Analisa Komputasi Paralel Mengurutan Data Dengan Metode *Radix* Dan *Selection*

Favorisen R. Lumbanraja, Aristoteles, Nadila Rizqi Muttaqina

Jurusan Ilmu Komputer FMIPA Universitas Lampung, Bandar Lampung 35145

**E-mail*: nadila.rizqi1135@students.unila.ac.id

Abstract - Increasing computing power is now achieved by replacing the programming paradigm with parallel programming. Parallel computing is a method of solving problems by dividing the computational load into small parts of the computation sub-process. This study describes the comparative analysis of parallel computations in the Selection Sort and Radix Sort algorithms. The data used are in the form of whole numbers and decimal numbers totaling 100 to 2 million data. The test was carried out with three scenarios, namely using two processors, four processors, and 3 computers connected to each other via a LAN network. The results showed that the parallel Selection Sort algorithm for small data was better than the parallel Radix Sort. On the other hand, parallel Radix Sort is better for millions of data than Selection Sort.

Keywords: *parallel computing, serial computing, selection sort, radix sort, grid computing*.

1. PENDAHULUAN

Pada tahun 1965, Gordon E. Moore salah satu pendiri Intel, membuat prediksi bahwa kompleksitas sebuah mikroporsesor akan mengalami peningkatan dua kali lipat setiap dua tahun sekali. Namun saat ini yang terjadi *clock rate* tidak meningkat sebanyak sebelumnya dikarenakan keterbatasan dari material-material penyusun *chip* itu sendiri.

Penyediaan sumber daya komputasi yang *powerful* didorong oleh kebutuhan pengaksesan informasi kapanpun dan dimanapun. Peningkatan kekuatan komputasi saat ini didapat dengan mengganti paradigma pemrograman menjadi pemrograman paralel dengan cara memecah proses menjadi sub-proses yang bisa memaksimalkan kapasitas *core* dan *memory*. Para peneliti telah menggunakan dua pendekatan di bidang komputasi berkinerja tinggi yaitu *supercomputer* dan *multicomputer*[1)].

Supercomputer yaitu membangun sebuah komputer dengan kemampuan perhitungan dan kapasitas tinggi dibandingkan dengan komputer biasa[2)]. Pendekatan ini umumnya berharga sangat mahal yang dapat dimiliki oleh segelintir pihak saja, namun akan menghasilkan sebuah komputer yang berkinerja tinggi. Sedangkan *multicomputer* yaitu membangun sebuah sistem yang terdiri dari beberapa komputer independen dihubungkan oleh jaringan telekomunikasi atau jaringan komputer[3)]. Pendekatan kedua ini terdiri dari beberapa komputer yang tergabung dan terkonfigurasi perangkat lunak yang digunakan dan menghasilkan kinerja yang bervariasi tergantung jumlah komputernya, dengan harga yang lebih terjangkau dibandingkan *supercomputer*[4)].

Pengolahan data yang baik sangat diperlukan dalam pengaksesan data yang baik, kuat, dan cepat pula. Mengenai pengolahan data, pengurutan data memegang peranan penting yang banyak dipertimbangkan agar keseluruhan permasalahan menjadi lebih baik dan cepat untuk diselesaikan sehingga menghasilkan data yang akurat[5)].

Pengurutan (*sorting*) merupakan suatu proses mengurutkan data dengan proses yang terjadi yaitu perbandingan data dan pertukaran data berdasarkan suatu aturan tertentu, sehingga tersusun secara teratur sesuai dengan aturan tersebut[6]. Pada dasarnya ada dua macam aturan pengurutan yang biasa digunakan yaitu *ascending* (pengurutan data terkecil ke terbesar) dan *descending* (pengurutan data terbesar ke terkecil)[7].

2. METODOLOGI

a. Komputasi Paralel

Komputasi paralel adalah metode komputasi untuk menyelesaikan permasalah komputasi dengan membagi beban komputasi ke dalam beberapa bagian kecil sub proses komputasi, dimana sub komputasi tersebut dijalankan pada prosesor yang berbeda secara bersamaan dan saling berinteraksi satu sama lain[8]. Tujuan dari komputasi paralel adalah meningkatkan kinerja komputer dalam menyelesaikan berbagai masalah dengan membagi sebuah masalah besar ke dalam beberapa masalah kecil, sehingga hal tersebut membuat kinerja menjadi cepat[9].

Peningkatan kecepatan dalam komputasi paralel menunjukkan seberapa cepat proses komputasi paralel dibandingkan dengan komputasi serial. Pada dasarnya, peningkatan kecepatan (*Speed Up*) dan Efisiensi dirumuskan pada persamaan (1)*Speed Up* dan (2)Efisiensi berikut ini.

$$S = \frac{TS}{TP} \quad (1)$$

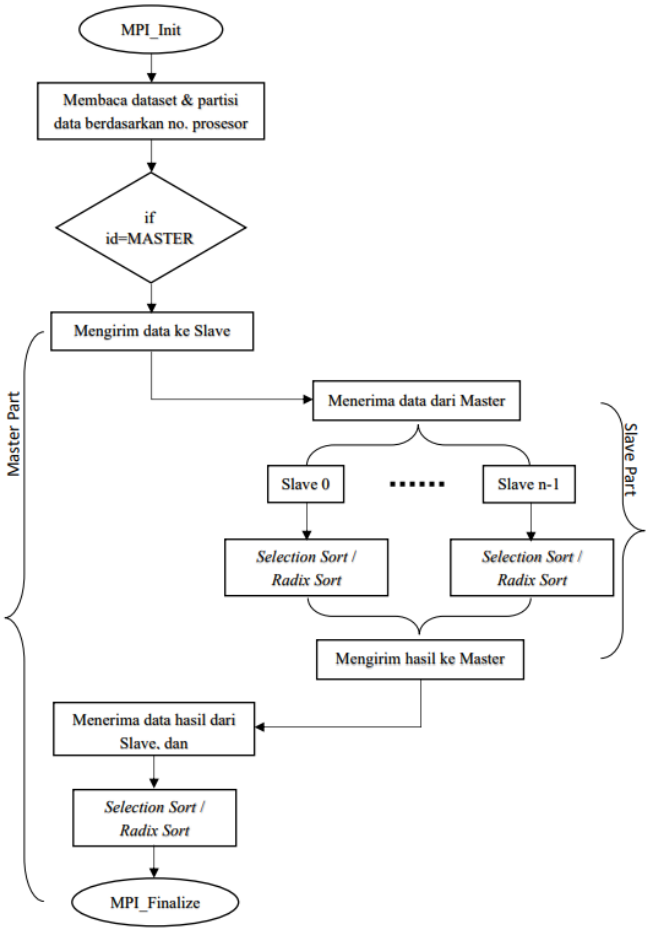
$$E = \frac{S}{P} \quad (2)$$

Keterangan:
TS : Waktu eksekusi dari komputasi serial
TP : Waktu eksekusi dari komputasi paralel menggunakan sebanyak P (prosesor)
S : Peningkatan kecepatan (*Speed Up*)
E : Efisiensi

b. Message Passing Interface (MPI)

Implementasi standar dari model ”*message passing*” komputasi yaitu *Message Passing Interface* (MPI). MPI adalah sebuah komputasi paralel terdiri dari sejumlah proses yang masing-masing bekerja pada beberapa data lokal[10]. Setiap proses mempunyai variable lokal, dan tidak ada mekanisme memori lain bisa diakses secara langsung oleh suatu proses. *Message Passing Interface* mempunyai tujuan yaitu untuk menyediakan standar pemakaian secara luas untuk menulis program pertukaran pesan[11].

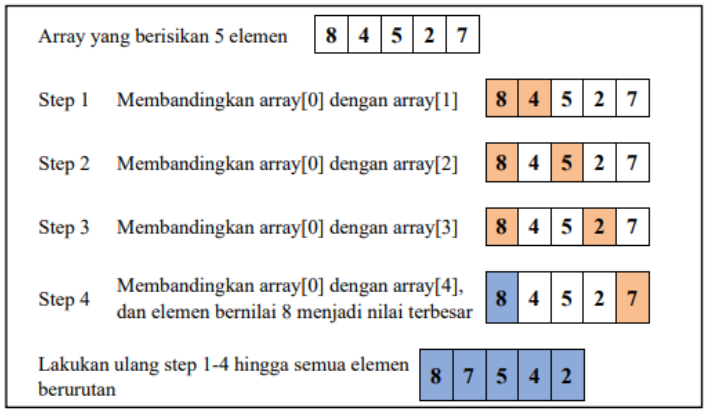
Message Passing Interface (MPI) adalah suatu spesifikasi *library* pemrograman untuk meneruskan pesan (*message-passing*), yang diajukan sebagai standar oleh berbagai komite dari vendor, pelaksana, dan pemakai. Suatu proses (*process*) merupakan sebuah pencacah program dan ruang alamat yang dapat memiliki banyak *thread* (pencacah program dan memori lokal) yang saling berbagi ruang alamat. Dalam hal ini, MPI berfungsi sebagai alat komunikasi di antara proses yang saling memiliki ruang terpisah terutama komunikasi berupa sinkronisasi dan perpindahan data antar proses. Konsep MPI yang digunakan dapat dilihat pada Gambar 1.



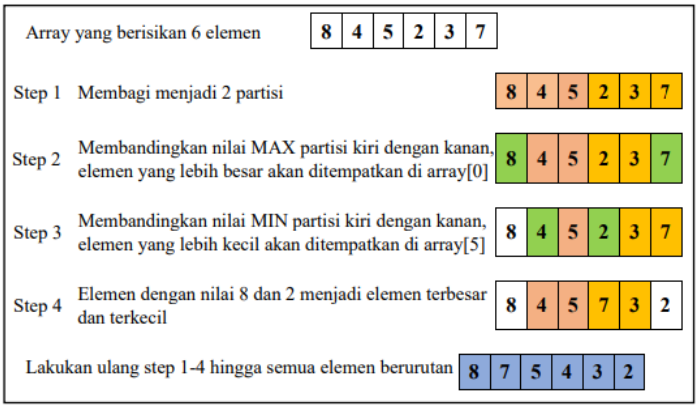
Gambar 1. Konsep MPI

c. Algoritma Selection Sort

Selection Sort merupakan suatu metode pengurutan yang membandingkan elemen paling kiri (elemen awal) dengan elemen-elemen berikutnya sampai dengan elemen yang terakhir pada setiap tahapnya[12]. Pada saat membandingkan, jika terdapat elemen lain yang lebih kecil (pengurutan *ascending*) atau lebih besar (pengurutan *descending*) dari elemen awal maka dicatat posisinya dan langsung ditukar. Proses pengurutan dilakukan terus hingga tahap terakhir dan tidak ada lagi pertukaran data. Kompleksitas dari algoritma *Selection Sort* adalah $O(n^2)$[13]. Ilustrasi algoritma *Selection Sort* dapat dilihat pada Gambar 2 dan Gambar 3.



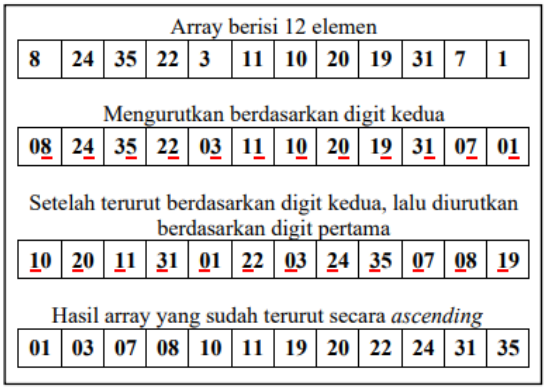
Gambar 2. Ilustrasi algoritma *Selection Sort* secara serial



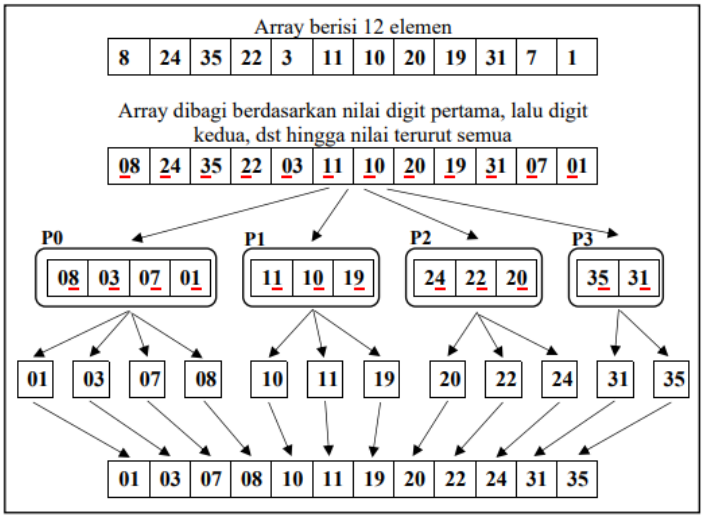
Gambar 3. Ilustrasi algoritma *Selection Sort* secara paralel

d. Algoritma Radix Sort

Radix Sort adalah metode pengurutan yang mengurutkan nilai tanpa melakukan perbandingan pada data yang dimasukkan[14]. Terdapat 2 jenis *Radix Sort* yaitu *Least Significant Digit* (LSD) dan *Most Significant Digit* (MSD). MSD bekerja dengan cara mengurutkan nilai-nilai input berdasarkan *radix* (digit) pertama, lalu pengurutan dilakukan lagi berdasarkan *radix* kedua, begitu seterusnya hingga data terurut. Sebaliknya, LSD berdasarkan digit terakhir hingga pertama. Kompleksitas algoritma *Radix Sort* yaitu O(n+kb) atau O(n)[14]. Simulasi algoritma *Radix Sort* ditunjukkan pada Gambar 4 dan Gambar 5.



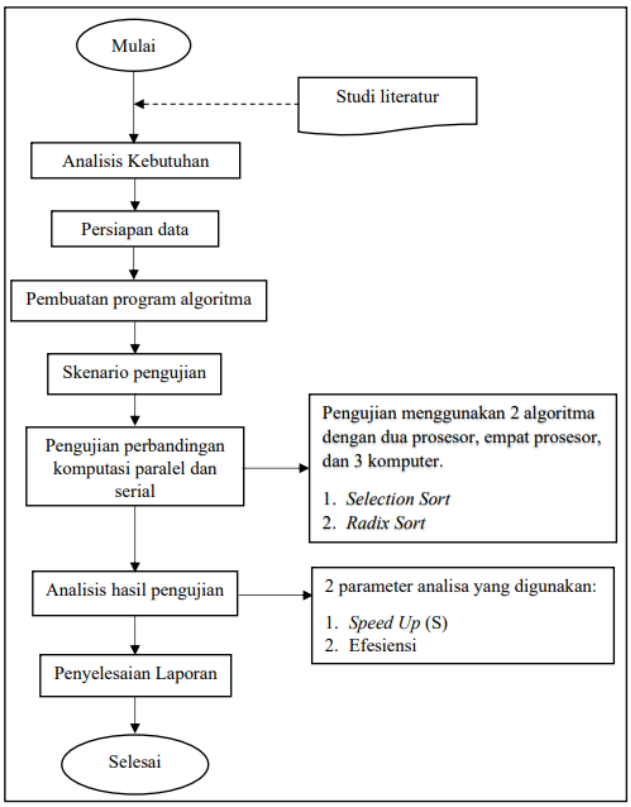
Gambar 4. Ilustrasi algoritma *Radix Sort* secara serial



Gambar 5. Ilustrasi algoritma *Radix Sort* secara paralel

e. Tahapan Penelitian

Tahapan penelitian yang dilakukan yaitu studi literatur, analisis kebutuhan, persiapan data, pembuatan program, pelaksanaan pengujian, analisis hasil pengujian, dan yang terakhir penyelesaian laporan. Adapun lebih jelasnya tahapan penelitian ditunjukkan dalam bentuk *flowchart* pada Gambar 6.



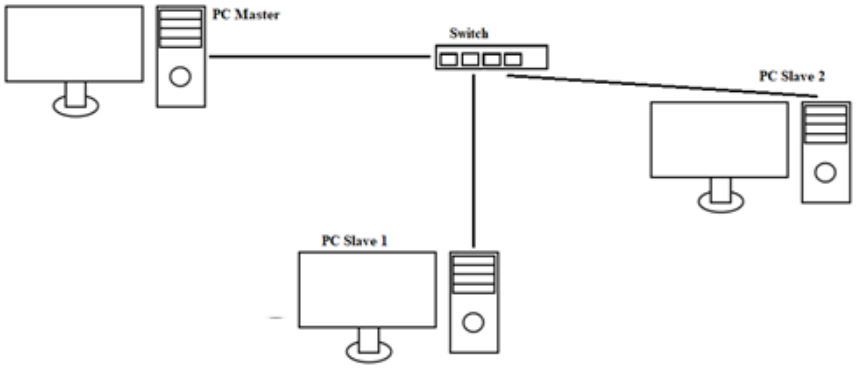
Gambar 6. *Flowchart* tahapan penelitian

i. Studi Literatur

Tahapan ini yaitu melakukan pencarian terhadap berbagai sumber tertulis, baik berupa buku-buku, arsip, majalah, artikel, jurnal, paper, atau dokumen-dokumen yang relevan dengan permasalahan komputasi paralel serta algoritma *Selection Sort* dan *Radix Sort*. Tujuan dari tahapan ini adalah memperoleh referensi teori serta penelitian yang telah dilakukan sebelumnya, sehingga dapat dijadikan rujukan untuk memperkuat argumentasi-argumentasi yang ada.

ii. Analisis Kebutuhan

Sebelum melakukan analisis dan pengujian perlu adanya suatu rumusan dan perencanaan yang jelas, sehingga dapat ditentukan tujuan dari pengujian yang dilakukan. Penelitian ini dilakukan dengan 2 pengujian, pertama pengujian menggunakan satu komputer dengan banyak prosesor, dan kedua pengujian menggunakan tiga komputer yang saling terhubung menggunakan jaringan LAN (*Local Arena Network*) dengan kata lain disebut *grid computing* (komputasi grid). Desain rancangan sistem grid yang dibuat dapat dilihat pada Gambar 7.



Gambar 7. Desain rancangan sistem *grid computing*

Pada perancangannya, tiap-tiap komputer akan diinstal beberapa perangkat lunak yang digunakan untuk membangun sistem komputasi grid. Sebagai perinciannya dijelaskan sebagai berikut.

a. Komputer master akan diinstal sistem operasi LINUX, Net-Tools, SSH (*Secure Shell*), NFS Server (*Network File System*), teks editor Gedit, perekam layar Kazam, dan OpenMPI sebagai API untuk menjalankan pemrosesan paralel.
b. Komputer slave akan diinstal sistem operasi LINUX, Net-Tools, SSH (*Secure Shell*), NFS Client (*Network File System*), teks editor Gedit, perekam layar Kazam, dan OpenMPI sebagai API untuk menjalankan pemrosesan paralel.

iii. Persiapan Data

Data yang digunakan adalah data bilangan bulat dan bilangan desimal yang disimpan di dalam file teks (.txt) dengan masing-masing jumlah data yang ingin diuji. Jumlah data berisi ratusan, ribuan, puluh ribuan, hingga jutaan.

iv. Pembuatan Program Algoritma

Program algoritma dibuat menjadi empat program yaitu program algoritma *Selection Sort* untuk bilangan bulat, *Selection Sort* untuk bilangan desimal, *Radix Sort* untuk bilangan bulat, serta *Radix Sort* untuk bilangan desimal.

v. Skenario Pengujian

Pengujian dilakukan dengan membandingkan nilai *Speed Up* dan Efisiensi dari sejumlah prosesor dengan banyaknya data yang diurutkan. Pengujian juga dilakukan dengan memperlihatkan hasil pengujian dengan menggunakan satu komputer dan 3 komputer. Pada pengujian satu komputer dilakukan dengan melakukan simulasi terhadap dua prosesor dan empat prosesor, sedangkan pengujian dengan tiga komputer menggunakan 12 prosesor.

Pengujian dengan tiga komputer ini, setiap PC terinstal sistem operasi Linux dengan *dual boot* yang sistem operasi utamanya yaitu Windows 10. Adapun skenario pengujian yang dilakukan dapat dilihat pada Tabel 1 sebagai berikut.

Tabel 1. Skenario pengujian

No	Jumlah Prosesor (p)	Banyak Data (n)	Parameter Analisa	
1.	p = 2	100, 1000, 10000, 50000, 100000, 5000000, 1000000, 2000000.	*Speed Up* (S)	Efisiensi (E)
2.	p = 4			
3.	3 Komputer, p = 12			

vi. Pengujian

Tahapan ini merupakan pelaksanaan pengujian yang membandingkan antara komputasi secara paralel dengan serial. Sebelum melakukan pengujian, komputer yang digunakan akan dilakukan penginstalan serta konfigurasi agar saling terhubung melalui jaringan LAN. Tahapan penginstalan komputer sebagai berikut.

a. Menginstal sistem operasi Linux Ubuntu versi 20.04 LTS.
b. Menginstal SSH (*Secure Shell*) yang berfungsi agar komputer master dapat mengakses komputer slave A dan slave B melalui koneksi yang terenkripsi.
c. Menginstal NFS (*Network File System*) yang berfungsi untuk *sharing* file dari komputer master ke komputer slave.
d. Menginstal OpenMPI yang berfungsi sebagai *library* program paralel.

Pengujian terhadap penelitian ini difokuskan pada dua aspek yaitu Efisiensi dan *Speed Up* yang merupakan nilai yang diperoleh dari pembagian waktu proses komputasi serial dengan waktu proses komputasi paralel. Efisiensi pada pengujian ini adalah pembagian antara nilai *Speed Up* dengan banyaknya prosesor yang digunakan.

vii. Analisis Hasil Pengujian

Tahap ini yaitu menganalisis dengan menghitung nilai *Speed Up* dan Efisiensi dari pemrograman secara serial dan paralel. Waktu eksekusi secara serial dan paralel ini yang akan dijadikan sebagai parameter untuk mendapatkan nilai *Speed Up* dan Efisiensi algoritma *Selection Sort* dan *Radix Sort*. Setelah mendapatkan nilai *Speed Up* dan Efisiensi, selanjutnya yaitu membandingkan nilai tersebut dari ketiga skenario pengujian yang telah dilakukan.

viii. Penyelesaian Laporan

Penulisan laporan ditujukan untuk dokumentasi seluruh kegiatan penelitian analisis perbandingan komputasi paralel dan serial pada algoritma *Selection Sort* dan *Radix Sort* di Laboratorium RLP Jurusan Ilmu Komputer Universitas Lampung.

3. PEMBAHASAN

a. Pengujian

Pada penelitian ini pengujian dilaksanakan selama 8 hari dari tanggal 7 September 2020 hingga 18 September 2020 di Laboratorium Rekayasa Perangkat Lunak (RPL) Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Detail tahapan pelaksanaan pengujian sebagai berikut.

1. Menginstall Sistem Operasi (SO) Linux Ubuntu versi 20.04 LTS di komputer ke-1 sebagai PC Master dan komputer ke-2 serta ke-3 sebagai PC Slave A dan B.
2. Menginstall SSH (Secure Shell), NFS (Network File System), dan OpenMPI.
3. *Running* program algoritma *Selection Sort* dan *Radix Sort*.
4. Mencatat waktu eksekusi baik secara paralel maupun serial.
5. Hitung *Speed Up* dan Efisiensi dari masing-masing data.

Tabel 2. Spesifikasi komputer

Komponen	Spesifikasi
Prosesor	Intel I7-8550U 1.8
RAM	8 GB
Sistem Operasi	Linux

Tabel 3. Kegiatan pengujian

No	Tanggal	Hari	Kegiatan
1.	7 September 2020	Senin	Menginstall Sistem Operasi (SO) Linux Ubuntu versi 20.04 LTS di 3 komputer (*dual boot*).
2.	8 September 2020	Selasa	Menginstall SSH, NFS, dan OpenMPI.
3.	9 September 2020	Rabu	• *Running* program algoritma *Selection Sort* secara serial maupun paralel. • Mengubah coding program algoritma *Selection Sort* karena terdapat kejanggalan dari hasil *running time* yang sudah dilakukan. • *Running* ulang program algoritma *Selection Sort*.
4.	14 September 2020	Senin	• *Running* data bilangan bulat dan decimal menggunakan algoritma *Selection Sort*. • *Running* program algoritma *Radix Sort*.
5.	15 September 2020	Selasa	• *Running* data algoritma *Selection Sort.* • Mengubah coding program dari algoritma *Radix Sort* karena sebelumnya terdapat masalah. • *Running* data 7 juta untuk algoritma *Selection Sort* menggunakan 3 komputer (komputer dibiarkan hidup).
6.	16 September 2020	Rabu	• *Running* data 7 juta untuk algoritma *Selection Sort* menggunakan 3 komputer belum selesai hingga sore hari.
7.	17 September 2020	Kamis	• *Running* data 7 juta tidak kunjung selesai hingga siang hari. • Melaporkan kondisi *running* data 7 juta yang juga belum selesai hingga dua hari kepada pak Aristoteles. • Memutuskan pengurangan data yang diuji menjadi maksimal 2 juta saja. • *Running* data algoritma *Selection Sort* dan *Radix Sort*.
8.	18 September 2020	Jumat	• Menyelesaikan *running* data yang belum.

b. Hasil Pengujian

Pada tahap ini akan disampaikan mengenai hasil pengujian dan analisa keempat program yakni pengurutan data bilangan bulat dan bilangan desimal dengan menggunakan algoritma *Selection Sort* dan *Radix Sort* baik yang dilakukan secara serial maupun secara paralel. Pengujian dilakukan menggunakan suatu skenario pengujian yang dapat membandingkan hasil dari keempat program tersebut.

Hasil pengujian dilakukan dengan menghitung nilai *Speed Up* dan Efisiensi penggunaan pemrograman secara serial dan paralel. Waktu eksekusi secara serial dan paralel ini yang akan dijadikan sebagai parameter untuk mendapatkan nilai *Speed Up* dan Efisiensi algoritma *Selection Sort* dan *Radix Sort*.

i. Pengujian Algoritma *Selection Sort* (p = 2)

Hasil pengujian dari algoritma *Selection Sort* menggunakan dua prosesor dengan menguji data bilangan bulat dan bilangan desimal dapat dilihat pada Tabel 4.

Tabel 4. Hasil pengujian algoritma *Selection Sort* dengan dua prosesor

Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)

Bilangan bulat	100	0.000203	0.000076	0.37438424	0.18719212
	1000	0.006614	0.005142	0.77744179	0.3887209
	10000	0.132795	0.135117	1.0174856	0.5087428
	50000	3.309182	2.95984	0.89443252	0.44721626
	100000	13.545971	11.834191	0.8736318	0.4368159
	500000	345.072925	321.197037	0.93080915	0.46540458
	1000000	1382.66316	1318.4576	0.95356385	0.47678192
	2000000	5773.86105	5282.46807	0.91489352	0.45744676
	Rata-rata	939.823988	867.132135	0.84208031	0.42104015
Bilangan desimal	100	0.000183	0.000075	0.40983607	0.20491803
	1000	0.006609	0.005154	0.77984567	0.38992283
	10000	0.130703	0.132766	1.01578388	0.50789194
	50000	3.152981	2.811933	0.89183316	0.44591658
	100000	13.215738	11.299437	0.85499856	0.42749928
	500000	339.485791	315.334428	0.92885899	0.46442949
	1000000	1408.70096	1299.62982	0.92257325	0.46128662
	2000000	6134.89913	5548.95054	0.90448929	0.45224464
	Rata-rata	987.449011	897.270518	0.83852736	0.41926368

ii. Analisis Pengujian Algoritma Selection Sort dengan Dua Prosesor

Pada Tabel 4 dapat dilihat bahwa untuk jumlah data 10.000 waktu komputasi secara paralel lebih, tetapi untuk data lainnya dari 100 hingga 2.000.000 waktu komputasi secara serial lebih kecil. *Speed Up* mengalami peningkatan dan penurunan seiring bertambahnya jumlah data, seperti peningkatan nilai *Speed Up* dari data 1.000 ke 10.000, namun menurun pada data 50.000. Terlihat bahwa nilai *Speed Up* tidak konstan naik atau turun. Hal ini berarti bahwa Algoritma *Selection Sort* dengan dua prosesor tidak cocok untuk komputasi secara paralel, terbukti dari hasil *Speed Up* yang didapat hampir keseluruhan masih menunjukkan nilai dibawah 1 yang berarti waktu komputasi secara serial masih lebih kecil dibandingkan waktu komputasi secara paralel.

iii. Pengujian Algoritma Selection Sort (p = 4)

Hasil pengujian dari algoritma *Selection Sort* menggunakan empat prosesor dengan menguji data bilangan bulat dan bilangan desimal dapat dilihat pada Tabel 5.

Tabel 5. Hasil pengujian algoritma *Selection Sort* dengan empat prosesor

Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)
Bilangan bulat	100	0.000108	0.000041	0.37962963	0.09490741
	1000	0.002249	0.002545	1.13161405	0.28290351
	10000	0.115987	0.149094	1.28543716	0.32135929
	50000	2.872051	3.746832	1.30458408	0.32614602
	100000	12.554171	15.460428	1.23149732	0.30787433
	500000	316.334332	429.446747	1.35757236	0.33939309
	1000000	1246.92713	1742.66552	1.39756805	0.34939201
	2000000	4945.62066	6692.44327	1.35320594	0.33830149
	Rata-rata	815.553336	1110.48931	1.18013858	0.29503464
Bilangan desimal	100	0.000108	0.000066	0.61111111	0.15277778
	1000	0.002679	0.002563	0.95670026	0.23917507
	10000	0.116352	0.143129	1.23013786	0.30753446
	50000	2.855971	3.536559	1.23830354	0.30957589
	100000	11.974052	14.445506	1.20640081	0.3016002
	500000	304.608218	411.902765	1.35223786	0.33805947

	1000000	1259.19215	1678.07175	1.33265741	0.33316435
	2000000	5420.82996	6975.38304	1.286774	0.3216935
	Rata-rata	874.947435	1135.43567	1.15179036	0.28794759

iv. Analisis Pengujian Algoritma Selection Sort dengan Empat Prosesor

Pada Tabel 5 dapat dilihat bahwa untuk jumlah data di atas 1000 waktu komputasi secara paralel lebih kecil dibandingkan secara serial. Terdapat peningkatan nilai *Speed Up* dari data 10.000 ke 50.000, namun mengalami penurunan pada data 100.000. Dengan bertambahnya jumlah data, nilai *Speed Up* pada algoritma *Selection Sort* dengan empat prosesor cenderung tidak konstan, tetapi masih menghasilkan nilai *Speed Up* lebih dari 1. Bila dibandingkan dengan menggunakan dua prosesor, rata-rata nilai *Speed Up* dari penggunaan empat prosesor adalah sebesar 1,18013858 (data bilangan bulat) dan 1,15179036 (data bilangan desimal), lebih besar dara rata-rata nilai *Speed Up* dengan dua prosesor yaitu sebesar 0.84208031 (data bilangan bulat) dan 0,83852736 (data bilangan desimal). Hal tersebut diperkirakan karena dipengaruhi oleh *overhead time* yang lebih besar pada penggunaan dua prosesor dibandingkan dengan empat prosesor. Perbandingan ini menyimpulkan penggunaan empat prosesor algoritma *Selection Sort* udah tepat untuk komputasi secara paralel.

v. Pengujian Algoritma Selection Sort (3 Komputer)

Hasil pengujian dari algoritma *Selection Sort* menggunakan tiga komputer yang saling terhubung dengan total 12 prosesor yang digunakan dengan menguji data bilangan bulat dan bilangan desimal dapat dilihat pada Tabel 6.

Tabel 6. Hasil pengujian algoritma *Selection Sort* dengan tiga komputer (p = 12)

Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)
Bilangan bulat	100	0.030619	0.000016	0.00052255	4.3546E-05
	1000	0.089333	0.002568	0.02874638	0.00239553
	10000	0.192962	0.154462	0.80047885	0.06670657
	50000	2.770201	3.60793	1.3024073	0.10853394
	100000	11.361699	14.713537	1.29501204	0.10791767
	500000	286.754206	424.245462	1.47947424	0.12328952
	1000000	1146.64364	1719.6714	1.49974354	0.12497863
	2000000	4936.88643	7231.38792	1.46476692	0.12206391
	Rata-rata	798.091137	1174.22291	0.98389398	0.08199116
Bilangan desimal	100	0.030035	0.000075	0.00249709	0.00020809
	1000	0.085306	0.005217	0.06115631	0.00509636
	10000	0.195931	0.154523	0.78866029	0.06572169
	50000	2.755546	3.623299	1.31491145	0.10957595
	100000	11.473691	14.800374	1.28994009	0.10749501
	500000	286.710935	423.159648	1.47591039	0.12299253
	1000000	1390.29461	2008.68573	1.44479142	0.12039928
	2000000	4976.80232	7192.09331	1.44512336	0.12042695
	Rata-rata	833.543547	1205.31527	0.9778738	0.08148948

vi. Analisis Pengujian Algoritma Selection Sort dengan Tiga Komputer

Pada pengujian data Algoritma *Selection Sort* proses pengurutan dengan tiga komputer, terlihat bahwa pada jumlah data 100 hingga 10.000 waktu komputasi secara serial masih lebih kecil. Namun, mulai pada jumlah data 50.000 waktu komputasi secara paralel lebih kecil dari serial. Walaupun terdapat penurunan pada nilai *Speed Up* di beberapa data, tetapi lebih banyak terjadi peningkatan nilai *Speed Up* seiring bertambahnya jumlah data. Hal ini diperkirakan semakin

banyak data yang diproses maka semakin besar pula nilai *Speed Up* yang dihasilkan meskipun masih terdapat penurunan nilai *Speed Up* dibeberapa jumlah data tertentu.

vii. Pengujian Algoritma Radix Sort (p = 2)

Hasil pengujian dari algoritma *Radix Sort* menggunakan dua prosesor dengan menguji data bilangan bulat dan bilangan decimal dapat dilihat pada Tabel 7.

Tabel 7. Hasil pengujian algoritma *Radix Sort* dengan dua prosesor

Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)
Bilangan bulat	100	0.000075	0.000012	0.16	0.08
	1000	0.000813	0.00045	0.55350554	0.27675277
	10000	0.009577	0.005892	0.61522397	0.30761199
	50000	0.014058	0.014289	1.01643192	0.50821596
	100000	0.0248	0.020013	0.80697581	0.4034879
	500000	0.121296	0.085099	0.70158126	0.35079063
	1000000	0.245798	0.164478	0.66915923	0.33457961
	2000000	0.684571	0.470416	0.68716904	0.34358452
	Rata-rata	0.1376235	0.09508113	0.65125585	0.32562792
Bilangan desimal	100	0.000559	0.000316	0.56529517	0.28264758
	1000	0.00106	0.000696	0.65660377	0.32830189
	10000	0.010961	0.008212	0.74920172	0.37460086
	50000	0.055272	0.045618	0.82533652	0.41266826
	100000	0.103651	0.071731	0.69204349	0.34602175
	500000	0.518284	0.357915	0.69057698	0.34528849
	1000000	1.038519	0.716015	0.68945778	0.34472889
	2000000	2.093373	1.446426	0.69095474	0.34547737
	Rata-rata	0.47770988	0.33086613	0.69493377	0.34746689

viii. Analisis Pengujian Algoritma Radix Sort dengan Dua Prosesor

Pada Tabel 7 dapat dilihat bahwa dari jumlah data 100 hingga 2.000.000 hampir keseluruhan waktu komputasi secara serial lebih kecil dibandingkan secara paralel. Terdapat peningkatan dan penurunan nilai *Speed Up* yang didapat, tetapi lebih banyak mengalami penurunan seperti pada data 50.000 ke 100.000. Algoritma *Radix Sort* dengan dua prosesor memiliki persamaan dengan algoritma *Selection Sort* dengan dua prosesor yang menghasilkan nilai *Speed Up* tidak konstan naik atau turun dan cenderung dibawah satu. Hal ini membuktikan bahwa algoritma *Radix Sort* tidak cocok untuk komputasi secara paralel.

ix. Pengujian Algoritma Radix Sort (p = 4)

Hasil pengujian dari algoritma *Radix Sort* menggunakan empat prosesor dengan menguji data bilangan bulat dan bilangan decimal dapat dilihat pada Tabel 8.

Tabel 8. Hasil pengujian algoritma *Radix Sort* dengan Empat Prosesor

Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)
Bilangan bulat	100	0.000089	0.000023	0.25842697	0.06460674
	1000	0.000413	0.000241	0.58353511	0.14588378
	10000	0.003039	0.002351	0.77360974	0.19340244
	50000	0.011395	0.011389	0.99947345	0.24986836
	100000	0.023785	0.028994	1.21900357	0.30475089
	500000	0.106116	0.119522	1.12633345	0.28158336
	1000000	0.213141	0.213789	1.00304024	0.25076006

	2000000	0.595473	0.597152	1.00281961	0.2507049
	Rata-rata	0.11918138	0.12168263	0.87078027	0.21769507
Bilangan desimal	100	0.000602	0.000333	0.55315615	0.13828904
	1000	0.002391	0.002122	0.88749477	0.22187369
	10000	0.011375	0.011365	0.99912088	0.24978022
	50000	0.050139	0.042497	0.84758372	0.21189593
	100000	0.09084	0.092828	1.02188463	0.25547116
	500000	0.454971	0.46041	1.01195461	0.25298865
	1000000	0.917712	0.926766	1.00986584	0.25246646
	2000000	1.845502	1.867933	1.01215442	0.2530386
	Rata-rata	0.4216915	0.42553175	0.91790188	0.22947547

x. Analisis Pengujian Algoritma Radix Sort dengan Empat Prosesor

Pada Tabel 8 dapat dilihat bahwa untuk jumlah data 100 hingga 50.000 waktu komputasi secara serial masih lebih kecil, sedangkan data 100.000 atau lebih menghasilkan waktu komputasi secara paralel lebih kecil. Terjadi peningkatan dan penurunan pada nilai *Speed Up* seiring bertambahnya data dan tidak konstan. Namun, perkiraan semakin banyak jumlah data yang diproses maka nilai *Speed Up* semakin besar pula walaupun tetap terdapat penurunan nilai *Speed Up* dibeberapa data tertentu.

xi. Pengujian Algoritma Radix Sort (3 Komputer)

Hasil pengujian dari algoritma *Radix Sort* menggunakan tiga komputer yang saling terhubung dengan total 12 prosesor yang digunakan dengan menguji data bilangan bulat dan bilangan desimal dapat dilihat pada Tabel 9.

Tabel 9. Hasil pengujian algoritma *Radix Sort* dengan tiga komputer (p = 12)

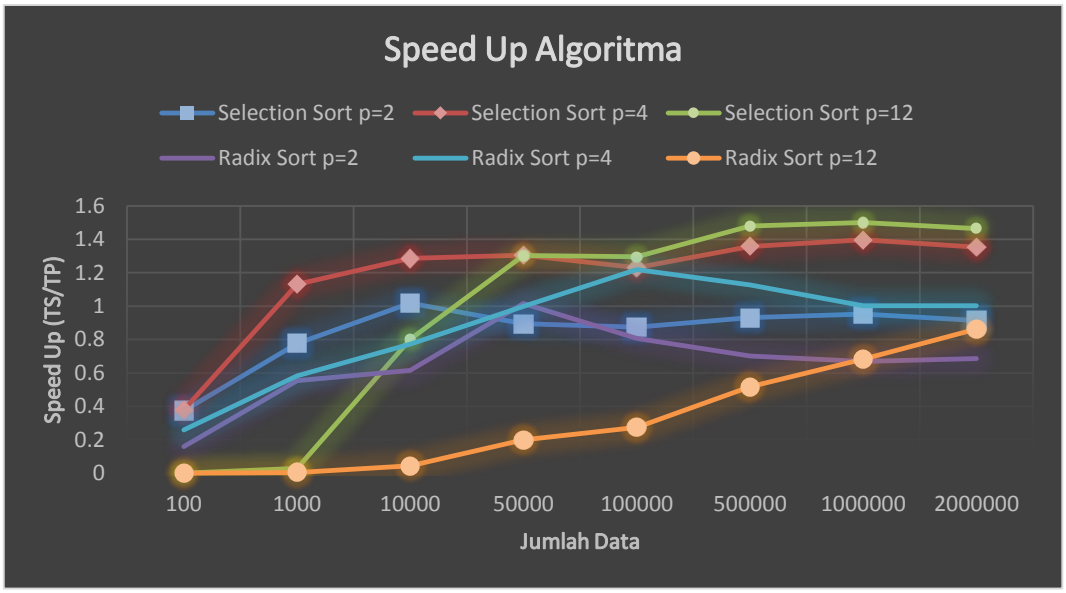
Jenis Data	Banyak Data (n)	(TP)	(TS)	*Speed Up* (S) TS/TP	Efisiensi (S/p)
Bilangan bulat	100	0.030564	0.000017	0.00055621	4.6351E-05
	1000	0.08976	0.000444	0.00494652	0.00041221
	10000	0.087047	0.003895	0.04474594	0.00372883
	50000	0.0948	0.01901	0.20052743	0.01671062
	100000	0.106989	0.029435	0.27512174	0.02292681
	500000	0.215582	0.111616	0.51774267	0.04314522
	1000000	0.316219	0.215722	0.68219177	0.05684931
	2000000	0.799083	0.689044	0.8622934	0.07185778
	Rata-rata	0.2175055	0.13364788	0.32351571	0.02695964
Bilangan desimal	100	0.030334	0.000199	0.0065603	0.00054669
	1000	0.08846	0.002113	0.0238865	0.00199054
	10000	0.10002	0.008593	0.08591282	0.0071594
	50000	0.122803	0.047832	0.38950189	0.03245849
	100000	0.170877	0.095422	0.55842507	0.04653542
	500000	0.50877	0.467506	0.91889459	0.07657455
	1000000	0.922676	0.941322	1.02020861	0.08501738
	2000000	2.020126	2.055131	1.01732813	0.08477734
	Rata-rata	0.49550825	0.45226475	0.50258974	0.04188248

xii. Analisis Pengujian Algoritma Radix Sort dengan Tiga Komputer

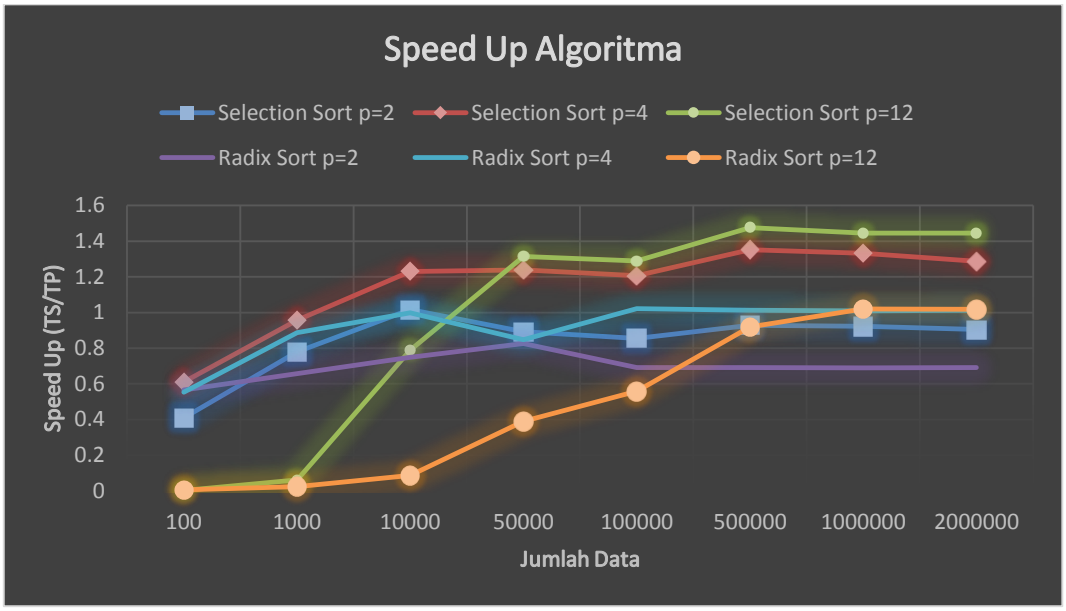
Pada Tabel 9 dapat dilihat bahwa untuk jumlah data 100 hingga 2.000.000 menghasilkan nilai *Speed Up* yang cenderung dibawah satu. Namun, seiring bertambahnya jumlah data nilai *Speed Up* cenderung selalu naik. Hal ini berarti proses paralel yang dilakukan memiliki hasil yang lebih baik apabila data berjumlah sangat banyak dibandingkan penggunaan dua prosesor maupun empat prosesor.

c. Hasil Perbandingan *Speed Up*

Hasil perbandingan *Speed Up* dari keenam skenario pengujian tersebut dapat dilihat dalam bentuk grafik seperti yang dapat dilihat pada Gambar 8 dan Gambar 9 berikut.



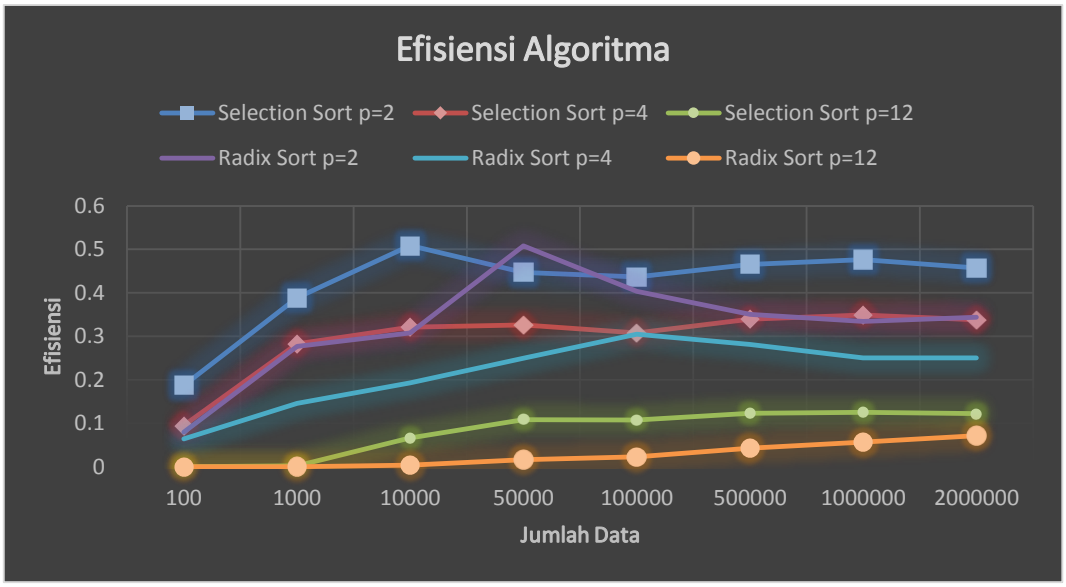
Gambar 8. Grafik perbandingan *Speed Up* algoritma *Selection Sort* dan *Radix Sort* untuk data bilangan bulat



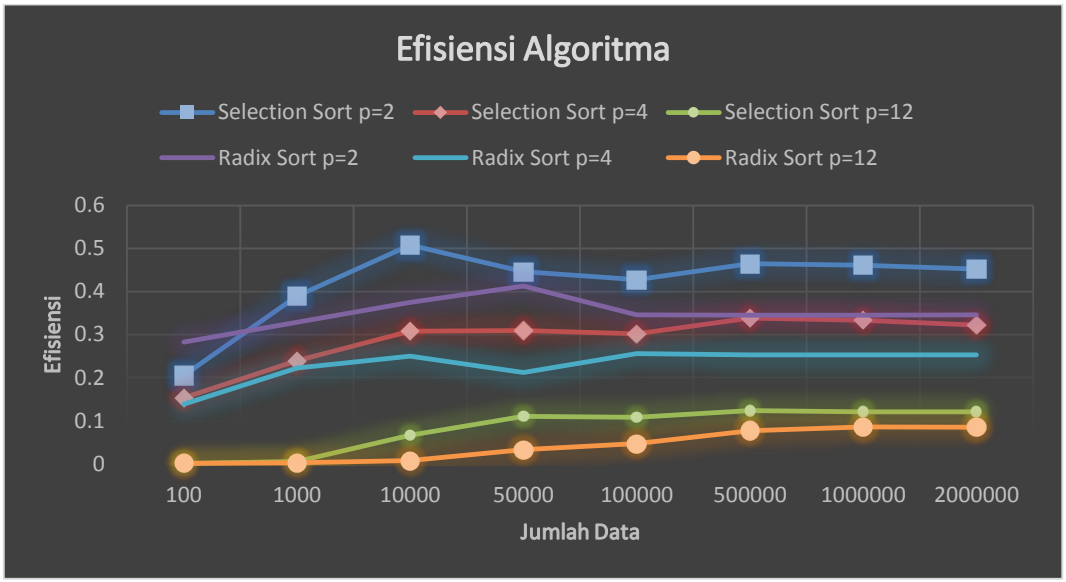
Gambar 9. Grafik perbandingan *Speed Up* algoritma *Selection Sort* dan *Radix Sort* untuk data bilangan desimal

d. Hasil Perbandingan Efisiensi

Hasil perbandingan Efisiensi dari keenam scenario pengujian tersebut dapat dilihat dalam bentuk grafik seperti yang dapat dilihat pada Gambar 10 dan Gambar 11 berikut.



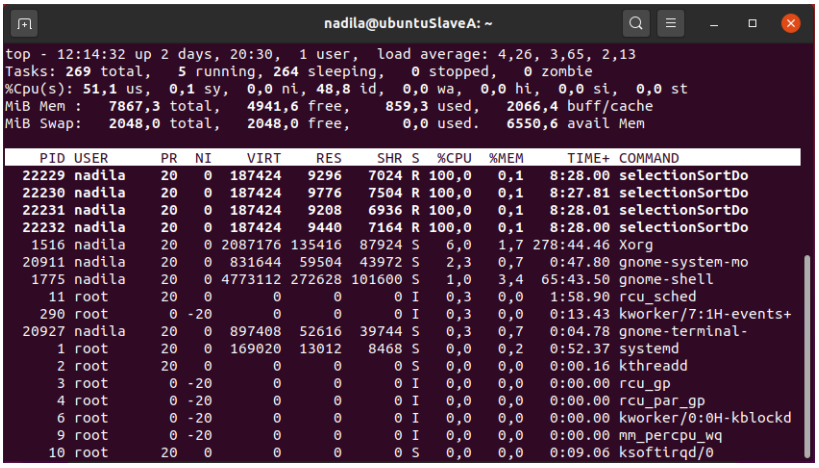
Gambar 10. Grafik perbandingan Efisiensi algoritma *Selection Sort* dan *Radix Sort* untuk data bilangan bulat



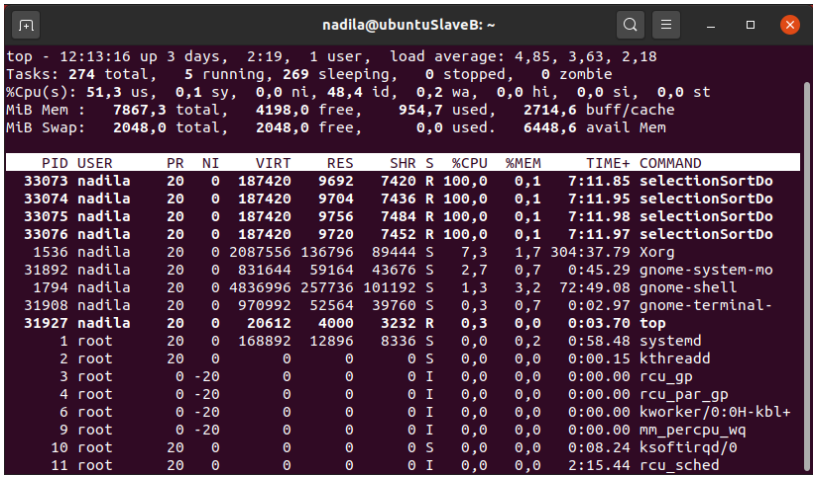
Gambar 11. Grafik perbandingan Efisiensi algoritma *Selection Sort* dan *Radix Sort* untuk data bilangan desimal

e. Hasil Perbandingan Efisiensi

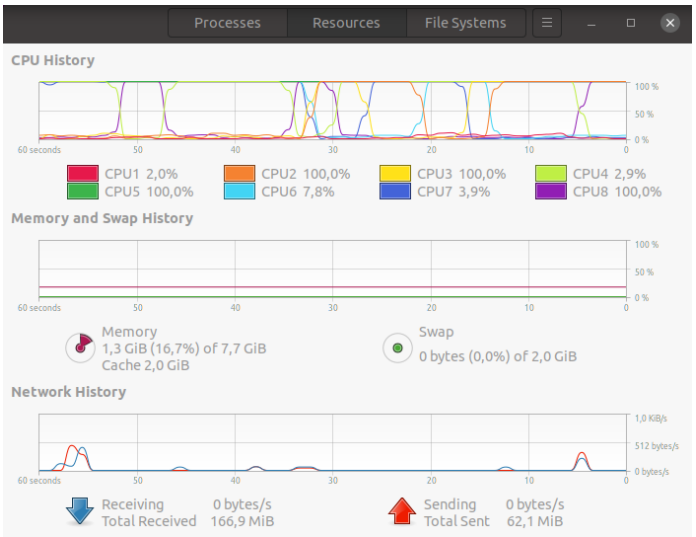
Pada proses pengujian menggunakan tiga komputer yang saling terhubung, penggunaan sumber daya dalam proses komputer Slave A dan Slave B dapat dilihat pada Gambar 12, Gambar 13, Gambar 14, dan Gambar 15.



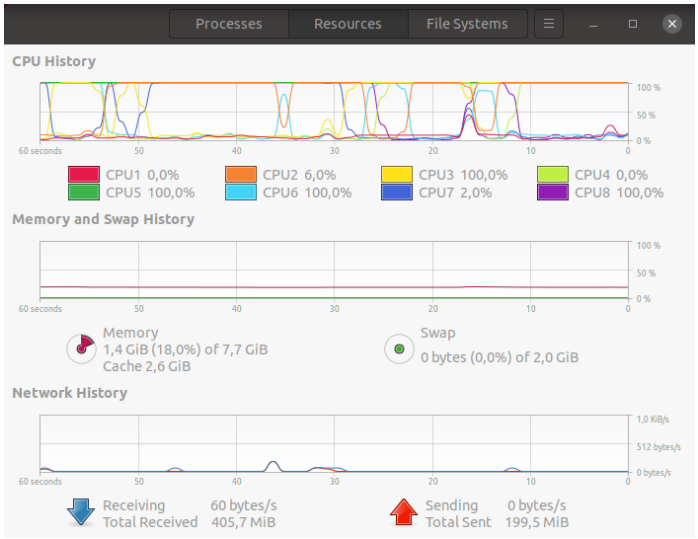
Gambar 12. Tampilan proses monitor dan penggunaan sumber daya sistem di PC Slave A pada saat *running* program data 2.000.000 dengan tiga komputer (12 prosesor)



Gambar 13. Tampilan proses monitor dan penggunaan sumber daya sistem di PC Slave B pada saat *running* program data 2.000.000 dengan tiga komputer (12 prosesor)



Gambar 14. Tampilan sistem monitor penggunaan sumber daya di PC Slave A dalam proses komputasi dengan tiga komputer



Gambar 15. Tampilan sistem monitor penggunaan sumber daya di PC Slave B dalam proses komputasi dengan tiga komputer

4. KESIMPULAN

Berdasarkan hasil pengujian yang dilakukan dapat mengambil simpulan sebagai berikut.

1. Secara umum untuk jumlah data yang tidak terlalu besar, waktu komputasi serial berjalan lebih cepat bila dibandingkan dengan waktu komputasi secara paralel.
2. Algoritma *Selection Sort* maupun *Radix Sort* dengan dua prosesor tidak menghasilkan nilai *Speed Up* baik yang berarti tidak cocok untuk komputasi secara paralel. Hal ini disebabkan karena komunikasi paralel mengalami overhead time yang besar.
3. Algoritma *Selection Sort* dengan empat prosesor menghasilkan nilai *Speed Up* cenderung konstan diatas 1 yang berarti waktu eksekusi komputasi paralel lebih cepat dibandingkan komputasi serial.
4. Algoritma *Radix Sort* dengan empat prosesor untuk jumlah data lebih dari 50.000 terbilang menghasilkan nilai *Speed Up* yang baik. Diperkirakan nilai *Speed Up* akan semakin besar seiring bertambahnya jumlah data, walaupun tetap terdapat penurunan nilai *Speed Up* dibeberapa data tertentu.
5. Algoritma *Selection Sort* dengan tiga komputer menghasilkan nilai *Speed Up* yang hampir sama dengan algoritma *Radix Sort* dengan empat prosesor untuk jumlah data yang relatif kecil waktu komputasi secara serial masih terbilang lebih cepat, tetapi untuk data besar waktu komputasi secara paralel lebih cepat dibandingkan komputasi secara serial.
6. Algoritma *Radix Sort* dengan tiga komputer menghasilkan nilai *Speed Up* cenderung selalu meningkat tanpa mengalami penurunan sama sekali. Namun, dari hasil nilai *Speed Up* rata-rata waktu komputasi secara serial masih terbilang jauh lebih cepat dibandingkan komputasi secara paralel. Disimpulkan bahwa untuk jumlah data dibawah 1.000.000, algoritma *Radix Sort* belum memerlukan penggunaan tiga komputer untuk komputasi secara paralel.
7. Hasil nilai Efesiensi yang didapat dari semua skenario pengujian membuktikan bahwa penggunaan dengan dua prosesor masih tergolong paling efesien dibandingkan empat prosesor dan 3 komputer baik itu algoritma *Selection Sort* maupun *Radix Sort*.
8. Simpulan terakhir, apabila jumlah data yang diproses terbilang kecil sebaiknya hanya menggunakan dua prosesor saja dengan menggunakan komputasi secara serial. Namun, jika data sudah memasuki jutaan sebaiknya menggunakan algoritma *Radix Sort* dengan empat prosesor secara paralel. Dan apabila data yang diproses sudah sangat besar seperti puluhan juta, sebaiknya menggunakan algoritma *Radix Sort* dengan 3 komputer atau lebih secara paralel.

REFERENSI

[1]. Reed D., Fujimoto R.M. 1987. *Multicomputer Networks: Message-Based Parallel Processing*. MIT Press.

[2]. Prianto B. 2008. *Cluster Komputer Sebagai Pengganti Super Komputer Tunggal untuk Pemodelan Kimia Komputasi*. Berita Dirgantara Vol 9 No 1.

[3]. Singh I., Sch. J. Eng. Tech. 2013. *Review on Parallel and Distributed Computing*. Scholars Journal of Engineering and Technology (SJET).

[4]. Yusman M., Aristoteles, & Anie Rose Irawati. 2012. *Analisis Komputasi Paralel dan Serial Pada Algoritma Merge Sort*. Jurnal Sains MIPA Vol 18 No 1.

[5]. Fauzi, Indrayana. 2005. *Perbandingan Kecepatan/Waktu Komputasi Beberapa Algoritma Pengurutan (Sorting)*. Institut Teknologi Bandung.

[6]. Utami E., Raharjo S., & Sukrisno. 2007. *Struktur Data Konsep & Implementasinya dalam Bahasa C & Free Pascal di GNU/LINUX*. Yogyakarta: Graha Ilmu.

[7]. Sitepu R.R., Machudor Y., & Febi E.F. 2017. *Implementasi Algoritma Bubble Sort dan Selection Sort Menggunakan Arraylist Multidimensi Pada Pengurutan Data Multi Prioritas*. Jurnal Komputasi Vol 5 No 1.

[8]. Wibawa I.P.A.P., Dwi G., & Made S. 2018. *Komputasi Paralel Menggunakan Model Message Passing Pada SIM RS (SIstem Informasi Manajemen Rumah Sakit)*. Majalah Ilmiah Teknologi Elektro Vol 17 No 3.

[9]. Setyawan C.A. 2015. *Metode Eliminasi Gauss dengan Komputasi Parallel*. Makalah IF2123 Aljabar Geometri.

[10]. Barney Blaise. 2011. *Message Passing Interface (MPI)*. Lawrence Livermore Nation Laboratory. Dikutip 22 September 2019 dari Computing llnl: https://computing.llnl.gov/tutorials/mpi/.

[11]. Liswandini Intan, Budhi Irawan, & Irzaman. 2005. *Studi Komparatif Antara Paralel Virtual Machine (PVM) dan Message Passing Interface (MPI) dengan Memanfaatkan Local Area Network (LAN)*. Universitas Komputer Indonesia.

[12]. Yahya S.Y. 2014. *Analisa Perbandingan Algoritma Bubble Sort dan Selection Sort dengan Perbandingan Eksponensial*. Jurnal Pelita Informatika Budi Darma Vol VI No 3.

[13]. Cormen T., C. Leiserson, R. Rivest, & C. Stein. 2001. *Introduction to Algorithms*. McGraw Hill, pp.320-330.

[14]. Alfatwa D.F., Eriek R.S.P., & Fahris M.A. 2015. *Implementasi Algoritma Radix Sort dalam Berbagai Kasus Bilangan Dibandingkan Algoritma Pengurutan yang lain*. Institut Teknologi Bandung.