IMPLEMENTASI *DYNAMIC VOLTAGE AND FREQUENCY SCALING* PADA *PARALLEL CLUSTER* ORANGE PI

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI REKASAYA KOMPUTER

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

RIDHA WEKATAMA

NIM. 125060307111028

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2018

LEMBAR PENGESAHAN

IMPLEMENTASI *DYNAMIC VOLTAGE AND FREQUENCY SCALING* PADA *PARALLEL CLUSTER* ORANGE PI

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI REKASAYA KOMPUTER

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

RIDHA WEKATAMA
NIM. 125060307111028 – 63

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
pada tanggal 6 Juni 2018

Mengetahui Ketua Jurusan Teknik Elektro	Menyetujui Dosen Pembimbing
Hadi Suyono, S.T., M.T., Ph.D., IPM.	**Waru Djuriatno, S.T., M.T.**
NIP. 19730520 200801 1 013	**NIP. 19690725 199702 1 001**

JUDUL SKRIPSI :

IMPLEMENTASI *DYNAMIC VOLTAGE AND FREQUENCY SCALING* PADA *PARALLEL CLUSTER* ORANGE PI

Nama Mahasiswa : Ridha Wekatama

NIM : 125060307111028

Program Studi : Teknik Elektro

Konsentrasi : Rekayasa Komputer

KOMISI PEMBIMBING :

Ketua : Waru Djuriatno, S.T., M.T. .....

TIM DOSEN PENGUJI :

Dosen Penguji 1 : Adharul Muttaqin, S.T., M.T. .....

Dosen Penguji 2 : Raden Arief Setiawan, S.T., M.T .....

Dosen Penguji 3 : Dr. Ir. Muhammad Aswin, M.T. .....

Tanggal Ujian : 24 Mei 2018

SK Penguji : 1091/UN10.F07/SK/2018

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan, dan masalah ilmiah yang diteliti dan diulas di dalam naskah skripsi ini adalah asli dari pemikiran saya. 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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 31 Mei 2018

Mahasiswa,

Ridha Wekatama

NIM 125060307111028

RINGKASAN

Ridha Wekatama, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Mei 2018, *Implementasi Skala Tegangan dan Frekuensi Dinamis pada Klaster Paralel Orange Pi*, Dosen Pembimbing: Waru Djuriatno.

Dalam kajian topik Big Data, terdapat banyak permasalahan yang masih membutuhkan berbagai macam solusi penyelesaiannya. Salah satunya adalah permasalahan untuk pengolahan data yang berukuran sangat besar. Dibutuhkan cara untuk meningkatkan kinerja dan efisiensi dalam pengolahan data. Komputasi paralel mampu meningkatkan performa komputasi dalam pengolaha data baik data yang kompleks maupun data yang berskala besar. Akan tetapi untuk memperoleh performa yang terbaik, tentu akan ada yang harus dikorbankan untuk mewujudkan hasil peningkatan performa tersebut, dalam hal ini adalah penggunaan energi. *Single-board computer* Orange Pi dihadirkan sebagai perangkat komputer berdimensi praktis dengan konsumsi daya yang lebih hemat dari komputer komoditas. Orange Pi mendukung implementasi Teknik Skala Tegangan dan Frekuensi Dinamis (*Dynamic Voltage and Frequency Scaling*). Dengan menggunakan Teknik DVFS, diharapkan dapat menghasilkan baik peningkatan performa maupun penghematan energi komputasi paralel sebagai solusi pengolahan Big Data. Dalam penelitian, hasil penelilitan akan menunjukkan peningkatan kinerja serta penghematan yang diperoleh dari sistem. Pengujian dilakukan menggunakan program penghitung cacah kata bebasis pemrogramam Java pada lingkup pengembangan *Hadoop framework*, mendukung sistem berkas terdistribusi, dengan memanfaatkan algoritma *mapreduce*.

Kata kunci— *big data, paralel computing, dvfs, mapreduce, hadoop framework.*

SUMMARY

Ridha Wekatama, Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, May 2018, *Implementation of Dynamic Voltage and Frequency Scaling on Orange Pi Parallel Cluster*, Academic Supervisor: Waru Djuriatno.

Big Data is one of an issues which may be faced by the advancement of information technology. Processing voluminous and complex larger scale data would need some methods to increase computation's performance and efficiency. Parallel Computation could increase computation's performance on Big Data's problems and complex data calculation problems. But in order to obtain greater performance, there'll an aspect for trade-off. In this case, power consumption become the trade-off. A better computation's performance will need a higher power consumption. Orange Pi, open source single-board computer, is a computer which has smaller dimension and lower power consumption than commodity computer. To improve performance of parallel computing, Dynamic Voltage and Frequency Scaling will be used within Parallel Cluster Orange Pi. By using DVFS, we will be able to know how power consumption is affected by the computation's performances. The challenge of this research is how to get the optimal performance with efficient power consumption. In this research, testing is done in Hadoop framework environment, support Hadoop distributed file system, by running word count program using Map Reduce algorithm.

Keywords— big data, paralel computing, dvfs, mapreduce, hadoop framework.

PENGANTAR

Puji syukur kehadirat Allah SWT, karena atas rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi Skala Tegangan dan Frekuensi Pada Klaster Paralel Orange Pi".

Tujuan dari pembuatan tugas akhir ini adalah sebagai salah satu syarat kelulusan yang harus dipenuhi oleh mahasiswa Teknik Elektro Fakultas Teknik Universitas Brawijaya untuk meraih gelar sarjana teknik dengan harapan dapat menjadi sumbangsih bagi ilmu pengetahuan di bidang Teknik Elektro khususnya bidang Rekayasa Komputer.

Dalam menyusun penelitian ini, penulis telah banyak mendapat bantuan dan bimbingan. Maka dari itu penulis ingin mengucapkan banyak terima kasih kepada:

1. Bapak Hadi Suyono, S.T., M.T., Ph. D dan Ibu Ir. Nurussa'adah, M.T., selaku Ketua dan Sekretaris Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya yang sangat membantu kelancaran tugas akhir ini.
2. Bapak Ali Mustofa, S.T., M.T, selaku Ketua Program Studi S1 Teknik Elektro Fakultas Teknik Universitas Brawijaya yang sangat membantu kelancaran tugas akhir ini.
3. Bapak Waru Djuriatno, S.T., M.T, selaku dosen pembimbing yang telah meluangkan waktu untuk membimbing dan memberikan saran serta membuka wawasan ilmu pengetahuan penulis.
4. Bapak Adharul Muttaqin, S.T., M.T, Bapak R. Arief Setiawan, S.T., M.T, Bapak Dr. Ir. Muhammad Aswin, M.T, selaku dosen rekayasa komputer yang telah meluangkan waktu memberikan saran serta masukan kepada penulis.
5. Bapak Raden Arief Setiawan, S.T., M.T, selaku dosen pembimbing akademik yang telah membimbing penulis selama perkuliahan.
6. Bapak dan Ibu dosen Teknik Elektro yang telah memberikan saran dan masukan kepada penulis.
7. Mas Rakhmad Romadhoni, selaku pihak dari Laboratorium Infomatika dan Komputer yang telah membantu penulis.
8. Kedua orang tua penulis serta kedua kakak yang telah memberikan dukungan secara penuh baik mental maupun materiil.
9. Alin, Arif dan Risto, yang merupakan kolega seperjuangan kuliah.
10. Seluruh Anggota RisTIE yang telah memperluas wawasan penulis dan melatih

kemampuan penulis dalam bekerja secara tim.

11. Seluruh Asisten Laboratorium Informatika dan Komputer serta kru Rekayasa Komputer yang telah menemani penulis saat melaksanakan penelitian.
12. Dan teman-teman terutama kekasih yang telah menyadarkan penulis untuk melepaskan diri dari zona nyaman, serta adik-adik yang senantiasa memberikan semangat bagi penulis.

Penulis pun menyadari bahwa penulis tidak terlepas dari kekurangan dan keterbatasan. Begitupun dalam penyusunan tugas akhir ini, dengan kerendahan hati penulis menantikan adanya masukan, baik berupa saran maupun kritik yang dapat bersifat membangun guna penyusunan laporan kedepannya.

Akhir kata penulis berharap agar tugas akhir ini dapat bermanfaat dan berguna bagi pembaca dan semua pihak yang memerlukan.

Malang, Mei 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring meningkatnya data yang tidak henti-hentinya mengalir dalam dunia komputer, peningkatan kebutuhan untuk penyimpanan, pengolahan dan pengambilan data dalam jumlah besar menjadi permasalahan yang harus diberikan solusi. Perkembangan teknologi dunia mengajukan *High Parallel Computing* sebagai salah satu solusi untuk memenuhi tantangan kebutuhan pengolahan *Big Data* seperti data saintifik, data eksperimen, atau data hasil simulasi sistem berskala besar. Kemajuan Sistem HPC menghasilkan peningkatan kinerja yang semakin besar pada pengolahan data (Edson L. Padoin, p1, 2014). HPC dapat menyelesaikan permasalahan komputasi yang lebih besar dan rumit namun semakin tinggi peningkatan kinerja komputasi, maka akan semakin tinggi pula penggunaan energi yang dibutuhkan.

Penghematan energi menjadi salah satu isu yang harus diperhatikan dalam pengembangan sistem komputasi paralel. Salah satu teknik yang dapat dimanfaatkan sebagai penghematan adalah teknik skala tegangan dan frekuensi dinamis atau DVFS. DVFS mampu melakukan penyesuaian penggunaan tegangan dan frekuensi sesuai dengan beban komputasi yang diperlukan secara dinamis. Selain digunakan sebagai teknik penghematan energi, DVFS juga dapat digunakan sebagai peningkat kinerja suatu perangkat komputasi, yakni dengan meningkatkan frekuensi proses komputasi yang seiring juga meningkatkan penggunaan energi (Maja Etinski et al, p1, 2009). Pembahasan pada penelitian ini mengarah pada penghematan menggunakan DVFS, dengan harapan komputasi dapat berjalan dengan efisiensi yang optimal tanpa menurunkan kinerja komputasi.

Arsitektur Hadoop dihadirkan sebagai salah satu solusi untuk menangani penyimpanan, pengolahan maupun penerimaan *Big Data* (Nan Zhu, p2 2014). Klaster Hadoop merupakan konfigurasi komputer paralel yang dikhususkan secara spesifik untuk menyimpan dan menganalisis data berskala besar yang tidak berstruktur pada lingkup komputasi terdistribusi. Hadoop merupakan sebuah *framework* berbasis *open source* dengan komponen utama yaitu *MapReduce* dan *Hadoop Distributed File System* (HDFS).

Penghematan energi juga dapat diwujudkan dengan memanfaatkan perangkat keras yang rendah konsumsi daya dibandingkan dengan komputer komoditas pada umumnya. *Open-source single board computer* merupakan mesin yang memiliki dimensi sebesa kartu kredit dan mampu berjalan layaknya komputer komoditas tetapi dengan penggunaan daya yang lebih rendah (Aaron M. Pfalzgraf, p1, 2014). Orange Pi merupakan sebuah mini PC berbasis *open source single board computer* berbasis ARM dengan spesifikasi yang cukup untuk menjalankan fungsinya sebagai komputer. Selain penggunaan daya listrik yang lebih rendah, Orange Pi juga memiliki harga yang terjangkau dan konfigurasi yang mudah.

Berdasarkan penelitian sebelumnya telah diterapkan pemodelan klaster menggunakan Orange Pi, akan tetapi terdapat kekurangan dalam berbagai sisi seperti penggunaan daya yang tidak terkontrol secara baik menyebabkan terjadinya pembuangan energi yang tidak bermanfaat, serta penurunan kinerja periferal perangkat keras dalam komunikasi data antar node paralel. Selain itu ditemukan adanya ketidakcocokan permasalahan komputasi paralel pada pemodelan sistem yang telah dirancang. Diharapkan konfigurasi klaster yang akan dilakukan pada penelitian ini dapat membuktikan penghematan energi tanpa mengurangi kinerja komputasi paralel.

1.2 Identifikasi Masalah

Dengan semakin tinggi peningkatan kinerja komputasi paralel akan semakin tinggi pula konsumsi energi yang digunakan pada sistem, maka perlu ditemukan suatu solusi untuk masalah tersebut. Salah satu solusi tersebut adalah dengan menggunakan konfigurasi penghematan pada sistem paralel. Sehingga masalah yang ditemukan adalah sebagai berikut:

1. Besarnya penggunaan daya pada suatu sistem komputer paralel.
2. Untuk menjaga kinerja komputer paralel agar tidak berubah ataupun turun, maka diperlukan metode untuk menghasilkan kestabilan pada sistem komputer paralel.
3. Untuk menghasilkan penghematan energi tanpa mempengaruhi kinerja komputasi dibutuhkan penyelesaian untuk meningkatkan efisiensi sistem komputer paralel.

1.3 Rumusan Masalah

Berdasarkan identifikasi masalah yang ditemukan, maka rumusan masalah yang akan dibahas adalah sebagai berikut:

1. Bagaimana pengaruh beban komputasi terhadap penggunaan energi pada sistem komputer paralel.
2. Bagaimana pengaruh teknik skala tegangan dan frekuensi dinamis terhadap penggunaan daya pada sistem komputer paralel.

3. Bagaimana penghematan energi yang dapat diperoleh dari implementasi teknik skala tegangan dan frekuensi pada sistem komputer paralel.

1.4 Batasan Masalah

Untuk memperjelas ruang lingkup penelitian, maka pembatasan masalah diberikan dalam penelitian ini adalah sebagai berikut.

1. Pengujian komputasi paralel menggunakan rutin program dengan bahasa *Java*.
2. Beban komputasi mengarah pada pembebanan *central procesing unit* (CPU).
3. Pengujian dilakukan menggunakan perangkat yang akan dirancang.
4. Perancangan, pengujian dan pengambilan data dilakukan di Laboratorium Informatika dan Komputer Jurusan Teknik Elektro, Universitas Brawijaya.
5. Kemampuan alat terbatas oleh rekomendasi pabrik sesuai dengan *datasheet*.

1.5 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut.

1. Untuk mengetahui faktor yang mempengaruhi penggunaan daya pada sistem komputer paralel.
2. Untuk mengetahui pengaruh skala tegangan dan frekuensi dinamis pada sistem komputer paralel.
3. Untuk mengetahui penghematan penggunaan energi yang diperoleh dari implementasi teknik skala tegangan dan frekuensi pada sistem komputer paralel.

1.6 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah sebagai berikut.

a. Bagi penyusun antara lain:
    1. Membangun sistem komputer paralel yang terjangkau dan hemat daya.
    2. Menerapkan ilmu yang telah diperoleh dari Konsentrasi Rekayasa Komputer Teknik Elektro.
    3. Menambah wawasan mengenai *High Parallel Computing*.
b. Bagi kalangan akademis antara lain:
    1. Memanfaatkan sistem komputer paralel secara efisien.
    2. Meningkatkan utilisasi sumber daya komputer.
    3. Sebagai referensi yang dapat digunakan dalam penelitian di bidang Komputer Paralel yang berkaitan dengan penghematan daya.

1.7 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan laporan penelitan ini sebagai berikut:

BAB I Pendahuluan

Membahas latar belakang, rumusan masalah, ruang lingkup, tujuan dan sistematika penulisan.

BAB II Tinjauan Pustaka

Membahas kajian pustaka dan dasar teori yang digunakan sebagai landasan teori dalam penelitian yang dilakukan.

BAB III Metode Penelitian

Menjelaskan tentang metode yang digunakan dalam penelitian dalam tahapan penyelesaian skripsi yang meliputi studi literatur, konsep perancangan dan pembuatan sistem, pengumpulan data, analisi, serta pengambilan kesimpulan.

BAB IV Perancangan dan Implementasi

Membahas tentang perincian perancangan dan pengerjaan sisitem komputer paralel untuk pengujian implementasi teknik yang digunakan dalam penelitian.

BAB V Pengujian dan Analisis

Membahas tentang pengujian sistem dan analisis pengujian sistem serta hasil pengujian pada sistem komputer paralel dengan konfigurasi yang diimplementasikan.

BAB VI Penutup

Memuat kesimpulan berdasarkan hasil yang telah didapatkan dan saran sehingga penelitin dapat dikembangkan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

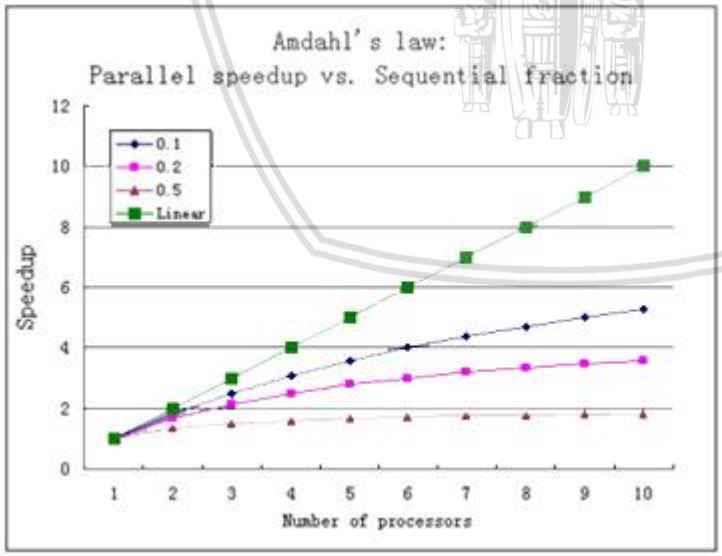
2.1 Komputasi Paralel

Komputasi paralel adalah eksekusi proses pengoperasian aritmatik atau logik yang sejenis secara bersamaan. (Reif, 2007). Sebuah masalah komputasi yang kompleks dibagi menjadi bagian-bagian yang lebih kecil, sederhana, dan dijalankan di beberapa prosesor secara bersamaan sehingga keseluruhan proses dapat diselesaikan dengan lebih cepat.

Tujuan utama komputasi paralel adalah peningkatan kecepatan komputasi. Hukum Amdahl (Gene Amdahl, 1967) dan Gustafon (John Gustafon, 1988) menjelaskan peningkatan kecepatan komputasi paralel. Hukum Amdahl menjelaskan kecepatan komputasi prosesor dalam Persamaan (2-1).

$$S = \frac{1}{\frac{1-\alpha}{P}+\alpha} \quad \text{..... (2-1)}$$

Kecepatan komputasi S dari beberapa prosesor dijabarkan, di mana α adalah waktu yang dibutuhkan untuk menyelesaikan rutin komputasi dengan 1 prosesor, P adalah cacah prosesor.



Gambar 2.1 Hukum Amdahl
Sumber: Reif John (2007)

Ruby Lee (1980) mendefinisikan beberapa parameter untuk mengevaluasi komputasi paralel. Beberapa parameter tersebut adalah konsep fundamental dalam

parallel processing. Perbandingan diantara faktor kinerja sering ditemui dalam aplikasi dunia nyata.

Efisiensi sistem untuk sebuah sistem n-prosesor didefinisikan dalam persamaan berikut:

$$\mathrm{E(n)} = \frac{S(n)}{n} = \frac{T(1)}{nT(n)} \quad \text{(2-2)}$$

dimana S(n), merupakan *speedup factor* dengan T(n) berupa lama eksekusi komputasi dalam satuan waktu. Efisiensi maksimum akan diperoleh ketika semua n prosesor dimanfaatkan sepenuhnya sepanjang periode eksekusi komputasi.

2.2 Hadoop Cluster

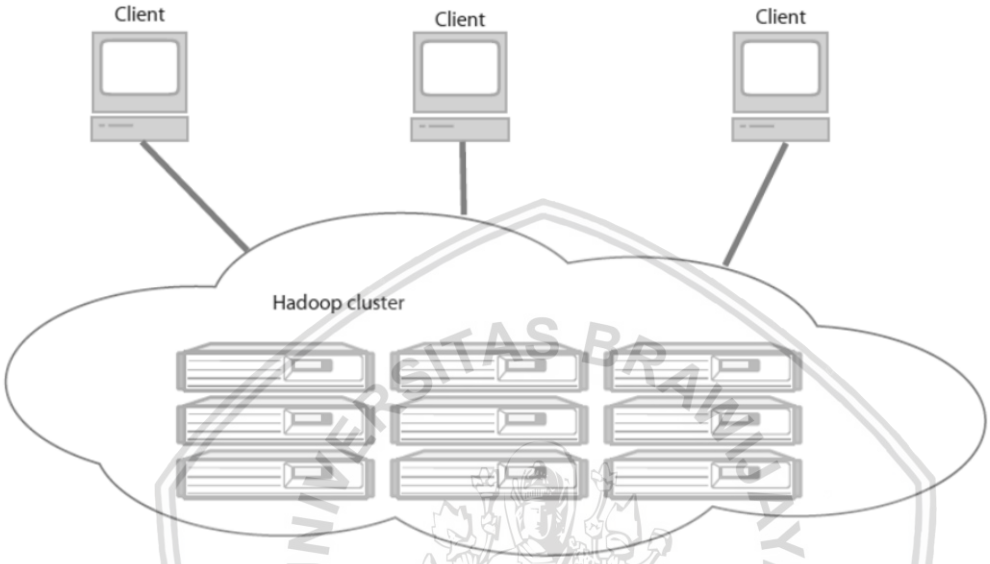
Hadoop cluster merupakan klaster yang didesain khusus dikonfigurasikan untuk menyimpan dan menganalisis data tidak berstruktur pada lingkup komputasi terdistribusi dalam jumlah yang besar secara spesifik. Konfigurasi tersebut menjalankan perangkat lunak komputasi terdistribusi *open source* pada komputer komoditas dan memiliki keuntungan dari stuktur dan metode implementasinya, antara lain: skalabilitas, konvergensi, fleksibilitas konfigurasi, *failover,* kemudahan, kecepatan, dan harga implementasi yang murah dibandingkan komputer konvensional.

Hadoop cluster dikenal akan kecepatannya dalam aplikasi analisis data. Sistem ini mendukung skalabilitas yang tinggi. Jika kemampuan proses sebuah klaster dibanjiri oleh volume data yang terus meningkat, penambahan klaster baru dapat meningkatkan *throughput*. *Hadoop cluster* memiliki toleransi kerusakan yang tinggi terhadap *failure* karena setiap data yang diproses pada setiap klaster dilakukan replikasi, yang mana bila ada data pada satu node mengalami kerusakan tidak akan mempengaruhi data pada node lainnya.

Hadoop mengimplementasikan paradigma komputasi *MapReduce*, di mana kerja dibagi menjadi banyak fragmen kecil dan dieksekusi pada node manapun dalam klaster. *Hadoop* juga menyediakan sistem berkas terdistribusi (HDFS) yang mampu menyimpan data pada node-node komputer, menyediakan *aggregate bandwidth* yang sangat tinggi di seluruh klaster. Pada *MapReduce* dan HDFS, apabila terjadi kerusakan pada node tertentu, maka *framework* akan secara otomatis menangani dengan bantuan cadangan dari node lainnya.

Dalam *hadoop cluster*, satu mesin dalam klaster ditunjuk sebagai *NameNode* dan mesin lainnya ditunjuk sebagai *JobTracker*. *JobTracker* bekerja sebagai pengatur rutin

tugas paralel yang akan dikerjakan dan *NameNode* mencatat perubahan *filesystem metadata*. Mesin-mesin tersebut adalah *Masters*. *TaskTracker* bekerja sebagai pekerja rutin tugas yang akan dibagikan. *DataNode* menyimpan blok-blok data dan replikasi data HDFS. Mesin lain dalam *cluster* yang tidak ditunjuk berperan sebagai *DataNode* dan *TaskTracker*. Mesin-mesin tersebut menjadi *Slaves*. Seluruh kumpulan node tersebut bekerja menjadi satu sistem *cloud* seperti Gambar 2.2.

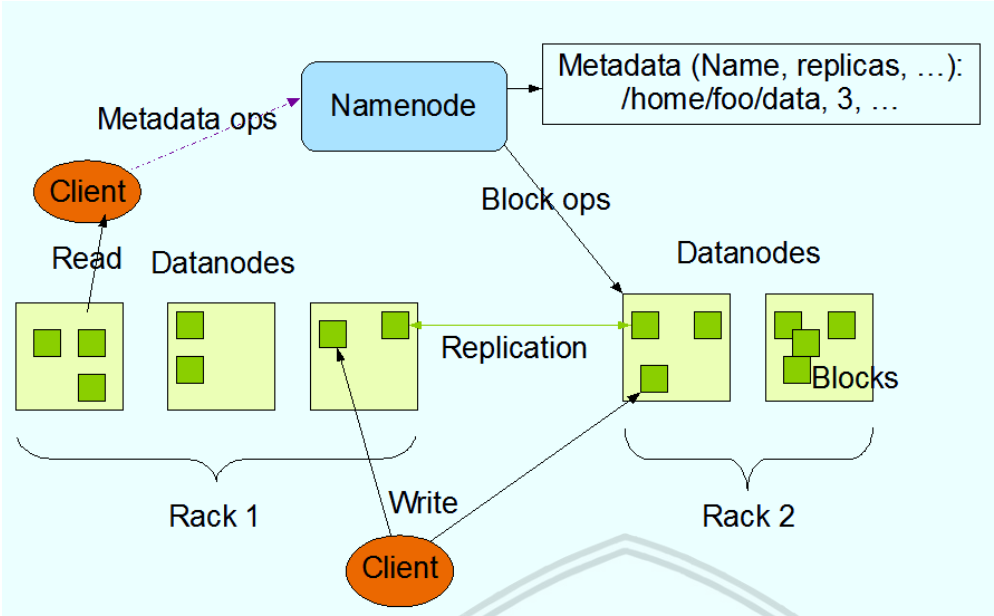


Gambar 2.2 Sebuah klaster Hadoop memiliki banyak mesin paralel yang menyimpan dan memproses kumpulan data yang besar. Komputer klien mengirimkan pekerjaan ke dalam *cloud.*

2.3 Arsitektur HDFS

Hadoop Distributed File System atau HDFS adalah sistem berkas terdistribusi yang didesain untuk berjalan pada perangkat komoditas. HDFS memiliki banyak kesamaan dengan sistem berkas terdistribusi lainnya. Yang menjadikan HDFS berbeda dengan yang lainnya adalah HDFS bersifat *highly fault-tolerance* dan didesain untuk disematkan dalam perangkat yang terjangkau. Desain HDFS sesuai untuk aplikasi yang menggunakan data besar.

HDFS memiliki arsitektur *master/slave* seperti pada Gambar 2.3. HDFS terdiri dari sebuah *NameNode* pada node *master* yang mengelola *filesystem metadata* dan regulasi akses berkas oleh klien dan ada bagian yang berupa *DataNodes* pada node *slave*.

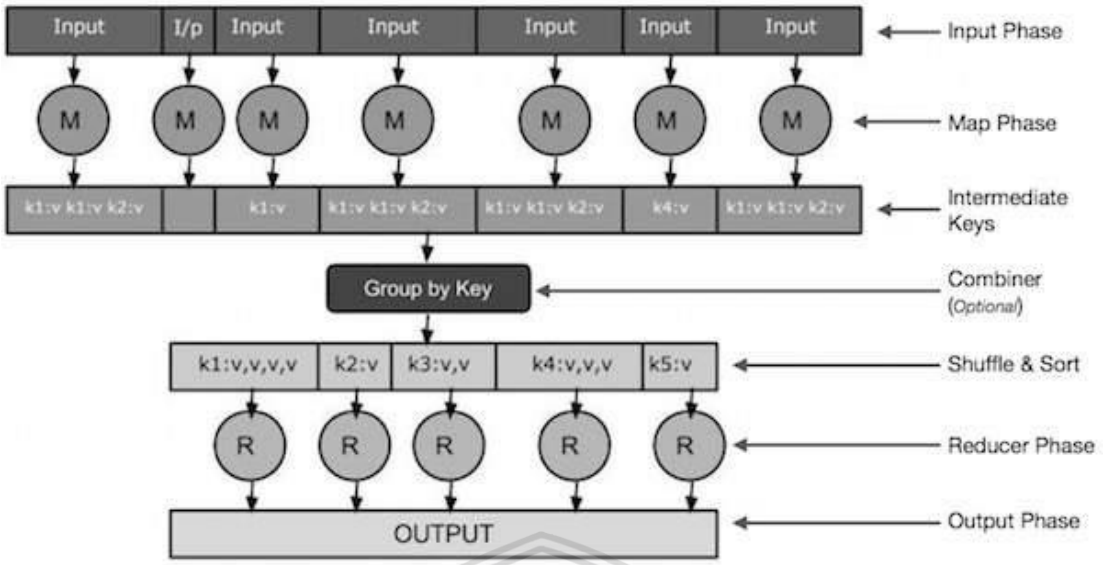


Gambar 2.3 Arsitektur HDFS

HDFS merupakan perangkat berbasis *Java*. Semua mesin yang mendukung *Java* dapat menjalankan *NameNode* atau *DataNode*. HDFS mendukung pengelolaan *traditional hierarchical file*. Seorang user atau aplikasi dapat membuat direktori maupun menyimpan berkas ke dalam direktori. Selain itu HDFS didesain secara reliabel untuk menyimpan berkas yang cukup besar dalam sebuah klaster. Blok dari sebuah berkas direplikasi untuk toleransi kerusakan. Untuk memantau ada atau tidaknya kerusakan pada berkas, *NameNode* secara periodik menerima *Heartbeat* dan *Blockreport* dari seluruh node dalam klaster. Seluruh protokol komunikasi HDFS berada pada layer TCP/IP.

2.4 MapReduce

MapReduce, diperkenalkan oleh Google pada tahun 2004, merupakan model pemrograman yang sekarang telah dikembangkan oleh Apache Hadoop, terdiri atas pembagian blok data yang besar dan fase '*Map*' & '*Reduce*'. *Map task* adalah mengambil sebuah set dari data dan mengubahnya menjadi set data lain, di mana elemen-elemen individu dipecah menjadi tupel (*key-value pairs*). *Reduce task* adalah mengambil keluaran dari *Map* sebagai masukan dan menggabungkan data tupel menjadi set tupel yang lebih kecil. Fase cara kerja *MapReduce* dapat dilihat pada Gambar 2.4.

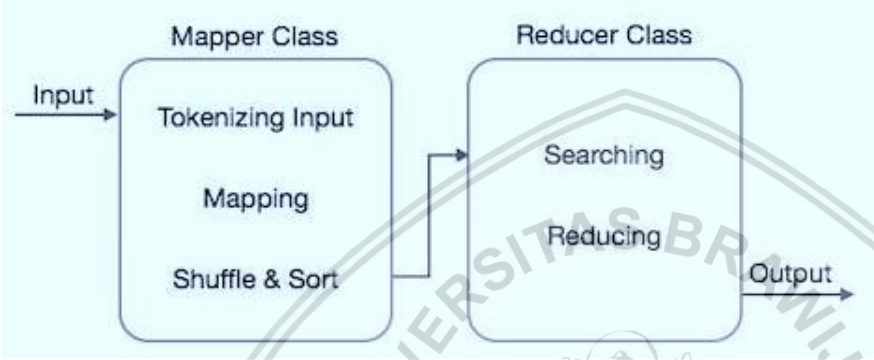


Gambar 2.4 Fase cara kerja dari *MapReduce*

- Input Phase − Terdapat *record reader* yang menerjemahkan tiap catatan dari berkas masukan dan mengirim data terparsir ke *mapper* dalam bentuk tupel.
- *Map* – Fungsi yang ditentukan pengguna, dengan mengambil serangkaian tupel dan memproses masing-masing untuk menghasilkan tupel bernilai nol atau lebih.
- *Intermediate Keys* – Tupel yang dihasilkan oleh *mapper* dikenal sebagai *intermediate keys*
- *Combiner* – Bekerja seperti *Reducer* lokal yang mengelompokkan data yang serupa dari fase *map* menjadi set yang dapat diidentifikasi. Lalu mengambil *intermediate keys* dari *mapper* sebagai input dan menerapkan kode yang ditentukan pengguna untuk mengumpulkan nilai dalam suatu lingkup kecil pada satu *mapper*. Fase ini opsional.
- *Shuffle and Sort* – Pada langkah ini, tupel yang sudah dikelompokkan dikirimkan ke *local machine* yang sedang menjalankan *Reduce*. Tupel individu diurutkan menjadi *data list* yang lebih besar. *Data list* mengelompokkan kunci yang setara agar nilainya dapat diiterasikan secara mudah dalam *Reduce task*.
- *Reducer* – *Reducer* mengambil tupel yang telah dikelompokkan sebagai masukan dan menjalankan fungsi *Reducer* pada masing-masing data. Pada fase ini data dapat diaggregasi, difilter, dan dikombinasikan dengan berbagai cara, dan membutuhkan pengolahan yang berbagai macam. Sampai eksekusi telah berakhir, tupel akan diberikan nilai nol atau lebih untuk tahap akhir.

- *Output Phase* – Fase ini memiliki *output formatter* yang berfungsi menerjemahkan tupel akhir dari fungsi *Reducer* dan menuliskannya pada berkas menggunakan *record writer*.

Diagram blok kerja *MapReduce* dapat dilihat pada Gambar 2.5. Algoritma *MapReduce* memiliki dua *task* penting yaitu *Map* dan *Reduce*. *Map task* dilakukan oleh fungsi *Mapper Class*, dan *Reduce task* dilakukan oleh fungsi *Reducer Class*. *MapReduce* mengimplementasikan berbagai algoritma matematik untuk membagi *task* menjadi bagian-bagian kecil dan mengarahkannya pada beberapa sistem.

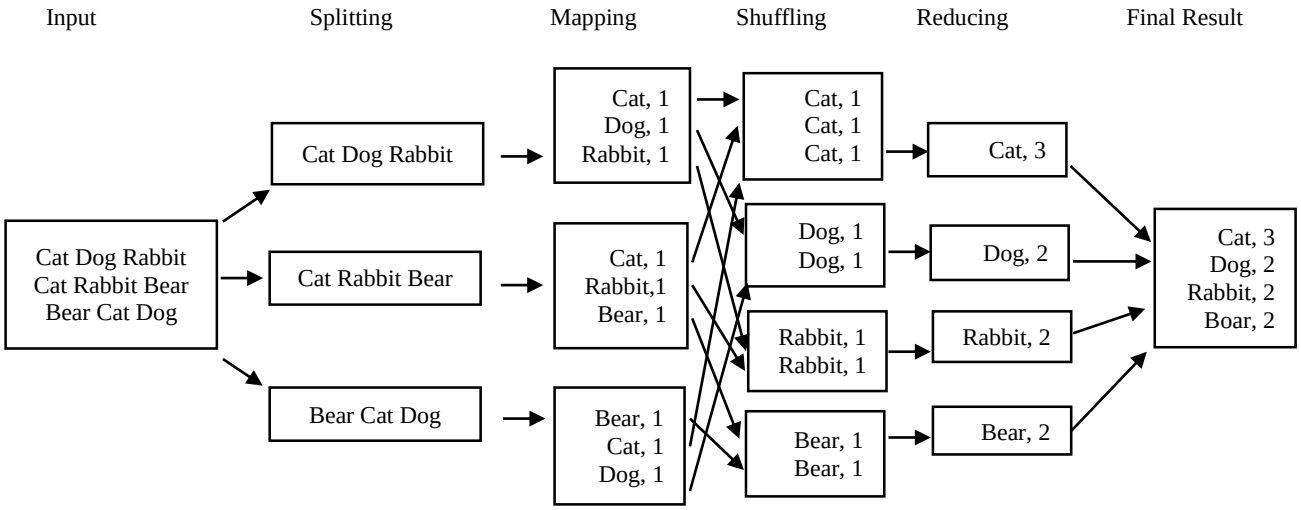


Gambar 2.5 Diagram blok kerja dari *MapReduce*
Sumber: http://www.tutorialspoint.com/map_reduce/map_reduce_algorithm.htm, diakses pada tanggal 9/11/2017)

2.5 WordCount

WordCount merupakan rutin program yang bertujuan untuk menghitung banyaknya cacah kata dalam suatu berkas teks. Keluaran dari rutin program berupa berkas teks, dengan tiap baris berupa sebuah kata dan cacah kata tersebut muncul dalam teks. *WordCount* termasuk rutin program *indexing*. Jumlah cacah kata menjadi beban perhitungan yang diproses oleh *WordCount*.

Pada Gambar 2.6, setiap *mapper* mengambil baris sebagai masukan dan memecahnya menjadi banyak kata. Kemudian tiap kata diberikan *key/value pair* dan angka 1 (untuk penjumlahan pada saat *Reduce*). Tiap reducer menjumlahkan untuk masing-masing kata dan diberikan *single key/value* dengan kata dan jumlah kata.

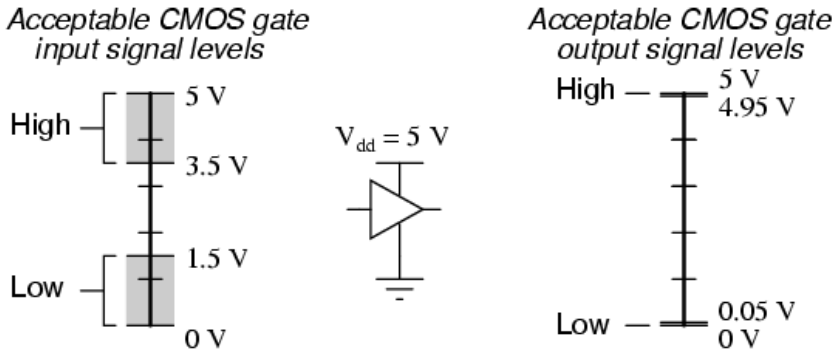


Gambar 2.6 Diagram blok kerja dari *WordCount* dengan *MapReduce*

2.6 Skala Tegangan dan Frekuensi Dinamis (DVFS)

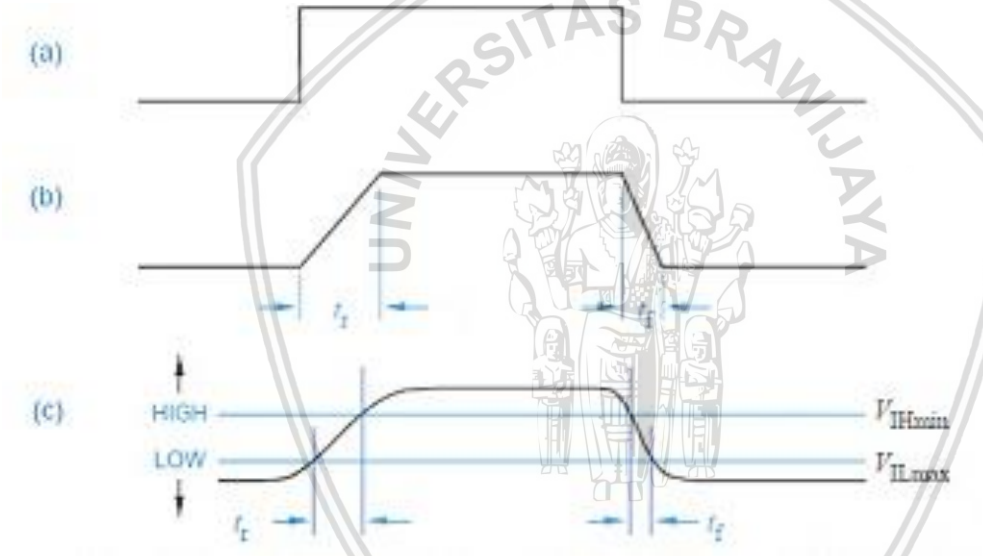
Skala tegangan dan frekuensi atau *Dynamic Voltage and Frequency Scaling* (DVFS) adalah sebuah teknik untuk mengatur tegangan dan kecepatan prosesor pada suatu perangkat komputasi, chip kontroler, dan perangkat periferal lainnya. DVFS bertujuan untuk mengoptimalkan pembagian jatah alokasi sumberdaya untuk rutin-rutin kerja dan memaksimalkan penghematan daya ketika sumberdaya tidak sedang terpakai. DVFS dapat dimanfaatkan sebagai peningkat kinerja komputasi, akan tetapi penggunaan daya akan menjadi lebih tinggi.

Di dalam teknik DVFS, tegangan yang digunakan pada prosesor ditambah atau dikurangi sesuai dengan kebutuhan. Peningkatan tegangan disebut sebagai *overvolting* dan untuk penurunan tegangan disebut *undervolting*. *Overvolting* dilakukan untuk meningkatkan kinerja komputer, sebaliknya untuk *undervolting* dimanfaatkan untuk melakukan konservasi daya.



Gambar 2.7 Skala Tegangan untuk State Gerbang Logika pada CMOS

Perubahan tegangan yang terjadi pada rangkaian digital akan mempengaruhi representasi dari *logical state*. Jika tegangan berada dalam keadaaan *high* maka akan merepresentasikan logika benaran 1, dan sebaliknya jika berada dalam keadaan *low* maka nilai logika benaran berupa 0. Pada rangkaian elektronik digital, logika benaran 1 maupun 0 memiliki batas nilai tegangan tersendiri. Salah satu contohnya pada rangkaian elektronik digital yang menggunakan CMOS sebagai gerbang logika, memiliki operasi yang berbeda untuk nilai penerimaan input dan output tegangan, seperti pada Gambar 2.7. Pada penerimaan nilai input, untuk "*high*" *logic state* dibaca dari batas 3.5V hingga 5V dan untuk "*low*" *logic state* dibaca dari batas 0V hingga 1.5 *volt*. Nilai output untuk "*high*" *logic state* dibaca dari batas 4.95V hingga 5V dan untuk "*low*" *logic state* dibaca dari batas 0V hingga 0.05V. Kecepatan sebuah rangkaian digital ditentukan dari kemampuan perubahan *state*, dari "low" ke "high" dan sebaliknya.



Gambar 2.8 Waktu transisi, (a) kondisi ideal *zero time switching* (b) pendekatan realistis (c) waktu aktual yang menunjukkan wakti *Rise* dan *Fall*

Perubahan logika dari *high* menjadi *low* dan sebaliknya, membutuhkan daya yang disebut juga sebagai *switching power*. *Switching power* berasal dari pengisian dan pelepasan keluaran dari kapasitansi. Besar konsumsi daya dapat dimodelkan secara matematis pada Persamaan (2-5) dan (2-6**)**.

$$P_{dyn} = \alpha C V^2 F \qquad (2\text{-}5)$$

$$P_{leak} = I_{leak} V \qquad (2\text{-}6)$$

dengan:

P_{dyn} : daya dinamis

P_{leak} : daya bocor

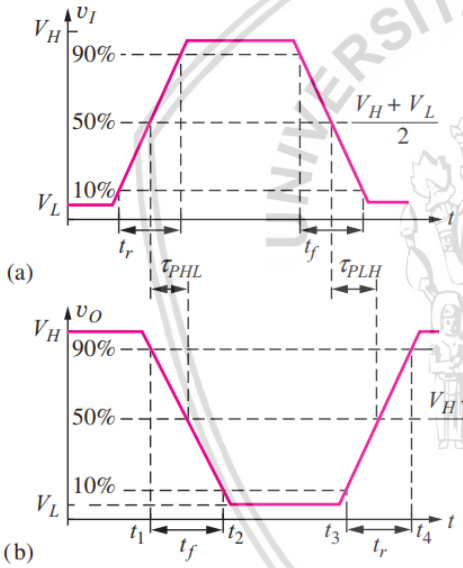
α : aktifitas *switching*

C : kapasitansi berubah tiap siklus *clock*

V : suplai tegangan

F : frekuensi *switching*

Konsumsi daya dinamis berasal dari adanya pengisian dan pelepasan kapasitansi beban dan arus hubung singkat. Daya bocor muncul karena adanya arus bocor yang mengalir meskipun perangkat berada dalam kondisi tidak aktif. Teknik DVFS bekerja dengan mengurangi daya dinamis dan daya bocor yang bertujuan untuk perubahan sistem hemat daya. Jika tingkat frekuensi yang berjalan terbatas, maka akan berdampak pada tingkat jumlah instruksi program yang dapat berjalan seiring kecepatan *clock* yang dapat berjalan. Hal tersebut akan mengakibatkan segmen program yang sedang berjalan memakan waktu yang lebih lama. Respon gerbang logika dapat dilihat pada Gambar 2.9.



Gambar 2.9 Respon Dinamis Gerbang Logika (a) *input voltage*, (b) *output voltage*

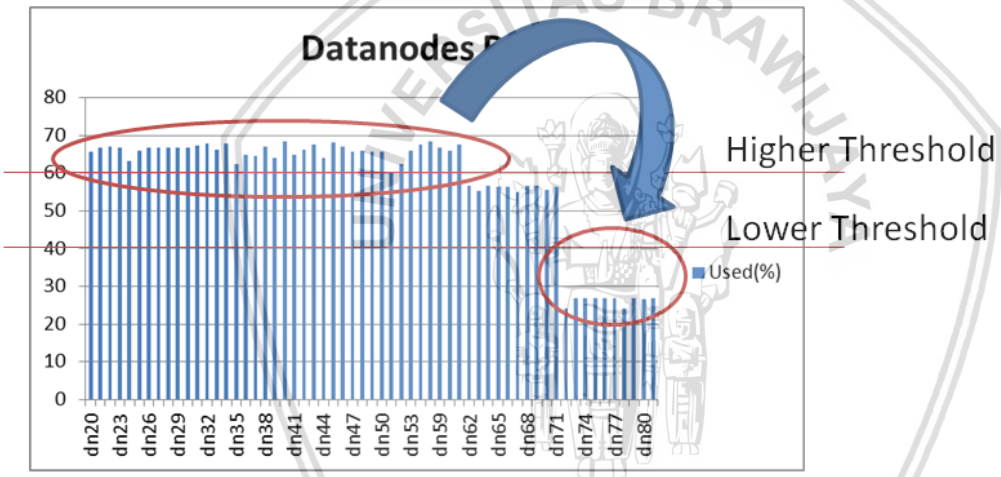
DVFS merupakan teknik untuk mengubah tegangan dan/atau frekuensi dari sistem komputasi berdasarkan kinerja dan kebutuhan daya. DVFS dapat digunakan untuk mencegah sistem komputer mengalami *overheating*, yang menyebabkan kerusakan dalam program maupun perangkat keras. Akan tetapi jika suplai tegangan diturunkan hingga melampaui batas aturan pabrik dapat menyebabkan ketidakstabilan pada perangkat.

2.7 Load Balancing

Load Balancing atau penyeimbangan beban merupakan salah satu teknik untuk meningkatkan kinerja suatu sistem paralel dalam komputasi dengan beban berskala besar. *Load Balancing* bertujuan untuk menyetarakan beban kerja untuk setiap node. Jika beban

kerja yang dikerjakan pada setiap node paralel adalah sama, maka beban komputasi untuk setiap node akan mendekati sama. Jika tidak ada perbedaan yang jauh antara beban tiap-tiap node, maka nilai efisiensi akan mendekati titik optimal, baik penggunaan prosesor maupun sumberdaya lainnya. Selain itu beban kerja dari tiap-tiap node perkerja dalam sebuah sistem paralel, komputasi paralel dapat diselesaikan lebih cepat dan perbandingan waktu penyelesaian komputasi dari tiap node menjadi kecil. Dengan demikian akan didapatkan peningkatan kinerja dan penghematan sumberdaya sistem paralel.

Sebuah klaster dianggap terbeban seimbang jika untuk setiap data node, perbandingan rasio ruang yang digunakan terhadap total kapasitas node berbeda dengan perbandingan rasio ruang yang digunakan terhadap total kapasitas klaster, dengan tidak lebih dari nilai ambang batas. Gambar 2.10 menunjukkan penyeimbangan yang terjadi jika perbandigan rasio melebihi nilai ambang batas.



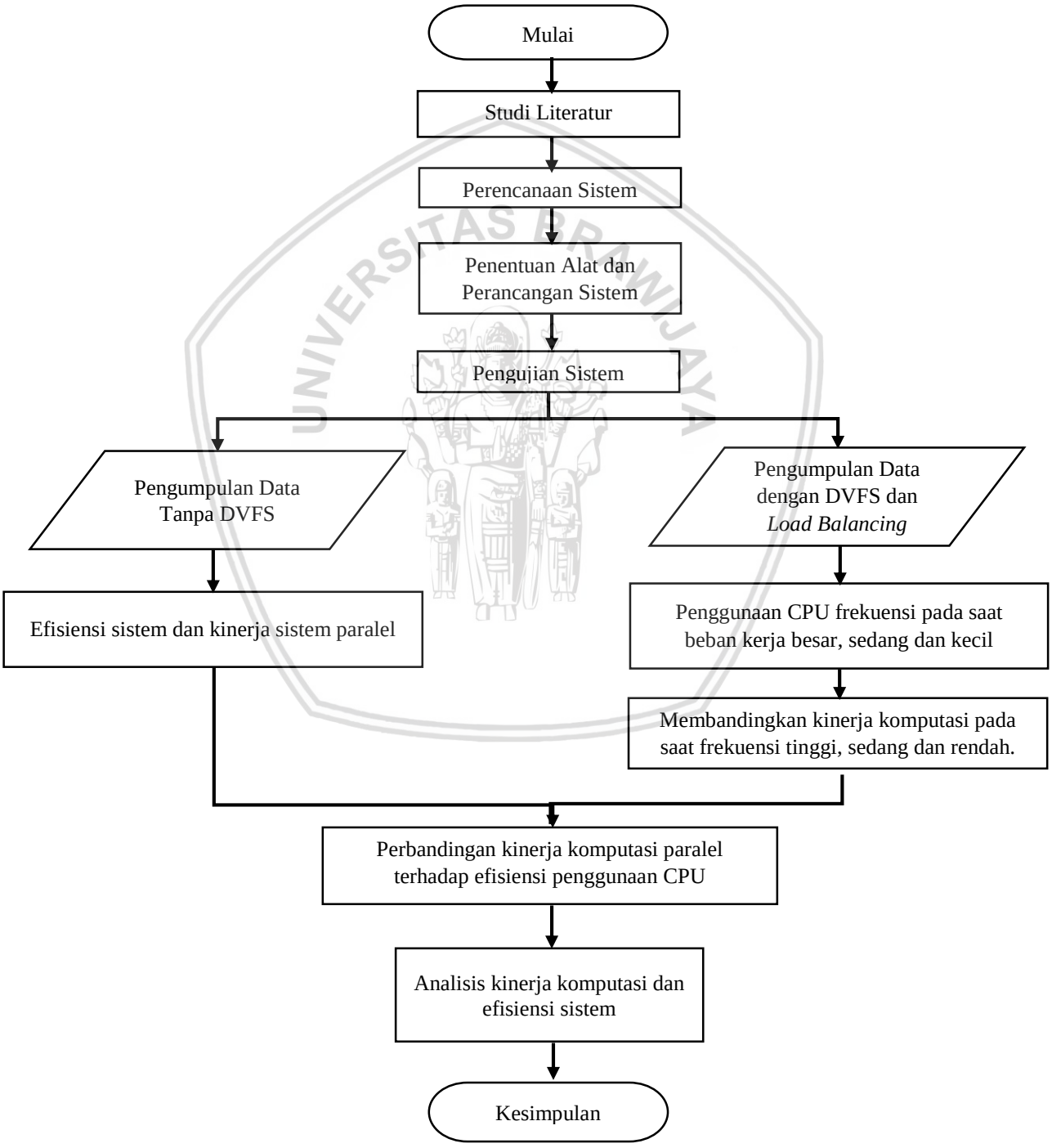
Gambar 2.10 Ilustrasi *load balancing* yang dilakukan jika terjadi perbedaan beban kerja melebihi batas prosentase *threshold*.

2.8 Orange Pi

Orange Pi merupakan sebuah mini PC berbasis *open source single board computer* dengan spesifikasi yang cukup untuk menjalankan fungsinya sebagai komputer. Orange Pi memiliki dimensi yang lebih ringkas jika dibandingkan dengan komputer pada umumnya dan menggunakan daya listrik yang jauh lebih rendah. Orange Pi menggunakan prosesor berbasi ARM, yang menjadikan perangkat ini sebagai perangkat yang hemat energi. Sistem operasi yang dapat disisipkan ke dalam media penyimpanan MMC menjadikannya cukup praktis dan fleksibel. Orange Pi memiliki *port ethernet* yang dapat dimanfaatkan sebagai media komunikasi data antar perangkat pada jaringan.

BAB III
METODE PENELITIAN

Dalam penyusunan skripsi ini, akan dirancang sistem komputer paralel hemat daya menggunakan teknik Skala Tegangan dan Frekuensi Dinamis. Penelitian dilakukan dengan memperhatikan karakteristik dari sistem.



Gambar 3.1 Diagram alir tahapan penelitian.

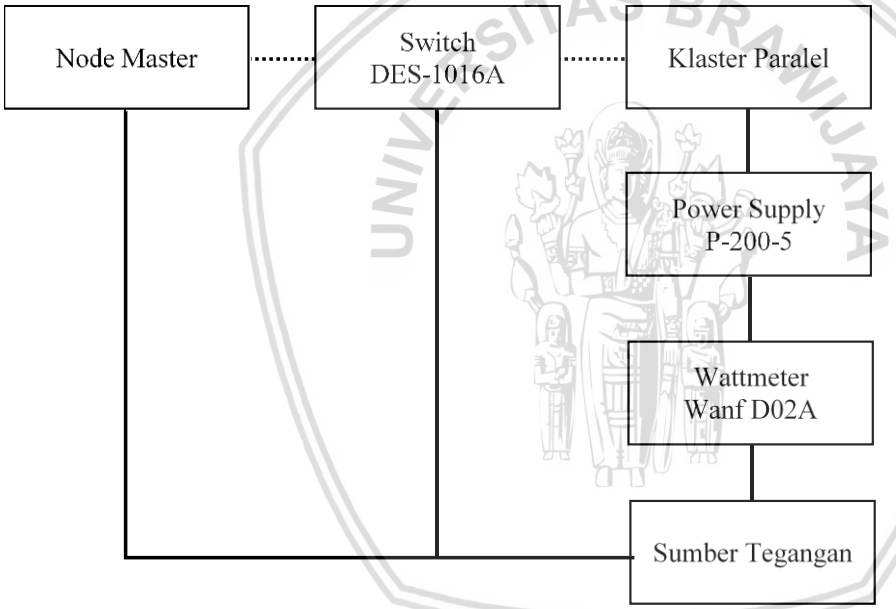
3.1 Studi Literatur

Studi literatur dilakukan dengan mengkaji berbagai sumber pustaka yang berhubungan dengan judul penelitian. Teori-teori yang berhubungan dengan judul penelitian ini meliputi:

1. Mempelajari hal-hal yang berkaitan dengan komputer paralel dan komputasinya.
2. Mempelajari arsitektur *Hadoop cluster* dan algoritma *MapReduce*.
3. Mempelajari teknik *Load Balancing* dan *Dynamic Voltage and Frequency Scaling*.

3.2 Perancangan dan Penentuan Spesifikasi Alat

Berdasarkan studi literatur dan tinjauan pustaka untuk menghasilkan sistem komputer paralel hemat daya dapat direncanakan dalam infrakstruktur sebagai berikut:

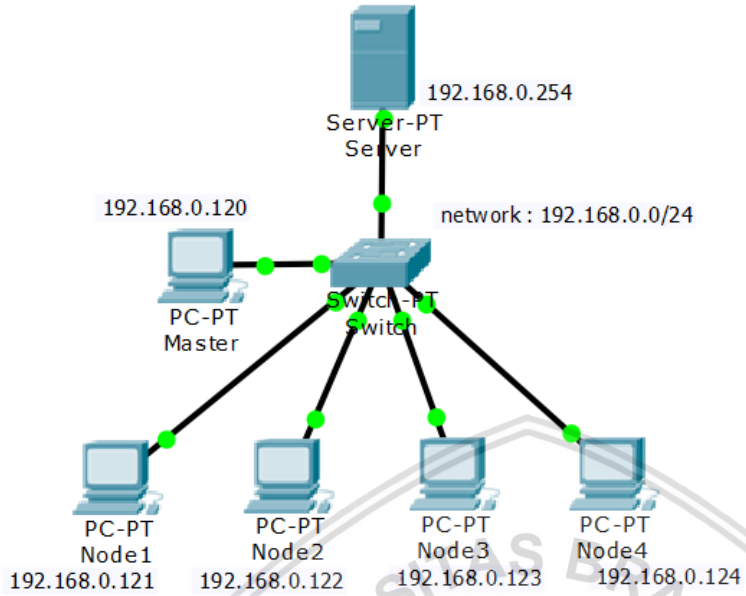


Gambar 3.2 Pemodelan Sistem.

Pada Gambar 3.2 terdapat node yang berperan sebagai *master* dan *slave* yang berada di dalam klaster paralel. *Master* (*JobTracker*) bekerja sebagai pengatur rutin tugas paralel yang akan dikerjakan dan *Slave* (*TaskTracker*) bekerja sebagai pekerja rutin tugas yang akan dibagikan. *Master* (*NameNode*) mencatat perubahan *filesystem metadata* dan *Slave* (*DataNode*) menyimpan blok-blok data dan replikasi data HDFS.

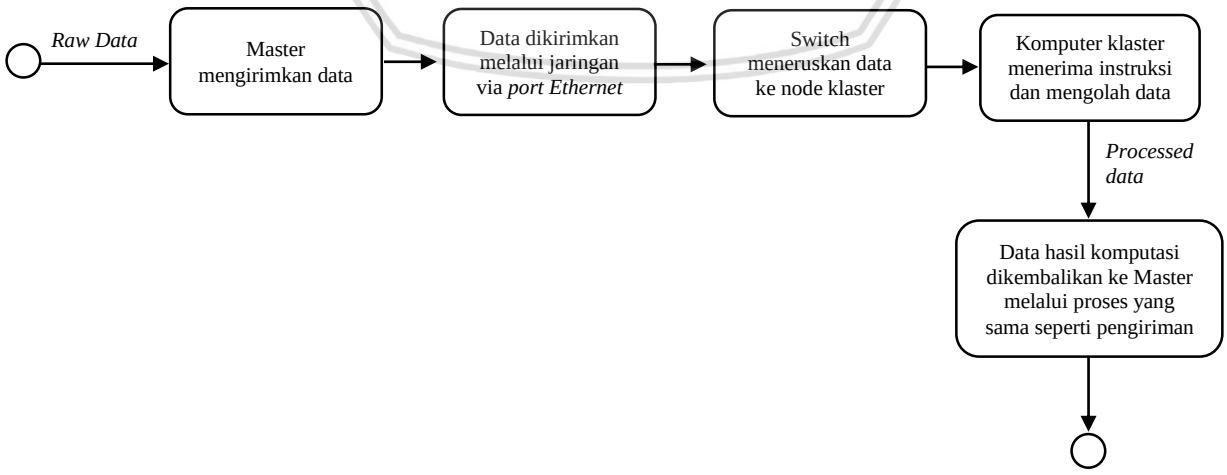
Perancangan sistem yang akan dikerjakan pada jaringan area lokal dapat dilihat pada Gambar 3.3. Protocol TCP/IP digunakan sebagai media komunikasi antar perangkat. Masing-masing perangkat terhubung melalui *port Ethernet* menuju satu *switch*

penghubung dengan alamat IP yang diatur secara *static*. Server berfungsi sebagai *internet backbone* serta sebagai DHCP server.



Gambar 3.3 Pemodelan jaringan lokal sistem

Dari perancangan sistem tersebut, adapun aliran data yang dapat dilakukan dalam penelitian yang dijelaskan pada Gambar 3.4**.** Data yang belum diolah akan dikirmkan oleh klien menuju node master. Kemudian master akan mengirimkan data yang perlu diolah kepada klaster paralel. Klaster paralel menyimpan data yang akan diolah di HDFS. Master mengirimkan set instruksi dari rutin program paralel kepada seluruh node untuk melakukan komputasi menggunakan jaringan lokal via *Ethernet*. Data yang telah diolah, akan disimpan ke HDFS dan diambil oleh master untuk diberikan pada klien.



Gambar 3.4 Aliran data pada sistem

Berdasarkan pemodelan dan perancangan sistem, maka diperlukan beberapa perangkat yang dapat mendukung sistem komputer paralel. Perangkat yang akan digunakan sebagai pengujian pada sistem komputer paralel adalah sebagai berikut:

a. Perangkat Keras

Adapun perangkat keras yang akan digunakan pada perancangan sistem sebagai berikut:

1. 5 unit Orange Pi PC (CPU H3 Quad-core Cortex-A7, DDR3 1 GB), dengan satu unit sebagai *master* dan empat unit sebagai *slave*.
2. 5 unit MicroSD card SanDisk 8 GB, berfungsi sebagai media penyimpanan.
3. 1 Switch 16 Port TP-LINK, berfungsi sebagai penghubung routing jaringan.
4. Kabel UTP Cat 5E, berfungsi sebagai media transmisi komunikasi pada jaringan.
5. Power Supply P-200-5, sebagai pengatur sumber daya pada sistem paralel klaster.
6. Wattmeter Wanf D02A, sebagai pengukur energi terpakai pada sistem paralel klaster.

b. Perangkat Lunak

Adapun perangkat lunak pendukung sistem komputer paralel yang akan dirancang sesuai dengan perumusan masalah sebagi berikut:

1. GNU/Linux Debian 8.1 (jessie) : Kernel 3.4.6
2. Java 8.
3. Javac 1.8
4. OpenSSH
5. Apache Hadoop 2.7.4
6. CPUFreq
7. Cpustat

Konfigurasi dilakukan pada setiap perangkat yang digunakan. Konfigurasi pada *master-sla*ve diuraikan sebagai berikut:

1. Pemasangan Sistem Operasi GNU/Linux Debian Jessie pada satu komputer master.
2. Pemasangan *library* dan *compiler* pada komputer master.
3. Pemasangan modul OpenSSH pada komputer master.

4. Pemasangan Java pada komputer master.
5. Pemasangaan Hadoop pada komputer master.
6. Pemasangan hasil kloning dari sistem operasi master sebagai sistem operasi yang dijalankan oleh komputer slave.
7. Pengaturan alamat IP secara statik pada setiap komputer.
8. Penyambungan seluruh node via *port Ethernet* dengan kabel UTP ke switch dalam satu jaringan yang terhubung dengan server.
9. Pengaturan kunci SSH pada master dan slave, untuk memudahkan akses secara *remote* pada setiap slave.
10. Pengaturan Hadoop pada setiap node yang akan dipekerjakan sebagai slave.
11. Pengaturan Skala Tegangan dan Frekuensi Dinamis pada setiap node yang digunakan.

3.3 Abstraksi Sistem

Sistem komputer paralel yang digunakan adalah konfigurasi komputer paralel dengan arsitektur *Hadoop cluster*. Dipilih *hadoop cluster* karena hadoop mampu melakukan analisis dan pengolahan dataset yang berukuran besar dalam permasalahan *Big Data*.

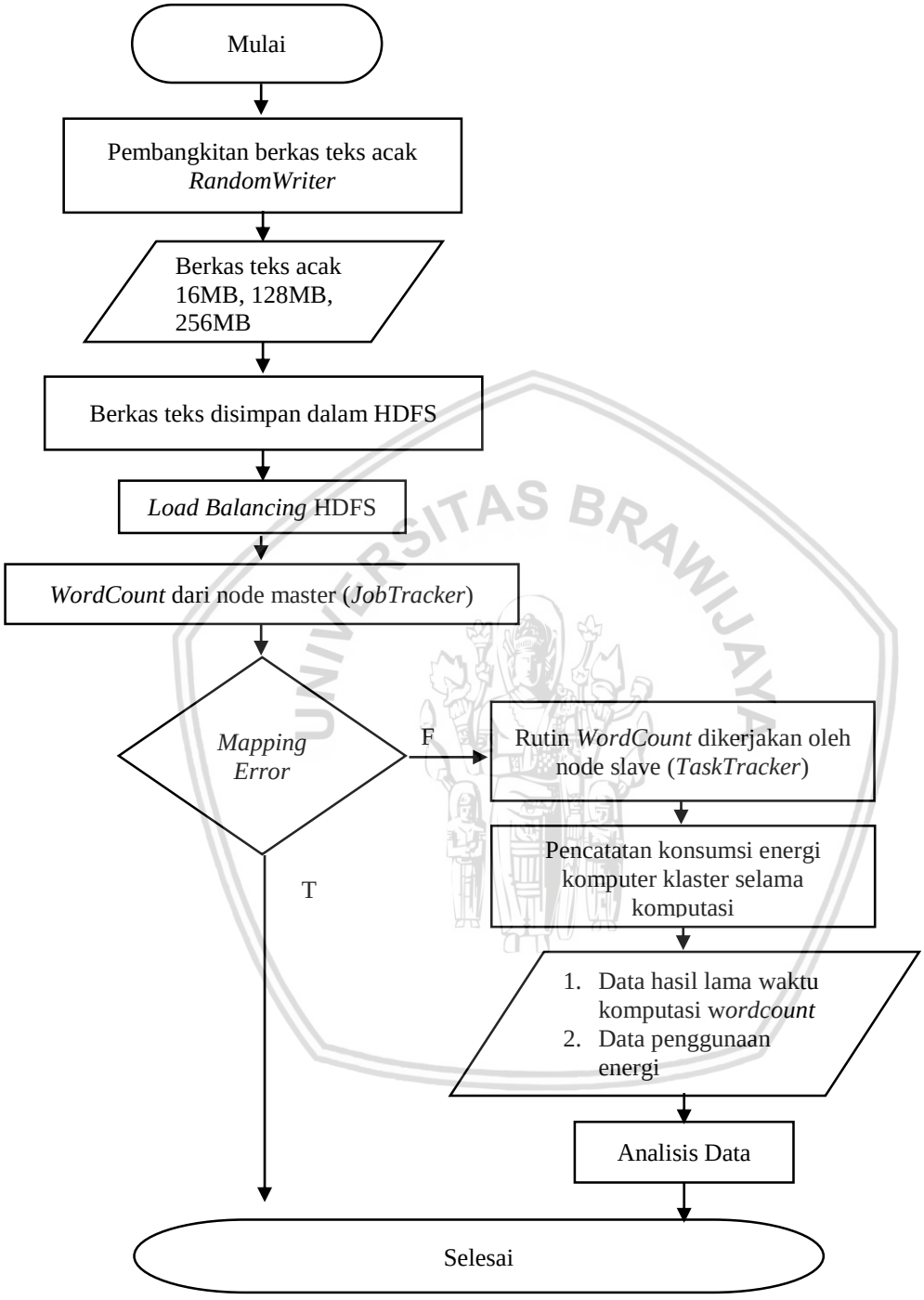
Pada sistem ini, akan dimanfaatkan *Balancer* sebagai *Load Balancing* pada HDFS dan Teknik *Dynamic Voltage and Frequency Scaling* pada komputer klaster sebagai pengatur penggunaan daya yang terpakai pada sistem komputer paralel. Kemudian penggunaan energi terpakai akan dicatat dan dibandingkan untuk tiap faktor yang mempengaruhinya. Pengambilan data akan dilakukan dengan membandingkan penggunaan proses energi terhadap skala data yang akan diolah pada klaster paralel.

Data yang akan diuji berupa dataset yang dibangkitkan dengan ukuran 256MB, 128MB, 16MB. Angka tersebut didapatkan melalui percobaan stress test pada mini pc terhadap alokasi memori, dengan test yang meliputi beban dari 4 kali ukuran memori hingga ¼ kali ukuran memori. Serta didapati ukuran maksimum *data block size* yang terdistribusi adalah sebesar 128MB. Ukuran data yang terlalu besar mengakibatkan lama pembacaan serta melebihi kemampuan *resource* pada perangkat yang digunakan.

Dengan menggunakan acuan penelitian sebelumnya, pada penelitian ini akan dilakukan analisis kinerja, karaktersitik sistem terhadap penggunaan daya. Diharapkan implementasi metode yang dirancang mampu menghemat daya tanpa mengurangi kinerja komputasi paralel.

3.4 Langkah Kerja Sistem

Langkah-langkah kerja sistem paralel dijelaskan dalam diagram alir berikut:



Gambar 3.5 Diagram alir langkah kerja komputasi paralel yang dijalankan pada sistem

Langkah-langkah kerja yang dijalankan di node master adalah sebagai berikut:

1. digunakan *randomwriter* sebagai data uji yang akan diolah pada klaster paralel,
2. mengatur skala kecepatan frekuensi untuk seluruh node sesuai frekuensi pengujian, dari nilai minimum hingga maksimum.
3. data yang akan diolah, dikirimkan ke HDFS menggunakan proses *mapping*,
4. program utama *wordcount* akan melakukan *mapreduce*, dan membagi rutin pekerjaan kepada setiap node slave, jika proses *mapping* gagal rutin program utama selesai,
5. pembagian beban kerja yang dikirimkan kepada setiap node slave akan diseimbangkan untuk tiap-tiap node slave menggunakan *load balancing* dengan *balancer* pada HDFS,
6. masing-masing node slave akan mendapatkan blok-blok beban kerja yang telah dibagi sesuai dengan *threshold balance* yang ditentukan.

Langkah-langkah kerja sistem paralel di masing-masing node slave adalah sebagai berikut:

1. kumpulan node slave bekerja sebagai *worker*,
2. mengatur skala kecepatan frekuensi untuk seluruh node sesuai frekuensi pengujian, dari nilai minimum hingga maksimum,
3. setiap node slave akan mendapatkan block data beban kerja dari node master, jika proses mapping gagal maka program utama selesai,
4. node slave melakukan komputasi *wordcount*, dengan menghitung banyak cacah kata pada bagian data yang dibebankan,
5. hasil komputasi dari node slave akan digabungkan menjadi satu keluaran melalui fase *Reduce*, yang ditempatkan pada HDFS,
6. tiap-tiap node slave yang akan mencatat penggunaan CPU yang terpakai saat komputasi berlangsung pada sebuah berkas bernama *cpu_usage*.

Rutin komputasi tahap akhir sistem paralel yang dijalankan di node master adalah:

1. keluaran program utama *WordCount* yang berupa hasil *log*, tersimpan di dalam HDFS, akan diambil oleh master dan kemudian akan dilakukan analisis data,
2. pengumpulan informasi mengenai hasil pencatatan penggunaan CPU dari seluruh node ke dalam satu berkas *cpu_usage_cluster* untuk tiap skenario DVFS yang diterapkan, terhadap perubahan frekuensi saat komputasi.

3.5 Pengujian Sistem

Pengujian dilakukan menggunakan rutin program *WordCount* untuk pengolahan data acak yang telah dibangkitkan menggunakan *RandomWriter*. *WordCount* menggunakan algoritma *MapReduce*. Ketika fase mapping data, jika terjadi pembagian blok data yang tidak merata, maka *Load Balancing* akan aktif dan dilakukan alokasi pemerataan terhadap pembagian blok data.

Pada pengujian, akan dilakukan perbandingan penggunaan energi terpakai dengan volume data yang akan diolah. Besar ukuran data yang akan diproses adalah sebesar 256MB, 128MB, dan 16MB. Serta pengaruh lama eksekusi komputasi terhadap perubahan frekuensi melalui skenario DVFS. Hasil lama waktu eksekusi komputasi akan menentukan efisiensi sistem dengan perhitungan matematik.

3.6 Analisis Data

Pengambilan data hasil pengujian sistem dilakukan dengan cara *Logging* serta pembacaan secara manual. Hasil *log file* mencatat total waktu eksekusi dan penggunaan sumberdaya dari sistem. Pembacaan manual dilakukan untuk mengukur konsumsi daya.

Setelah semua data yang dibutuhkan terkumpul maka dilakukan analisis data dan pembahasan yang mengacu pada rumusan masalah. Berikut ini analisis data yang akan dilakukan dalam penelitian ini yaitu:

a. Komputasi paralel terhadap beban komputasi
   Membahas pengaruh besar beban komputasi terhadap penggunaan energi.
b. Komputasi paralel dengan teknik Skala Tegangan dan Frekuensi
   Membahas pengaruh kinerja komputasi untuk tiap konfigurasi pada masing-masing skala frekuensi yang ditentukan.
c. Perhitungan efisiensi energi pada sistem komputer paralel
   Membahas perbandingan efisiensi dengan kinerja sistem komputer paralel.

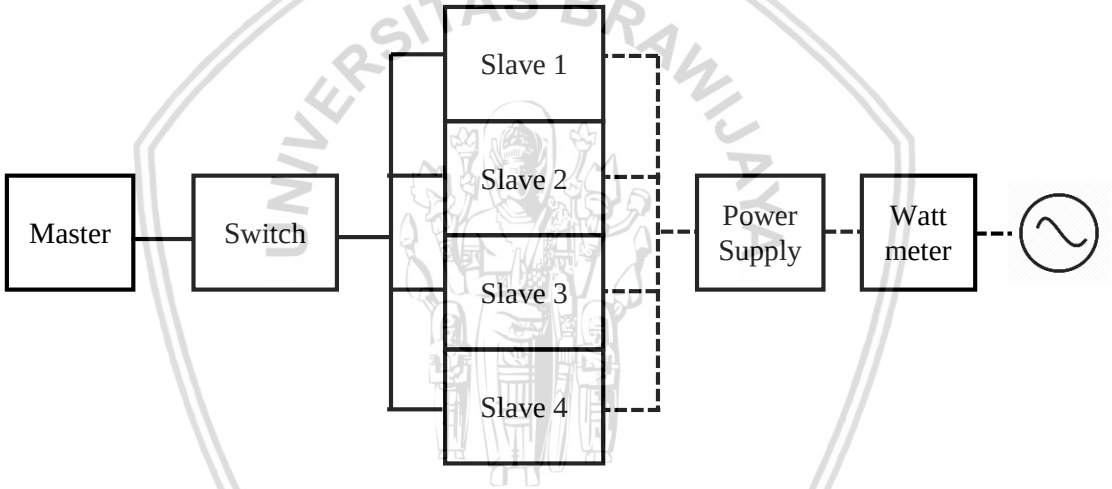
3.7 Pengambilan Kesimpulan dan Saran

Pada tahap ini, kesimpulan dan analisis dari pengujian dipaparkan. Analisis dilakukan untuk mengamati hasil pengujian serta untuk mengetahui pengaruh komputasi terhadap penggunaan daya. Tahap selanjutnya adalah pembuatan saran untuk perbaikan dan pengembangan penelitian selanjutnya. Kesimpulan dari kesangkilan sistem, struktur dan desain pada penerapa sistem komputer paralel.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

4.1 Konfigurasi Perangkat Keras

Sistem komputer paralel menggunakan 5 unit komputer *single-board*: 1 *master* dan 4 node *slave*. Jaringan area lokal digunakan sebagai media komunikasi data dengan *switch FastEthernet* melalui perkabelan Cat 5E. Klaster paralel dilengkapi dengan energimeter Wanf D02A sebagai pengukur daya yang terkonsumsi pada saat komputasi paralel dilakukan, seperti skema perancangan pada Gambar 4.1.



Gambar 4.1 Skema perancangan sistem komputer klaster paralel Orange Pi PC, dimana garis lurus menunjukkan pemasangan jaringan lokal pada sistem dan garis putus-putus menujukkan pemasangan sumber tegangan listrik.

Node merupakan bagian pekerja yang disebut sebagai *slave*, terhubung ke node *master* melalui *switch*. Server berguna sebagai peladen konfigurasi jaringan dalam sistem *gateway* penyedia hubungan jaringan lokal ke jaringan yang lain.

4.1.1 Perangkat Keras Node *Master*

Node master merupakan komputer tunggal yang digunakan sebagai antarmuka pengguna dan pengontrol sistem komputer paralel. Pada node master digunakan komputer *open-source single board computer* Orange Pi PC dengan spesifikasi perangkat keras yang diuraikan dalam Tabel 4.1.

Tabel 4.1 Perangkat Keras Node *Master*

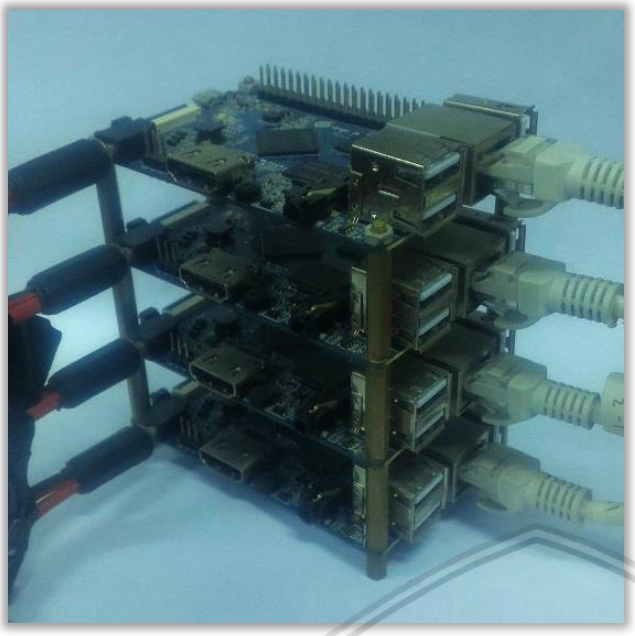
Processor	Allwinner H3 (sun8iw7p1) SoC
	CPU Quad-Core ARM Cortex-A7 @ 1.296GHz
	GPU Mali400 MP2 @ 600Mhz
Memory	1 GB DDR3L
Storage	MicroSD SanDisk 8GB
NIC	10/100 Ethernet

Node *master* diperlihatkan pada Gambar 4.2.

Gambar 4.2 Orange Pi sebagai Node *Master*

4.1.2 Perangkat Keras Node *Slave*

Node *slave* merupakan komputer jamak yang digunakan sebagai sistem komputasi paralel. Pada node *slave* digunakan komputer *open-source single board*, komputer Orange Pi PC. Spesifikasi perangkat komputer untuk tiap node *slave* diuraikan dalam Tabel 4.1. Masing-masing Orange Pi PC dihubungkan pada satu *power supply* dan terhubung ke dalam jaringan lokal. *Node slave* diperlihatkan pada Gambar 4.3.



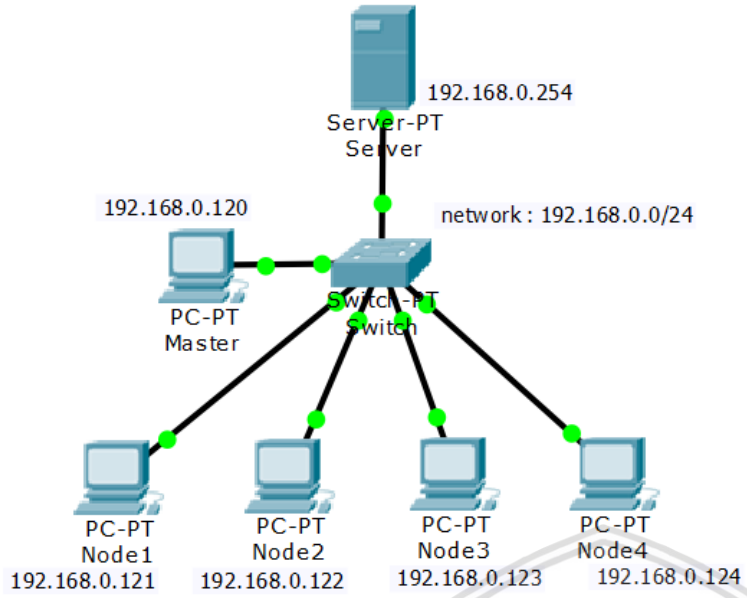
Gambar 4.3 Klaster Paralel dengan 4 unit Orange Pi PC sebagai Node *Slave*

4.1.3 Konfigurasi Jaringan Lokal

Jaringan sistem komputer paralel menggunakan jaringan lokal yang disusun sesuai topologi star.

Gambar 4.4 Perangkat *Switch* DES-1016A

Setiap perangkat terhubung langsung dengan *switch* DES-1016A, seperti pada Gambar 4.4, sebagai media komunikasi data yang mengakomodasi kecepatan transfer data 10/100 Mbps melalui kabel UTP cat 5e. Gambar 4.5 menunjukkan topologi jaringan pada sistem.



Gambar 4.5 Topologi Jaringan Lokal dan Alamat IP

4.2 Konfigurasi Perangkat Lunak

Konfigurasi perangkat lunak sistem komputer paralel melingkupi sistem operasi, konfigurasi jaringan, sistem berkas terdistribusi, kompilasi dan instalasi paket program, implementasi komputasi paralel dan otomasi pengujian.

4.2.1 Konfigurasi Orange Pi PC

Komputer master bertanggungjawab atas kontrol sistem komputer paralel. *Master* memiliki tugas utama, salah satunya adalah melakukan distribusi berkas rutin program melalui sistem berkas terdistribusi, kompilasi dan eksekusi program komputasi paralel.

Seluruh node menggunakan sistem operasi GNU/Linux Debian 8.1 (Jessie) dengan kernel 3.4.6. Konfigurasi ini dimudahkan dengan cara melakukan kloning sistem operasi dari *master* untuk node *slave* dalam klaster paralel, dengan tahapan sebagai berikut.

dd if=/dev/sdX of=/dev/sdY bs=4M

Dengan X sebagai perangkat yang menjadi sumber kloning dan Y sebagai target media penyimpanan hasil kloning.

4.2.2 Konfigurasi Alamat IP dan Hosts

Master menggunakan nama host masteropi dengan alamat IP 192.168.0.120. Sistem Operasi Debian menggunakan berkas /etc/hosts untuk mengasosiasikan nama host dan alamat IP. Seluruh komputer dalam sistem komputer paralel menggunakan berkas /etc/hosts yang berisi daftar seluruh hosts dalam satu jaringan sebagai pengenalan hosts lainnya.

Pengaturan alamat IP master dilakukan dengan menambahkan entri sebagai berikut ke dalam berkas /etc/network/interface.

auto eth0

iface eth0 inet static

address 192.168.0.120

netmask 255.255.255.0

gateway 192.168.0.254

Pengaturan nama host dilakukan dengan penambahan berkas entri ke dalam /etc/hostname.

masteropi

Pengaturan alamat IP *slave* tidak jauh berbeda dengan *master*, dengan perubahan pada baris *address* untuk N node komputer.

address 192.168.0.12N

Pengaturan nama host untuk tiap slave dilakukan penambahan entri ke dalam berkas /etc/hostname, dengan N merupakan nomor node.

slaveopiN

4.2.3 Konfigurasi User hdpi

Sistem komputer paralel membutuhkan satu buah user identik untuk menjalankan komputasi paralel. Hal ini mengharuskan adanya contoh user dengan uid dan gid yang identik pada seluruh hosts dalam klaster paralel. Selain mencegah terjadinya kerusakan perangkat lunak pada sistem, penggunaan satu buah user yang sama dapat mempermudah kendali pada tiap-tiap nodenya. Untuk menambahkan user identik bernama hdpi ditambahkan entri berikut sebagai root user.

adduser hdpi

Program adduser akan menampilkan prompt untuk melakukan konfigurasi user secara praktis dan akan menyediakan direktori home untuk user yang telah ditambahkan.

4.2.4 Konfigurasi SSH

Eksekusi program paralel membutuhkan *login* secara *remote* melalui protokol SSH. Untuk menjalankan *remote login* tanpa *prompt* sandi lewat, diperlukan duplikasi kunci publik user hdpi pada tiap node.

/home/hdpi/.ssh/authorized_keys

```
# ssh-keygen -t rsa -P ""
# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# scp ~/.ssh/authorized_keys hdpi@slaveopi1:~/.ssh/authorized_keys
```

4.2.5 Konfigurasi Java

Lingkungan pengembangan yang digunakan dalam sistem komputer paralel berbasis Java. Dengan memasukkan entri sebagai berikut dalam *console*, pemampatan java akan secara otomatis dilakukan dan berada dalam pengaturan secara *default*.

apt-get install oracle-java8-installer javac

4.2.6 Konfigurasi dan Instalasi *Hadoop Framework*

Konfigurasi dan pemasangan *Hadoop framework* dilakukan menggunakan *bash script* installhadoop.sh untuk memudahkan proses penyematan pada sistem. Di dalam script juga disertakan konfigurasi untuk *Hadoop distributed filesystem* (HDFS) juga disertakan. Entri yang ditambahkan untuk konfigurasi tersebut dilampirkan pada halaman konfigurasi *Hadoop framework* (lampiran 1).

4.3 Konfigurasi Skala Tegangan dan Frekuensi Dinamis (*DVFS*)

Konfigurasi skala tegangan dan frekuensi dinamis dilakukan dengan menambahkan *binary script* yang akan dibaca oleh Orange Pi saat dilakukan proses *booting*. *Binary script* dienkripsi menggunakan *fex file* yang telah disediakan oleh pabrikan. Penambahan entri berkas yang dienkripsi untuk konfigurasi DVFS terlampir pada halaman lampiran konfigurasi DVFS (lampiran 2). Orange PI akan menjalankan *binary script* saat *booting* dilakukan. Berikut adalah skala tegangan yang akan dikonfigurasikan pada sistem komputer paralel.

```
[dvfs_table]
pmuic_type = 2
pmu_gpio0 = port:PL06<1><1><2><1>
pmu_level0 = 11300
pmu_level1 = 576
extremity_freq = 1536000000
max_freq = 1536000000
min_freq = 180000000
LV_count = 11
LV1_freq = 1536000000
LV1_volt = 1500
LV2_freq = 1296000000
LV2_volt = 1340
LV3_freq = 1200000000
LV3_volt = 1320
LV4_freq = 1008000000
LV4_volt = 1200
LV5_freq = 916000000
LV5_volt = 1100
LV6_freq = 816000000
LV6_volt = 1100
LV7_freq = 648000000
LV7_volt = 1040
LV8_freq = 504000000
```

LV8_volt = 1040
LV9_freq = 480000000
LV9_volt = 1040
LV10_freq = 240000000
LV10_volt = 1040
LV11_freq = 180000000
LV11_volt = 1040

[gpu_dvfs_table]
G_LV_count = 3
G_LV0_freq = 312000000
G_LV0_volt = 1200000
G_LV1_freq = 384000000
G_LV1_volt = 1200000
G_LV2_freq = 456000000
G_LV2_volt = 1200000

4.4 Program Komputasi *WordCount*

Program komputasi paralel *WordCount* untuk pengujian diimplementasikandengan bahasa pemrograman berbasis Java. Kode program wc.java terlampir (lampiran 3).

Kompilasi program dilakukan dengan program javac:

hadoop com.sun.tools.javac.Main WordCount.java

jar cf wc.jar WordCount*.class

4.5 Program *Bash Shell Script* untuk Otomasi Pengujian

Beberapa *shell script* digunakan untuk melakukan otomasi pengujian secara *batch*. Skrip paralelstatus.sh digunakan untuk mengetahui keadaan kecepatan clock frekuensi dan temperatur prosesor dari seluruh *slave*. Skrip sync_hadoop_config.sh digunakan untuk sinkronisasi konfigurasi *Hadoop framework* pada tiap *slave*. Skrip reformat_hdfs.sh digunakan untuk melakukan format pada sistem berkas terdistribusi. Skrip cpulog_paralel.sh digunakan untuk melakukan logging penggunaan sumberdaya dari prosesor.

Skrip controldvfs.sh digunakan untuk melakukan control kecepatan clock frekuensi prosesor pada tiap node dalam klaster paralel. Skrip rutin.sh digunakan untuk menjalankan program komputasi paralel *WordCount* dengan beberapa volume data uji yang berbeda.

BAB V
PENGUJIAN DAN ANALISIS

Kecepatan komputasi sebuah prosesor dipengaruhi oleh nilai kecepatan frekuensi prosesor. Besaran nilai frekuensi tersebut dibatasi oleh besar tegangan inti yang ada pada prosesor. Perubahan pada nilai tegangan akan mempengaruhi *switching power*.

Perubahan nilai frekuensi mempengaruhi siklus *sampling* clock pada prosesor. Semakin besar frekuensi akan semakin banyak set instruksi yang dapat dikerjakan oleh prosesor dalam satu waktu.

5.1 Skala Tegangan dan Frekuensi

Skala tegangan diatur dalam berkas *script.bin* yang diakses oleh kernel saat *booting* dilakukan. Pengaturan frekuensi dilakukan secara langsung menggunakan *shell script* control_dvfs.sh dari frekuensi tertinggi dengan nilai frekuensi sebesar 1536 MHz hingga nilai frekuensi terendah dengan nilai 120 MHz. Pengukuran penggunaan energi dilakukan dengan pembacaan secara manual pada Wattmeter. Pada Tabel 5.1 diperlihatkan skala tegangan yang diatur untuk tiap level frekuensi yang bekerja.

Tabel 5.1 Skala Tegangan dan Frekuensi Sistem Komputer Paralel

Frekuensi (MHz)	**Tegangan (V_{core})**
1536	1500
1296	1340
1200	1320
1008	1200
916	1100
816	1100
648	1040
504	1040
480	1040
240	1040
180	1040

5.2 Pengujian Pengaruh Volume Data terhadap Energi

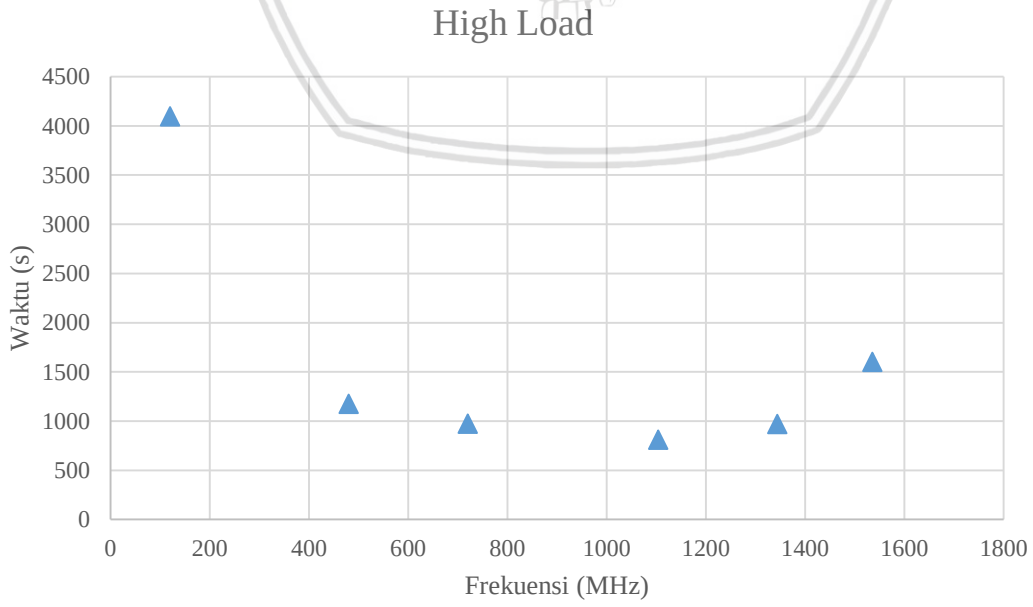
Pada komputasi penghitung cacah kata atau *wordcount* akan digunakan 3 jenis beban uji yaitu beban besar, beban sedang, dan beban kecil. Beban uji dibangkitkan menggunakan algoritma *mapreduce*. Berkas yang dihasilkan akan diolah dan dibagikan secara rata menggunakan *Load Balancing*.

5.2.1 Analisis Komputasi Beban Besar dan Perubahan Frekuensi

Pada pengujian komputasi beban besar disediakan berkas untuk pengujian yang dibangkitkan menggunakan *randomwriter* dengan algoritma *mapreduce*. Besar volume berkas uji adalah sebesar 256 MB. Nilai tersebut didapatkan dari uji kemampuan memori dari *single-board computer*. Hasil pengujian komputasi paralel pada beban uji besar diperlihatkan pada Tabel 5.2

Tabel 5.2 Hasil Komputasi Paralel Data 256 MB

Frekuensi (MHz)	Waktu Eksekusi (s)	Waktu CPU (ms)	Suhu Prosessor (°C)
1536	1600.9	772564	66.4
1344	971.1	799023	70.5
1104	811.5	891911	57.1
720	973.6	1336171	50.9
480	1174.9	1967891	50.6
120	4096.9	8526792	48.5



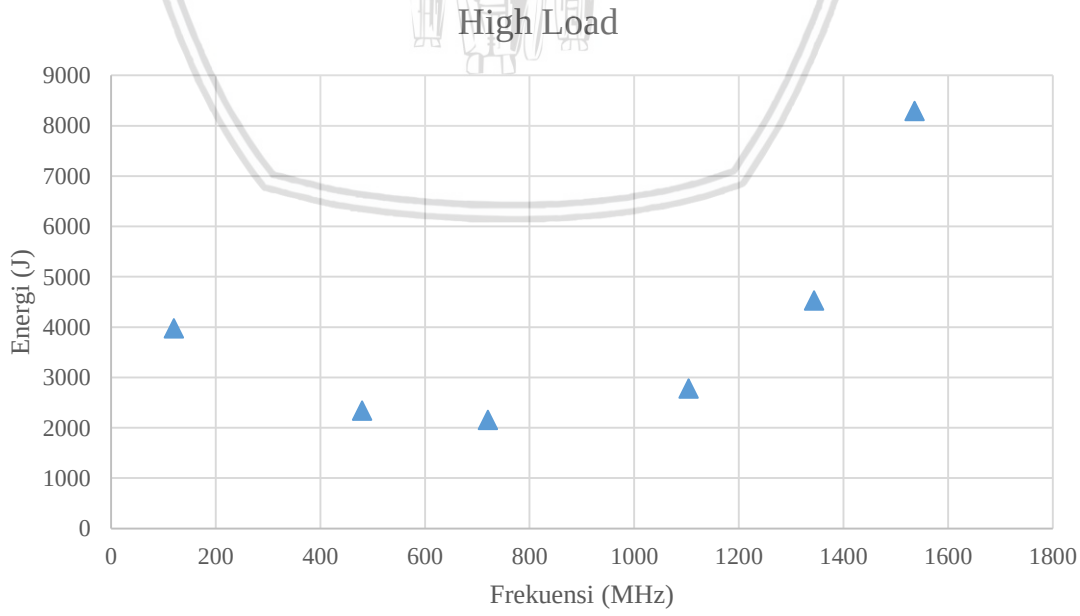
Gambar 5.1 Grafik peningkatan kecepatan komputasi terhadap frekuensi pada komputasi paralel beban besar.

Pada Gambar 5.1 dapat dilihat bahwa pada saat komputasi dilakukan menggunaakan frekuensi paling rendah membutuhkan waktu penyelesaian yang lebih lama. Hal ini dikarenakan waktu sampling instruksi dalam satuan siklus pada prosesor menjadi kecil. Dan dapat diperhatikan pula pada penggunaan frekuensi paling tinggi terjadi penurunan waktu penyelesaian. Hal ini dikarenakan kemampuan kecepatan prosesor terhambat oleh pengiriman data dari periferal menuju ke prosesor.

Penggunaan energi yang terpakai oleh sistem dalam komputasi paralel bisa didapatkan dari hasil selisih energi saat kondisi aktif dengan energi saat kondisi *idle* kali lama waktu eksekusi komputasi. Pada Gambar 5.2 dapat dilihat bahwa terjadi peningkatan konsumsi energi pada saat dilakukan komputasi dengan frekuensi tertinggi.

Tabel 5.3 Penggunaan Energi Komputasi Paralel Beban Besar

Frekuensi (MHz)	**Daya Komputasi (W)**	**Waktu Eksekusi (s)**	**Energi (W s)**
1536	5.18	1600.9	8292.66
1344	4.66	971.1	4525.33
1104	3.42	811.5	2775.33
720	2.21	973.6	2154.9
480	1.99	1174.9	2338.05
120	0.97	4096.9	3973.99



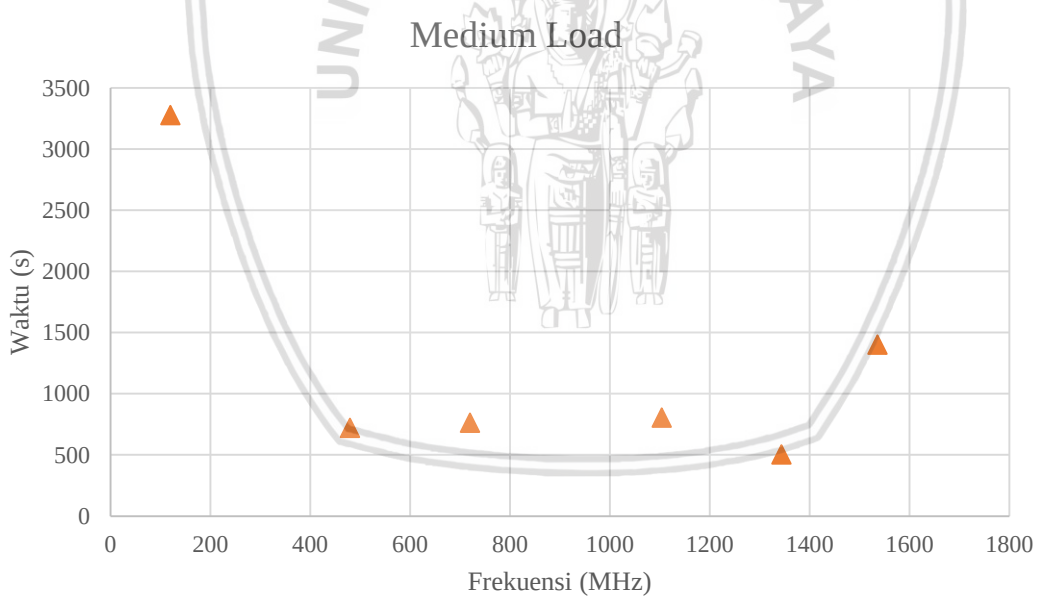
Gambar 5.2 Grafik penggunaan energi untuk tiap perubahan frekuensi pada komputasi paralel beban besar.

5.2.2 Analisis Beban Uji Sedang Terhadap Perubahan Frekuensi

Pada pengujian komputasi beban sedang, digunakan metode yang sama seperti pengujian beban besar. Berkas uji dibangkitkan menggunakan *randomwriter* dengan volume sebesar 128 MB. Hasil pengujian komputasi paralel pada beban uji sedang diperlihatkan pada Tabel 5.4

Tabel 5.4 Hasil Komputasi Paralel Beban 128 MB

Frekuensi (MHz)	Waktu Eksekusi (s)	Waktu CPU (ms)	Suhu Prosessor (℃)
1536	1401.3	407184	62.7
1344	502.9	417762	70.5
1104	804.22	481550	57.1
720	762.5	720187	51.1
480	719.8	1030573	49.5
120	3277	4674973.75	48.3

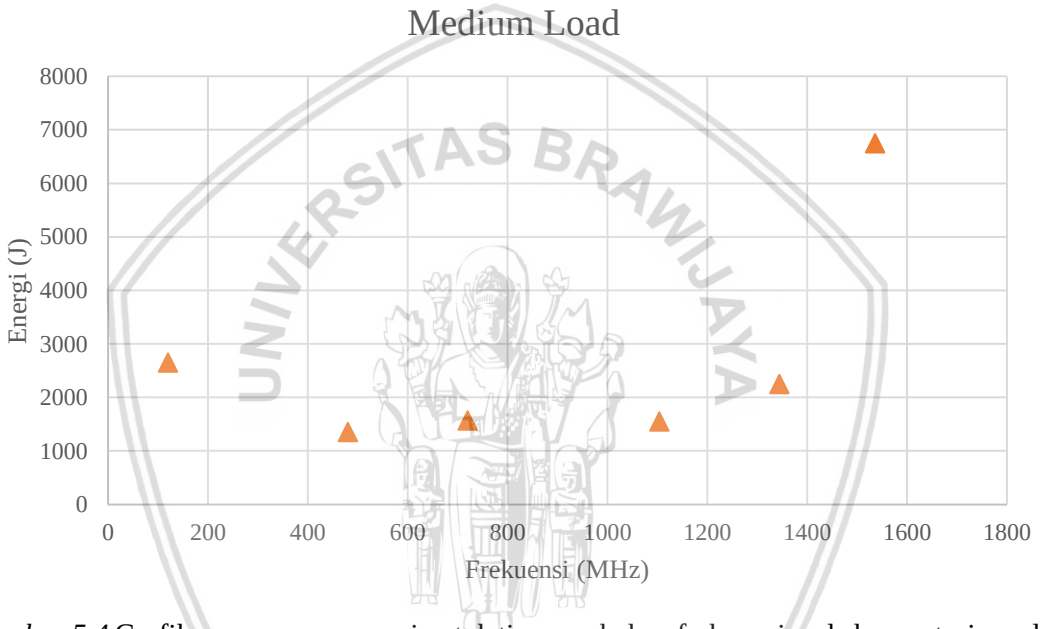


Gambar 5.3 Grafik peningkatan kecepatan terhadap frekuensi pada komputasi paralel beban sedang.

Pada Gambar 5.3 dapat dilihat bahwa komputasi paralel beban sedang menggunakan frekuensi rendah membutuhkan waktu eksekusi lebih lama. Peningkatan kecepatan frekuensi menunjukkan pengurangan waktu eksekusi komputasi paralel, namun pada frekuensi tertinggi terjadi pelambatan.

Tabel 5.5 Penggunaan Energi Komputasi Paralel Beban Sedang

Frekuensi (MHz)	Daya Komputasi (W)	Waktu Eksekusi (s)	Energi (W s)
1536	4.82	1401.3	6754.27
1344	4.48	502.9	2252.99
1104	1.93	804.22	1552.15
720	2.06	762.5	1570.75
480	1.88	719.8	1353.22
120	0.81	3277	2654.37



Gambar 5.4 Grafik penggunaan energi untuk tiap perubahan frekuensi pada komputasi paralel beban sedang.

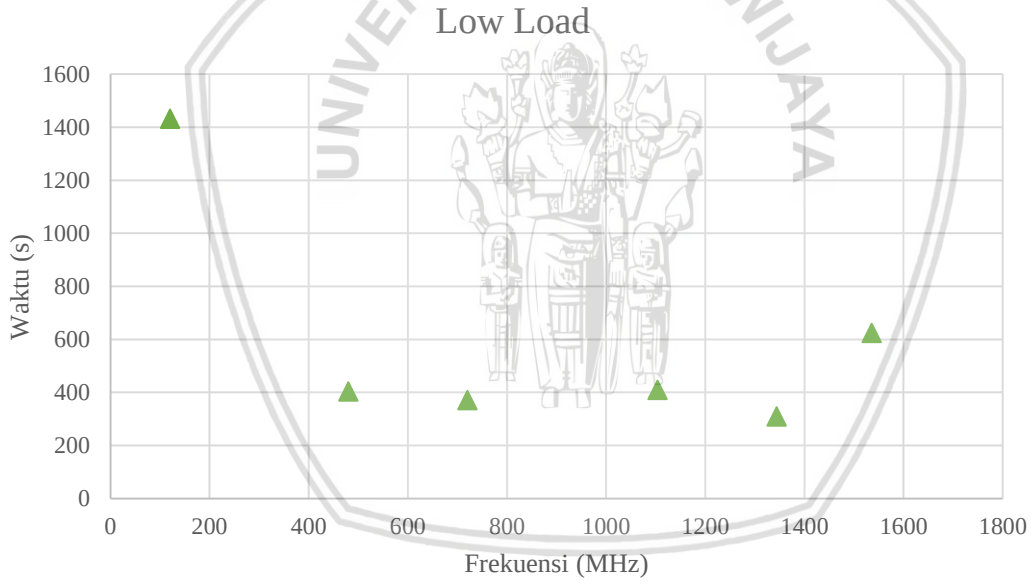
Pada Gambar 5.4 dapat dilihat bahwa peningkatan frekuensi menyebabkan peningkatan penggunaan energi. Dari analisis data, dapat diketahui bahwa penyebab peningkatan energi pada frekuensi tertinggi dikarenakan terjadinya pelambatan waktu komputasi paralel.

5.2.3 Analisis Beban Uji Kecil Terhadap Perubahan Frekuensi

Pada pengujian komputasi beban kecil, digunakan berkas uji dengan volume sebesar 16 MB. Hasil pengujian komputasi paralel pada beban uji kecil untuk setiap perubahan frekuensi diperlihatkan pada Tabel 5.6.

Tabel 5.6 Hasil Komputasi Paralel Beban 16 MB

Frekuensi (MHz)	Waktu Eksekusi (s)	Waktu CPU (ms)	Suhu Prosessor (°C)
1536	625.4	94800	64.1
1344	309.9	93964	66.2
1104	409.9	123280	57.6
720	371	152226	51.6
480	403.4	225803	51.8
120	1432.2	991570	49.5

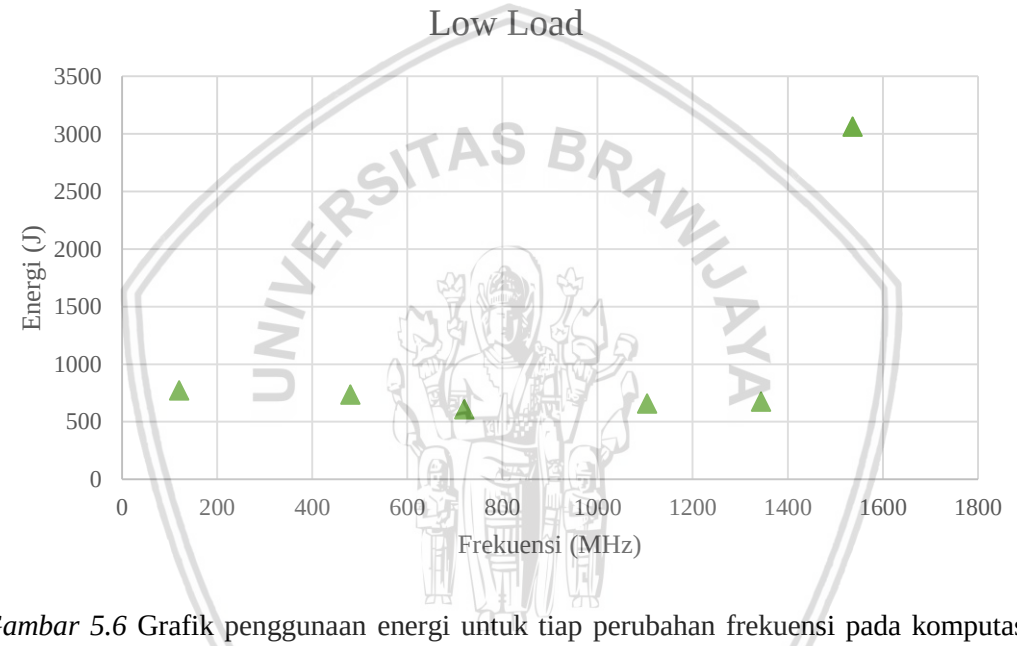


Gambar 5.5 Grafik peningkatan kecepatan terhadap frekuensi pada komputasi paralel beban kecil.

Pada Gambar 5.5 dapat dilihat bahwa pada penggunaan frekuensi rendah, kecepatan komputasi membutuhkan waktu yang lama. Semakin tinggi kecepatan frekuensi prosesor, akan semakin mempersingkat waktu yang dibutuhkan untuk menyelesaikan komputasi paralel. Namun pada frekuensi tertinggi terjadi pelambatan kecepatan komputasi.

Tabel 5.7 Penggunaan Energi Komputasi Paralel Beban Kecil

Frekuensi (MHz)	Daya Komputasi (W)	Waktu Eksekusi (s)	Energi (W s)
1536	4.9	625.4	3064.46
1344	2.18	309.9	675.58
1104	1.61	409.9	659.94
720	1.64	371	608.44
480	1.83	403.4	738.22
120	0.54	1432.2	773.39



Gambar 5.6 Grafik penggunaan energi untuk tiap perubahan frekuensi pada komputasi paralel beban kecil.

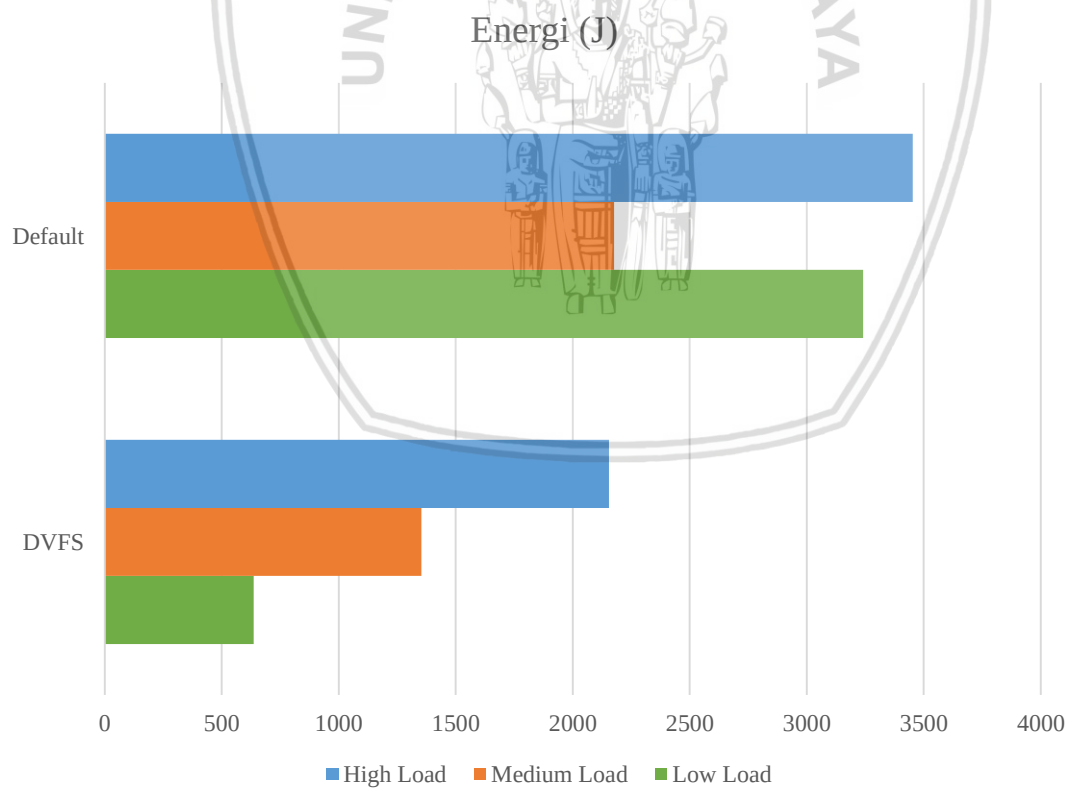
Pada Gambar 5.6 dapat dilihat bahwa penggunaan energi pada komputasi paralel beban kecil untuk tiap perubahan frekuensinya relatif sama. Terjadi *spike* pada grafik hasil pengujian menggunakan frekuensi tertinggi dikarenakan adanya inkonsistensi lama waktu eksekusi komputasi paralel.

5.3 Analisis Penghematan Energi

Penghematan energi pada sistem komputer paralel dapat dilihat melalui perbandingan komputasi paralel menggunakan konfigurasi *default* terhadap konfigurasi DVFS. Pada konfigurasi *default*, tegangan inti pada prosesor bernilai sebesar 1.2V dengan frekuensi yang bekerja secara aman hingga sebesar 1008MHz. Berdasarkan *datasheet*, tegangan absolut maksimum untuk prosesor adalah sebesar 1.5V dan rekomendasi maksimum besar 1.4V. Penghematan yang akan dibandingkan adalah penggunaan energi minimum antara konfigurasi DVFS dengan konfigurasi tanpa skala DVFS.

Tabel 5.8 Penggunaan Energi Minimum Komputasi Paralel

Konfigurasi	**Energi (J)**		
	Low Load	**Medium Load**	**High Load**
Default	3240.7	2176	3452.8
DVFS	635.98	1353.22	2154.9



Gambar 5.7 Perbandingan penggunaan energi minimum komputasi pada sistem komputer paralel konfigurasi *default* dan sistem komputer paralel DVFS.

Dari hasil tersebut, maka dapat diketahui besar prosentasi penghematan energi dengan menggunakan persamaan berikut,

$$\eta = \frac{E_{default} - E_{dvfs}}{E_{default}} \; x \; 100\%$$

dimana $E_{default}$ adalah besar penggunaan energi komputasi pada konfigurasi secara *default* dan E_{dvfs} merupakan besar penggunaan energi komputasi pada konfigurasi DVFS dalam satuan joule. Maka berdasarkan persamaan di atas, dapat dihitung prosentasi penghematan adalah sebagai berikut:

- Beban besar

  $$\eta = \frac{3452.8 - 2154{,}9}{3452.8} \; x \; 100\%$$

  $$\eta = 0.3759 \; x \; 100\%$$

  $$\eta = 37.59 \; \%$$

- Beban sedang

  $$\eta = \frac{2176 - 1353{,}22}{2176} \; x \; 100\%$$

  $$\eta = 0.3781 \; x \; 100\%$$

  $$\eta = 37.81 \; \%$$

- Beban kecil

  $$\eta = \frac{3240.7 - 635{,}98}{3240.7} \; x \; 100\%$$

  $$\eta = 0.8038 \; x \; 100\%$$

  $$\eta = 88.38 \; \%$$

Berdasarkan perhitungan maka penghematan yang dapat diperoleh dengan menggunakan teknik skala tegangan dan frekuensi dinamis sebesar 37.59 % untuk beban uji besar, 37.81 % untuk beban uji sedang dan 88.38 % untuk beban uji kecil.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan pengujian dan analisis sistem komputer paralel dengan skala tegangan dan frekuensi dinamis yang telah dilakukan, didapatkan beberapa kesimpulan sebagai berikut.

1. Ukuran beban kerja komputasi yang dijalankan oleh sistem komputer paralel mempengaruhi besar penggunaan energi. Semakin besar beban kerja komputasi, maka penggunaan energi akan semakin meningkat.
2. Peningkatan skala tegangan pada prosesor dapat meningkatkan kecepatan frekuensi prosesor untuk mempersingkat lama waktu eksekusi komputasi, dan sebaliknya penurunan tegangan dapat menurunkan kecepatan frekuensi prosesor namun memperpanjang lama waktu eksekusi komputasi.
3. Penggunaan skala tegangan dan frekuensi dinamis yang optimal dapat menghasilkan penghematan konsumsi energi pada sistem komputer paralel.

6.2 Saran

Adapun beberapa saran yang dapat dipertimbangkan sebagai pengembangan untuk penelitian lebih lanjut sebagai berikut.

1. Penggunaan sistem komputer paralel dengan spesifikasi yang lebih superior.
2. Penggunaan cacah node klaster paralel yang lebih banyak.
3. Penggunaan jenis algoritma komputasi paralel lainnya.
4. Penggunaan *scheduling* untuk pengaturan sumberdaya sistem.
5. Analisis kemampuan dan konsistensi pembacaan data dari memori ke prosesor.
6. Komputasi paralel *ubiquitous* menggunakan perangkat *mobile* atau *embedded*.
7. Studi kasus komputasi paralel untuk solusi Big Data lainnya.

DAFTAR PUSTAKA

A. M. Pfalzgraf and J. A. Driscoll. (2014). *A low-cost computer cluster for high-performance computing education*. IEEE International Conference on Electro/Information Technology, Milwaukee, WI, pp. 362-366.

A. P. Florence and V. Shanthi. (2014). *Energy aware load balancing for computational cloud*. 2014 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, pp. 1-3.

Barry, Wilkinson and Michael, Allen. (2010). *Parallel Pogramming*. Yogyakarta: Penerbit ANDI

Christian, Ari B. (2017). *Konfigurasi Jaringan Bertingkat Pada Cluster Paralel Orange Pi*. Skripsi tidak dipublikasikan. Malang: Universitas Brawijaya.

M. Etinski, J. Corbalan, J. Labarta, M. Valero and A. Veidenbaum. (2009). *Power-aware load balancing of large scale MPI applications.* 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, pp. 1-8.

G. Michael Noll. (2017). *Applied Research. Big Data. Distributed Systems. Open Source.* http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/ (diakses tanggal 11 September 2017)

G. Kecskemeti, W. Hajji and F. P. Tso. (2017). *Modelling Low Power Compute Clusters for Cloud Simulation*. 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), St. Petersburg, pp. 39-45.

George S. Almasi and Allan Gottlieb. (1994). *Highly Parallel Computing (2nd Ed.)*. Benjamin-Cummings Publ. Co., Inc., Redwood City, CA, USA.

G. Terzopoulos and H. Karatza. (2016). *Power-aware load balancing in heterogeneous clusters*. 2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Toronto, ON, 2013, pp. 148-154.

J. A. Issa. (2015). *Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount*. IEEE Access, vol. 3, pp. 2784-2793, 2015.

M. U. K. Khan, M. Shafique, A. Gupta, T. Schumann and J. Henkel. (2016). Power-efficient load-balancing on heterogeneous computing platforms. 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016. pp. 1469-1472.

N. Zhu, X. Liu, J. Liu and Y. Hua. (2014). *Towards a cost-efficient MapReduce: Mitigating power peaks for Hadoop clusters*. Tsinghua Science and Technology, vol. 19, no. 1, pp. 24-32, Feb. 2014.

P. Trancoso and M. Efstathiou. (2017). *Low-Cost Sub-5W Processors for Edge HPC*. 2017 Euromicro Conference on Digital System Design (DSD), Vienna, 2017, pp. 529-532.

Padoin, Edson & Martínez Abaunza, Víctor Eduardo & Navaux, Philippe & Méhaut, Jean-François. (2017). *Using Power Demand and Residual Load Imbalance in the Load Balancing to Save Energy of Parallel Systems*. Procedia Computer Science. 108. 695-704. 10.1016/j.procs.2017.05.215.

Padoin, Edson & Bastos Castro, Marcio & Lima Pilla, Laércio & Navaux, Philippe & Méhaut, Jean-François. (2014*). Saving Energy by Exploiting Residual Imbalances on Iterative Applications*. 2014 21st International Conference on High Performance Computing, HiPC 2014. 10.1109/HiPC.2014.7116895.

Sanguthevar Rajasekaran and John Reif. (2007). *Handbook of Parallel Computing: Models, Algorithms and Applications* Chapman & Hall/Crc Computer & Information Science Series 1 ed.

Richard C. Jaeger and Travis Blalock. (2010). *Microelectronic Circuit Design 4th Edition*. McGraw-Hill Science/Engineering/Math.

Shelepov, Daniel & Fedorova, Alexandra. (2012). *Scheduling on Heterogeneous Multicore Processors Using Architectural Signatures*. Vancouver: Simon Fraser University.

Laman situs "http://hadoop.apache.org/docs/r2.7.4/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html" diakses pada tanggal 02-11-2017 14:20.

LAMPIRAN

Lampiran 1 Script Shell untuk Konfigurasi Hadoop Framework

```
echo -e "\e[32m Configuration Files\e[0m"
echo -e "\e[32m##########################################\n\e[0m"

set -xv
sudo update-alternatives --auto java
java -version
javac -version
cp ~/.bashrc ~/.bashrc.bak
sed -i -e '/#HADOOP VARIABLES START/,+11d' ~/.bashrc
cat << 'EOT' >> ~/.bashrc
#SET JDK
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:jre/bin/java::")
#HADOOP VARIABLES START
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
#HADOOP VARIABLES END
EOT

sed -i.bak -e 's/export JAVA_HOME=${JAVA_HOME}/export JAVA_HOME=$(readlink -f
\/usr\/bin\/java | sed "s:jre\/bin\/java::")/g' /usr/local/hadoop/etc/hadoop/hadoop-env.sh

sed -n -i.bak '/<configuration>/q;p'  /usr/local/hadoop/etc/hadoop/core-site.xml
cat << EOT >> /usr/local/hadoop/etc/hadoop/core-site.xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
EOT

sed -n -i.bak '/<configuration>/q;p' /usr/local/hadoop/etc/hadoop/yarn-site.xml
cat << EOT >> /usr/local/hadoop/etc/hadoop/yarn-site.xml
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
EOT
```

```
cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml
sed -n -i.bak '/<configuration>/q;p'  /usr/local/hadoop/etc/hadoop/mapred-site.xml
cat << EOT >> /usr/local/hadoop/etc/hadoop/mapred-site.xml
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
EOT

mkdir -p ~/hadoop_store/hdfs/namenode
mkdir -p ~/hadoop_store/hdfs/datanode
sed -n -i.bak '/<configuration>/q;p'  /usr/local/hadoop/etc/hadoop/hdfs-site.xml
cat << EOT >> /usr/local/hadoop/etc/hadoop/hdfs-site.xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/$USER/hadoop_store/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/$USER/hadoop_store/hdfs/datanode</value>
  </property>
</configuration>
EOT
set +xv

sleep 2s
echo -e "\n\n"

set +euo pipefail

source ~/.bashrc &>/dev/null

clear
echo -e "\e[32mHadoop configuration was successful!\e[0m"
```

Lampiran 2 Konfigurasi DVFS

```
[product]
version = "100"
machine = "orangepi-plus"

[cooler_table]
cooler_count = 8
cooler0 = "1536000 4 4294967295 0"
cooler1 = "1296000 4 4294967295 0"
cooler2 = "1200000 4 4294967295 0"
cooler3 = "1080000 4 4294967295 0"
cooler4 = "912000 4 4294967295 0"
cooler5 = "816000 4 4294967295 0"
cooler6 = "648000 3 4294967295 0"
cooler7 = "504000 2 4294967295 0"
cooler8 = "480000 1 4294967295 0"

[corekeeper]
corekeeper_enabled = 1

[dvfs_table]
pmuic_type = 2
pmu_gpio0 = port:PL06<1><1><2><1>
pmu_level0 = 11300
pmu_level1 = 576
extremity_freq = 1536000000
max_freq = 1536000000
min_freq = 180000000
LV_count = 11
LV1_freq = 1536000000
LV1_volt = 1500
LV2_freq = 1296000000
LV2_volt = 1340
LV3_freq = 1200000000
LV3_volt = 1320
LV4_freq = 1008000000
LV4_volt = 1200
LV5_freq = 916000000
LV5_volt = 1100
LV6_freq = 816000000
LV6_volt = 1100
LV7_freq = 648000000
LV7_volt = 1040
LV8_freq = 504000000
LV8_volt = 1040
LV9_freq = 480000000
LV9_volt = 1040
LV10_freq = 240000000
LV10_volt = 1040
LV11_freq = 180000000
LV11_volt = 1040
```

Lampiran 3 Kode Program WordCount

```
import java.io.IOException;
import java.io.PrintStream;
import java.util.Date;
import java.util.Iterator;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount
{
  public static void main(String[] paramArrayOfString)
    throws Exception
  {
    Configuration localConfiguration = new Configuration();
    String[] arrayOfString = new GenericOptionsParser(localConfiguration,
paramArrayOfString).getRemainingArgs();
    if (arrayOfString.length < 2)
    {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job localJob = Job.getInstance(localConfiguration, "wordcount");
    localJob.setJarByClass(WordCount.class);
    localJob.setMapperClass(WordCount.TokenizerMapper.class);
    localJob.setCombinerClass(WordCount.IntSumReducer.class);
    localJob.setReducerClass(WordCount.IntSumReducer.class);
    localJob.setOutputKeyClass(Text.class);
    localJob.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < arrayOfString.length - 1; i++) {
      FileInputFormat.addInputPath(localJob, new Path(arrayOfString[i]));
    }
    FileOutputFormat.setOutputPath(localJob, new Path(arrayOfString[(arrayOfString.length -
1)]));

    Date localDate1 = new Date();
    System.out.println("Job started: " + localDate1);
    int j = localJob.waitForCompletion(true) ? 0 : 1;
    Date localDate2 = new Date();
    System.out.println("Job ended: " + localDate2);
    System.out.println("The job took " + (localDate2.getTime() - localDate1.getTime()) / 1000L
+ " seconds.");

    System.exit(localJob.waitForCompletion(true) ? 0 : 1);
```

```
  }

  public static class IntSumReducer
    extends Reducer<Text, IntWritable, Text, IntWritable>
  {
    private IntWritable result = new IntWritable();

    public void reduce(Text paramText, Iterable<IntWritable> paramIterable, Reducer<Text,
IntWritable, Text, IntWritable>.Context paramReducer)
      throws IOException, InterruptedException
    {
      int i = 0;
      IntWritable localIntWritable;
      for (Iterator localIterator = paramIterable.iterator(); localIterator.hasNext(); i +=
localIntWritable.get()) {
        localIntWritable = (IntWritable)localIterator.next();
      }
      this.result.set(i);
      paramReducer.write(paramText, this.result);
    }
  }

  public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>
  {
    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object paramObject, Text paramText, Mapper<Object, Text, Text,
IntWritable>.Context paramMapper)
      throws IOException, InterruptedException
    {
      StringTokenizer localStringTokenizer = new StringTokenizer(paramText.toString());
      while (localStringTokenizer.hasMoreTokens())
      {
        this.word.set(localStringTokenizer.nextToken());
        paramMapper.write(this.word, one);
      }
    }
  }
}
```

Lampiran 4 Shell Script Otomasi Pendukung

1. controldvfs.sh

```
#!/bin/bash
# Controlling DVFS For Your Sikiripsi, master
cmpid=$BASHPID;

echo -e "\e[32m Capturing Data: Frequency and CPU Usage \e[0m"
echo -e "\e[32m##########################################\n\e[0m"

username=root
hosts="masteropi slaveopi1 slaveopi2 slaveopi3 slaveopi4"
script="echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor; echo 480000 >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed;"

for hostname in ${hosts}; do
   ssh -l ${username} ${hostname} "${script}"
done

sleep 2;
kill $cmpid
```

2. rutin.sh

```
#!/bin/bash
# The whole purpose to make easier for your sikiripsi, master
#cmpid=${BASHPID}
set -xv

echo Pengambilan data 256mb
n=2
while [ $n -lt 11 ]; do
   echo ============PENGAMBILAN DATA KE $n============
   hadoop fs -rm -r sampleout256
   ./capturing_data.sh
   nohup ./cpulog_paralel.sh &
   hadoop jar wc.jar WordCount sample256/ sampleout256/
   ./paralelstatus.sh
   ./capturing_data_finish.sh
   ./collecting_data
   cp -r output_all/ output_all$n
   let n=n+1
  done
sleep 5

echo Pengambilan data 128mb
i=11
while [ $i -lt 21 ]; do
   echo ============PENGAMBILAN DATA KE $i============
   hadoop fs -rm -r sampleout128
   ./capturing_data.sh
   nohup ./cpulog_paralel.sh &
   hadoop jar wc.jar WordCount sample128/ sampleout128/
   ./paralelstatus.sh
   ./capturing_data_finish.sh
   ./collecting_data
   cp -r output_all/ output_all$i
   let i=i+1
  done
sleep 5

echo Pengambilan data 16mb
counter=21
while [ $counter -lt 31 ]; do
   echo ============PENGAMBILAN DATA KE $counter============
```

```
    hadoop fs -rm -r sampleout16
    ./capturing_data.sh
    nohup ./cpulog_paralel.sh &
    hadoop jar wc.jar WordCount sample16/ sampleout16/
    ./paralelstatus.sh
    ./capturing_data_finish.sh
    ./collecting_data
    cp -r output_all/ output_all$counter
    let counter=counter+1
   done

set +xv
echo TEST FOR RESEARCH COMPLETED - HAVE A NICE DAY
#kill cmpid
```

3. cpustatus.sh

```
#!/bin/bash
# cpustatus untuk tiap node
# The temperature is reported in degrees Celsius (C) while
# the CPU speed is calculated in megahertz (MHz).

function convert_to_MHz {
    let value=$1/1000
    echo "$value"
}

#untuk membaca tegangan (orangpi pc tidak tersedia vcgencmd untuk mengukur tegangan)
#function calculate_overvolts {
    # We can safely ignore the integer
    # part of the decimal argument
    # since it's not realistic to run the Pi
    # at voltages higher than 1.99 V
#    let overvolts=${1#*.}-20
#   echo "$overvolts"
#}
#temp=$(vcgencmd measure_temp)
temp=$(cat /sys/devices/virtual/thermal/thermal_zone0/temp)
#temp=${temp:5:4}
#volts=$(vcgencmd measure_volts)
#volts=${volts:5:4}
#if [ $volts != "1.20" ]; then
#    overvolts=$(calculate_overvolts $volts)
#fi

minFreq=$(cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq)
minFreq=$(convert_to_MHz $minFreq)
maxFreq=$(cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq)
maxFreq=$(convert_to_MHz $maxFreq)
freq=$(cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq)
freq=$(convert_to_MHz $freq)
governor=$(cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor)

echo "Temperature:   $temp C"
#echo -n "Voltage:       $volts V"
#[ $overvolts ] && echo " (+0.$overvolts overvolt)" || echo -e "\r"
echo "Min speed:     $minFreq MHz"
echo "Max speed:     $maxFreq MHz"
echo "Current speed: $freq MHz"
echo "Governor:      $governor"

exit 0
```

Lampiran 5 Analisis Uji Komputasi Berdasarkan Perubahan Frekuensi

1. Skenario Performance (Frekuensi F = 1536 MHz)

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	690	759200	15.5	19.1	67	2095132672	3854667776	2484
2	high_load	1367	791640	15.7	21.3	68	2016137216	3853926400	7655.2
3	high_load	2884	737920	15.6	22.1	65	1957601280	3853828096	18746
4	high_load	2393	896940	15.6	21.4	65	1875693568	3854958592	13879.4
5	high_load	1651	706510	15.6	20.1	63	1991970816	3855572992	7429.5
6	high_load	2143	750560	15.8	21.1	67	1820758016	3853565952	11357.9
7	high_load	2004	768150	15.6	21.1	65	1923149824	3853746176	11022
8	high_load	874	757590	15.7	19	67	2015973376	3854741504	2884.2
9	high_load	895	763700	15.7	22.3	71	2003722240	3854499840	5907
10	high_load	1108	793430	15.6	20.7	66	2041360384	3855368192	5650.8
	avg	1600.9	772564	15.64	20.82	66.4	1974149939	3854487552	8292.66

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	1962	427740	15.5	19.4	66	1547517952	3854073856	7651.8
2	med_load	940	399140	15.4	20.6	65	1733894144	3854778368	4888
3	med_load	2738	398150	15.4	19.4	62	1468395520	3854258176	10952
4	med_load	2799	433950	15.3	20.1	61	1522114560	3853647872	13435.2
5	med_load	1746	415630	15	20.2	61	1650307072	3852787712	9079.2
6	med_load	847	374960	15.2	19.8	63	1746792448	3854340096	3896.2
7	med_load	981	398760	15.3	20	64	1764696064	3854323712	4610.7
8	med_load	605	405340	15.3	19.9	59	1738661888	3855671296	2783
9	med_load	693	406130	15.4	20	64	1751662592	3854684160	3187.8
10	med_load	702	412040	15.5	22.1	62	1827090432	3853160448	4633.2
	avg	1401.3	407184	15.33	20.15	62.7	1675113267	3854172570	6754.27

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	837	91390	15.4	20.4	63	1305915392	3853594624	4185
2	low_load	588	92520	15.8	20.1	66	1248583680	3854196736	2528.4
3	low_load	412	91750	15.9	20.7	63	1533104128	3854991360	1977.6
4	low_load	622	93700	15.6	18.2	62	1387307008	3854594048	1617.2
5	low_load	344	84350	15.7	20.1	63	1513349120	3854532608	1513.6
6	low_load	356	91070	15.7	23.6	64	1574649856	3853631488	2812.4
7	low_load	758	91370	15.8	19.9	66	1414828032	3854684160	3107.8
8	low_load	1157	112700	15.7	20.4	63	1244336128	3854569472	5437.9
9	low_load	702	108390	15.7	22.4	63	1273712640	3854827520	4703.4
10	low_load	478	90760	15.7	20.2	68	1470275584	3853422592	2151
	avg	625.4	94800	15.7	20.6	64.1	1396606157	3854304461	3064.46

2. Skala Frekuensi F = 1344 MHz

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	1150	835220	15.7	20.7	74	2001362944	3855081472	5750
2	high_load	1132	796700	15.6	21.6	72	2024796160	3851751424	6792
3	high_load	1161	840900	15.6	19.4	72	2047799296	3853549568	4411.8
4	high_load	1137	781580	16	19.6	73	2131357696	3854118912	4093.2
5	high_load	798	802320	15.8	19.9	70	1996591104	3852480512	3271.8
6	high_load	1081	812500	15.7	19.6	68	1955307520	3854188544	4215.9
7	high_load	702	804630	15.7	19.5	68	2016210944	3856187392	2667.6
8	high_load	854	769730	15.6	21.7	67	2103599104	3854422016	5209.4
9	high_load	951	770650	15.7	20.4	73	2173673472	3851862016	4469.7
10	high_load	745	776000	15.7	21.3	68	2128539648	3853668352	4172
	avg	971.1	799023	15.71	20.37	70.5	2057923789	3853731021	4525.326

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	442	407970	15.7	19.1	68	1951547392	3854544896	1502.8
2	med_load	430	419280	15.6	19.5	73	1909227520	3853074432	1677
3	med_load	435	404000	15.6	19.6	75	1930055680	3852926976	1740
4	med_load	452	429620	15.7	21.7	76	1931722752	3851014144	2712
5	med_load	447	408260	15.9	19.7	73	1926127616	3853668352	1698.6
6	med_load	527	421340	15.7	20.4	71	1919299584	3853471744	2476.9
7	med_load	425	410990	15.8	20	68	1946255360	3854114816	1785
8	med_load	739	416400	15.6	20.3	68	1893478400	3853045760	3473.3
9	med_load	520	455920	15.7	21.3	66	1818435584	3853156352	2912
10	med_load	612	403840	15.7	20.2	67	1928347648	3854532608	2754
	avg	502.9	417762	15.7	20.18	70.5	1915449754	3853355008	2252.992

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	201	98560	15.7	19.1	70	1617829888	3853631488	683.4
2	low_load	360	93400	15.8	18.2	65	1520500736	3853029376	864
3	low_load	251	90840	15.8	17.4	65	1405628416	3852214272	401.6
4	low_load	249	91060	15.8	18.8	67	1551085568	3854557184	747
5	low_load	225	88150	15.7	18.8	65	1606291456	3854524416	697.5
6	low_load	215	87340	15.9	18.2	66	1594544128	3853176832	494.5
7	low_load	383	89560	15.7	17.5	64	1522614272	3854692352	689.4
8	low_load	273	97370	15.8	17.1	66	1586049024	3853959168	354.9
9	low_load	263	91500	15.7	17.2	69	1517817856	3854684160	394.5
10	low_load	679	111860	15.8	17.2	65	1294589952	3850588160	950.6
	avg	309.9	93964	15.77	17.95	66.2	1521695130	3853505741	675.58

3. Skala Frekuensi F = 1104 MHz

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	827	887840	14.5	17.5	57	2110914560	3853619200	2481
2	high_load	832	881890	14.5	17.2	56	2069024768	3853176832	2246.4
3	high_load	841	893640	14.5	17.8	56	2067230720	3852742656	2775.3
4	high_load	650	888730	14.5	18.3	56	2112438272	3853430784	2470
5	high_load	714	890060	14.6	18	57	2039316480	3853549568	2427.6
6	high_load	950	890040	14.5	18.1	59	1964130304	3853975552	3420
7	high_load	860	880730	14.5	17.9	58	2114379776	3850874880	2924
8	high_load	800	913370	14.5	17.9	56	2031656960	3854602240	2720
9	high_load	838	898900	14.5	18.4	57	2099470336	3855405056	3268.2
10	high_load	803	893910	14.6	18.3	59	2049789952	3852345344	2971.1
	avg	811.5	891911	14.52	17.94	57.1	2065835213	3853372211	2775.33

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	948	481360	14.5	15.2	57	1858433024	3853041664	663.6
2	med_load	805	510250	14.4	16.8	56	1780404224	3853041664	1932
3	med_load	1034	475540	14.6	16.7	56	1822105600	3855552512	2171.4
4	med_load	976	464020	14.5	17	56	1824849920	3853721600	2440
5	med_load	514	474910	14.5	16.7	59	1783468032	3853676544	1130.8
6	med_load	992	499160	14.6	16.7	56	1794805760	3851653120	2083.2
7	med_load	645	469280	14.5	16.4	59	1883815936	3854958592	1225.5
8	med_load	667	486660	14.5	16.3	57	1818218496	3853656064	1200.6
9	med_load			14.5	16.1	55			0
10	med_load	657	472770	14.4	16.4	60	1854668800	3853815808	1314
	avg	804.222222	481550	14.5	16.43	57.1	1824529977	3853679730	1552.149

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	331	110980	14.5	15.2	59	1426636800	3851640832	231.7
2	low_load	340	113120	14.6	15.1	56	1543258112	3854155776	170
3	low_load	255	109070	14.5	15.3	58	1591365632	3853197312	204
4	low_load	280	106420	14.5	15.2	59	1542406144	3854381056	196
5	low_load	394	113530	14.4	15.5	57	1469026304	3854680064	433.4
6	low_load	409	116630	14.4	16.5	56	1300131840	3854036992	858.9
7	low_load	392	111640	14.5	17.1	56	1298890752	3854041088	1019.2
8	low_load	428	109990	14.5	17.2	59	1545101312	3854217216	1155.6
9	low_load	557	118880	14.5	16.9	59	1425076224	3854647296	1336.8
10	low_load	713	123280	14.5	17	57	1317228544	3853942784	1782.5
	avg	409.9	113354	14.49	16.1	57.6	1445912166	3853894042	659.939

4. Skala Frekuensi F = 720 MHz

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	922	1333460	13.8	16.2	52	2115698688	3853672448	2212.8
2	high_load	1143	1370470	13.8	15.8	51	1955741696	3854385152	2286
3	high_load	911	1353060	13.9	16	50	2131554304	3852337152	1913.1
4	high_load	1216	1385630	14	16.3	51	2100744192	3850768384	2796.8
5	high_load	996	1328770	13.8	16.5	49	2127994880	3853365248	2689.2
6	high_load	899	1300950	13.9	16.1	51	2008567808	3851862016	1977.8
7	high_load	888	1319020	13.9	15.8	51	2012962816	3854299136	1687.2
8	high_load	799	1337610	13.8	16.3	51	2026848256	3852529664	1997.5
9	high_load	934	1330240	13.9	16.5	51	2089390080	3855302656	2428.4
10	high_load	1028	1302500	13.9	16.2	52	2067755008	3852775424	2364.4
	avg	973.6	1336171	13.87	16.08333333	50.9	2063725773	3853129728	2154.901

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	776	730720	13.9	15.4	52	1772244992	3852075008	1164
2	med_load	512	693710	13.9	15.8	52	1944121344	3854082048	972.8
3	med_load	733	695840	13.8	16	51	1864773632	3855568896	1612.6
4	med_load	937	708100	13.9	15.7	52	1859616768	3855216640	1686.6
5	med_load	887	725290	13.9	16.1	54	1761878016	3854127104	1951.4
6	med_load	913	715520	13.9	16.2	49	1740861440	3853357056	2099.9
7	med_load	678	733860	13.9	16.4	50	1747767296	3855372288	1695
8	med_load	731	736170	13.8	16.2	52	1795379200	3852529664	1754.4
9	med_load	667	741330	13.9	15.9	50	1807978496	3855400960	1334
10	med_load	791	721330	13.9	15.7	49	1852698624	3854663680	1423.8
	avg	762.5	720187	13.88	15.94	51.1	1814731981	3854239334	1570.75

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	366	155580	13.8	15.5	51	1473806336	3854942208	622.2
2	low_load	627	151450	13.8	15.3	51	1405640704	3854655488	940.5
3	low_load	335	153800	13.9	15.6	52	1588789248	3853185024	569.5
4	low_load	319	157090	13.8	15.4	51	1576296448	3855179776	510.4
5	low_load	440	158100	13.9	15.6	51	1487187968	3855405056	748
6	low_load	297	148000	13.8	15.1	51	1609756672	3853496320	386.1
7	low_load	301	149660	13.9	15.4	52	1605332992	3852877824	451.5
8	low_load	398	146030	13.9	15.5	52	1497862144	3854831616	636.8
9	low_load	274	151000	13.8	15.7	52	1599680512	3853520896	520.6
10	low_load	353	151550	13.9	15.8	53	1471762432	3853357056	670.7
	avg	371	152226	13.85	15.49	51.6	1531611546	3854145126	608.44

5. Skala Frekuensi F = 480 MHz

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	1764	1983620	13.5	15.3	55	2002001920	3854483456	3175.2
2	high_load	1078	1942910	13.5	15.4	49	2134638592	3854368768	2048.2
3	high_load	1012	1957850	13.6	15.7	49	2068873216	3854782464	2125.2
4	high_load	1463	1985680	13.6	15.6	49	2053169152	3851776000	2926
5	high_load	1050	2008660	13.5	15.2	53	1975697408	3854213120	1785
6	high_load	1107	2034420	13.5	15.7	50	2065428480	3854426112	2435.4
7	high_load	1088	1980260	13.5	15.3	49	2097078272	3854376960	1958.4
8	high_load	1031	1908310	13.5	15.6	51	2239676416	3853836288	2165.1
9	high_load	1114	1911060	13.5	15.5	50	2136403968	3853443072	2228
10	high_load	1042	1966140	13.5	15.8	51	2105327616	3853586432	2396.6
	avg	1174.9	1967891	13.52	15.51	50.6	2087829504	3853929267	2338.051

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (℃)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	665	1030970	13.6	15.7	52	1975066624	3854880768	1396.5
2	med_load	557	1036780	13.5	15.5	49	1866219520	3851862016	1114
3	med_load	756	1014590	13.6	15.8	50	1952550912	3852357632	1663.2
4	med_load	846	1057860	13.5	15.4	48	1795747840	3854860288	1607.4
5	med_load	605	1026120	13.5	15	49	1949933568	3853578240	907.5
6	med_load	896	1004190	13.5	15.5	49	1860825088	3852439552	1792
7	med_load	692	1028210	13.5	15.6	51	1901051904	3851939840	1453.2
8	med_load	682	1039670	13.6	15.2	49	1942990848	3853271040	1091.2
9	med_load	688	1067610	13.5	15.1	49	1757868032	3853025280	1100.8
10	med_load	811	999730	13.6	15.4	49	1991290880	3853119488	1459.8
	avg	719.8	1030573	13.54	15.42	49.5	1899354522	3853133414	1353.22

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	369	233200	13.5	15.5	48	1553530880	3854233600	738
2	low_load	411	238880	13.4	15.1	48	1519570944	3853807616	698.7
3	low_load	289	214050	13.5	15.4	52	1546145792	3852423168	549.1
4	low_load	482	232490	13.5	15.3	53	1517162496	3854241792	867.6
5	low_load	442	225030	13.6	15.4	54	1427832832	3854147584	795.6
6	low_load	479	227040	13.5	15.6	53	1498394624	3853381632	1005.9
7	low_load	333	214850	13.6	15.1	53	1592721408	3853512704	499.5
8	low_load	352	220340	13.5	15.3	52	1531121664	3853164544	633.6
9	low_load	293	216710	13.6	15.5	53	1478877184	3853488128	556.7
10	low_load	584	235440	13.5	15.3	52	1444065280	3853914112	1051.2
	avg	403.4	225803	13.52	15.35	51.8	1510942310	3853631488	738.222

6. Skenario Powersave (Frekuensi F = 120 MHz)

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	3070	8577260	13.2	14.2	48	2033549312	3854602240	3070
2	high_load	3090	8778880	13.2	14.5	47	2089709568	3853877248	4017
3	high_load	3251	8856230	13.3	14.1	49	2121269248	3852566528	2600.8
4	high_load	3325	8380860	13.4	13.9	48	2003468288	3853864960	1662.5
5	high_load	3012	8625370	13.4	14.3	49	1944596480	3852558336	2710.8
6	high_load	6309	8490710	13.2	14.2	49	2016915456	3853672448	6309
7	high_load	5871	8672070	13.3	14.6	49	1836109824	3856322560	7632.3
8	high_load	3359	8075870	13.4	14.4	50	1898172416	3855028224	3359
9	high_load	4587	8509890	13.4	14.3	49	2041180160	3853479936	4128.3
10	high_load	5095	8300780	13.4	14.4	47	2000650240	3855077376	5095
	avg	4096.9	8526792	13.32	14.29	48.5	1998562099	3854104986	3973.99

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	3414	4797990	13.4	14.5	47	1677287424	3855368192	3755.4
2	med_load	2480		13.3	14.1	46			1984
3	med_load	4071	4716940	13.4	14.4	48	1541992448	3854606336	4071
4	med_load	5228	4894130	13.4	14.3	50	1564139520	3853905920	4705.2
5	med_load	4273	4736300	13.4	14.1	49	1633538048	3855114240	2991.1
6	med_load	3146	4510180	13.3	14.3	50	1560555520	3852443648	3146
7	med_load	2559		13.4	14.1	49			1791.3
8	med_load	2946	4388720	13.5	14	48	1684553728	3851714560	1473
9	med_load	2524	4731940	13.4	14	48	1629270016	3854319616	1514.4
10	med_load	2129	4623590	13.4	14.2	48	1711374336	3855921152	1703.2
	avg	3277	4674973.75	13.39	14.2	48.3	1625338880	3854174208	2654.37

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	1042	977170	13.3	13.8	48	1344749568	3853385728	521
2	low_load	1651	1012460	13.4	13.9	47	1249722368	3852857344	825.5
3	low_load	1496	956660	13.4	13.7	48	1280483328	3851427840	448.8
4	low_load	1717	1055800	13.5	14	50	1191550976	3853815808	858.5
5	low_load	1741	1042100	13.4	13.9	50	1214390272	3853918208	870.5
6	low_load	1067	934250	13.4	14.1	50	1305014272	3854069760	746.9
7	low_load	1723	-	13.4	14.1	50	-	-	1206.1
8	low_load	1508	975400	13.3	14	50	1351524352	3851649024	1055.6
9	low_load	1261	931120	13.4	13.9	52	1353506816	3855007744	630.5
10	low_load	1116	1039170	13.4	13.9	50	1225728000	3852259328	558
	avg	1432.2	991570	13.39	13.93	49.5	1279629995	3853154532	773.39

7. Skenario Konfigurasi *Default* (frekuensi *interactive*)

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	high_load	4263	925270	15.7	23.3	67	1783283712	3854045184	32398.8
2	high_load	664	757720	15.8	21	63	2039218176	3850919936	3452.8
3	high_load	1197		15.8	21.6				6942.6
4	high_load	2211		15.8	22				13708.2
5	high_load	928		15.8	20				3897.6
6	high_load	2409	838130	15.8	19.6	68	1939308544	3854315520	9154.2
7	high_load	2055	768550	15.8	20	55	1964793856	3853070336	8631
8	high_load	986	802890	15.8	19.9	75	1941970944	3854901248	4042.6
9	high_load	3149	797920	15.8	20.2	67	1929031680	3856039936	13855.6
10	high_load	1462		15.8	20.4				6725.2
	avg	1932.4	815080	15.79	20.8	65.83333	1932934485	3853882027	9681.324

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	med_load	435	389050	15.4	19.8	63	1825673216	3855142912	1914
2	med_load	823		15.6	20.2				3785.8
3	med_load	927		15.5	18.5				2781
4	med_load	1086		15.2	19.2				4344
5	med_load	1161		15.5	20.4				5688.9
6	med_load	1355	433510	15.6	19.3	58	1774686208	3853053952	5013.5
7	med_load	738	394590	15.7	19	65	1796526080	3853377536	2435.4
8	med_load	544	385080	15.6	19.6	54	1888481280	3854221312	2176
9	med_load	576	407670	15.6	19.7	58	1826992128	3853705216	2361.6
10	med_load	1017	467640	15.7	19.8	57	1676533760	3852296192	4169.7
	avg	866.2	412923.3333	15.54	19.55	59.16667	1798148779	3853632853	3473.462

No	Work Load	Job Time Elapsed (s)	CPU Time Spent (ms)	Power Usage (Idle) (Watt)	Power Usage (Load) (Watt)	Temp (°C)	Pmem Used (byte)	Vmem Used (byte)	Energy (Joule)
1	low_load	1409	95960	15.8	18.1	57	1499791360	3854540800	3240.7
2	low_load	1701		15.5	19.7				7144.2
3	low_load	1697		15.6	18.7				5260.7
4	low_load	1764		15.2	18.1				5115.6
5	low_load	1553		15.1	19.1				6212
6	low_load	1874	127640	15.2	18	53	1122971648	3851907072	5247.2
7	low_load	2263	126440	15.2	19.6	62	1119907840	3852046336	9957.2
8	low_load	1444	130410	16.1	19.7	64	1163489280	3854233600	5198.4
9	low_load	1388	122830	15.2	20.1	57	1310666752	3855765504	6801.2
10	low_load	1426	121130	15.7	20.3	60	1129979904	3851907072	6559.6
	avg	1651.9	120735	15.46	19.14	58.83333	1224467797	3853400064	6078.992