data bervolume besar dan beragam. Sedangkan dari *non-functional testing*, *Framework* yang dikembangkan dapat menerima 443 data /detik dari IGD*(Internet Gateway Device*), dapat mengirimkan 173 data/detik ke IoT Apps dan *respon time* yang didapat dibawah 1 detik.

Saat ini telah terdapat 225 jenis NoSQL database yang telah dikembangkan. Setiap tipe database pada NoSQL memiliki mekanisme implementasi, karakteristik penyimpanan, konfigurasi dan optimasi yang berbeda-beda. Ini yang menyebabkan semakin terbukanya tantangan dalam pemilihan NoSQL yang akan digunakan(EnqiqTang & Yushun Fan 2016). Oleh sebab itu, NoSQL MongoDB yang digunakan pada penelitian sebelumnya (Pramukantoro. Dkk.,2017) perlu dibandingkan dengan NoSQL database yang lain.

Diantara 225 jenis NoSQL database, HBase juga cukup populer digunakan untuk mengelola data dalam jumlah besar yang setiap harinya dihasilkan dari bermacam - macam sumber. Dalam penelitian ini Hbase dipilih karena didesain untuk dapat mengelola data berukuran bersar dalam satu sistem terdistribusi karena berjalan diatas Hadoop, lebih tepatnya *Hadoop Distributed File System*(HDFS) dan memiliki fungsi *sharding original* bawaan yang dapat bekerja secara otomatis maupun manual.

Hbase memiliki karakteristik '*fault tolerance*' atau mampu menjamin keutuhan data meskipun terjadi kegagalan pada beberapa komputer yang dikerjakannya (Wijaya, Wayan, M, 2015). Hbase mempunyai keunggulan untuk menyimpan data yang beragam karena *less scema atau free structure dan writing speed* yang cukup bagus dalam pengujian kinerja *random writing test*. (Naheman, W & Wei, J, 2013).

Dengan demikian akan dilakukan perbandingan kinerja database yang akan buat pada penelitian saat ini yaitu Hbase dengan mongoDB sebagai media penyimpanan data IoT. Penelitian ini dibuat pada lingkungan *(environment / Framework)* yang telah dikembangkan oleh peneliti sebelumnya. Hasil pengujian nantinya akan didapat sebagai penilaian kinerja kedua database. Adapun parameter ujinya menggunakan *Runtime, Throughtput, CPU Usage, Memory Usage dan Disk I/O* dari database server ketika melakukan operasi *insert*. Sedangkan untuk pengujian *media* penyimpanan data dengan Hbase, dilakukan pengujian penyimpanan *database*. Penelitian ini diharapkan menjadi pertimbangan pengguna *database sql* maupun *nosql* dalam membangun sistem pada penyimpanan data IoT.

1.2 Rumusan masalah

Berdasarkan penjelasan latar belakang di atas, maka rumusan masalah yang dapat diangkat yaitu :

1. Bagaimana menerapkan *data storage NoSQL* Hbase pada *environment* penelitian mongoDB "*Topic Based IoT Data Storage Framework for Heterogeneous Sensor data*"?
2. Bagaimana hasil kinerja database *NoSQL HBase dan MongoDB* pada *IoT data Storage*?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian :

1. Menggunakan Hbase pada lingkungan sebelumnya dengan menggunakan data yang beragam dari node sensor.
2. Menguji kinerja berdasarkan *Runtime, Throughput, CPU Usage, Memory Usage dan Disk I/O* pada databases *NoSqL mongoDB* dan *HBase* pada media penyimpanan *IoT.*
3. Mengetahui hasil dari perbandingan kinerja *NoSqL mongoDB dan HBase* sebagai backend IoT datastorage.

1.4 Manfaat

Manfaat yang diperoleh dari penelitian adalah untuk mengetahui kinerja database antara HBase dan mongoDB dalam penyimpanan data sensor dengan volume besar

dan beragam pada *IoT data Storage*. Selain itu menjadi pertimbangan pengguna database sql maupun nosql dalam membangun media penyimpanan data.

1.5 Batasan masalah

Batasan masalah dalam penelitian ini diberikan dengan tujuan agar pembahasan lebih rinci dan tidak meluas. Adapun penelitian ini dibatasi oleh hal-hal sebagai berikut :

1. Lingkungan penelitian yang digunakan adalah lingkungan penelitian pada MongoDB.
2. Penelitian menggunakan hbase untuk membandingkan database pada penelitian MongoDB.
3. Lingkungan pengujian kinerja database dilakukan pada Virtual Server yang ada di FILKOM.
4. Pengujian pada penelitian ini lebih difokuskan pada sisi media penyimpanan data.
5. Pengujian dilakukan dengan dua scenario, yaitu pengujian penyimpanan data dan pengujian kinerja.
6. Parameter uji yang digunakan yaitu *Runtime, Throughput, CPU Usage, Memory Usage dan Disk I/O*.
7. Data uji yang digunakan untuk melakukan pengujian kinerja menggunakan data *Payload*.

1.6 Sistematika pembahasan

Pada penyusunan dan pembahasan penulisan sistematika laporan penelitian dapat dijabarkan secara garis besar sebagai berikut :

BAB I PENDAHULUAN

Pada bab I pendahuluan akan dijelaskan mengenai latar belakang penulis melakukan penetian, rumusan masalah hasil ekstraksi dari latar belakang, tujuan penelitian, manfaat penelitan yang dapat diambil, serta batasan penelitian supaya peneliti fokus pada hal yang ingin dibahas, dan sistematika penulisan.

BAB II LANDASAN KEPUSTAKAAN

Pada bab II Landasan Kepustakaan berisi tentang teori dan referensi yang berkaitan dengan topik penelitian berdasarkan penelitian sebelumnya ataupun penelitian penunjang lainnya yang bersifat ilmiah sebagai dasar acuan.

BAB III METEDOLOGI PENELITIAN

Pada bab III Metedologi membahas langkah kerja yang akan dilakukan dalam penelitian analisis kinerja. Tahapan berikut meliputi studi litelatur, lingkungan penelitian dan persiapan scenario, pengujian, pembahasan, dan hipotetis dibuat atas dasar penelitian yang dilakukan.

BAB IV PERANCANGAN LINGKUNGAN UJI

Pada bab IV Perancangan akan menjelaskan *environment* penelitian dan pembuatan scenario untuk tahap selanjutnya yaitu pengujian.

BAB V PENGUJIAN

Pada bab V pembahasan akan dilakukan pengujian dari setiap scenario yang telah dibuat pada bab sebelumnya yaitu bab perancangan.

BAB VI PEMBAHASAN DAN HASIL

Pada bab VI Penutup terdapat kesimpulan yang diambil berdasarkan tahapan yang sudah dilakukan mulai dari persiapan scenario, pengujian, hingga pembahasan yang mengacu pada rumusan masalah penelitian.

BAB VII PENUTUP

Bab ini berisi kesimpulan yang telah diperoleh dari penelitian yang telah dilakukan untuk menjawab rumusan masalah serta memuat saran untuk penelitian pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

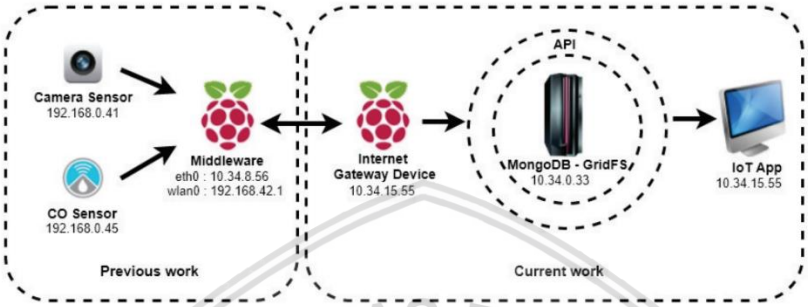
Pada bab ini membahas tentang landasan kepustakaan yang berisi penjelasan dan pembahasan tentang kajian pustaka dan dasar teori. Kajian pustaka membahas tentang penelitian sebelumnya yang berkaitan dengan penelitian ini, dan dasar teori serta pemahanan yang diperlukan untuk mencapai tujuan penelitian ini.

2.1 Kajian Pustaka

Kajian pustaka akan membahas tentang penelitian-penelitian yang terkait dengan penelitian yang akan dilakukan. Kajian pustaka berfungsi untuk mengetahui perbedaan penelitian saat ini dengan yang sudah dilakukan sebelumnya, serta untuk membatu sebagai kepustakaan dan refrensi dalam penelitian ini. Berikut penjelasan beberapa penelitian terkait pada Tabel 2.1.

Nomor	Nama Judul, Penulis dan Tahun	Hasil Penelitian
1	Performance Comparison Between Five NoSQL Databases. EnqingTang, Yushun Fan(2016).	1. Menggunakan 5 buah virtual machine untuk pengujian kinerja. 2. Dataset yang digunakan menggunakan data ASCII dari YCSB. 3. Parameter yang digunakan untuk menguji yaitu Througput dan *Runtime*. 4. Banyak data yang digunakan sebanyak 10.000 dan 100.000 data
2	Review of NoSQL database and performance testing on Hbase. Naheman W, Wei J(2013).	1. Menggunakan 3 buah server berisi Hadoop cluster sebagai distributed file system. 2. menggunakan delapan machines yang dibuat pada Hbase Cluster. 3. menguji column family, sorting, random read and write, dan query test
3	Based IoT Data Storage *Framework* For Heterogeneous Sensor Data. Pramukantoro Eko, Dkk(2017).	1. Membangun mongoDB sebagai IoT data storage. 2. Menguji kinerja MongoDB

Pada penelitian ketiga (Pramukantoro, Dkk,. 2017) telah dikembangkan sebuah IoT datastorage *Framework* untuk menyimpan data yang beragam dari node sensor. *Framework* tersebut menggunakan mongoDB dan GridFS sebagai media storagenya. Protocol pengiriman data menggunakan MQTT. Untuk topologi sistem dapat dilihat pada Gambar 2.1.



Gambar 2.1 Topologi Sistem *(Pramukantoro, E., dkk., 2017)*

Untuk pengujian yang dilakukan pada penelitian ini adalah pengujian skalabilitas API webservice, *respon time* pada IoT Application dan pengujian fungsional pada sistem yang dibuat.

Pada penelitian kali ini, hbase akan menggunakan *Framework* IoT datastorage MongoDB sebagai media penyimpanannya. *Framework* yang digunakan berdasarkan penelitian yang sudah dilakukan (Pramukantoro, Dkk., 2017). Kemudian hbase dan mongodb akan dibandingkan kinerja data storagenya dengan lima parameter yaitu *Runtime*, *throughput*, *CPU Usage*, *Memory Usage*, serta *Disk I/O* dan lima variasi data yaitu 10.000, 30.000, 50.000, 70.000 dan 100.000 untuk pengujian *text*. Sedangkan pengujian gambar memakai lima variasi data yaitu 1000, 3000, 5000, 7000, 10.000.

2.2 Dasar Teori

Berdasarkan beberapa informasi dan kajian pustaka, maka dalam penulisan ini terdapat beberapa teor, antara lain :

2.2.1 Internet Of *Things*

Internet of *Things* (IoT) merupakan konsep yang memiliki tujuan untuk menghubungkan benda fisik dan virtual dengan sebah konektivitas internet yang tersambil sehingga dapat mencakup pada setiap aspek kehidupan manusia.

IoT bisa disebut sebagai sebuah infrastruktur jaringan global. Infrastruktur terdiri dari jaringan yang telah ada dan internet berikut pengembangan jaringannya. IoT adalah ketika internet dan jaringan memperluas lingkupnya hingga ke tempat-tempat seperti jaringan energy, fasilitas kesehata, dan transportasi (Cisco, 2017).

Menurut analisa McKinsey Global Institute, Internet of *Things* adalah sebuah teknologi yang memungkinkan kita untuk menghubungkan mesin. peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator untuk memperoleh data dan mengelola kinerjanya sendiri, sehingga memungkinkan mesin untuk berkolaborasi dan bahkan bertindak berdasarkan informasi baru yang diperoleh secara independen. Cara kerja dari Internet of *Things* cukup mudah. Setiap benda harus memiliki sebuah IP Address. IP Address adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, IP address dalam benda-benda tersebut akan dikoneksikan ke jaringan internet.

2.2.2 Internet Of *Things* Middleware

Untuk mempermudah *Things* dalam berkomunikasi dengan internet, IoT membutuhkan sebuah middleware yakni software atau hardware yang menyediakan *interface* bagi *Things* untuk mengirimkan atau mendapatkan data. Dalam IoT, penggunaan middleware sangat beragam misalkan, middleware untuk menghubungkan sensor dengan *Cloud*, middleware untuk smart-device terhubung dengan smart-phone ataupun untuk komunikasi machine-to-machine (M2M). Pada umumnya middleware memiliki *interface* bagi sensor untuk mengirimkan data dan juga *interface* bagi aplikasi untuk membaca data tersebut. Hal yang membedakan antara middleware satu dengan middleware lainnya yakni bagaimana ia menerapkan *interface* tersebut, melakukan manajemen data, serta menjaga keamanan dan privasi data tersebut. Dengan kondisi IoT saat ini, middleware yang dikembangkan harus dapat beradaptasi dan menyelesaikan tantangan yang ada. Beberapa penelitian telah dilakukan untuk merumuskan tantangan tersebut dan bagaimana menyelesaikannya. Penelitian ini meliputi survei dan analisis tentang middleware yang sudah saat ini, pola-pola perancangan serta konsep arsitektur middleware (Ngu & Gutierrez, 2016).

2.2.3 Database

Database adalah kumpulan data. Data merupakan fakta yang diketahui, dapat direkam dan memiliki makna tersirat. Sebuah *Database* biasanya merepresentasikan beberapa objek asli dari dunia nyata dan digunakan untuk tujuan tertentu oleh satu atau lebih kelompok pengguna. Untuk menerapkan dan memelihara suatu *database* secara terkomputerisasi, pengguna dapat menggunakan suatu *Database Management System* (*DBMS*). *DBMS* adalah perangkat lunak yang berfungsi untuk membuat dan memanajemen suatu *database*.

Sebagai contoh, nama, nomor telepon, dan alamat dari orang yang kita kenal. Data-data tersebut biasanya disimpan dalam sebuah buku alamat berindeks atau mungkin disimpan pada *hard drive*, menggunakan komputer pribadi dan perangkat lunak seperti *Microsoft Access* atau *Excel*. Koleksi data dengan makna implisit ini bisa

disebut dengan *database*, sedangkan perangkat lunak seperti *Microsoft Access* dan *Excel* ini bisa disebut sebagai *DBMS* (Elmasri dan Navathe, 2011).

2.2.4 Not Only SQL

Database Not Only SQL (NoSQL) merupakan model baru *database* non-relasional, yang pada perkembangannya menarik perhatian masyarakat penelitian dan perusahaan dengan cepat. *NoSQL Database* dapat menangani semua tuntutan penyimpanan dan pengaksesan data pada suatu sistem perusahaan / aplikasi web yang berbeda-beda dengan cepat dan efisien. *NoSQL Database* memiliki fitur tambahan yang membedakannya dengan *Relational Database Management Systems (RDBMS)*, seperti tidak membutuhkan skema *database* kaku untuk didefinisikan dan memiliki skalabilitas horizontal yang mudah.

NoSQL Database dirancang untuk menyediakan skalibilitas *shared-nothing* horizontal yang ditunjukan dengan distribusi data melalui *server* yang berbeda. Dengan demikian, *NoSQL Database* dikembangkan untuk dapat melakukan skalabilitas (penambahan beban) dengan mudah dan dapat mendistribusikan data dengan efisien. Ketika jumlah *server* meningkat, sistem ini justru mampu melayani permintaan dengan lebih efisien. Semua permintaan dieksekusi secara paralel, sehingga menghasilkan *Throughput* yang lebih tinggi dan waktu eksekusi *query* yang lebih rendah (Abramova, Bernardino dan Furtado, 2014).

Ada 4 tipe *NoSQL Database* berdasarkan model penyimpanannya, yaitu:

1. *Key-Value based*

   Pada tipe ini, data disimpan dalam bentu kunci-isi berpasangan, maksudnya pada tipe ini terdapat dua bagian bagian yaitu sebuah string yang mempresentasikan sebuah kunci (*key*) dan data aktual yang merupakan nilai (*value*) sehinga akan membentuk suatu pasangan kunci-isi (*key-value*). Contoh *database* yang menggunakan konsep *key-value* based yaitu *DynamoDB* dan *Aerospike*.

2. *Column-oriented based*

   Pada tipe ini, data disimpan dalam kolom-kolom, walaupun menggunakan konsep kolom, tetapi tipe ini tidak seperti konsep relasional kolom *database* pada umumnya. Data tidak disimpan dalam tabel tetapi disimpan dalam suatu arsitektur distribusi yang sangat besar. Pada tipe ini, setiap *key* dihubungkan dengan satu atau lebih atribut (kolom). Contoh *database* yang menggunakan konsep *Column-oriented* adalah *HBase* dan *Cassandra*.

3. Document-store based

   Pada tipe ini, setiap satu objek data disimpan dalam dalam satu bentuk dokumen. Dokumen pada *document-store database* sama dengan *record* pada *relational database*. Dokumen yang disimpan dapat berupa format *PDF*, *XML*, *JSON* dan lain-lain. Dokumen sendiri bisa terdiri dari *key-value, unique key* digunakan

untuk mempresentasikan dokumen. *Key* bisa berupa sebuah *string* sederhana atau sebuah *string* yang mengarahkan ke sebuah *URI* atau *path*. Contoh *database* yang menggunakan konsep *document-store* adalah *CouchDB* dan *MongoDB*.

4. *Graph based*

   Pada tipe ini, penyimpanan data dilakukan dalam bentuk graph. Sebuah graph terdiri dari *node* dan *edge*. *Node* menggambarkan sebuah objek dan *edge* menggambarkan sebuah relasi antar objek. Setiap *node* memiliki *direct pointer*, titik tersebut merupakan *node* yang berdekatan. *Database Graph* memiliki fitur *schema less* dan penyimpanan efisien pada data yang semi terstruktur. Contoh *database* yang menggunakan konsep *graph* adalah *Neo4j*.

5. *Object oriented based*

   Pada tipe ini, data disimpan dan direpresentasikan sebagai sebuah objek. Tipe ini menggabungkan antara konsep *Object Oriented Programming (OOP)* dengan dasar *database*. Objek data disimpan dengan fitur yang sama dengan *OOP* seperti *encapsulation, polymorphism* and *inheritance*. Kelas dapat disamakan dengan tabel. Setiap objek mempunyai sebuah *identifier* untuk membedakan objek satu dengan objek yang lainnya. Contoh *database* yang menggunakan konsep ini adalah *db4o* (Nayak, Poriya dan Poojary, 2013).

2.2.5 *Cloud Computing*

Cloud Computing. *Cloud Computing* adalah sebuah model komputasi yang terkonfigurasi, dimana sumber daya seperti processor/*Computing* power, storage, network, dan sofware menjadi abstrak (virtual) serta diberikan sebagai layanan di jaringan/internet menggunakan pola akses remote (Purbo, 2011). NIST (National Institute of Standarts and Technology) sebagai badan nasional standar dan teknologi Amerika Serikat memberikan definisi *Cloud Computing* yaitu suatu model untuk memberikan kenyamanan, on-demand akses jaringan untuk memanfaatkan bersama suatu sumber daya komputasi yang terkonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat secara cepat diberikan dan dirilis dengan upaya manajemen yang minimal atau interaksi penyedia layanan. *Cloud Computing* menyediakan 3 model layanan yaitu (Mell & Grance, 2011):

a*) Software as a Service (SaaS)*

b) Platform as a Service (PaaS)

c) Infrastructure as a Service (IaaS) (Purbo, 2011)

Sedangkan model penyebaran *Cloud Computing* menurut NIST (National Institute of Standarts and Technology) terdiri dari empat model, yaitu (Mell & Grance, 2011):

a*) Private Cloud*

b) Community Cloud

c) Public Cloud

d) Hybrid Cloud

2.2.6 MongoDB Dan GridFS

MongoDB adalah sebuah database yang bersifat Open Source yang memiliki high performance. MongoDB merupakan sebuah database dengan konsep manajemen database berorietasi dokumen yang dibuat menggunakan bahasa pemrograman C++. Database Berorientasi Dokumen adalah sebuah program komputer yang dirancang untuk menyimpan, mengambil dan mengelola data yang berorientasi dokumen. database berorientasi Dokumen adalah salah satu dari kategori database yang di kenal dengan istilah populer NoSQL. NoSQL singkatan dari Not Only SQL, artinya sebuah sistem basis data yang tidak harus menggunakan perintah SQL (*Structure Query Language*) untuk melakukan proses manipulasi data. MongoDB merupakan basis data yang tidak relasional, hal ini membuat MongoDB sangat cepat saat melakukan proses manipulasi data dari pada sistem basis data relasional (RDBMS), selain itu MongoDB berbasis dokumen sehingga tidak memiliki struktur yang teratur seperti tabel. Kelebihan MongoDB dibandingkan database yang lain adalah dapat melakukan searching lebih cepat, tidak perlu membuat struktur tabel karena MongoDB otomatis membuatkannya, jadi hanya perlu melakukan *insert* saja, mempercepat proses CRUD (create, read, update, delete), digunakan oleh banyak website-website besar (Chodorow & Dirolf, 2010).

Menurut Seguin (2012,p4), MongoDB adalah data storage dimana tiap table tidak memiliki relasi dengan tabel lainnya. MongoDB berisi *collection*. Setiap *collection* terdiri dari documents. Setiap *documents* terdiri dari fields. Sebuah *collections* dapat di *indexes*, yang meningkatkan kinerja pengurutan data. Ada 6 konsep yang harus di mengerti adalah :

1. MongoDB memiliki konsep yang sama dengan data storage lainnya seperti MySQL atau Oracle. MongoDB dapat tidak memiliki data storage atau lebih dari satu data storage, masing-masingnya bertindak sebagai *"high level containers".*

2. Sebuah data storage dapat tidak memiliki collection atau lebih dari satu collection. Sebuah collection memiliki banyak kesamaan dengan tabel tradisional pada data storage seperti MySQL. Sebuah collection dan tabel tradisional dalam hal ini dapat di anggap sama.

3. Sebuah data storage dapat tidak memiliki documents atau lebih dari satu documents. Sebuah documents dapat dianggap sama dengan sebuah row pada tabel tradisional.

4. Sebuah documents terdiri dari satu atau lebih fields atau columns.

6. Cursors pada MongoDB di gunakan untuk meminta atau memanggil data.

GridFS merupakan spesifikasi MongoDB untuk menyimpan dan mengambil file besar, yang merupakan jenis sistem file untuk menyimpan file namun data yang disimpan dalam koleksi MongoDB.

2.2.7 HBase

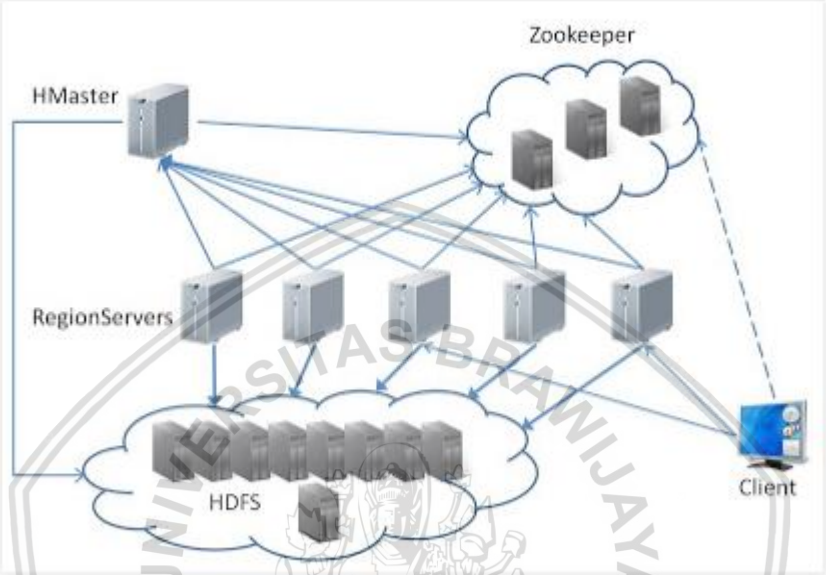
Menurut Naheman W, dan Wei J(2013) HBase adalah sejenis database NoSQL yang mempunyai karakteristik column-oriented, distributed, *high reliability, dan horizontal scalable.* Tekniknya dikembangkan dan diturunkan oleh Apache Software Foundation dengan Bahasa pemrograman Java. HBase mampu membuktikan ketersediaan tinggi, dapat di perluas skalanya, konsiten, solusi rendah biaya untuk penyimpanan yang besar.

Hbase adalah program yang berjalan diatas Hadoop Distributed File System(HDFS) yang mampu memproses data dalam skala besar secara interaktif. HBase merupakan implementasi dari konsep Google Bigtable.

Menurut WIjaya, W. M pada buku "Teknologi Bigdata" HBase memiliki beberapa arsitektur sesuai dengan gambar 2.4 yaitu:

1. HMaster adalah server pada Hbase yang berfungsi untuk men-start HBase, mendistribusikan Region ke RegionServer yang terdaftar, mendeteksi dan memulihkan RegionServer yang bermasalah. HMaster tidak menangani data yang disimpan pada HBase.
2. RegionServer adalah server yang bertugas menyimpan dan mengelola Region-region yang terimanya dari HMaster, menangani permintaan client, dan mempartisi Region yang sudah melewati ukuran maksimal, kemudian melaporkan Region yang telah dipartisi kepada HMaster.
3. ZooKeeper bertugas mengelola informasi pokok tentang kondisi HBase itu sendiri. Selain itu bisa mengetahui kondisi terkini dari pada RegionServer, yang kemudian memberikan informasi kepada HMaster. Berdasarkan informasi dari ZooKeeper ini, HMaster mengatur pembagian Region ke RegionServer dan memulihkan RegionServer yang bermasalah. ZooKeeper juga menyimpan informasi lokasi RootKatalog dan alamat HMaster. Koneksi pertama kali client yang ingin mengakses HBase dimulai dengan koneksi melalui ZooKeeper.
4. HDFS (Hadoop Distributed File System) berfungsi sebagai media penyimpanan data bagi HBase. Semua data yang diloading ke HBase dan data log HBase disimpan dalam HDFS.

5. Client berfungsi untuk mencari keperluan yang sesuai dengan lingkungan pada RegionServer service. Client berkomunikasi dengan HMaster melalui ZooKeeper untuk kemudian dapat berkomunikasi dengan HMaster.



Gambar 2.3 Arsitektur HBase *(ApacheHbase)*

2.2.8 *Web Service*

Dalam Microsoft (2000) dinyatakan bahwa *Web Service* merupakan tahapan ketiga dari tahapan evolusi ASP (*Application Service Provider*) dimana pada tahapan pertama ditekankan pada penyediaan aplikasi desktop sedangkan pada tahapan kedua ditekankan pada penyediaan aplikasi berbasis client-server. Pada tahapan ketiga ini, komponen-komponen atau building blocks *software* disediakan sebagai service dan disebarkan lewat jaringan internet untuk diintegrasikan dengan aplikasi-aplikasi lain. Menurut Kreger (2001) *Web Service* diartikan sebagai sebuah antar muka (*interface*) yang menggambarkan sekumpulan operasi-operasi yang dapat diakses melalui jaringan, misalnya internet, dalam bentuk pesan XML. Sedangkan menurut Manes (2001), *Web Service* diartikan sebagai sepotong atau sebagian informasi atau proses yang dapat diakses oleh siapa saja, kapan saja dengan menggunakan piranti apa saja, tidak terikat dengan sistem operasi atau bahasa pemrograman yang digunakan.

Web Service dapat dibangun dengan menggunakan bahasa pemrograman apa saja dan juga dapat diimplementasikan pada platform manapun. Kita dapat

membangun *Web Service* pada Windows 2000 dan menjalankannya melalui Windows, Linux, Unix, Mac, PalmOS dan WinCE (Hamids, 2000). Hal ini dimungkinkan karena *Web Service* berkomunikasi menggunakan sebuah standar format data yang universal yaitu XML dan menggunakan protokol SOAP. Karena *Web Service* menggunakan format data XML, maka *Web Service* juga mewariskan sifat multi-tier dari XML sehingga memungkinkan terjadinya integrasi antar *Web Service* atau aplikasi (Microsoft, 2001).

Menurut Meiyanto (2001) pada sistem multi-tier, aplikasi maupun dokumen XML dapat dilewatkan ke pihak lain dan diolah oleh pihak tersebut. Dalam sistem ini dimungkinkan suatu aplikasi dapat mengambil data dari satu sumber tanpa harus tahu bahwa sebenarnya data tersebut dihasilkan melalui proses pengolahan oleh sistem lain sehingga dapat terjadi integrasi data maupun aplikasi yang sering disebut dengan A2A *(application to application*).

Menurut Kreger (2001) dikatakan bahwa model dari sebuah *Web Service* didasarkan pada interaksi antara 3 komponen yang berperan dalam *Web Service*, yaitu: service provider, service registry dan service requestor/consumer. Interaksi yang terjadi antara ketiga komponen tersebut juga melibatkan operasi publish, find dan bind. Service provider menyediakan service yang dapat diakses melalui jaringan komputer, misalnya internet. Kemudian, service provider mendeskripsikan service yang dibangun dan mem-publish-kan service description tersebut ke service registry atau secara langsung ke service consumer. Service requestor/consumer menggunakan operasi find untuk mendapatkan service description secara local maupun melalui service registry. Service description yang diperoleh itu kemudian digunakan untuk men-bind service provider dan berinteraksi dengan implementasi *Web Service* yang akan digunakan tersebut

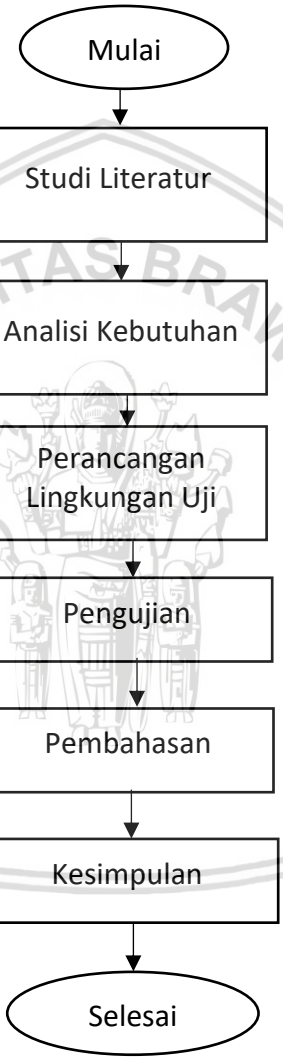
2.2.9 Apache JMter

Apache JMeter adalah sebuah perangkat lunak *open-source* yang dikembangkan oleh *Apache Software Foundation*. *JMeter* merupakan aplikasi berbasis *Java* yang dirancang untuk melakukan tes fungsional dan mengukur kinerja suatu sistem.

Pada awalnya, aplikasi ini dirancang untuk pengujian Aplikasi Web. Namun saat ini sudah diperluas untuk menguji fungsi lainnya. Selain itu, juga terdapat banyak pengembang dari pihak ketiga yang ikut berkontribusi membuat *plugin-plugin* tambahan yang dapat digunakan untuk menguji kinerja suatu sistem/aplikasi menggunakan *JMeter* ini (*JMeter*, 2017).

BAB 3 METODOLOGI

Dalam bab ini akan dijelaskan tahapan dan metode yang digunakan untuk mengerjakan penelitian perbandingan kinerja database NoSQL HBase dan MongoDB pada IoT data Storege. Berikut adalah tampilan diagram alur metedologi penelitian yang ditunjukan pada gambar berikut :



Gambar 3.1 Diagram Blok Medelogi Penelitian

3.1 Studi Literatur

Pada Penelitian ini dibutuhkan studi literature yang bertujuan untuk mencari dasar-dasar teori dan kajian pustaka yang digunakan untuk menunjang penulisan skripsi yang dilakukan yaitu Perbandingan Kinerja NoSQL Database HBase dan MongoDB pada IoT data Storage. Teori-teori pedukung tersebut didapat dari jurnal, e-book, artikel, website dan dokumentasi project pada penelitan sebelumnya. Landasan kepustakaan yang digunakan untuk mendukung penulisan penelitian ini sebagai berikut :

1. *Database*
2. *Not Only SQL(NoSQL)*
3. *Cloud Computing*
4. *MongoDB & GridFS*
5. *Apache Hbase*
6. *Web Service*
7. *Apache J-Meter*

3.2 Rekayasa Kebutuhan

3.2.1 Rekayasa Kebutuhan Perangkat Keras

Bagian ini menjelaskan berbagai macam rekayasa perangkat keras yang digunakan pada penelitian. Kebutuhan perangkat keras pada penelitian ini adalah sebagai berikut :

Kebutuhan perangkat keras adalah sebagai berikut :

a. Laptop
   - Operating System : Windows 10 Home (64-bit)
   - Prosesor : Inte(R) Core(TM) i3-5005U
   - RAM : 4 GB
   - Hardisk : 500 GB

b. Raspberry Pi digunakan sebagai perangkat untuk menjalankan middleware.

c. Node MCU Perangkat ini digunakan sebagai mikrokontroler bagi sensor karbon monoksida yang dibuat. Mikrokontroler ini nantinya dirangkai dengan modul DHT11 dan diprogram sehingga dapat mengirimkan data kelembapan sensor.

d. Sensor kamera adalah modul sensor untuk mengambil gambar

3.2.2 Rekayasa Kebutuhan Perangkat Lunak

Bagian ini menjelaskan berbagai macam perangkat lunak yang digunakan pada penelitian. Kebutuhan perangkat lunak pada penelitian ini adalah sebagai berikut :

Kebutuhan perangkat lunak yang digunakan adalah sebagai berikut :

a. Virtual Private Server(VPS)

   Adapun spesifikasi dari VM node database server, yaitu :

   - Operating System : Ubuntu Server 16.04
   - Prosesor : Genui Intel Westmare e56xx
   - RAM : 4 GB
   - Harddisk : 40 GB

b. MobaXterm adalah software remote konsol berfungsi untuk me-remote koneksi komputer melalui port SSH

c. Apache Hadoop adalah software untuk menyimpan data dalam hdfs

d. Apache HBase adalah software yang berjalan diatas Hadoop

e. MongoDB adalah software yang digunakan untuk menyimpan data sensor

f. RoboMongo adalah software yang digunakan untuk mengatur, management dan mengembangkan database pada mongodb

Apache Jmeter adalah software open source berbasis java yang digunakan untuk melakukan performance test os metrics pada database server.

3.3 Perancangan Lingkungan Uji

Pembangunan lingkungan uji merupakan tahap yang digunakan penulis untuk menyiapkan berbagai kebutuhan yang diperlukan untuk melakukan penelitian. Kebutuhan tersebut dibagi menjadi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Kebutuhan perangkat keras berisikan daftar perangkat yang akan digunakan dalam penelitian. Kebutuhan perangkat lunak berisikan daftar program yang digunakan dalam penelitian. Sebelum melakukan lingkungan uji, diperlukan instalasi Hbase pada server yang digunakan. Setelah instalasi selesai, maka dilanjutkan tahap konfigurasi Hbase. Untuk proses konfigurasi pada Hbase dapat dilihat pada lampiran. Proses konfigurasi bertujuan agar Hbase yang akan digunakan dapat berjalan dengan baik.

3.4 Pengujian

Pengujian ni dilakukan untuk mengetahui apakah Hbase yang diusulkan dapat bekerja sesuai fungsinya yaitu menyimpan data yang dirancang serta didapatkan data

berupa nilai yang nantinya digunakan untuk mengukur kinerja dari sisi database yang digunakan. Untuk menguji kinerja parameter yang digunakan adalah *Runtime*, *Throughput*, *CPU Usage*, *Memory Usage* dan *Disk I/O*. yakni pengujian *insert* data *Payload* dengan Hbase dan MongoDB.

3.4.1 Skenario 1: Pengujian *insert Payload* data dengan data storage Hbase

Pengujian skenario 1 dilakukan dengan melakukan *insert* data *text* dan data file dengan data storage Hbase. Proses *insert* data *Payload* menggunakan python dengan variasi jumlah baris data *text* sebanyak 10.000, 30.000, 50.000, 70.000, dan 100.000 baris data serta variasi jumlah baris data file sebanyak 1.000, 3.000, 5.000 dan 10.000 baris data. Untuk data *text* yang digunakan mengacu pada data yang dihasilkan oleh sensor DHT11.

3.4.2 Skenario 2: Pengujian *insert Payload* data dengan MongoDB

Pengujian skenario 2 dilakukan dengan melakukan *insert* data *text* dan data file dengan data storage MongoDB. Proses *insert* data *Payload* menggunakan python dengan variasi jumlah baris data *text* sebanyak 10.000, 30.000, 50.000, 70.000, dan 100.000 baris data serta variasi jumlah baris data file sebanyak 1.000, 3.000, 5.000 dan 10.000 baris data. Untuk data *text* yang digunakan mengacu pada data yang dihasilkan oleh sensor DHT11.

3.5 Pembahasan

Setelah melakukan proses pengujian penyimpanan dan pengujian kinerja database, maka akan dilakukan pembahasan pada hasil pengujian yang didapat. Pembahasan dilakukan untuk mengetahui fungsi Hbase dapat berjalan sesuai serta mengetahui kinerja dari database Hbase dan MongoDB sebagai *backend* IoT *data storage* dengan melihat dan membandingkan *Runtime*, *Throughput*, *CPU Usage*, *Memory Usage* dan *Disk I/O*. Untuk mengetahui perbandingan tersebut digunakan dengan diagram whisker yang menampilkan hasil perhitungan lima kali pengujian.

3.6 Kesimpulan

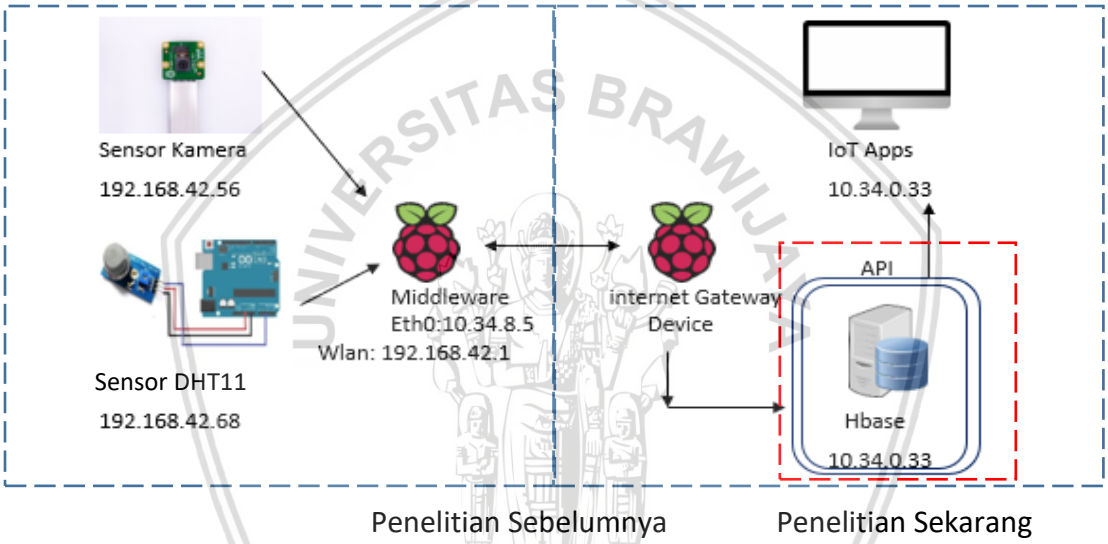
Kesimpulan dan Saran merupakan tahapan penutup dari proses penelitian setelah melalui setiap tahapan-tahapan dalam penelitian. Kesimpulan dibuat untuk memberikan jawaban terhadap rumusan masalah. Saran diberikan agar dapat dijadikan rujukan untuk memperbaiki atau mengembangkan penelitian ini maupun sejenis.

BAB 4 PERANCANGAN LINGKUNGAN UJI

Perancangan dibutuhkan dalam pengembangan system agar dapat berjalan dengan sistematis karena menyiapkan berbagai kebutuhan yang diperlukan untuk penelitian. Penelitian ini dilakukan di Lab Jaringan Komputer, Fakultas Ilmu Komputer.

4.1 Topologi Lingkungan Sistem

Topologi jaringan antar komponen sistem dibuat agar sistem dapat berjalan sesuai dengan harapan. Topologi jaringan ini menggambarkan bagaimana proses komunikasi dimulai dari sensor sampai data masuk ke dalam database Hbase. Topologi dapat dilihat pada gambar 4.1



Gambar 4.1 Topologi Lingkungan Sistem

Data yang dihasilkan dari topologi diatas bersumber pada sensor camera dan sensor DHT11. Data yang dihasilkan oleh sensor DHT11 berupa data suhu dan kelembapan sedangkan data yang dihasilkan oleh sensor camera berupa data gambar. Kedua sensor tersebut mempublish data menuju ke middleware. Middleware tersebut menggunakan 3 port ketika dijalankan yaitu port 5683 untuk CoAP, 1883 untuk MQTT, serta 3000 untuk Websocket. Pada penelitian ini hanya menggunakan protocol MQTT. Selanjutnya IDG (Internet Gateway Device) mensubscribe topic dari middleware untuk mendapatkan data dari sensor dht11 dan sensor camera. Data yang disubscribe berdasarkan nama topic tersebut akan dikirimkan dari IGD ke API. Kemudian selanjutnya webservice akan melakukan post dan get data terhadap Hbase. Untuk proses get, data akan ditampilkan melalui IoT Apps yang telah dibuat pada penelitian sebelumnya (Pramukantoro, Eko dkk 2017). Dari topologi diatas dibagi

menjadi penelitian sebelumnya dan saat ini. Penelitian saat ini hanya berfokus pada penilaian kinerja database lingkungan menggunakan hbase.

4.2 Perancangan Kode

Untuk mendukung proses berjalannya lingkungan uji, maka perlu dibuat suatu kode program adalah kode subscriber, kode webservice, dan kode pengujian. Kode subscriber digunakan untuk mensubscribe data yang berasal dari middleware. Kode webservice digunakan untuk post data yang sebelumnya sudah di subscribe, kemudian data tersebut disimpan di database. sedangkan kode pengujian digunakan untuk menguji sistem dari sisi *backend* storage. Untuk perancangan kode pengujian bisa dilihat pada bab lampiran.

Tabel 4.1 Kode Webservice.py

Algoritme 1: Webservice.py

```
1    def postdataJSON():
2        global rowkey
3     conn.open()
4    data = request.get_json()
5        print "diterima data",data
6        if "sensor" in data:
7            table = conn.table('ujit')
8            protocol = data['protocol']
9            temperature = data['temperature']['value']
10           humidity = data['humidity']['value']
11           topic = data['topic']
12           ip = data['sensor']['ip']
13           indeks = str(data['sensor']['index'])
14           timestamp = data['timestamp']
15         data_sensor = {
16      b'cb:protocol':protocol,
17                          b'cb:temperature':temperature,
18                          b'cb:humidity':humidity,
19                          b'cb:topic':topic,
20                            b'cb:ip':ip
21                      }
22       else:
23           table = conn.table('ujig')
24           file  = data['Data']
25         fname = data['Name']
26    data_sensor = {
27                              b'cb:data':file,
28                              b'cb:fname':fname,
29                            }
30       table.put(str(rowkey),data_sensor)
31       conn.close()
32       return "Berhasil menulis data :"+repr(data_sensor)
33
34    if __name__ == '__main__':
35     app2.run(host='192.168.100.10',debug=True,port=5001)
```

Tabel 4.2 Kode Subscriber.py

Algoritme 2: Subscriber.py

```
1     import paho.mqtt.client as mqtt
2     import json
3     import httplib
4     mqttc = mqtt.Client("server", clean_session=False)
5     mqttc.connect("10.34.8.5", 1883)
6     conn = httplib.HTTPConnection("10.34.0.33:5001")
7     #SUBSCRIBER
8     def on_message(mqttc,obj,msg):
9             headers = {"Content-type": "application/json"}
10            params = msg.payload
11            conn.request("POST","/api/postdataJSON",
12 params,headers)
13            response = conn.getresponse()
14            print response.read()
15
16 mqttc.on_message = on_message
17    mqttc.subscribe("office/roomA13")
18    mqttc.subscribe("office/gambarA13")
19
20    if __name__ == '__main__':
21          mqttc.loop_forever()
```

4.3 Perancangan pengujian

Perancangan pengujian digunakan sebagai langkah untuk menguji database yang digunakan dengan sistem yang sudah dikembangkan serta menguji kinerja dari databasenya. Pada penelitian ini menggunakan dua model pengujian yaitu pengujian penyimpanan dan pengujian kinerja.

4.3.1 Pengujian penyimpanan

Pengujan penyimpanan dilakukan untuk mengetahui apakah database yang digunakan sudah memenuhi kebutuhan penyimpanan yang ditentukan. Apabila semua kebutuhan penyimpanan terpenuhi, maka dilanjutkan pengujian berikutnya. Namun jika kebutuhan penyimpanan yang belum berjalan sesuai harapan, maka dilkukan pengecekan kembali terhadap rancangan dan kode yang dibuat kemudian melakukan pengujian sekali lagi, begitu seterusnya hingga kebutuhan penyimpanan terpenuhi. Untuk skenario pengujian penyimpanan yang dilakukan dapat dilihat pada tabel 4.3

Tabel 4.3 pengujian penyimpanan

No	Deskripsi Pengujian	Skenario
UF_001	API Webservice dapat mengirim data sensor dht11 dari Internet	1.Middleware sudah berjalan 2.Pengguna menjalankan kode subscriber

	Gateway Device ke database Hbase	3.Pengguna menjalankan kode service 4.kode menampilkan pesan data sensor dht11 berhasil di POST
UF_002	API Webservice dapat mengirim data sensor kamera dari Internet Gateway Device ke database Hbase	1.Middleware sudah berjalan 2.Pengguna menjalankan kode subscriber 3.Pengguna menjalankan kode service 4.kode menampilkan pesan data sensor kamera berhasil di POST
UF_003	Database Hbase Dapat menyimpan data dari sensor dht11	1.Pengguna melihat data pada tabel penyimpanan data
UF_004	Database Hbase dapat menyimpan data dari sensor kamera	1.Pengguna melihat data pada tabel penyimanan data

4.3.2 Pengujian Kinerja

Penelitian ini menggunakan beberapa parameter pengujian untuk menguji perfroma *Hbase* dan *MongoDB*. Adapun parameter uji yang digunakan antara lain:

- *Runtime* merupakan waktu eksekusi query dari suatu *database*.
- *Throughput* merupakan jumlah operasi yang dieksekusi *database* pada waktu tertentu.
- *Memory Usage* merupakan persentase kapasitas memory (RAM) yang dipakai oleh *database* server ketika mengeksekusi suatu operasi transaksi data.
- *CPU Usage* merupakan persentase kapasitas prosesor (*CPU*) yang dipakai oleh *database* server ketika mengekseskusi suatu operasi transaksi data.
- *Disk I/O* merupakan kemampuan Disk yang dipakai oleh *database* server ketika mengeksekusi suatu operasi transaksi data.

Pengujian datastorage dilakukan untuk mengetahui kinerja database yang digunakan dalam menyimpan data *Payload* yang menyerupai data sensor dht11 dan dan sensor kamera. Untuk parameter yang digunakan dalam mengambil nilainya yaitu *Runtime*, *Throughput*, *CPU Usage*, *Memory Usage* dan *Disk I/O*. Pengujian dilakukan dengan skenario perubahan jumlah baris data dalam melakukan operasi *insert Payload* data. Proses operasi *insert* dilakukan menggunakan API Hbase dan API MongoDB dengan menggunakan bahasa pemrograman python. Selain menggunakan API database, pengujian pada penelitian ini juga melibatkan Apache Jmeter sebagai tools untuk mengukur *CPU Usage*, *Memory Usage* dan *Disk I/O* pada database server

dengan menambahkan plugins PerfMon (Servers Performance Monitoring). Plugins ini bekerja dengan cara mengumpulkan resource metrics dari server target. Server target harus menjalankan tool ServerAgent agar plugins PerfMon dapat melakukan binding terhadap server target dari PerfMon. Untuk skenario jumlah data dapat dilihat pada Tabel 4.4, tabel 4.5, tabel, Tabel 4.6 dan tabel 4.7.

Tabel 4.4 Skenario Jumlah Baris Data *text*

Nomor	**Hbase**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	10.000 data					
2	30.000 data					
3	50.000 data					
4	70.000 data					
5	100.000 data					

Dirancang table untuk scenario uji data sensor DHT11, sesuai dengan Tabel 4.4, yang berisi jumlah baris data dan 5 parameter.

Tabel 4.5 Skenario Jumlah Baris Data *text*

Nomor	**MongoDB**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	10.000 data					
2	30.000 data					
3	50.000 data					
4	70.000 data					

5	100.000 data					

Dirancang table untuk scenario uji data sensor DHT11, sesuai dengan Tabel 4.5, yang berisi jumlah baris data dan 5 parameter.

Tabel 4.6 Skenario Jumlah Baris Data File Gambar

Nomor	**Hbase**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data					
2	3.000 data					
3	5.000 data					
4	7.000 data					
5	10.000 data					

Dirancang table untuk scenario uji data sensor gambar, sesuai dengan Tabel 4.6, yang berisi jumlah baris data dan 5 parameter.

Tabel 4.7 Skenario Jumlah Baris Data File Gambar

Nomor	**MongoDB**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data					
2	3.000 data					
3	5.000 data					
4	7.000 data					
5	10.000 data					

Dirancang table untuk scenario uji data sensor gambar, sesuai dengan Tabel 4.7, yang berisi jumlah baris data dan 5 parameter.

4.4 Data Uji

Data uji digunakan untuk menguji sistem dari sisi *backend* storage. Data uji yang digunakan yaitu data *Payload* yang menyerupai data sensor dht11 dan data gambar.

Untuk *Payload* sensor dht11 tipe datanya disamakan dengan data sensor asli yaitu *String*, begitu juga sensor gambar yaitu *.Jpg.* Selain tipe datanya sama besaran data pada sensor gambar dan payload disamakan .

Pengujian dilakukan dengan memasukkan operasi *insert* data *Payload* ke dalam database dengan parameter uji antara lain *Runtime, Throughput, CPU Usage, Memory Usage*, dan *Disk I/O.* Pengujian akan dilakukan sebanyak 5 kali untuk mendapatkan hasil yang tepat. Untuk pengambilan parameter *CPU Usage*, *Memory Usage* dan *Disk I/O* digunakan tools *Apache Jmeter.*

Contoh *Payload* data uji yang digunakan bisa dilihat pada gambar 4.3 dan gambar 4.4. Besaran data gambar yang digunakan bisa dilihat pada gambar 4.5, untuk *Payload* gambar 4.6.

```
b'cb:protocol':protocol,
b'cb:temperature':temperature,
b'cb:humidity':humidity,
b'cb:topic':topic,
b'cb:ip':ip
```

Gambar 4.3 Contoh *Payload* dari Sensor DHT11

```
b'cb:data':file,
b'cb:fname':fname,
```

Gambar 4.4 Contoh *Payload* data Gambar

```
pi@raspberrypi:~/camera/foto $ ls -lh
total 236K
-rw-r--r-- 1 pi pi 235K Aug  3 15:00 image0.jpg
```

Gambar 4.5 ukuran data gambar sensor kamera

```
ubuntu@ega:~/hbase-ws$ ls
2.jpg     cassandra_insert_file.py  hbase_insert_file.py  sub.py       websocket_hbase.py
baru.jpg  cassandra_insert_text.py  hbase_insert_text.py  sub.py.save  ws.py
ubuntu@ega:~/hbase-ws$ ls -lh baru.jpg
-rw-rw-r-- 1 ubuntu ubuntu 237K Jul 17 17:23 baru.jpg
```

Gambar 4.6 ukuran data *Payload* gambar

Gambar 4.5 yaitu gambar sensor asli yang terdapat pada sensor kamera, mempunyai ukuran data sebesar 235kb, sedangkan gambar 4.6 adalah data *Payload* yang mempunyai ukuran data sebesar 237kb.

BAB 5 PENGUJIAN

Pada bab ini dijelaskan hasil dari pengujian yang telah dilakukan dalam penelitian ini. Mendeskripsikan bagaimana database Hbase bekerja dalam hal menyimpan dan menamplkan data berupa data sensor dht11 dan data file berupa gambar serta kinerjanya jika ditinjau dari parameter *Runtime*, Througput, *CPU Usage*, *Memory Usage* dan *Disk I/O*. pengujian database dibagi menjadi dua tipe yaitu pengujian penyimpanan dan pengujian kinerja. Pengujian akan dilakukan sebanyak tiga kali pada setiap tipe.

5.1 Pengujian Penyimpanan Data

Pengujian penyimpanan data berfungsi untuk menguji fungsi dari database itu sendiri bahwa bisa menerima dan menyimpan data dari sensor dht11 dan sensor kamera. Hasil pengujian ada dibawah ini sebagai berikut :

5.1.1 Hasil Pengujian Data Sensor DHT11

Pada gambar 5.1 menunjukan bahwa API webservice dapat mengirim sensor dht11 ke database Hbase.

```
ubuntu@ega:~/hbase-ws$ python ws.py
 * Running on http://192.168.100.10:5001/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 289-325-841
diterima data {u'protocol': u'mqtt', u'temperature': {u'unit': u'celcius', u'value': u'26.0'}, u'timestamp': 1532877200, u'humidi
ty': {u'unit': u'%', u'value': u'36.0'}, u'topic': u'office/roomA13', u'sensor': {u'index': 8456747, u'tipe': u'esp8266', u'modul
e': u'dht11', u'ip': u'192.168.42.51'}}
10.34.0.33 - - [29/Jul/2018 22:15:23] "POST /api/postdataJSON HTTP/1.1" 200 -
diterima data {u'protocol': u'mqtt', u'temperature': {u'unit': u'celcius', u'value': u'26.0'}, u'timestamp': 1532877380, u'humidi
ty': {u'unit': u'%', u'value': u'36.0'}, u'topic': u'office/roomA13', u'sensor': {u'index': 8456747, u'tipe': u'esp8266', u'modul
e': u'dht11', u'ip': u'192.168.42.51'}}
10.34.0.33 - - [29/Jul/2018 22:16:20] "POST /api/postdataJSON HTTP/1.1" 200 -
diterima data {u'protocol': u'mqtt', u'temperature': {u'unit': u'celcius', u'value': u'26.0'}, u'timestamp': 1532877561, u'humidi
ty': {u'unit': u'%', u'value': u'35.0'}, u'topic': u'office/roomA13', u'sensor': {u'index': 8456747, u'tipe': u'esp8266', u'modul
e': u'dht11', u'ip': u'192.168.42.51'}}
10.34.0.33 - - [29/Jul/2018 22:19:21] "POST /api/postdataJSON HTTP/1.1" 200 -
diterima data {u'protocol': u'mqtt', u'temperature': {u'unit': u'celcius', u'value': u'26.0'}, u'timestamp': 1532877740, u'humidi
ty': {u'unit': u'%', u'value': u'34.0'}, u'topic': u'office/roomA13', u'sensor': {u'index': 8456747, u'tipe': u'esp8266', u'modul
e': u'dht11', u'ip': u'192.168.42.51'}}
```

Gambar 5.1 Data Dht11 berhasil diterima oleh Hbase

5.1.2 Hasil Pengujian Data Sensor Kamera

Pada gambar 5.2 menunjukan bahwa webservice dapat mengirim sensor gambar ke database Hbase berhasil diterima.

```
10.34.0.33 - - [19/Jul/2018 21:41:43] "POST /api/postdataJSON HTTP/1.1" 200 -
diterima data {u'Data': u'S\'\\xff\\xd8\\xff\\xe1c\\xfaExif\\x00\\x00MM\\x00*\\x00\\x00\\x00\\x08\\x00\\n\\x01\\x00\\x00\\x04\\x0
0\\x00\\x00\\x01\\x00\\x00\\x02\\xbc\\x01\\x01\\x00\\x04\\x00\\x00\\x00\\x01\\x00\\x02X\\x01\\x0f\\x00\\x02\\x00\\x00\\x00\\
x0c\\x00\\x00\\x00\\x86\\x01\\x10\\x00\\x02\\x00\\x00\\x00\\n\\x00\\x00\\x00\\x92\\x01\\x1a\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x
00\\x00\\x9c\\x01\\x1b\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\xa4\\x01(\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x02\\x00\\x00\
\x012\\x00\\x02\\x00\\x00\\x00\\x14\\x00\\x00\\x00\\xac\\x02\\x13\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x01\\x00\\x00\\x87i\\x00\\x
04\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\xc0\\x00\\x00\\x03|RaspberryPi\\x00RP_ov5647\\x00\\x00\\x00\\x00H\\x00\\x00\\x00\\x01\\x00
\\x00\\x00H\\x00\\x00\\x00\\x012018:05:15 09:03:03\\x00\\x00\\x17\\x82\\x9a\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x00\\x01\\xda\\x8
2\\x9d\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x00\\x01\\xe2\\x88"\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x03\\x00\\x00\\x88\\\'\\x00\\x
03\\x00\\x00\\x00\\x01\\x00d\\x00\\x00\\x90\\x00\\x00\\x07\\x00\\x00\\x00\\x040220\\x90\\x03\\x00\\x02\\x00\\x00\\x00\\x14\\x00\\
x00\\x01\\xea\\x90\\x04\\x00\\x02\\x00\\x00\\x00\\x14\\x00\\x00\\x01\\xfe\\x91\\x01\\x00\\x07\\x00\\x00\\x00\\x04\\x01\\x02\\x03\
\x00\\x92\\x01\\x00\\n\\x00\\x00\\x00\\x01\\x00\\x00\\x02\\x12\\x92\\x02\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x00\\x02\\x1a\\x92\\
x03\\x00\\n\\x00\\x00\\x00\\x01\\x00\\x00\\x02*\\x92\\x05\\x00\\x05\\x00\\x00\\x00\\x01\\x00\\x00\\x02*\\x92\\x07\\x00\\x03\\x00\
\x00\\x00\\x01\\x00\\x02\\x00\\x00\\x92\\t\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\x00\\x92\\n\\x00\\x05\\x00\\x00\\x00\\x0
1\\x00\\x00\\x022\\x92|\\x00\\x07\\x00\\x00\\x010\\x00\\x00\\x02:\\xa0\\x00\\x00\\x07\\x00\\x00\\x00\\x040100\\xa0\\x01\\x00\\x03
\\x00\\x00\\x00\\x01\\x00\\x01\\x00\\x00\\xa0\\x02\\x00\\x04\\x00\\x00\\x00\\x01\\x00\\x00\\x02\\xbc\\xa0\\x03\\x00\\x04\\x00\\x0
0\\x00\\x01\\x00\\x00\\x02X\\xa0\\x05\\x00\\x04\\x00\\x00\\x00\\x01\\x00\\x00\\x03j\\xa4\\x02\\x00\\x03\\x00\\x00\\x00\\x01\\x00\
\x00\\x00\\x00\\xa4\\x03\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00R\\xd8\\x00\\x0fB@\\x00\\
x00q8\\x00\\x00\\'\\x102018:05:15 09:03:03\\x002018:05:15 09:03:03\\x00\\x00T\\xd3\\xcf\\x00\\x0fB@\\x00\\x00w\\xf1\\x00\\x00\\
'\\x10\\x00\\x00\\x01\\x95\\x00\\x00\\x00d\\x00\\x00w\\xf1\\x00\\x00\\'\\x10\\x00\\x00\\x8c\\x88\\x00\\x00\\'\\x10ev=-1 mlux=-1
```

Gambar 5.2 Data Gambar berhasil diterima oleh Hbase

5.1.3 Hasil Pengujian Menyimpan Data Sensor DHT11

Berikut adalah gambar 5.3 menunjukan bahwa database Hbase dapat menyimpan data sensor dht11 pada data table.

```
ROW                                   COLUMN+CELL
 20136ab0-b737-4237-8a9c-ba3cdb47 column=cb:humidity, timestamp=1532007812108, value=36.0
 8998
 20136ab0-b737-4237-8a9c-ba3cdb47 column=cb:ip, timestamp=1532007812108, value=192.168.42.51
 8998
 20136ab0-b737-4237-8a9c-ba3cdb47 column=cb:protocol, timestamp=1532007812108, value=mqtt
 8998
 20136ab0-b737-4237-8a9c-ba3cdb47 column=cb:temperature, timestamp=1532007812108, value=22.0
 8998
 20136ab0-b737-4237-8a9c-ba3cdb47 column=cb:topic, timestamp=1532007812108, value=office/roomA13
 8998
 5589a278-7542-4393-beb6-30b99dc3 column=cb:humidity, timestamp=1532012671923, value=36.0
 e94e
 5589a278-7542-4393-beb6-30b99dc3 column=cb:ip, timestamp=1532012671923, value=192.168.42.51
 e94e
 5589a278-7542-4393-beb6-30b99dc3 column=cb:protocol, timestamp=1532012671923, value=mqtt
 e94e
 5589a278-7542-4393-beb6-30b99dc3 column=cb:temperature, timestamp=1532012671923, value=21.0
 e94e
 5589a278-7542-4393-beb6-30b99dc3 column=cb:topic, timestamp=1532012671923, value=office/roomA13
 e94e
 99281d81-b4e6-4845-aa13-5739a205 column=cb:humidity, timestamp=1532877740494, value=34.0
 fa6a
 99281d81-b4e6-4845-aa13-5739a205 column=cb:ip, timestamp=1532877740494, value=192.168.42.51
 fa6a
 99281d81-b4e6-4845-aa13-5739a205 column=cb:protocol, timestamp=1532877740494, value=mqtt
 fa6a
 99281d81-b4e6-4845-aa13-5739a205 column=cb:temperature, timestamp=1532877740494, value=26.0
 fa6a
 99281d81-b4e6-4845-aa13-5739a205 column=cb:topic, timestamp=1532877740494, value=office/roomA13
 fa6a
 9eca5da8-2b74-4eab-bd83-bc07dcef column=cb:humidity, timestamp=1532007271848, value=36.0
 4d48
```

Gambar 5.3 Data sensor DHT11 berhasil tersimpan ke dalam Hbase

5.1.4 Hasil Pengujian Menyimpan Data Sensor Kamera

Berikut adalah gambar 5.4 bahwa hbase berhasil menyimpan data gambar dari sensor kamera pada data table

```
hbase(main):006:0> scan 'ujig'
ROW                                   COLUMN+CELL
 5589a278-7542-4393-beb6-30b99dc3 column=cb:data, timestamp=1532012549332, value=\x5Cxff\x5Cxd8\x5Cxff\x5Cxe1c\x5CxfaExif\x5Cx0
 e94e                             0\x5Cx00MM\x5Cx00*\x5Cx00\x5Cx00\x5Cx00\x5Cx08\x5Cx00\x5Cn\x5Cx01\x5Cx00\x5Cx00\x5Cx04\x5Cx00\x
                                  5Cx00\x5Cx00\x5Cx01\x5Cx00\x5Cx00\x5Cx02\x5Cxbc\x5Cx01\x5Cx01\x5Cx00\x5Cx04\x5Cx00\x5Cx00\x5Cx0
                                  0\x5Cx01\x5Cx00\x5Cx00\x5Cx02X\x5Cx01\x5Cx0f\x5Cx00\x5Cx02\x5Cx00\x5Cx00\x5Cx00\x5Cx0c\x5Cx00\x
                                  5Cx00\x5Cx00\x5Cx86\x5Cx01\x5Cx10\x5Cx00\x5Cx02\x5Cx00\x5Cx00\x5Cx00\x5Cn\x5Cx00\x5Cx00\x5Cx00\
                                  5Cxca\x5Cxed\x5Cxd7\x5Cxc4\x5Cxb7w\x5Cxcb\x5Cxae\x5Cxb5\x5Cxf7\x5Cxd5\x5Cxe9\x5Cxb2Z$\x5Cx91\x5
                                  Cx19`\x5CxacT\x5Cx88\x5Cxb1\x5Cxfc,\x5Cx18s\x5Cxb7=?\x5Cx8bvI\x5Cx0c3\x5Cxb7\x5Cxb1\x5Cx07qZth\
                                  x5Cx98;\x5Cxe3\x5Cx8c\x5Cx9c\x5Cxf5<\x5Cxe4`c\x5Cx932\x5Cxe7\x5Cx8e\x5Cxf8\x5Cx15J_\x5Cxbd\x5Cx
                                  17\x5Cxfb\x5Cxf2\x5Cxff\x5Cx00\x5Cxe8Uo\x5Cxb0\x5Cxff\x5Cx00u\x5Cx7f\x5Cxf4\x5Cx11ZFII\x5Cxde1\
                                  x5Cx92\x5Cxb2\x5Cxd1\x5Cxf3Z\x5Cxf7z\x5Cxfb\x5Cxb2\x5Cx8b\x5Cxbd\x5Cx95\x5Cxb7\x5Cxb7V\x5Cxaf\x
                                  5Cxa9\x5Cx8c\x5Cxa5RQ\x5Cxa5\x5Cxc9Z\x5Cxbd\x5Cx198\x5Cxb7)\x5Cxd2\x5Cxaf[\x5Cx9avP\x5Cxd2^\x5C
                                  xd6u"\x5Cxf5\x5Cx93zE5\x5Cxa2MGC\x5Cxff\x5Cxd9'\x0Ap0\x0A.
 cb912459-9a7d-4690-99ba-56b4408a column=cb:fname, timestamp=1532012243274, value=images0.jpg
 8929
```

Gambar 5.4 Data sensor gambar berhasil tersimpan ke dalam Hbase

Pada pengujian penyimpanan, data dht11 dan gambar berhasil tersimpan kedalam hbase dan ditampilkan pada table hbase ‘ujig’ yang sudah didefinisikan pada awal webservice mengambil data.

5.2 Pengujian Kinerja Hbase dan MongoDB

Pengujian database berfungsi menguji kinerja database tersebut berdasarkan parameter yang telah dibuat sebelumnya, yaitu *Runtime*, Througput, *CPU Usage*, *Memory Usage*, dan *Disk I/O*. Pengujian database ini didapat dari hasil *generate* data *text* dan gambar dengan perulangan sebanyak parameter dalam program python yang disimpan dalam masing-masing database. hasil pengujian sebagai berikut :

5.2.1 Pengujian Hbase dan MongoDB *Insert* Data *Text*

Pengujian kinerja *insert* data *text* kedalam Hbase dan MongoDB dengan operasi baris data sebanyak 10.000, 30.000, 50.000, 70.000 dan 100.000 data. Dengan 5 parameter yaitu *Runtime*, *throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O*. pengujian dilakukan sebanyak masing-masing 5 kali untuk mendapatkan hasil yang berbeda kemudian dirata-ratakan. Hasil pengujian database terdapat pada table berikut :

Tabel 5.1 Pengujian Hbase file *text* pertama

No	Hbase					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	20	496	85	55	284
2	30.000 data	35	861	65	57	848
3	50.000 data	53	945	54	58	781
4	70.000 data	73	954	58	63	478
5	100.000 data	99	1.006	54	67	619

Pengujian hbase *insert* file *text* pertama pada tabel 5.1. *Runtime* dan *throughput* naik stabil sesuai dengan banyaknya jumlah data, sedangkan *CPU Usage* mengalami penurunan penggunaan, berkebalikan dengan *memory* mengalami kenaikan penggunaan, dan *Disk I/O* yang naik dan turun.

Tabel 5.2 Pengujian Hbase file *text* kedua

No	Hbase					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	12	874	55	80	41
2	30.000 data	37	800	68	84	90
3	50.000 data	52	969	54	86	95
4	70.000 data	70	1.000	54	76	103
5	100.000 data	100	1.001	59	96	1249

Pengujian hbase *insert* file *text* kedua pada tabel 5.2. *Runtime* dan *throughput* naik stabil sesuai dengan banyaknya jumlah data, sedangkan *CPU Usage* mengalami naik turun penggunaan, berkebalikan dengan memory mengalami kenaikan penggunaan, dan *Disk I/O* yang naik stabil.

Tabel 5.3 Pengujian Hbase file *text* ketiga

No	Hbase					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	11	874	41	80	70
2	30.000 data	31	959	54	85	95
3	50.000 data	52	955	54	83	92
4	70.000 data	70	997	55	91	631
5	100.000 data	100	1.004	55	93	353

Pengujian hbase *insert* file *text* ketiga pada tabel 5.3. *Runtime* dan *throughput* naik stabil sesuai dengan banyaknya jumlah data, sedangkan cpu dan *Memory Usage* yang cenderung meningkat penggunaannya, dan *Disk I/O* yang naik stabil.

Tabel 5.4 Pengujian Hbase file *text* keempat

No	Hbase					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	11	920	25	80	320
2	30.000 data	31	971	53	86	792
3	50.000 data	51	977	55	83	92
4	70.000 data	71	981	56	92	693
5	100.000 data	99	1.015	54	93	698

Pengujian hbase *insert* file *text* keempat pada tabel 5.4. *Runtime* dan *throughput* naik stabil sesuai dengan banyaknya jumlah data, sedangkan cpu dan *Memory Usage* yang cenderung meningkat penggunaannya, dan *Disk I/O* yang naik turun.

Tabel 5.5 Pengujian Hbase file *text* kelima

No	Hbase					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	11	927	35	80	24
2	30.000 data	30	992	51	86	108
3	50.000 data	48	1.038	54	85	95
4	70.000 data	70	1.003	54	93	481

5	100.000 data	101	933	56	86	2.902

Pengujian hbase *insert* file *text* lima pada tabel 5.5. *Runtime* yang naik stabil sesuai dengan banyaknya data, sedangkan *Throughput*, *CPU Usage*, *Memory Usage* dan *Disk I/O* mengalami naik dan turun nilai.

Tabel 5.6 Pengujian MongoDB file *text* pertama

No	MongoDB					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	6	1.627	44	63	1.483
2	30.000 data	19	1.582	46	64	280
3	50.000 data	31	1.617	49	67	266
4	70.000 data	44	1.605	56	69	1.251
5	100.000 data	64	1.559	47	72	266

Pengujian mongodb *insert* file *text* pertama pada tabel 5.6, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan, akan tetapi saat jumlah data memasuki 100.000 terjadi penurunan penggunaan *CPU Usage*.

Tabel 5.7 Pengujian MongoDB file *text* kedua

No	MongoDB					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	6	1.618	46	63	123
2	30.000 data	18	1.651	43	64	279
3	50.000 data	32	1.573	43	67	1.347

4	70.000 data	43	1.614	50	69	2.033
5	100.000 data	63	1.579	48	72	1.551

Pengujian mongodb *insert* file *text* dua pada tabel 5.7. Nilai *Runtime* yang naik sesuai dengan jumlah data yang dimasukan, sedangkan pada parameter *throughput*, *CPU Usage*, *Memory Usage* dan *Disk I/O* mempunyai nilai yang tidak stabil cenderung naik dan turun.

Tabel 5.8 Pengujian MongoDB file *text* ketiga

Nomor	MongoDB					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	6	1.633	45	53	414
2	30.000 data	19	1.620	45	64	276
3	50.000 data	32	1.574	49	67	519
4	70.000 data	43	1.647	53	69	1.783
5	100.000 data	65	1.534	52	72	275

Pengujian mongodb *insert* file *text* ketiga pada tabel 5.8, dengan nilai masing – masing parameter yang terus naik hingga data 70.000, kemudian turun nilai *Disk I/O* pada data 100.000.

Tabel 5.9 Pengujian MongoDB file *text* keempat

Pengujian No	MongoDB					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	6	1.695	46	63	571
2	30.000 data	18	1.653	46	64	970

3	50.000 data	31	1.594	46	67	1.405
4	70.000 data	44	1.576	49	69	646
5	100.000 data	63	1.577	47	72	1.745

Pengujian *insert* file *text* keempat pada tabel 5.8, dengan pengujian 10.000 sampai 70.000 kinerjanya masih naik stabil. Akan tetapi saat uji 100.000 kinerja *Disk I/O* menurun.

Tabel 5.10 Pengujian MongoDB file *text* kelima

Pengujian No	**MongoDB**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	10.000 data	6	1.555	40	63	845
2	30.000 data	20	1.535	50	65	270
3	50.000 data	32	1.582	54	67	2.308
4	70.000 data	44	1.592	47	69	734
5	100.000 data	64	1.571	46	72	799

Pengujian *insert* file *text* kelima pada tabel 5.10, dengan pengujian 10.000 sampai 70.000 kinerjanya masih naik stabil. Akan tetapi saat uji 100.000 kinerja *Disk I/O* menurun.

Tabel 5.11 Rata-rata nilai pengujian HBase file *text*

Nomor	**Hbase**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	10.000 data	13	818	48	75	148

2	30.000 data	33	917	58	79	387
3	50.000 data	51	977	54	79	231
4	70.000 data	71	987	56	83	477
5	100.000 data	100	992	55	87	1164
	Rata-rata	54	938	54	81	481

Pengujian rata-rata nilai dari hbase yang mengalami kenaikan nilai *Runtime*, *throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* sesuai dengan banyaknya data yang masuk. Terlihat pada tabel 5.11.

Tabel 5.12 Rata-rata nilai pengujian MongoDB file *text*

Nomor	**MongoDB**					
	Jumlah Baris Data	***Runtime* (Sec)**	***Throughput* (Ops/Sec)**	***CPU Usage* (%)**	***Memory Usage* (%)**	***Disk I/O* (kb)**
1	10.000 data	6	1.626	44	61	687
2	30.000 data	19	1.608	46	64	415
3	50.000 data	31	1.588	48	67	1.169
4	70.000 data	44	1.607	51	69	1.289
5	100.000 data	64	1.564	48	72	927
	Rata-rata	33	1.599	48	67	898

Pengujian rata-rata nilai dari hbase yang mengalami kenaikan nilai *Runtime*, *throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* sampai data ke 70.000, saat data uji 100.000 througput dan *Disk I/O* mengalami penurunan nilai. Terlihat pada tabel 5.12.

5.2.2 Pengujian Hbase dan MongoDB *Insert* Data Gambar

Pengujian kinerja *insert* data gambar kedalam Hbase dan MongoDB dengan operasi baris data sebanyak 1.000, 3.000, 5.000, 7.000 dan 10.000 data. Dengan 5 parameter yaitu *Runtime*, *throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O*. pengujian dilakukan sebanyak masing-masing 5 kali untuk mendapatkan hasil yang berbeda kemudian dirata-ratakan. Hasil pengujian database terdapat pada table berikut :

Tabel 5.13 Pengujian Hbase file gambar pertama

Pengujian No	Hbase					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	14	69	40	78	28.157
2	3.000 data	30	99	77	94	98.862
3	5.000 data	65	77	90	95	115.636
4	7.000 data	87	81	90	91	120.867
5	10.000 data	120	83	87	95	114.367

Pengujian Hbase *insert* file gambar pertama pada tabel 5.13, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan, akan tetapi saat jumlah data memasukin 70.000 mengalami penurunan penggunaan memory dan pada 100.000 data terjadi penurunan penggunaan *CPU Usage*.

Tabel 5.14 Pengujian Hbase file gambar kedua

Pengujian No	Hbase					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	11	89	73	95	48.587
2	3.000 data	28	106	72	95	98.400
3	5.000 data	57	87	95	95	107.271
4	7.000 data	96	73	92	95	119087
5	10.000 data	125	79	84	93	104.391

Pengujian Hbase *insert* file gambar kedua pada tabel 5.14, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.15 Pengujian Hbase file gambar ketiga

Pengujian No	Hbase					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	10	96	68	93	51.462
2	3.000 data	26	113	78	95	93.989
3	5.000 data	71	71	81	86	93.471
4	7.000 data	97	72	89	95	123.763
5	10.000 data	115	87	85	95	121.631

Pengujian Hbase *insert* file gambar ketiga pada tabel 5.15, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.16 Pengujian Hbase file gambar keempat

Pengujian No	Hbase					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	15	67	87	95	119.120
2	3.000 data	44	69	83	90	71.302
3	5.000 data	53	94	80	94	113.732

4	7.000 data	81	86	85	95	84.514
5	10.000 data	142	70	93	95	103.031

Pengujian Hbase *insert* file gambar keempat pada tabel 5.16, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.17 Pengujian Hbase file gambar kelima

Pengujian No	**Hbase**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	10	99	89	94	122.129
2	3.000 data	36	84	89	94	118.883
3	5.000 data	63	79	87	94	98.963
4	7.000 data	79	88	83	93	99.571
5	10.000 data	132	76	87	91	103.884

Pengujian Hbase *insert* file gambar lima pada tabel 5.17, dengan nilai yang cenderung naik sesuai dengan jumlah data yang dimasukan pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.18 Pengujian MongoDB file gambar pertama

Pengujian No	**MongoDB**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	11	92	44	45	13.883

2	3.000 data	33	92	69	84	25.052
3	5.000 data	57	88	67	95	22.622
4	7.000 data	80	88	76	95	30.306
5	10.000 data	118	85	56	96	24.963

Pengujian MongoDB *insert* file gambar pertama pada tabel 5.18, dengan nilai yang cenderung naik pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.19 Pengujian MongoDB file gambar kedua

Pengujian No	**MongoDb**					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	11	91	38	46	14.993
2	3.000 data	35	85	69	84	28.407
3	5.000 data	56	90	54	95	23.683
4	7.000 data	79	89	77	95	25.471
5	10.000 data	118	85	57	95	26.562

Pengujian MongoDB *insert* file gambar kedua pada tabel 5.19, dengan nilai yang cenderung naik pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.20 Pengujian MongoDB file gambar ketiga

Pengujian No	MongoDB					
	Jumlah Baris Data	*Runtime (Sec)*	*Throughput (Ops/Sec)*	*CPU Usage (%)*	*Memory Usage (%)*	*Disk I/O (kb)*
1	1.000 data	10	98	31	47	12.757
2	3.000 data	32	94	76	84	28.402
3	5.000 data	56	90	76	96	28.779
4	7.000 data	80	87	60	95	27.857
5	10.000 data	118	84	77	95	30.600

Pengujian MongoDB *insert* file gambar ketiga, dengan nilai yang cenderung naik pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.21 Pengujian MongoDB file gambar keempat

Pengujian No	MongoDB					
	Jumlah Baris Data	*Runtime (Sec)*	*Throughput (Ops/Sec)*	*CPU Usage (%)*	*Memory Usage (%)*	*Disk I/O (kb)*
1	1.000 data	10	100	10	47	20.659
2	3.000 data	35	85	23	85	56.930
3	5.000 data	57	88	29	95	71.908
4	7.000 data	78	90	22	95	53.135
5	10.000 data	118	85	29	95	69.670

Pengujian MongoDB *insert* file gambar keempat, dengan nilai yang cenderung naik pada parameter *Runtime*, akan tetapi pada parameter *Throughput*, *CPU Usage*, *Memory Usage*, dan *Disk I/O* mengalami naik turun nilai.

Tabel 5.22 Pengujian MongoDB file gambar kelima

Pengujian No	MongoDB					
	Jumlah Baris Data	*Runtime (Sec)*	*Throughput (Ops/Sec)*	*CPU Usage (%)*	*Memory Usage (%)*	*Disk I/O (kb)*
1	1.000 data	10	102	10	50	21.602
2	3.000 data	33	90	25	95	73.055
3	5.000 data	57	88	26	95	71.092
4	7.000 data	79	88	29	95	77.737
5	10.000 data	117	85	29	95	67.010

Pengujian mongodb *insert* gambar kelima *Runtime* naik dengan stabil, *throughput* mengalami penurunan, tidak banyak menggunakan cpu dan *Memory Usage*, dan *Disk I/O* yang naik dan turun. Terlihat pada tabel 5.22

Tabel 5.23 Rata-rata nilai pengujian Hbase file gambar

Nomor	Hbase					
	Jumlah Baris Data	*Runtime (Sec)*	*Throughput (Ops/Sec)*	*CPU Usage (%)*	*Memory Usage (%)*	*Disk I/O (kb)*
1	1.000 data	12	84	71	91	73.891
2	3.000 data	33	94	80	94	96.287
3	5.000 data	62	82	86	93	105.815
4	7.000 data	88	80	88	94	109.560
5	10.000 data	127	79	87	94	109.461
	Rata-rata	64	84	83	93	99.003

Pengujian rata-rata nilai hbase, dengan *Runtime* yang semakin besar sesuai dengan banyak data. *Throughput* yang sempat naik pada 3000 data dan kemudian turun pada

5000 sampai 10.000 data. *CPU Usage* yang naik sesuai dengan jumlah data. *Memory Usage* yang naik dan turun. Terkahir *Disk I/O* yang naik dengan nilai cukup tinggi. Terlihat pada tabel 5.23.

Tabel 5.24 Rata-rata nilai pengujian MongoDB file gambar

Nomor	MongoDb					
	Jumlah Baris Data	***Runtime (Sec)***	***Throughput (Ops/Sec)***	***CPU Usage (%)***	***Memory Usage (%)***	***Disk I/O (kb)***
1	1.000 data	10	97	27	47	16.779
2	3.000 data	34	89	52	87	42.369
3	5.000 data	56	89	50	95	43.617
4	7.000 data	79	88	53	95	42.901
5	10.000 data	118	85	50	95	43.761
	Rata-rata	60	90	46	84	37.885

Pengujian rata-rata nilai mongodb pada tabel 5.24, dengan *Runtime* yang naik sesuai dengan banyaknya data. *Throughput* yang justru menurun. *CPU Usage* naik dan turun. *Memory Usage* cenderung naik, dan *Disk I/O* naik dengan nilai cukup tinggi.

BAB 6 PEMBAHASAN

Pada bab ini akan dibahas hasil uji dari scenario yang diterapkan pada dua database yaitu Hbase dan MongoDB. Pengujian dibagi menjadi dua yaitu pengujian penyimpanan dan pengujian database.

6.1 Pembahasan Pengujian Penyimpanan Data

Pada hasil pengujian penyimpanan data ditentukan oleh dua nilai yaitu berhasil atau tidak berhasil. Berikut adalah table hasil pengujian penyimpanan data :

Tabel 6.1 Hasil Pengujian Penyimpanan Data

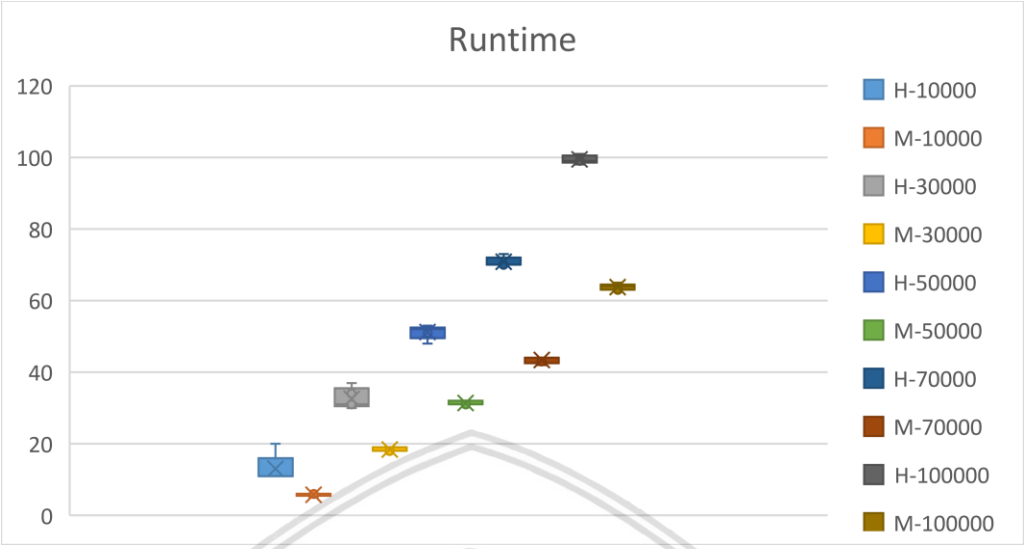
No	Deskripsi Pengujian	Hasil
UP_001	API Webservice dapat mengirim data sensor dht11 dari Internet Gateway Device ke database Hbase	Berhasil
UP_002	API Webservice dapat mengirim data sensor kamera dari Internet Gateway Device ke database Hbase	Berhasil
UP_003	Database Hbase Dapat menyimpan data dari sensor dht11	Berhasil
UP_004	Database Hbase dapat menyimpan data dari sensor kamera	Berhasil

6.2 Pembahasan Pengujian Data Storage Hbase dan MongoDB

Pada bab ini akan dibahas tentang hasil rata-rata Hbase dan MongoDB, yang akan ditampilkan hasil kinerjanya dalam bentuk grafik. Grafik yang digunakan menggunakan grafik whisker, grafrik ini berfungsi untuk mengetahui sebaran data yang dilakukan selama lima kali pengujian. Selain mengetahui sebaran data, grafik whisker berfungsi untuk mengetahui nilai tengah atau rata-rata, nilai terendah dan nilai tertinggi. Akan tetapi pembahasan data pengujian akan menampilkan hasil rata-ratanya saja.

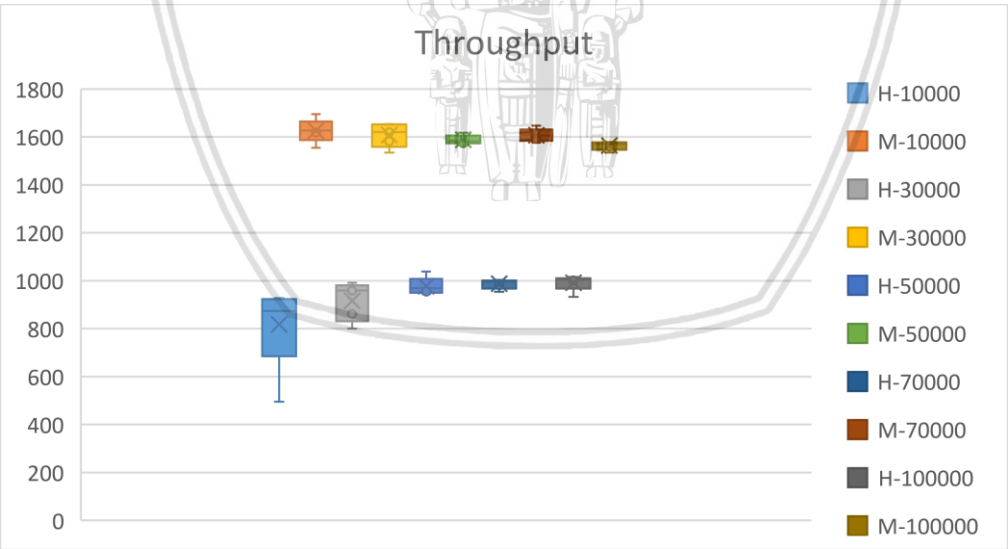
6.2.1 Pengujian Hbase dan MongoDB *Insert* Data *Text*

Pengujian *insert* data *text* pada hbase dan mongodb terdapat pada grafik chart dibawah dengan masing-masing parameternya :



Gambar 6.1 *Insert* data *text* dengan parameter *Runtime*

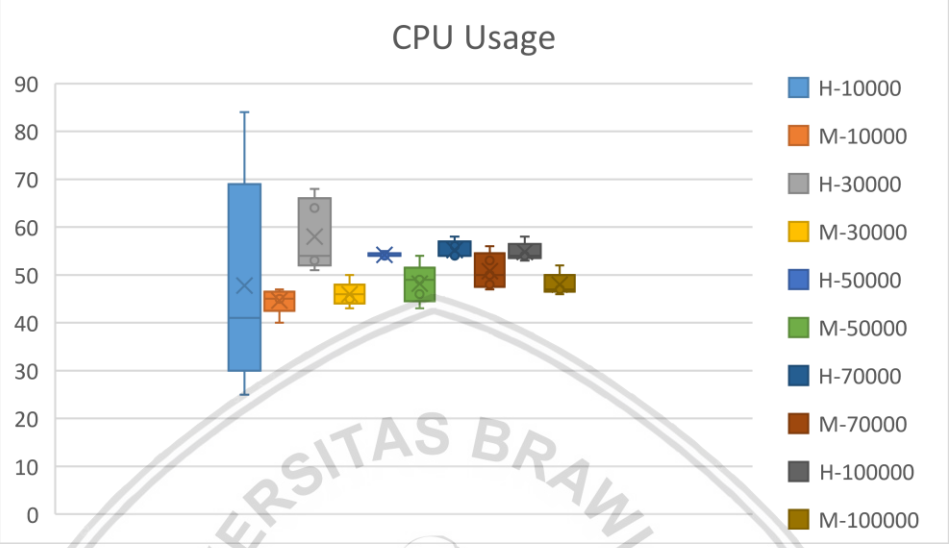
Terlihat pada parameter *Runtime* mongodb yang lebih cepat dibandingkan dengan dibandingkan hbase. Nilai rata-rata pada hbase diantara lain 13s, 33s, 51s, 71s, dan 100s, sedangkan pada mongoDB yaitu 6s , 19s, 31s, 44s, dan 64s untuk masing-masing banyak data sebesar 10.000, 30.000, 50.000, 70.000 dan 100.000 *text* dengan satuan detik(second/s). jadi dapat disimpulkan bahwa peningkatan jumlah variasi data dalam melakukan operasi *insert*, maka *Runtime* semakin naik.



Gambar 6.2 *Insert* data *text* dengan parameter Througput

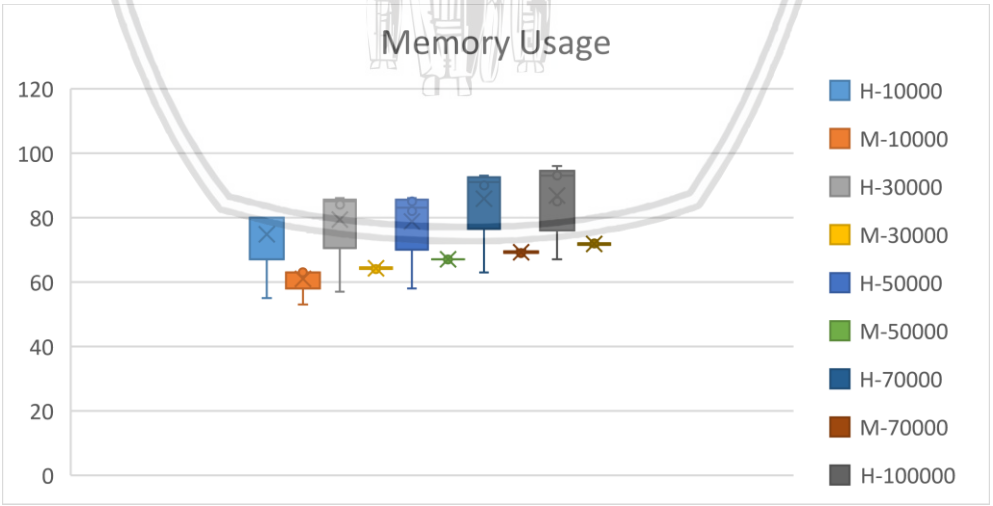
Terlihat pada parameter througput kinerja hbase lebih lambat dibandingkan dengan mongoDB yang cenderung lebih cepat. Nilai rata-rata pada hbase diantara lain 818 ops/s, 917 ops/s, 977 ops/s, 984 ops/s, dan 992 ops/s, sedangkan pada mongoDB lebih cepat yaitu 1626 ops/s , 1608 ops/s, 1588 ops/s, 1607 ops/s, dan 1564 ops/s

untuk masing-masing banyak data sebesar 10.000, 30.000, 50.000, 70.000 dan 100.000 *text* dengan satuan operasi(ops)/second(s). jadi dapat diambil kesimpulan bahwa, besar kecilnya *throughput* dipengaruhi oleh nilai *Runtime*.



Gambar 6.3 *Insert* data *text* dengan parameter *CPU Usage*

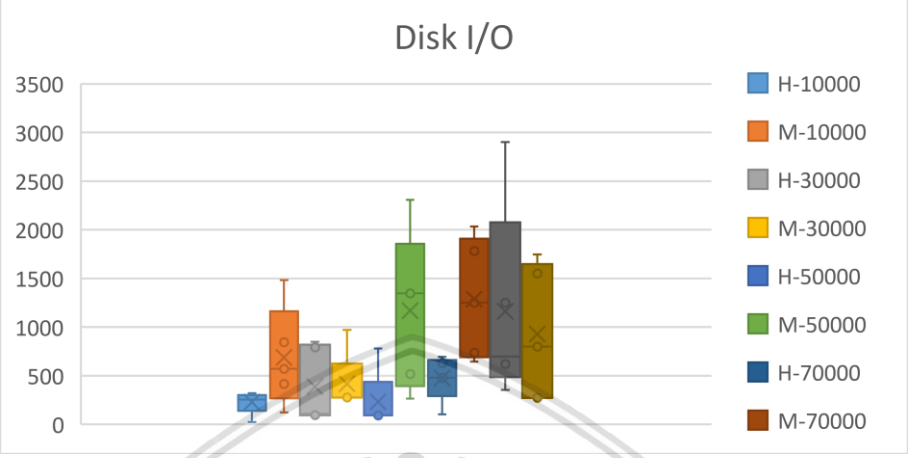
Terlihat pada parameter *CPU Usage* kinerja hbase yang besar dalam pemakaian dibandingkan dengan mongoDB lebih kecil pemakaian cpunya. Nilai rata-rata pada hbase diantara lain 48%, 58%, 54%, 56%, dan 55%, sedangkan pada mongoDB yaitu 44% , 46%, 48%, 51%, dan 48% untuk masing-masing banyak data sebesar 10.000, 30.000, 50.000, 70.000 dan 100.000 *text* dengan persen (%).



Gambar 6.4 *Insert* data *text* dengan parameter *Memory Usage*

Terlihat pada parameter *Memory Usage* kinerja hbase yang besar dalam pemakaian dibandingkan dengan mongoDB lebih kecil pemakaian cpunya. Nilai rata-rata pada hbase diantara lain 75%, 79%, 79%, 83%, dan 87%, sedangkan pada

mongoDB yaitu 61% , 64%, 67%, 69%, dan 72% untuk masing-masing banyak data sebesar 10.000, 30.000, 50.000, 70.000 dan 100.000 *text* dengan persen (%).

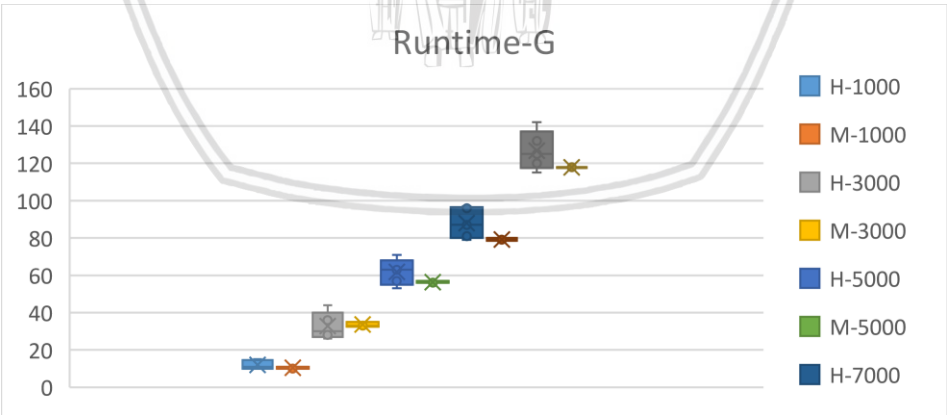


Gambar 6.5 *Insert* data *text* dengan parameter *Disk I/O*

Terlihat pada parameter *Disk I/O* kinerja hbase yang kecil dalam pemakaian dibandingkan dengan mongoDB lebih besar *Disk I/O*. Nilai rata-rata pada hbase diantara lain 148kb, 387kb, 231kb, 477kb, dan 1164kb sedangkan pada mongoDB yaitu 687kb, 415kb, 1169kb, 1289b, dan 927kb untuk masing-masing banyak data sebesar 10.000, 30.000, 50.000, 70.000 dan 100.000 *text* dengan persen (kb).

6.2.2 Pengujian Hbase dan MongoDB *Insert* Data Gambar

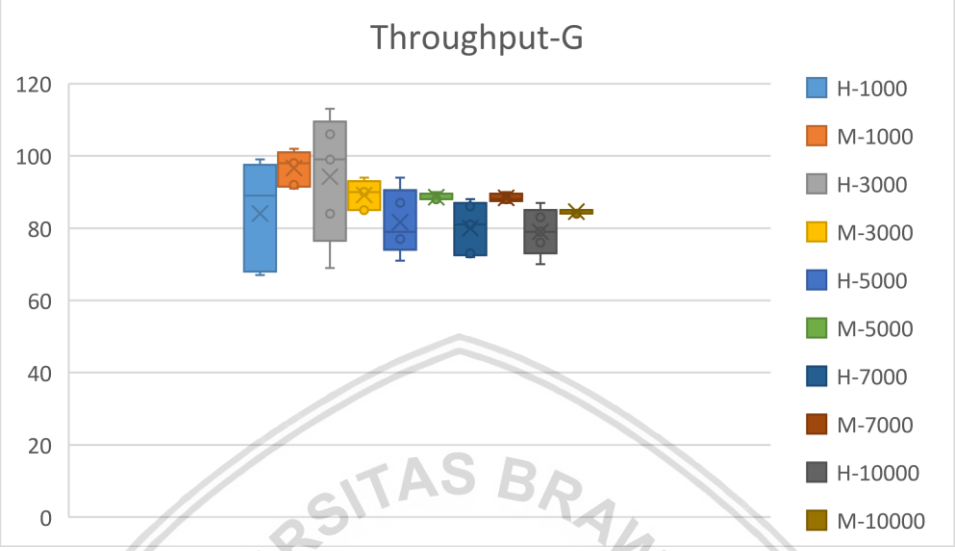
Pengujian *insert* data gambar pada hbase dan mongodb terdapat pada grafik chart dibawah dengan masing-masing parameternya :



Gambar 6.6 *Insert* data gambar dengan parameter *Runtime*

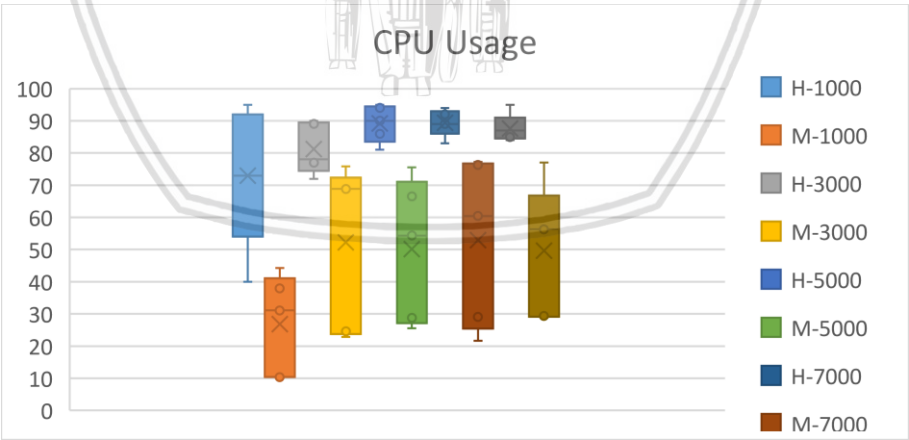
Terlihat pada parameter *Runtime* kinerja hbase dan mongoDB tidak jauh berbeda terlihat data meningkat sesuai jumlah data yang dimasukan. Nilai rata-rata pada hbase diantara lain 12s, 33s, 62s, 88s, dan 127s, sedangkan pada mongoDB terlihat data meningkat sebesar 10s , 34s, 56s, 79s, dan 118s untuk masing-masing banyak

data sebesar 1.000, 3.000, 5.000, 7.000 dan 10.000 *text* dengan satuan detik(second/s).



Gambar 6.7 *Insert* data gambar dengan parameter *Throughput*

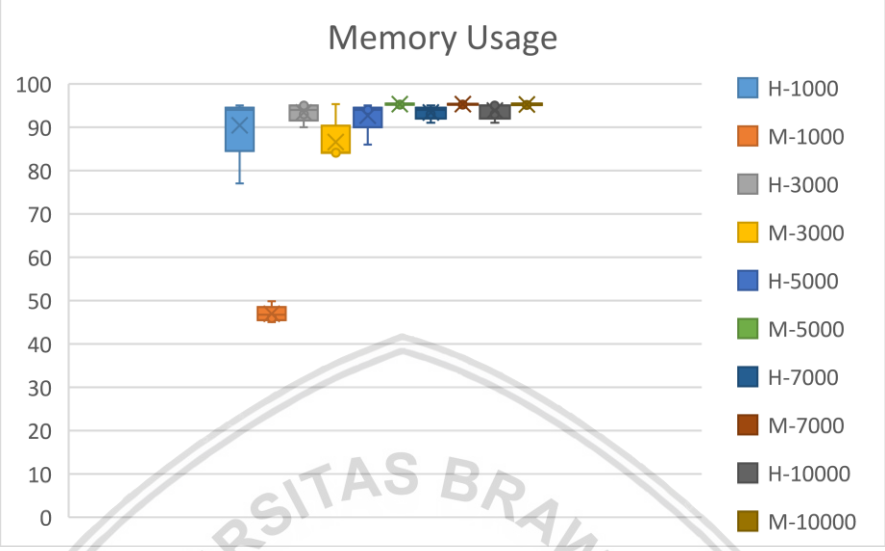
Terlihat pada parameter througput kinerja hbase lebih cepat diawal dibandingkan dengan mongoDB yang cenderung stabil. Nilai rata-rata pada hbase diantara lain 84 ops/s, 94 ops/s, 82 ops/s, 80 ops/s, dan 79 ops/s, sedangkan pada mongoDB tidak stabil yaitu 97 ops/s , 89 ops/s, 89 ops/s, 88 ops/s, dan 85 ops/s untuk masing-masing banyak data sebesar 1.000, 3.000, 5.000, 7.000 dan 10.000 *text* dengan satuan operasi(ops)/second(s).



Gambar 6.8 *Insert* data gambar dengan parameter *CPU Usage*

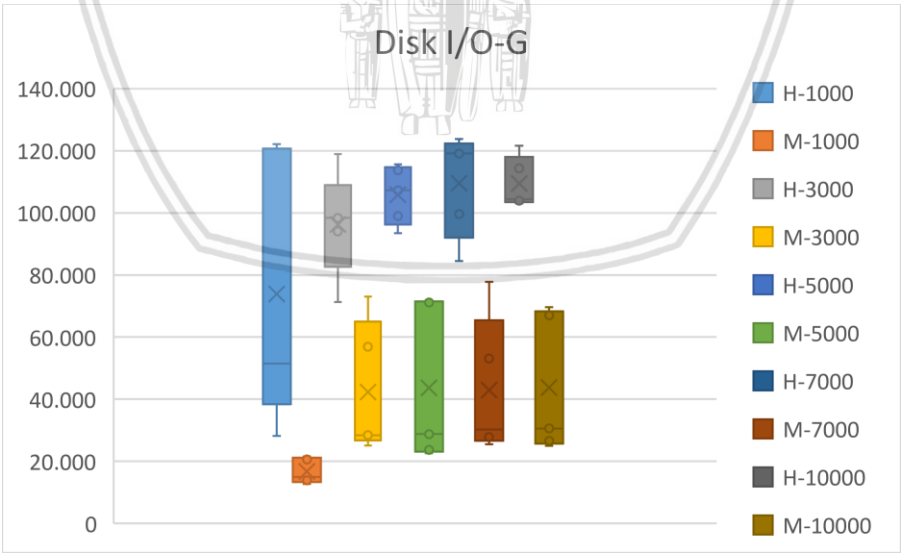
Terlihat pada parameter *CPU Usage* kinerja hbase yang besar dalam pemakaian dibandingkan dengan mongoDB lebih kecil pemakaian cpunya. Nilai rata-rata pada hbase diantara lain 60%, 76%, 88%, 90%, 85%, sedangkan pada mongoDB yaitu 18% ,

34%, 38%, 38%, dan 38% untuk masing-masing banyak data sebesar 1.000, 3.000, 5.000, 7.000 dan 10.000 gambar dengan satuan persen (%).



Gambar 6.9 *Insert* data gambar dengan parameter *Memory Usage*

Terlihat pada parameter *Memory Usage* kinerja hbase yang besar dalam pemakaian dibandingkan dengan mongoDB lebih kecil pemakaian cpunya. Nilai rata-rata pada hbase diantara lain 91%, 94%, 93%, 94%, dan 94%, sedangkan pada mongoDB yaitu 47% , 87%, 95%, 95%, dan 95% untuk masing-masing banyak data sebesar 1.000, 3.000, 5.000, 7.000 dan 10.000 gambar dengan persen (%).



Gambar 6.10 *Insert* data gambar dengan parameter *Disk I/O*

Terlihat pada parameter *Disk I/O* kinerja hbase yang besar dibandingkan dengan kinerja *Disk I/O* mongoDB. Nilai rata-rata pada hbase diantara lain 73.891kb,

96.287kb, 105.815kb, 109.560kb, dan 109.461kb sedangkan pada mongoDB yaitu 16.779kb, 42.369kb, 43.617kb, 42.901kb, dan 43.761kb untuk masing-masing banyak data sebesar 1.000, 3.000, 5.000, 7.000 dan 10.000 gambar dengan parameter kilobytes(kb).

6.3 Pembahasan Pengujian Kinerja

1. *Runtime* :

   pengujian data *text* menggunakan parameter *Runtime* dengan banyak data 10.000 menghasilkan 13s hbase, 6s mongodb. *Text* dengan parameter *Runtime* dengan banyak data 30.000 menghasilkan 33s hbase, 19s mongodb. *Text* dengan parameter *Runtime* banyak data 50.000 menghasilkan 51s hbase, 62s mongodb. *Text* dengan parameter *Runtime* dengan banyak data 70.000 menghasilkan 71s hbase, 33s mongodb. *Text* dengan parameter *Runtime* dengan banyak data 100.000 menghasilkan 100s hbase, 64s mongodb.

   pengujian data gambar menggunakan parameter *Runtime* dengan banyak data 1.000 menghasilkan 12s hbase, 10s mongodb. Gambar dengan parameter *Runtime* dengan banyak data 3.000 menghasilkan 33s hbase, 34s mongodb. Gambar dengan parameter *Runtime* banyak data 5.000 menghasilkan 62s hbase, 56s mongodb. gambar dengan parameter *Runtime* dengan banyak data 7.000 menghasilkan 88s hbase, 79s mongodb. gambar dengan parameter *Runtime* dengan banyak data 10.000 menghasilkan 127s hbase, 118s mongodb.

2. *Throughput* :

   pengujian data *text* menggunakan parameter *throughput* dengan banyak data 10.000 menghasilkan 818 ops/s hbase, 1.626 ops/s mongodb. *Text* dengan parameter *throughput* dengan banyak data 30.000 menghasilkan 917 ops/s hbase, 1.608 ops/s mongodb. *Text* dengan parameter *throughput* banyak data 50.000 menghasilkan 977 ops/s hbase, 1.588 ops/s mongodb. *Text* dengan parameter *throughput* dengan banyak data 70.000 menghasilkan 987 ops/s hbase, 1.607 ops/s mongodb. *Text* dengan parameter *throughput* dengan banyak data 100.000 menghasilkan 992 ops/s hbase, 1.564 ops/s mongodb.

   pengujian data gambar menggunakan parameter *throughput* dengan banyak data 1.000 menghasilkan 84 ops/s hbase, 97 ops/s mongodb. Gambar dengan parameter *throughput* dengan banyak data 3.000 menghasilkan 94 ops/s hbase, 89 ops/s mongodb. Gambar dengan parameter *throughput* banyak data 5.000 menghasilkan 82 ops/s hbase, 89 ops/s mongodb. Gambar dengan parameter *throughput* dengan banyak data 7.000 menghasilkan 80 ops/s hbase, 88 ops/s mongodb. Gambar dengan parameter *throughput* dengan banyak data 10.000 menghasilkan 79 ops/s hbase, 85 ops/s mongodb.

3. *CPU Usage* :

   Pengujian data *text* menggunakan parameter *CPU Usage* dengan banyak data 10.000 menghasilkan 48% hbase, 44% mongodb. *Text* dengan parameter *CPU Usage* dengan banyak data 30.000 menghasilkan 58% hbase, 46% mongodb. *Text* dengan parameter *CPU Usage* banyak data 50.000 menghasilkan 54% hbase, 48% mongodb. *Text* dengan parameter *CPU Usage* dengan banyak data 70.000 menghasilkan 56% hbase, 51% mongodb. *Text* dengan parameter *CPU Usage* dengan banyak data 100.000 menghasilkan 55% hbase, 48% mongodb.

   Pengujian data gambar menggunakan parameter *CPU Usage* dengan banyak data 1.000 menghasilkan 71% hbase, 27% mongodb. Gambar dengan parameter *CPU Usage* dengan banyak data 3.000 menghasilkan 80% hbase, 52% mongodb. Gambar dengan parameter *CPU Usage* banyak data 5.000 menghasilkan 86%, 50% mongodb. Gambar dengan parameter *CPU Usage* dengan banyak data 7.000 menghasilkan 88% hbase, 53% mongodb. Gambar dengan parameter *CPU Usage* dengan banyak data 10.000 menghasilkan 87% hbase, 50% mongodb.

4. *Memory Usage*:

   Pengujian data *text* menggunakan parameter *Memory Usage* dengan banyak data 10.000 menghasilkan 75% hbase, 61% mongodb. *Text* dengan parameter *Memory Usage* dengan banyak data 30.000 menghasilkan 79% hbase, 64% mongodb. *Text* dengan parameter *Memory Usage* banyak data 50.000 menghasilkan 79% hbase, 67% mongodb. *Text* dengan parameter *Memory Usage* dengan banyak data 70.000 menghasilkan 83% hbase, 69% mongodb. *Text* dengan parameter *Memory Usage* dengan banyak data 100.000 menghasilkan 87% hbase, 72% mongodb.

   Pengujian data gambar menggunakan parameter *Memory Usage* dengan banyak data 1.000 menghasilkan 91% hbase, 47% mongodb. Gambar dengan parameter *Memory Usage* dengan banyak data 3.000 menghasilkan 94% hbase, 87% mongodb. Gambar dengan parameter *Memory Usage* banyak data 5.000 menghasilkan 93%, 95% mongodb. Gambar dengan parameter *Memory Usage* dengan banyak data 7.000 menghasilkan 94% hbase, 95% mongodb. Gambar dengan parameter *Memory Usage* dengan banyak data 10.000 menghasilkan 94% hbase, 95% mongodb.

5. *Disk I/O*:

   Pengujian data *text* menggunakan parameter *Disk I/O* dengan banyak data 10.000 menghasilkan 204kb hbase, 687kb mongodb. *Text* dengan parameter *Disk I/O* dengan banyak data 30.000 menghasilkan 387kb hbase, 415kb mongodb. *Text* dengan parameter *Disk I/O* banyak data 50.000 menghasilkan 231kb hbase, 1.169kb mongodb. *Text* dengan parameter *Disk I/O* dengan banyak data 70.000 menghasilkan 477kb hbase, 1.289kb mongodb. *Text* dengan parameter *Disk I/O* dengan banyak data 100.000 menghasilkan

1164kb hbase, 927kb mongodb.

Pengujian data gambar menggunakan parameter *Disk I/O* dengan banyak data 1.000 menghasilkan 73.891kb. hbase, 16.779kb mongodb. Gambar dengan parameter *Disk I/O* dengan banyak data 3.000 menghasilkan 96.287 kb hbase, 87% mongodb. Gambar dengan parameter *Disk I/O* banyak data 5.000 menghasilkan 105.815 kb, 43.617kb mongodb. Gambar dengan parameter *Disk I/O* dengan banyak data 7.000 menghasilkan 109.560kb hbase, 42.901kb mongodb. Gambar dengan parameter *Disk I/O* dengan banyak data 10.000 menghasilkan 109.461kb hbase, 43.761kb mongodb.

MongoDB dan Hbase memiliki kekurangan dan kelebihan masing – masing, jika dilihat dari kinerja diatas MongoDB tentu lebih baik. Akan tetapi Hbase pun masih layak digunakan untuk menyimpan data sensor pada *environment* yang sudah ada.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan diambil dari jawaban atas pertanyaan yang mengacu pada rumusan masalah yang sudah dijelaskan pada bab pertama. Dapat ditarik kesimpulan dari pengujian diatas sebagai berikut :

1. Hbase dibangun menggunakan VPS yang berada di FILKOM Universitas Brawijaya. VPS tersebut memiliki spesifikasi yang sama dengan penelitian MongoDB. Hbase dapat menyimpan dan menampilkan data yang heterogen dari node sensor dalam lingkungan (*environment)* sebelumnya yaitu dengan menggunakan lingkungan mongoDB.
2. Dari hasil pengujian hbase dan mongodb menggunakan dua pengujian *text* dan gambar dengan parameter yaitu *Runtime, throughput, CPU Usage, Memory Usage, dan Disk I/O* dan lima data uji menghasilkan perbandingan data yaitu rata-rata data *Text* dengan parameter *Runtime* sebesar 54s untuk hbase, 33s untuk mongodb. Parameter *throughput* sebesar 938 ops/s untuk hbase, 1599 ops/s untuk mongodb. Parameter *CPU Usage* sebesar 54% untuk hbase, 48% untuk mongodb. Parameter *Memory Usage* sebesar 81% untuk hbase, 67% untuk mongodb. Dan Parameter *Disk I/O* sebesar 482kb untuk hbase, 4354kb untuk mongodb. Sedangkan rata-rata data gambar parameter *Runtime* sebesar 64s hbase, 60s mongodb, parameter *Throughput* 84 ops/s hbase, 90 ops/s mongodb. Parameter *CPU Usage* 83% hbase, 46% mongo. Parameter *Memory Usage* 93% hbase, 84% mongodb. Dan parameter *Disk I/O* 99.003kb hbase, 37.885kb mongodb. Dari perbandingan rata-rata nilai pengujian MongoDB lebih banyak unggulnya dibandingkan dengan Hbase pada penyimpanan data *text* dan data gambar. Akan tetapi jika dilihat dari menyimpan data gambar dengan parameter *Disk I/O* Hbase berkinerja baik dibanding mongodb.

7.2 Saran

Saran yang dapat disampaikan untuk pengembangan lebih lanjut pada penelitian terkait adalah sebagai berikut :

1. Pengujian dapat dilakukan dengan menambah parameter uji lain, seperti tingkat keamanan data dalam database.
2. Penelitian berikutnya dapat membandingkan database NoSQL MongoDB dan Hbase dengan database NoSQL lainnya, untuk menyimpan data IoT yang beragam dan besar volumenya.
3. Penelitian diterapkan pada lingkungan yang berbeda, yang menyediakan penyimpanan data server yang lebih besar untuk menyimpan lebih banyak data.