ISSN: 2089-1121

EKSPLOITASI ALGORITMA PARALLEL UNTUK PEMBENTUKAN DECISION TREE ID3

Hartanto Hadinata
Teknologi Informasi
Sekolah Tinggi Teknik Surabaya
hrtanto1986@gmail.com

ABSTRAK

Klasifikasi merupakan salah satu cara pembelajaran utama dalam Data Mining. Dengan menggunakan Klasifikasi komputer dapat mempelajari pattern atau ciri dari suatu data sehingga terbentuk pengetahuan berupa Decision Tree. Dalam praktek-nya klasifikasi memerlukan waktu yang sangat lama dalam pembentukan pengetahuan. Dengan memanfaatkan algoritma Parallel diharapkan dapat mempercepat proses pembentukan rule-rule tersebut.

Sampai saat ini, algoritma parallel ID3 masih terus dikembangkan hingga diharapkan bisa mencapai fine-grained parallelism. Dimana tingkat kompleksitas algoritma, komunikasi antar proses, dan waktu eksekusi-nya lebih cepat. Berdasarkan research sebelumnya dari Karsten Steinhaeuser, pembentukan decision tree secara parallel pada count matrix masih kurang mengalami reduksi yang cukup signifikan pada dataset yang besar antara 65%-75% pada mesin Multi-Processor Cray MTA.

Ide awal Pembentukan Decision Tree ID3 pertama kali ditemukan oleh Ross J. Quinlan. Algoritma ini berkerja secara recursive dengan model penelusuran tree secara depth first. Recursive ini bekerja dari root parent kemudian turun ke node dibawahnya secara depth first sampai semua dataset tercover secara penuh. Awalnya, pengembangan algoritma parallel ini didesain dengan menerapkan teori divide and conquer dengan model pembuatan tree secara breadth first.

Penerapan algoritma parallel ini pada pembentukan tree ini memiliki peningkatan yang cukup signifikan dibandingkan dengan serialnya. Dengan memanfaatkan algoritma parallel pada kasus pembentukan decision tree ID3 kecepatan pembentukan tree dapat dipersingkat. Persentase untuk pembentukannya berkurang 50% dari algoritma serial atau recursive.

Untuk pengembangan lebih lanjut terkait dengan parallel computing dan classification ada beberapa saran dan ide. Penanganan attribute continuous dalam komputasi secara parallel. Pemberian fitur incremental learning dimana tree bisa dibuat secara tahap demi tahap hingga terbentuknya tree secara utuh. Kedua ide ini dapat ditambahkan supaya mampu menangani dataset yang beragam dan kompleks.

Kata kunci: Permasalahan Parallel, Decision Tree, Iterative Dichotomiser, Pemrogramman Parallel, Klasifikasi Parallel.

ABSTRACT

Classification is one of important tasks in data mining. Using classification a computer can learn the pattern from a dataset. The result is a knowledge represented by a set of rules. Practically, Classification takes so much time to build this knowledge.

By applying Parallel Algorithm, It is expected that building rule process can be cut down in a half of the serial processes.

Until now, Parallel Algorithm of ID3 is still being developed. Progresively, This research is developed with hope to find fine-grained parallelism. Which Communication over processes is small, low complexity algorithm, and faster time execution. Based on previous research by Karsten Steinhaeuser, There is fact that the reduction rate of parallel algorithm is around 65%-75% on Multi-Processor Architecture of Cray MTA Machine.

The first idea of decision tree ID3 builder is found by Ross J. Quinlan. This Algorithm works in recursive which is explored in depth first model. This recursive pattern work from root parent drop down to node below in serial ways until all dataset covered. Firstly, This idea of parallel algorithm designed with divide and conquer theory in breadth first model.

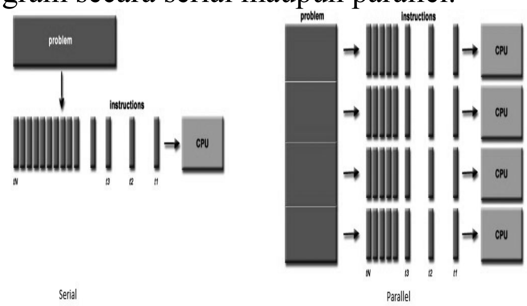
This implementation of parallel algorithm to build decision tree has an improved performance rather than its serial process. With parallel algorithm, speed of building decision tree ID3 can be reduced. The percentage of performance ratio is 50% from algorithm work in serial or recursive.

For future works related to this parallel computing and classification there is some ideas and suggestions. Handling continuous attribute in parallel computation. Feature incremental learning which the tree can be developed gradually. Two of this feature can be added to cover more various and complex dataset.

Keywords: Parallel Problem, Decision Tree, Iterative Dichotomiser, Parallel Programming, Parallel Classification

1. PENDAHULUAN

Secara tradisional komputasi suatu algoritma atau program selalu dilakukan secara serial. Problem atau masalah yang ada pada komputer biasanya diselesaikan secara satu per satu. Berikut ini adalah gambaran ilustrasi perbandingan penyelesaian masalah pada suatu program secara serial maupun parallel.



Gambar 1. Parallel Problem dan Proses

Dari gambar 1 dapat dibandingkan cara kerja komputer dalam menyelesaikan masalah. Dalam tesis ini problem utama yang akan dikemukakan adalah klasifikasi dalam ilmu Data Mining.

Klasifikasi adalah sebuah task yang sangat utama dalam data mining. Salah satu bentuk knowledge yang dipakai dalam klasifikasi ini adalah decision tree. Decision tree sering dipakai dalam data mining sebagai bentuk representasi sebuah knowledge karena mudah dipelajari dan dipahami.

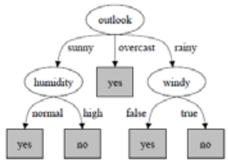
ISSN: 2089-1121

Meskipun memiliki tingkat kompleksitas yang rendah dalam komputasi-nya. Algoritma ID3 ini masih memakan waktu yang cukup lama untuk pembentukkan tree-nya terutama pada dataset dengan skala besar. Meskipun demikian, proses pembentukkan tree ini nantinya bisa dipersingkat waktunya dengan pembentukkan tree secara parallel. Dengan pembentukkan secara parallel ini diharapkan waktu pembentukkan tree-nya dapat dipersingkat meskipun tingkat kompleksitas untuk komputasi-nya lebih besar.

Sampai sekarang algoritma parallel ini dalam pembentukan decision tree ini masih topik research terutama untuk membentuk fine-grained parallelization. Dimana diharapkan apabila sudah sampai tahap fine-grained maka diharapkan running time untuk konstruksi Decision Tree-nya bisa dipotong menjadi setengahnya.

Dalam kasus dataset sederhana yaitu Playing Tennis yang biasanya digunakan sebagai contoh dalam beberapa Decision Tree. Berikut ini adalah dataset-nya dan bentuk tree yang lengkap dimana merupakan hasil training set.

outlook	temperature	humidity	windy	play tennis?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no



Gambar 2. Dataset Playing Tennis dan Knowledge Tree

Gambar 2 adalah sebuah gambar dataset playing tennis dengan sebuah gambar lain yaitu decision tree yang mewakili hasil dari training data playing tennis. Dalam membentuk decision tree tersebut diperlukan evaluasi per-feature(outlook,humidity,windy) untuk dipilih best split dengan cara menghitung information gain-nya.

Dengan adanya arsitektur multi-prosesor ini diharapkan evaluasi ini bisa diminimalisir dengan memanfaatkan processor yang sedang idle. Dari kompleksitas algoritma yang memiliki notasi O(MN) ini nilai M bisa dibuat sekecil-kecilnya sehingga menjadi 1 jadi notasi-nya O(N) tergantung dari resource processor yang tersedia.

2. TINJAUAN PUSTAKA

Parallel computing atau programming adalah sebuah tehnik komputasi secara simultaneous(bersama-sama) untuk memecahkan sebuah problem atau masalah[1]. Problem atau masalah ini kemudian dibagi-bagi menjadi sub-problem sehingga nantinya sub-problem tersebut diselesaikan secara parallel.

Permasalahan yang terjadi pada parallel computing isu yang terjadi. Permasalahan tersebut antara lain adalah semaphore, dining philoshoper, dan producer and consumer problem. Isu-isu diatas adalah masalah-masalah klasuk dalam akses memory pada multi process atau parallel.

[1] http://en.wikipedia.org/wiki/Parallel_computing, Mei 2011.

ISSN: 2089-1121

Hardware architecture sangat mempengaruhi tinggi-nya percepatan yang didapatkan oleh sebuah proses atau komputasi. Apabila dilihat dari architecture hardware-nya maka sistem processor dapat dibagi menjadi dua bagian yaitu[2] :

1. Shared Memory Systems, adalah sebuah sistem dimana terdiri dari beberapa processor tetapi masih dalam rangkaian memory yang sama.
2. Distributed Memory Systems, adalah sebuah sistem dimana terdiri dari beberapa processor yang terhubung satu persatu dengan memory yang ada. Biasanya antar sistem terhubung melalui sebuah network atau jaringan.

Perbedaan arsitektur komputer ini mengakibatkan adanya perbedaan cara melakukan pemogramman kedalam setiap mesin yang sudah diklasifikasikan oleh Flynn. Berikut ini adalah paradigma-paradigma yang digunakan dalam pemogramman secara parallel. Berikut ini adalah paradigma pemogramman yang dipakai pada parallel computing.

1. Shared Memory Model tanpa Threads
2. Threads Model
3. Message Passing Model
4. Data Parallel Model
5. Hybrid Parallel Model

Pada pemrograman secara parallel terdapat beberapa cara untuk melakukan pemindahan data. Pemindahan data ini sudah direncanakan terlebih dahulu oleh programmer yang bersangkutan. Implementasi yang efektif dan efisien dari komunikasi ini dapat meningkatkan performa, mengurangi usaha dan biaya untuk pengembangan, dan meningkatkan kualitas software. Setiap cara pemindahan data dari process ke process mempunyai cost dalam bentuk waktu dan usaha. Pemindahan data tersebut terjadi antara lain melalui:

1. One-to-All Broadcast and All-to-One Reduction
2. All-to-All Broadcast and Reduction
3. All-Reduce and Prefix-Sum Operations
4. Scatter and Gather
5. All-to-All Personalized Communication

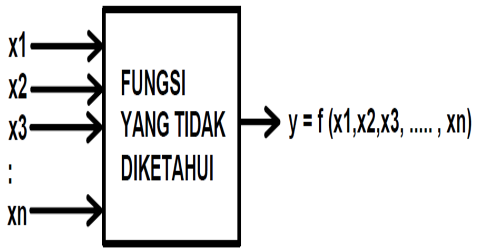
Parallel dapat berjalan pada mesin single-core, karena mesin single core mungkin masih bisa menggunakan lebih dari satu thread. Apabila digunakan single-core ini maka yang terjadi adalah variabel MaximumDegreeOfParallelism = 1. Artinya, proses yang dilakukan dalam pembuatan thread (multiple processing) tidak dilakukan dan hanya melakukan proses.

Algoritma ID3 adalah algoritma yang digunakan untuk klasifikasi dengan model supervised learning. Dimana terdapat training dataset yang kemudian dipakai untuk menentukan test case yang baru. Algoritma ID3 singkatan dari Iterative Dicothomiser atau Induction of Decision "3", pertama kali ditemukan oleh Ross Quinlan sejak akhir dekade 70-an.

Salah satu kelebihan dari algoritma ID3 ini adalah tahap belajar yang cepat, kompleksitas waktu-nya rendah, dan ketelitian dan akurasi yang tinggi. Berikut ini adalah ilustrasi Algoritma ID3.

[2] http://www.edcenter.sdsu.edu/training/workshop99/june29_ppt/JBois/sld008.htm, Mei 2011.

ISSN: 2089-1121



Gambar 3. Fungsi Algoritma ID3

Dimana nantinya algoritma ini dipakai untuk menemukan fungsi tidak diketahui dan menemukan fungsi y. Notasi x1,x2,…,xn merupakan attribute yang dipakai untuk pengukuran atau sebagai input dalam suatu fungsi. Dimana x1,x2,...,xn mewakili attribut dati dataset yang dipakai.

Problem dari algoritma ID3 ini adalah bagaimana caranya mendapatkan decision tree (ID3) terbaik yang konsisten dari sekumpulan data. Problem seperti ini disebut sebagai NP-Hard/Completeness.

NP-hard (*non-deterministic polynomial-time hard*), adalah computational complexity theory, dimana merupakan sebuah klas/kategori dari sebuah problem, yang mengatakan bahwa, "*at least as hard as the hardest problems in NP*". Ada sebuah problem H disebut sebagai NP-hard jika dan hanya jika problem tersebut merupakan NP-complete problem L yang merupakan problem waktu polynomial menggunakan Turing-reducible menjadi H ($L \leq TH$). Dengan kata lain, L dapat dipecahkan menjadi fungsi polynomial oleh sebuah oracle machine dengan oracle untuk H. Secara informal, kita dapat berpikir mengenai sebuah algoritma yang mampu memanggil mesin oracle adalah subroutine untuk memecahkan permasalahan H, dan memecahkan L sebagai fungsi waktu, Jika subroutine tersebut mampu menyelesaikan masalah hanya menggunakan satu iterasi saja untuk komputasinya. NP-hard problems bisa berbagai macam tipe: decision problems, search problems, atau optimization problems. Algoritma ID3 ini dikategorikan sebagai algoritma yang kompleksitasnya tinggi. Karena tidak mungkin menemukan set of rule hanya dalam satu iterasi saja.

3. ANALISA KEBUTUHAN

Ada beberapa kebutuhan yang diperlukan dalam software untuk membuktikan hipotesa yang ada. Kebutuhan-kebutuhan ini akan dijelaskan secara fitur per fitur program yang digunakan untuk membuktikan hipothesis. Selain dari kebutuhan tersebut, nantinya juga akan dianalisa cara kerja algoritma dan desain algoritma pembentukkan decision tree.

Kebutuhan lain dari penelitian ini adalah kebutuhan akan hardware yang dipakai. Hardware yang dipakai juga mempengaruhi waktu yang diperlukan untuk membentuk Decision Tree ID3 apabila program tersebut menggunakan pemrograman secara parallel. Adapun kendala lain dalam melakukan penelitian ini adalah dimana arsitektur parallel komputer mempunyai ragam yang sangat banyak.

Untuk melakukan pelacakan terhadap aktivitas dan affinitas processor. Dibutuhkan sebuah software untuk melacak apakah benar program yang sudah dibuat benar-benar menggunakan resource hardware yang sudah tersedia. Contohnya, pada multicore prosesor dimana komputer tersebut terdiri dari beberapa prosesor apakah resource tersebut juga digunakan sepenuhnya oleh proses yang bersangkutan.

ISSN: 2089-1121

Program yang dibutuhkan untuk membuktikan hipotesa ini adalah Program pembentukan decision tree ID3 secara serial dan secara parallel. Software ini perlu dibuat guna membandingkan waktu proses kedua-nya. Adapun fitur-fitur yang harus dibuat untuk pembuktian hipotesa antara lain :

1. Fitur membaca file berformat *.arff (dataset).
   Untuk melakukan perhitungan decision tree diperlukan pembacaan data untuk mendapatkan beberapa row data yang difilter pada kolom tertentu.
2. Fitur menghitung Gain untuk setiap kolom dataset.
   Untuk melakukan perhitungan decision tree diperlukan perhitungan Gain dan Entropy. Pertama dicari terlebih dahulu entropy dari masing-masing kolom yang terkait pada tree yang sudah ditelusuri baru setelah itu dihitung gain-nya untuk menentukan best split-nya.
3. Fitur filter examples.
   Fitur ini digunakan untuk memfilter dataset dari example-example yang ada pada file dataset arff. Filter ini nanti-nya menggunakan beberapa parameter yang dipakai untuk mendapatkan data dari input filter. Dari hasil filter ini nantinya examples bisa dipassing-kan kedalam sebuah variabel tertentu untuk ditampung dan diproses.
4. Fitur pemilihan best split untuk kolom data continous.
   Fitur ini digunakan untuk melakukan pemilihan best split dari node data yang berupa dataset continous atau numeric. Apabila dataset yang digunakan adalah dataset continous atau numeric maka fitur ini digunakan untuk melakukan pencarian nilai dengan gain yang paling tinggi.
5. Fitur generate Decision Tree secara serial.
   Merupakan fungsi untuk membentuk Decision Tree ID3 secara serial. Fitur ini dibuat untuk membandingkan kecepatan waktu antara algoritma serial dan parallel.
6. Fitur generate Decision Tree secara parallel.
   Merupakan fungsi untuk membentuk Decision Tree ID3 secara parallel. Ini adalah fitur utama dari thesis yang dibuat dimana nantinya tree yang dibuat harus mempunyai hasil yang sama dengan tree yang ada pada fitur decision tree serial.
7. Fitur timer.
   Merupakan fungsi dari timer ini digunakan untuk mengukur kecepatan waktu yang dibutuhkan dalam pembentukkan decision tree ID3. Fitur Timer ini nantinya menggunakan Visual Studio 2010 menggunakan Class yang ada.

4. IMPLEMENTASI

Untuk membuat alat uji coba akan dijelaskan secara spesifik mengenai algoritma-algoritma yang dibuat untuk mendukung fitur-fitur yang ada diatas. Berikut ini adalah daftar algoritma-algoritma hasil dari perancangan software yang bersangkutan.

1. Algoritma Inisialisasi Dataset
2. Algoritma Read Dataset
3. Algoritma Hitung Gain
4. Algoritma Filter Examples
5. Algoritma Calculate Best Split

ISSN: 2089-1121

6. Algoritma Generate Decision Tree in Serial
7. Algoritma Generate Decision Tree in Parallel
8. Algoritma Generate Decision Tree Viewer

Terkait dengan desain struktur data yang dipakai disesuaikan dengan kebutuhan dari aplikasi. Maka diperlukan struktur data untuk beberapa masalah :

1. Penyimpanan working area
2. Penyimpanan relasi rules
3. Penyimpanan data hasil rules

5. SKENARIO UJI COBA

Skenario uji coba yang dilakukan untuk membandingkan performa antara algoritma serial ID3 dengan algoritma parallel ID3. Subjek tes yatu pada dataset, dataset ini berkisar antara 10 data, 35.000 data, 80.000 data. Berikut ini adalah tahapan skenario uji coba yang dilakukan untuk pembuktian hipotesa.

1. Uji coba tahap pertama adalah uji coba performa algoritma dalam pembentukan count matrix secara parallel dan serial.
2. Uji coba tahap kedua adalah uji coba untuk membandingkan akurasi knowledge yang dihasilkan.
3. Uji coba tahap ketiga adalah uji coba untuk membandingkan performa algoritma parallel ID3 dengan algoritma serial ID3.

Uji coba diatas dilakukan pada beberapa hardware architecture diantaranya adalah core 2 duo, core 2 quad, dan core i7. Dengan percobaan tersebut diharapkan bisa melihat performa dari algoritma parallel pada berbagai multi-processor architecture.

6. PENUTUP

Dari proses uji coba tersebut terbentuk beberapa tabel nilai waktu antara performa dan hasil set of rules yang dihasilkan. Berikut ini adalah nilai hasil uji coba performa pada pembentukan count matrix sercara serial dan parallel.

Tabel 1. Hasil Uji Coba Performa Pembentukan Count Matrix

	Serial	Parallel
Hasil Data	**Dataset Playing Tennis** sunny,yes,2 sunny,no,3 overcast,yes,4 overcast,no,0 rain,yes,3 rain,no,2	**Dataset Playing Tennis** sunny,yes,2 overcast,yes,4 sunny,no,3 rain,no,2 overcast,no,0 rain,yes,3
Kecepatan Eksekusi	**Dataset Playing Tennis** Pada Pentium 4 : 0.73 Pada Core 2 Duo : 0.72 **Dataset Nursery** Pada Pentium 4 : 5.66 Pada Core 2 Duo : 5.44 **Dataset Connect-4** Pada Pentium 4 : 01:50.01 Pada Core 2 Duo : 01:49.01	**Dataset Playing Tennis** Pada Pentium 4 : 0.73 Pada Core 2 Duo : 0.43 **Dataset Nursery** Pada Pentium 4 : 5.66 Pada Core 2 Duo : 3.66 **Dataset Connect-4** Pada Pentium 4 : 01:50.01 Pada Core 2 Duo : 01:19.27

ISSN: 2089-1121

Tabel 1 memberikan informasi tentang hasil performa dari pembentukan count matrix secara parallel dan serial pada beberapa processor. Dari tabel 1 tampak bahwa dengan memanfaatkan algoritma parallel untuk pembentukan decision tree mengalami peningkatan performa yang cukup signifikan antara 60% - 70%. Berikut ini adalah hasil uji coba akurasi data antara algoritma serial dan parallel.

Tabel 2. Hasil Uji Coba Akurasi Data

Hasil Serial (Original)	Hasil Parallel (Modification)
if outlook = sunny and humidity = high then result = no if outlook = sunny and humidity = normal then result = yes if outlook = overcast then result = yes if outlook = rain and wind = weak then result = yes if outlook = rain and wind = strong then result = no	if outlook = overcast then result = yes if outlook = sunny and humidity = high then result = no if outlook = sunny and humidity = normal then result = yes if outlook = rain and wind = weak then result = yes if outlook = rain and wind = strong then result = no

Tabel 2 adalah hasil coba uji akurasi data dari algoritma yang dihasilkan. Hasil dari algoritma parallel ini menyebabkan aturan yang dihasilkan acak atau *random*. Hal ini karena proses urutan pengerjaan yang mana yang lebih dulu dikerjakan untuk setiap waktu eksekusi berbeda. Berikut ini adalah hasil uji coba performa pada berbagai macam processor architecture pada dataset skala kecil yaitu playing_tennis.

Tabel 3. Hasil Uji Coba Performa pada Dataset Kecil(*milliseconds*)

Small Dataset	Single	Core 2 Duo	Core 2 Quad	Core i7
Serial	6	6	6	3
Paralel	6	4	4	1

Tabel 3 adalah hasil uji coba performa pada dataset kecil pada berbagai macam processor architecture. Tabel diatas diukur pada timer internal processor dengan menggunakan ukuran milliseconds sebagai pengukurannya. Berikut ini adalah hasil uji coba performa pada medium dataset yaitu nursery. Dengan jumlah record 37000 dan attribut input sebanyak 8.

Tabel 4. Hasil Uji Coba Performa pada Dataset Menengah(*minutes*)

Medium Dataset	Single	Core 2 Duo	Core 2 Quad	Core i7
Serial	130	61	45	23
Paralel	128	45	33	10

ISSN: 2089-1121

Tabel 4 adalah hasil uji coba performa pada dataset menengah pada berbagai processor architecture. Pada dataset nursery terjadi overhead yang cukup besar pada pembentukan decision tree ini. Dengan semakin naiknya spesifikasi hardware yang dipakai demikian pula dengan waktu pembentukannya. Berikut ini adalah hasil uji coba performa pada large dataset yaitu connect-4 pada beberapa mesin yang multi-core architecture.

Tabel 5. Hasil Uji Coba Performa pada Dataset Besar(*minutes*)

Large Dataset	**Single**	**Core 2 Duo**	**Core 2 Quad**	**Core i7**
Serial	260	122	90	51
Paralel	256	90	70	28

Dari hasil analisa terhadap data-data yang sudah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Dengan menggunakan algoritma parallel pada processor arsitektur Intel Core i7. Kinerja algoritma parallel untuk pembentukan decision tree ID3 mengalami reduksi sebesar 50% dibandingkan dengan algoritma serial.
2. Apabila menggunakan parallel programming maka penambahan jumlah core berdampak pada jalannya aplikasi. Semakin banyak jumlah core-nya, maka semakin besar pula reduksinya. Hal ini tidak terjadi pada algoritma yang berjalan secara serial reduksi yang didapat tidak terlalu signifikan.
3. Terdapat anomali dimana pembentukan decision tree ID3 mengalami penurunan performa dibandingkan dengan algoritma serial. Hal ini disebabkan oleh lama waktu komunikasi yang terjadi antar proses dalam thread dan lama waktu query processor untuk membentuk subset of dataset.

Untuk pengembangan lebih lanjut pada penelitian ini dapat dilakukan beberapa penambahan fitur, antara lain:

1. Penanganan missing values pada algoritma parallel. Dengan penanganan ini diharapkan rule-rule yang tidak mengcover dataset dapat dihilangkan.
2. Penambahan fitur incremental learning untuk pembentukan decision tree akan menghemat waktu tunggu yang diperlukan apabila dataset yang digunakan skala sangat besar.
3. Penanganan attribut continuous secara parallel dapat ditambahkan untuk menangani beragam tipe dataset.

7. DAFTAR PUSTAKA

Giuseppe Di Fatta, Michael R. Berthold, dan Srinivasan Parthasarathy. 2006. *Proceedings of the International Workshop on Parallel Data Mining*. Berlin, Germany.

Gunawan, *Diktat ID3 : Induction Decision Tree*. STTS, Indonesia.

https://computing.llnl.gov/tutorials/parallel_comp/#Abstract, *Introduction to Parallel Computing*, 8 April 2011.

http://en.wikipedia.org/wiki/Parallel_computing, *Parallel Computing Concept*, 2011.

http://media.wiley.com/product_data/excerpt/95/04704959/0470495995-108.pdf , *Task-Based Programming Concept*. 2011.

Mellor-Crummey, John. *Analytical Modeling of Parallel Performance*. Rice University, Houston. 2011.

Kai Zheng, Zhiyong Liang, dan Yi Ge, *Parallel Packet Classification via Policy Table Pre-Partitioning*. Tsinghua University, P.R.China.

Karsten Steinhaeuser, Nitesh V. Chawla, dan Peter M. Kogge. *Exploiting Thread-Level Parallelism to Build Decision Trees*. Notre Dame: Department of Computer Science & Engineering.

Karsten J. K. Steinhaeuser, B.S. *Scalable Learning With Thread-Level Parallelism*. Notre Dame, Indiana. 2007.

Quinlan, J. R.: *Induction of Decision Trees*. Journal of Machine Learning 1 (1986) 81–106.