Pemilihan Graphics Processing Unit Nvidia dan AMD Menggunakan Algoritma Simple Additive Weighting

Electing Graphics Processing Nvidia and AMD Using Simple Additive Weighting

Jeffrey Jahja[1], Rizky Albert Hartono[2], Kristien Margi Suryaningrum[3]
[1,2,3]Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jl. Lodan Raya No. 2 Ancol, Jakarta Utara 14430
e-mail: [1]jeffrey.511999@gmail.com, [2]rizky.albert@gmail.com, [3]kristienmargi@gmail.com

Abstrak

Graphics Processing Unit (GPU) merupakan komponen komputer yang bekerja untuk mengelola grafis pada komputer baik dalam bentuk video, gambar, animasi. Pemilihan Graphics Processing Unit bagi pengguna komputer merupakan hal yang harus diperhatikan demi penggunaan yang maksimal. Penelitian dilakukan bertujuan untuk membangun sistem pengambilan keputusan untuk pemilihan Graphics Processing Unit. Metode yang digunakan dalam penelitian ini adalah (SAW) Simple Additive Weighting. Konsep dari Simple Additive Weighting adalah dengan memberikan bobot pada setiap kriteria yang digunakan kemudian dilanjutkan dengan perangkingan untuk mendapatkan hasil yang sesuai dengan bobot kriteria yang paling sesuaai. Penelitian ini dilakukan dengan urutan analisis, desain, program, serta pengujian. Kesimpulan yang didapat dalam penelitian ini adalah dengan dibuatnya sistem pendukung keputusan untuk pemilihan (GPU) Graphics Processing Unit mempermudah proses pemilihan (GPU) bagi penggunannya.

Kata Kunci : GPU, Simple Additive Weighting (SAW), Sistem Pendukung Keputusan.

Abstract

Graphics Processing Unit (GPU) is a computer component that works for managing graphics on computer in form of video, picture, animation. Picking Graphics Processing Unit for user is a must thing for maximal performance. This research was maded for building decision-making system for picking Graphics Processing Unit. Method used in this research is (SAW) Simple Additive Weighting. The concept of Simple Additive Weighting is by giving weight on each used criteria then continued with ranking to get the accurate result equal as the weight criteria. The conclusion in this research is by making decision-making system for picking (GPU) Graphics Processing Unit are making the process easier picking the GPU for user.

Keywords : GPU, Simple Additive Weighting (SAW), Decision Support System.

1. PENDAHULUAN

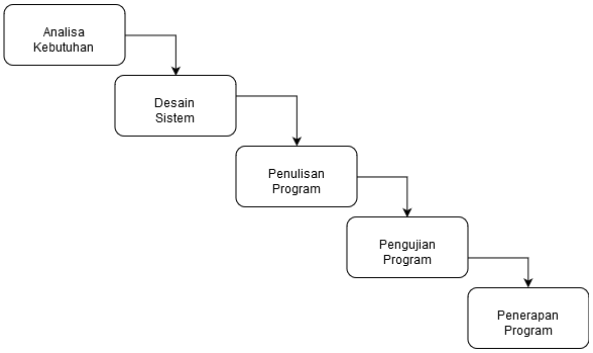
Kebutuhan grafis saat ini merupakan hal yang sering dipertimbangkan oleh banyak orang pada saat akan membeli sebuah komputer maupun membangun sebuah komputer[1]. Namun dikarenakan perkembangannya yang terbilang cepat *Graphics Processing Unit* (GPU) pun memiliki banyak ragam fitur baru maupun kelebihan pada fitur – fitur tertentu yang

bergantung pada setiap perusahaan yang mengembangkannya[2]. Timbul masalah dalam pemilihan GPU untuk digunakan dalam beberapa kegiatan mulai dari harga yang ditawarkan, kecepatan GPU memproses grafis, memori yang digunakan dan lainnya[3]. Berdasarkan masalah yang timbul maka judul penelitian ini adalah "Pemilihan *Graphics Processing Unit* Nvidia dan AMD menggunakan Algoritma *Simple Additive Weighting*" Yang sekiranya dapat membantu proses pemilihan GPU yang sesuai dengan kebutuhan setiap orang.

Penelitian ini terdiri dari beberapa tahapan, yakni pengumpulan data, perancangan sistem, pengujian sistem, pada penelitian ini penulis menggunakan aplikasi berbasis *mobile* (dapat digunakan dimana saja)[4], aplikasi yang dibuat dalam penelitian ini merupakan sistem pengambilan keputusan yaitu sistem yang saling berinteraksi dengan sistem bahasa, sistem pengetahuan, dan sistem pemrosesan masalah[5],[6],[7]. Algoritma SAW digunakan karena dapat menentukan bobot setiap atribut kemudian dilakukan proses perangkingan yang akan memberikan alternatif terbaik[8],[9], atau bisa juga disebut sebagai kombinasi bobot linear atau metode *scoring*[10],[11],[12]. Aplikasi ini dibuat dengan bahasa pemrograman berorientasi objek bertujuan untuk membuat kode program yang lebih terstruktur, terkelompokkan berdasarkan objek-objek yang terlibat sehingga bagian-bagiannya dapat digunakan untuk pembuatan aplikasi lain[13],[14].

2. METODE PENELITIAN

2.2 Metode Analisis dan Perancangan



Gambar 1. Metode Pengembangan Sistem *Waterfall.*

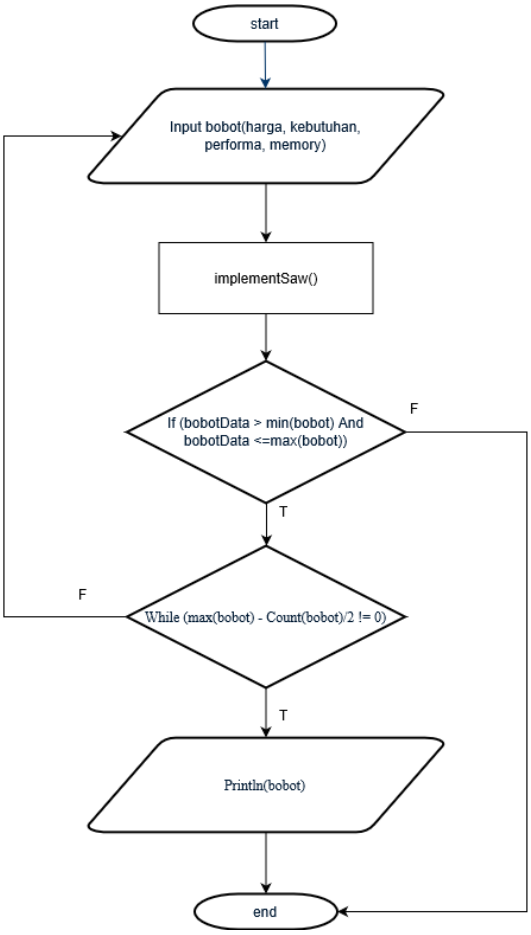
Pengembangan aplikasi dalam penelitian ini menggunakan metode *waterfall* dimana sering juga disebut model sekuensial linier (*sequential* linier) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian, dan tahap pendukung (*support*)[15].

3. HASIL DAN PEMBAHASAN

3.1 Perancangan

3.1.1 *Flowchart*

Flowchart pada gambar 2 menggambarkan alur kerja program, dimana *user* harus melakukan penginputan bobot, yang kemudian dengan metode SAW akan dipilih data dari basis data yang bobotnya sesuai dengan ketentuan bobot yang dimasukan oleh *user* dan kemudian ditampilkan.



Gambar 2. *Flowchart.*

1. Ketika aplikasi dijalankan pengguna diminta untuk memasukan kebutuhan, harga, dan kriteria kebutuhan yang nantinya akan digunakan dalam penentuan bobot.
2. Data masukan akan diproses menggunakan metode SAW dimana pembobotan dan normalisasi akan dilakukan.
3. Dilakukan pengecekan dan menentukan nilai terbaik berdasarkan hasil pengolahan dengan metode SAW , jika hasil tidak didapatkan maka proses akan kembali ke tahap meminta masukan pengguna.
4. Hasil terbaik ditampilkan di aplikasi.

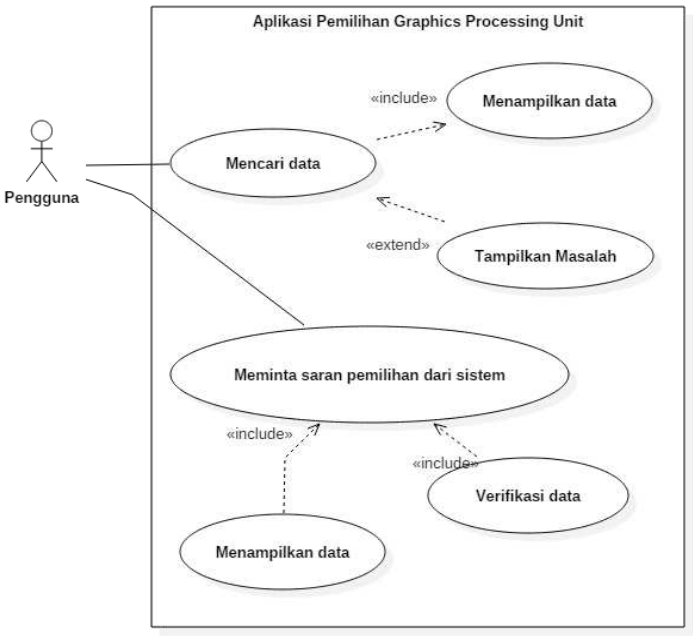
3.1.2 *Use Case Diagram*

Fitur dalam aplikasi yang dikembangkan oleh penulis adalah *user* dapat melakukan hal-hal seperti pada gambar 3 :

1. *User* dapat mencari data

   Ketika *user* mencari data maka data akan ditampilkan, jika terjadi kesalahan maka akan muncul pesan *error.*
2. *User* dapat meminta saran dari sistem

   Ketika *user* meminta saran secara otomatis sistem akan melakukan verifikasi data yang diinputkan *user* dan menampilkan hasil.

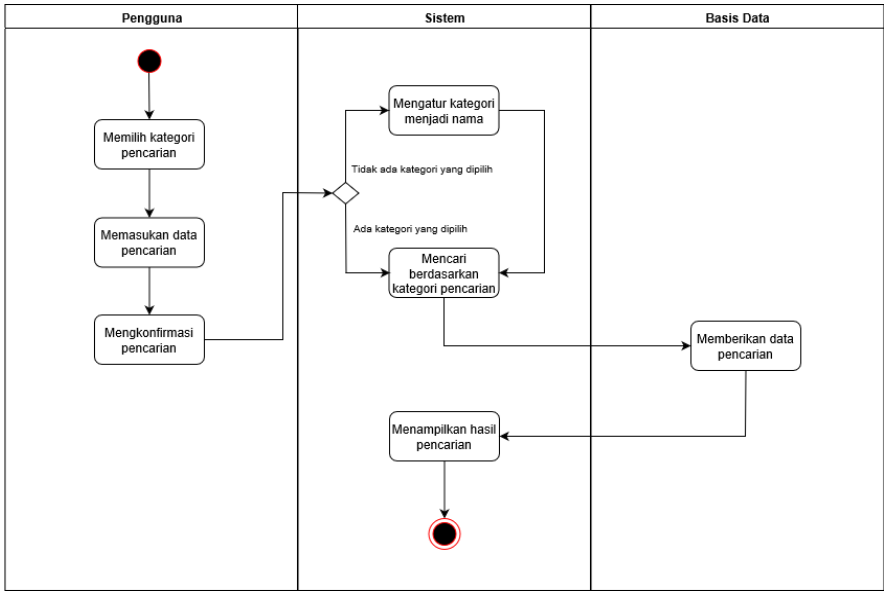


Gambar 3. *Use Case Diagram.*

3.1.3 *Activity Diagram*

Aktivitas yang terjadi dalam sistem ketika *user* menggunakan fitur pencarian dapat digambarkan seperti pada gambar 4.

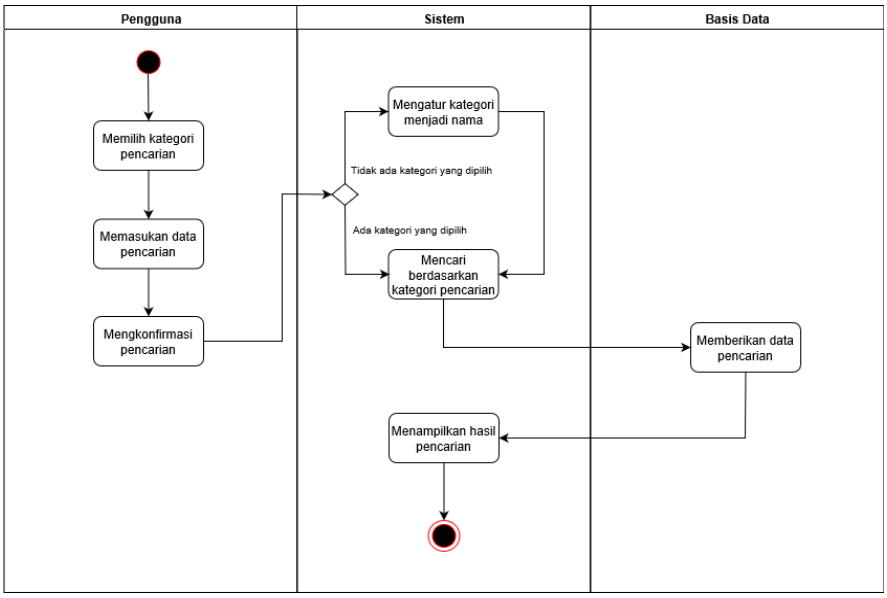
1. Ketika akan melakukan pencarian *user* akan memilih kategori dan memasukan data pencarian berdasarkan kategori, diikuti dengan mengkonfirmasi pencarian.
2. Sistem mengecek apakah kategori telah diatur oleh *user*, jika tidak maka sistem akan mengatur pencarian berdasarkan nama, diikuti dengan meminta data ke *database*.
3. *Database* akan memberikan data ke sistem untuk ditampilkan.



Gambar 4. *Activity Diagram* Fitur Pencarian.

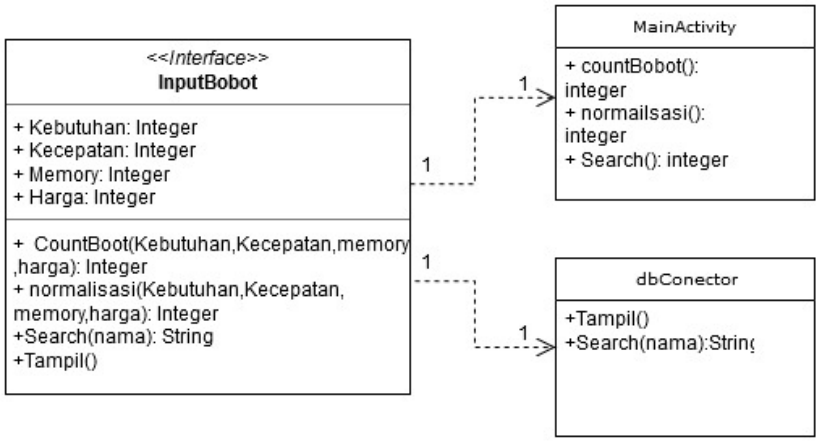
Aktivitas yang terjadi dalam sistem ketika *user* menggunakan fitur Meminta saran dapat digambarkan seperti pada gambar 5.

1. Ketika *user* meminta saran kepada sistem, *user* akan diminta untuk memasukan bobot, anggaran, dan menentukan penggunaan GPU, yang diikuti konfirmasi dasar penyaranan.
2. Sistem akan memproses data menggunakan metode SAW, sekaligus meminta data ke *database*.
3. *Database* akan memberikan data ke sistem untuk ditampilkan.



Gambar 5. *Activity Diagram* Fitur Penyaranan.

3.1.4 *Class Diagram*

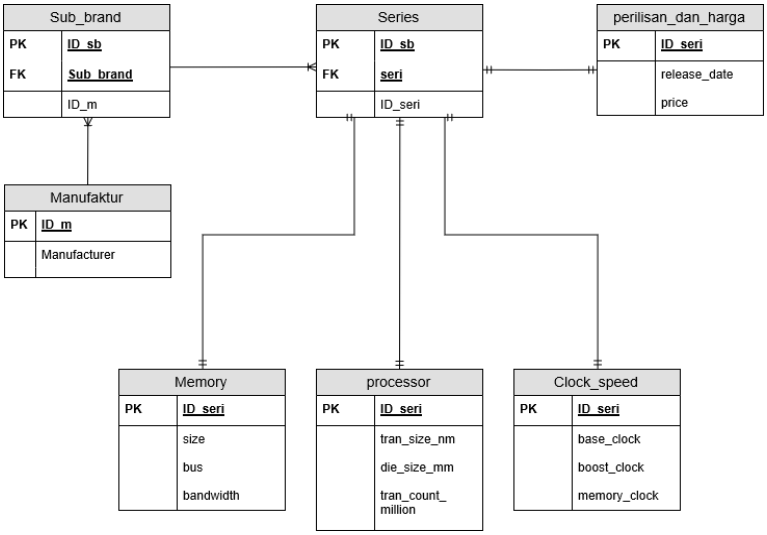


Gambar 6. *Class Diagram.*

Class diagram pada gambar 6 menggambarkan bagaimana *class* "inputBobot" memiliki beberapa data yang diperlukan yaitu bobot dengan kategori kebutuhan, kecepatan, memori, dan

harga kemudian menggunakan beberapa metode yang diambil dari “MainActivity” dan “dbConector”.

3.1.5 *ERD* (*Entity Relationship Diagram)*



Gambar 7. ERD Basis Data GPU.

ERD pada gambar 7 memiliki hubungan sebagai berikut :

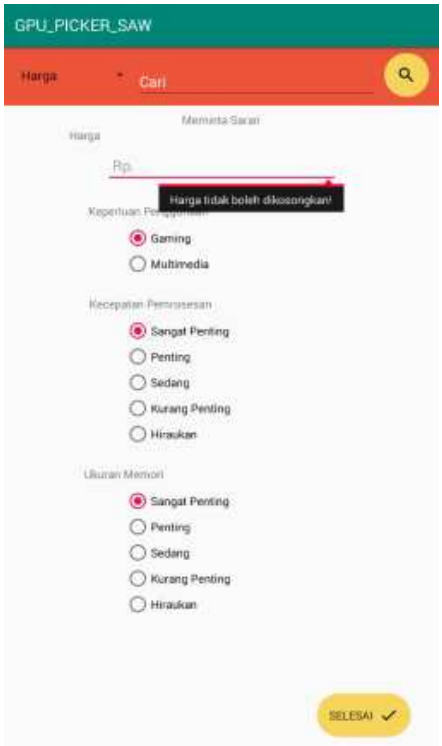
1. Setiap satu manufaktur memiliki banyak “sub_brand”.
2. Setiap satu “sub_brand” memiliki banyak “series”.
3. Setiap satu “series” memiliki satu “perilisan_dan_harga”, “Processor”, “Memory”, dan “Clock_speed”.

3.1.6 Implementasi Antarmuka Pengguna

Gambar 8. Halaman Utama.

Halaman utama merupakan halaman dimana pengguna diminta untuk memasukan data yang akan digunakan untuk pengolahan data mulai dari harga (dalam rupiah), keperluan penggunaan GPU, tingkat kepentingan kecepatan GPU bagi pengguna dan kepentingan ukuran memori pada GPU.

Halaman utama juga dilengkapi dengan fitur pencarian yang dapat dilakukan dalam beberapa kategori diantaranya harga, dan nama manufaktur.



Gambar 9. Pemberitahuan Wajib Memasukan Harga.

Tombol selesai terdapat pada pojok kanan bawah tampilan dimana ketika pengguna telah selesai melakukan penginputan data dan menekan tombol selesai maka aplikasi akan melakukan pemrosesan data yang telah dimasikan pengguna.

3.1.7 Implementasi Metode atau Algoritma

Gambar 10. Masukan data uji coba.

Implementasi algoritma SAW pada aplikasi yang dibuat oleh penulis diterapkan dengan cara sebagai berikut :

- Pencarian data berdasarkan harga guna memperkecil jumlah data gpu yang akan dipilih.
- Pengguna diminta untuk memilih keperluan penggunaan yaitu gaming dan multimedia, guna menentukan manufatkur sesuai dengan kebutuhan.
- Pengguna diminta untuk memilih kepentingan kecepatan, dan ukuran memori untuk dapat menntukan hasil gpu yang dibutuhkan oleh pengguna.

Kriteria (C) pada aplikasi ini ditentukan oleh penulis menggunakan radio button sebagai opsi pilihan dengan isi seperti pada gambar 10, yaitu Sangat Penting, Penting, Sedang, Kurang Penting, dan hiraukan. Pada setiap kriteria (kecepatan pemrosesan, ukuran memori, dan waktu rilis) penulis menetapkan kriteria kecepatan pemrosesan, dan ukuran memori memiliki nilai maksimal 40% dengan waktu rilis merupakan 100% dikurangi dengan jumlah persen terhadap kecepatan pemrosesan dan ukuran memori guna memastikan bahwa jumlah bobot total adalah 100%.

Penentuan bobot pada aplikasi ini penulis menggunakan nilai maksimal, minimal, dan median dari data yang telah dicari berdasarkan harga dan manufaktur. Pertama dicari nilai maksimal, dan nilai minimal dari data yang telah diperoleh guna mencari nilai median, kemudian dicari nilai tengah dari nilai maksimal dan median, diikuti nilai median dengan nilai minimal yang nantinya akan digunakan untuk memperoleh bobot. Berikut adalah pseudo-code untuk mencari nilai maksimal dan minimal yang digunakan penulis:

Nilai acuan bobot dapat dicari setelah nilai maksimal dan minimal telah didapatkan, dengan penghitungan nilai median dan nilai tengah dari median terhadap minimal dan maksimal dengan rumus dan kode program sperti yang terdapat pada gambar 11:

```
// Membuat nilai acuan bobot

Batasbc5, batasbc4, batasbc3, batasbc2, batasbc1 : double
Maxbc, Minbc :  double { Pencatat nilai terbesar, nilai terkecil dari "Base
Clock"}

ALGORITMA
read ( Maxbc ) { nilai terbesar dari data di basis data }
read ( Minbc ) { nilai terkecil dari data di basis data }

batasbc5 ← Maxbc {mengisi nilai batasbc5 dengan nilai maksimal}
batasbc4 ← (Maxbc +( (Maxbc + Minbc)/2)) /2
batasbc3 ← (Maxbc + Minbc)/2 { mencari nilai tengah dengan rumus median}
batasbc2 ←(batasbc3 +Minbc)/2
batasbc1 ← Minbc
```

Gambar 11. *Pseudo-code* Membuat Nilai Acuan Bobot.

Bobot dibuat setelah nilai acuan didapatkan, dengan pembobotan dengan menggunakan pengondisian sperti yang terdapat pada gambar 12:

```
//Membuat bobot bc

batasbc5, batasbc4, batasbc3, batasbc2, batasbc1 : double
baseclock : Array of double {data dari basis data}
bobotbc : Array of Integer
i : Integer {Pencacah banyaknya pengulangan}
JumlahRow : Integer  { Jumlah baris / record pada basis data }

ALGORITMA
read( jumlahRow ) {Membaca banyaknya baris dari basis data}
read( baseclock[] ) {Membaca data dari basis data}
for i ← 0   to jumlahRow  do
        if baseclock[i] >= batasbc4 then
                bobotbc[i] ← 5
        else if  baseclock[i] >= batasbc3 then
                bobotbc[i] ← 4
        else if  baseclock[i] >= batasbc2 then
                bobotbc[i] ← 3
        else if  baseclock[i] >= batasbc1 then
                bobotbc[i] ← 2
        else bobotbc[i] ← 1
        end if
end for
```

Gambar 12. *Pseudo-code* Membuat Nilai Bobot.

Penentuan nilai bobot pada aplikasi ini dibagi menjadi 5, dimana 5 sebagai nilai terbaik, dan 1 sebagai nilai terendah dimana jika data yang didapat lebih besar sama (>=) dengan batas 4 (nilai tengah dari maksimal dan median) maka bobot sama dengan 5, jika data yang didapat lebih kecil dari batas 4, maka dilakukan pengujian apakah data yamg didapat lebih besar sama

dengan (>=) batas 3, jika iya maka bobot sama dengan 4, jika tidak dilakukan lagi pengujian dengan batas 2. Jika lebih besar batas sama dengan 2 maka bobot sama dengan 3, jika tidak maka data akan dibandingkan lagi dengan batas 1 jika nilainya lebih besar sama dengan (>=) batas 1 maka bobot sama dengan 2, selain itu bobot sama dengan 1. Setelah nilai bobot didapatkan, dicarilah nilai maksimal dari bobot yang ada guna untuk dilakukan normalisasi.

Normalisasi kriteria dapat dilakukan dengan menggunakan rumus :

$$r_{ij} = \frac{X_{ij}}{Max_i X_{ij}} \text{ ; untuk } j = \text{ benefit} \quad (1)$$

$$r_{ij} = \frac{Min_i X_{ij}}{X_{ij}} \text{ ; Untuk } j = \text{cost} \quad (2)$$

Dalam penelitian ini penulis hanya mencari *benefit* menggunakan rumus $r_{ij} = \frac{X_{ij}}{Max_i X_{ij}}$; untuk $j =$ *benefit*, menggunakan algoritma sebagai berikut :

```
// Normalisasi C

baseclock, MemorySize, Release : Array of double {data dari basis data}
bobotbc, bobotms,bobotrd : Array of Integer { nilai bobot terhadap data di basis data }
maxbobotbc, maxbobotms, maxbobotrd : Integer { nilai maksimal dari bobot pada
setiap data }
i : Integer {Pencacah banyaknya pengulangan}
JumlahRow : Integer { Jumlah baris / record pada basis data }
normCbc, normCms, normCrd : Array of Integer

ALGORITMA
for i ← 0  to jumlahRow do
        normCbc[i] ← bobotbc[i]/ maxbobotbc
        normCms[i] ← bobotms[i]/ maxbobotms
        normCbrd[i] ← bobotrd[i]/ maxbobotrd
```

Gambar 13. *Pseudo-code* Normalisasi C.

Normalisasi (C) akhir dilakukan dengan pengalian hasil normalisasi (C) terhadap kriteria yang telah ditentukan di awal, pada kasus ini penulis menggunakan kriteria kecepatan pemrosesan = 40% = 0.4, ukuran memori = 40% = 0.4, dan waktu rilis = 20% = 0.2, maka hasil normalisasi dikalikan dengan nilai kriteria yang ada seperti pada *pseudo-code* sebagai berikut :

```
// Normalisasi C Akhir

i : Integer {Pencacah banyaknya pengulangan}
JumlahRow : Integer { Jumlah baris / record pada basis data }
normCbc, normCms, normCrd : Array of Integer { hasil normalisasi C }

ALGORITMA
for i ← 0  to jumlahRow do
        normCbc[i] ← normCbc[i]*InputKecepatan { pengalian normalisasi C
dengan inputan kriteria }
        normCms[i] ← normCbc[i]*InputMemori
        normCbrd{i] ← normCbc[i]*InputRilis
end for
```

Gambar 14. *Pseudo-code* Normalisasi C Akhir.

Penentuan data GPU terbaik dapat ditentukan dengan mencari nilai terbesar dari hasil pengalian data normalisasi dengan kriteria, diikuti dengan penjumlahan masing-masing kriteria untuk mencari nilai terbaik, setelah penjumlahan nilai tersebut dicari nilai terbaik dengan cara mencari nilai terbesar dari data yang ada seperti pada *pseudo-code* sebagai berikut:

```
// Penentuan Hasil Terbaik

i : Integer {Pencacah banyaknya pengulangan}
JumlahRow : Integer  { Jumlah baris / record pada basis data }
normCbc, normCms, normCrd : Array of Integer { hasil normalisasi C }
A[i] : Array of Integer
terbaik : Integer

ALGORITMA
for i ← 0   to jumlahRow do
        A[i] ← normCbc[i] + normCms[i] + normCbrd{i]
end for

terbaik ← A[0] {mengasumsikan nilai sama dengan nilai awal A}
for i ← 0   to jumlahRow do
        if A[i] > terbaik then
                terbaik ← A[i]
        end if
end for
```

Gambar 15. *Pseudo-code* Penentuan Hasil Terbaik.

4. KESIMPULAN

Berdasarkan hasil penelitian yang dilakukan oleh penulis maka penulis menarik kesimpulan sebagai berikut :

1. Dengan adanya aplikasi pemilihan *graphics processing unit* nvidia dan amd menggunakan algoritma SAW (*Simple Additive Weighting*) berbasis *mobile* proses pemilihan GPU dapat dilakukan.
2. Dengan adanya aplikasi pengguna dimudahkan untuk mencari GPU yang sesuai dengan kegunaan dan anggaran yang dimiliki.
3. Penggunaan tipe data *double* dapat menimbulkan masalah dalam proses pemilihan alternatif akhir.

5. SARAN

Berikut adalah saran yang penulis usulkan untuk pengembangan dikemudian hari adalah sebagai berikut :

1. Hasil pengolahan terhadap nilai terbaik kedepannya perlu ditambah tingkat akurasinya dengan mengatasi masalah terhadap tipe data *double* dimana angka spesifik di belakang koma tidak terdapat kelebihan jumlah hasil perhitungan.
2. Tampilan aplikasi dikembangkan menjadi lebih dinamis dan lebih mudah digunakan oleh pengguna.
3. Fitur pencarian dibuat lebih fleksibel terhadap masukan pengguna sehingga pengguna dapat lebih mudah mencari data yang sesuai.

DAFTAR PUSTAKA

[1] Utomo, Tegar Mochamad dan Hindayati Mustafidah. 2016. Penentuan Spesifikasi Komputer Berdasarkan Kebutuhan Pemakai Dan Harga Menggunakan Basis Data *Fuzzy*. Vol. IV No. 1. ISSN 2086-9398.

[2] Rizaldi, Ilyas Hilmi. dkk. 2017. Optimasi Proses Rendering Objek Game 3D Menggunakan Pemrograman CUDA pada Game Sandbox Craft. Vol. 4 No. 3. ISSN 2528-6579.

[3] Kurniawan, Bagus. dkk. 2015. Analisis Perbandingan Komputasi GPU dengan CUDA dan Komputasi CPU untuk *Image* dan *Video Procesing*. ISSN 1907-5022.

[4] Syani, Mamay dan Nindi Werstantia. 2018. Perancangan Aplikasi Pemesanan Catering Berbasis Mobile Android. Vol. 1 No. 2. ISSN 2615-0387.

[5] Hidayat, Rachmat. 2013. Metode Simple Additive Weighting Sebagai Sistem Pendukung Keputusan Penerima Beasiswa Murid Berprestasi. Vol. 2 No. 2. ISSN 2541-2019.

[6] Ernawati. dkk. 2017. Rancang Bangun Sistem Pendukung Keputusan Kenaikan Jabatan Pegawai dengan Metode Simple Additive Weighting. ISSN 1979-0767.

[7] Situmorang, Harold. 2015. Sistem Pendukung Keputusan Pemilihan Calon Peserta Olimpiade Sains Tingkat Kabupaten Langkat pada Madrasah Aliyah Negeri (MAN) 2 Tanjung Pura dengan Menggunakan Metode Simple Additive Weighting (SAW). Vol. IV No. 2. ISSN 2337-3601.

[8] Haswan, Febri. 2019. Application of Simple Additive Weighting Method to Determine Outstanding School Principals. Vol. 3 No. 2. ISSN 2541-2019.

[9] Afshari, Alireza. dkk. 2010. Simple Additive Weighting approach to Personnel Selection Problem. Vol. 1 No. 5. ISSN : 2010-0248.

[10] Agustina, Ina. dkk. 2017. Implementasi Metode SAW (Simple Additive Weighting) pada Perancangan Sistem Pendukung Keputusan Penerimaan Beasiswa Berbasis Web. Vol. 4. ISSN 2089-1083.

[11] Marpaung, Nasrun. 2018. Penerapan Metode Simple Additive Weighting pada Sistem Pendukung Keputusan untuk Menentukan Kenaikan Gaji Karyawan. Vol. IV No. 2. ISSN 2550-0201.

[12] Hidayati, Risky. dkk. 2016. Sistem Pendukung Keputusan Penerimaan Beasiswa di SMK N 1 Sukoharjo dengan Metode *Simple Additive Weighting* (SAW). Vol. 4 No. 1. ISSN 2338-4018.

[13] Hartono, Roni. 2012. "Aplikasi Laporan Laba-Rugi *Online* Berbasis Web Studi Kasus PT Andalusia Persada Indonesia". Fakultas Teknologi dan Desain. Universitas Bunda Mulia. Jakarta.

[14] Retnoningsih, Endang. dkk. 2017. Pembelajaran Pemrograman Berorientasi Objek (Object Oriented Programming) Berbasis *Project Based Learning*. Vol. 2 No. 1. ISSN 2548-3412.

[15] Tabrani, Muhamad dan Eni Pujiarti. 2017. Penerapan Metode Waterfall pada Sistem Informasi Inventori PT. Pangan Sehat Sejahtera. Vol. 1 No. 2. ISSN: 2615-3645.