Evaluating performance of Grid based on Soft Computing

KUN GAO, ZHONGWEI CHEN, WAN ZHONG
Computer Science and Information Technology College
Zhejiang Wanli University
No. 8, South Qian Hu Road, 315100, Ningbo, Zhejiang
P. R. China

Abstract: As well known, it is very hard to evaluate the performance of applications in distributed environment, especially in grid computing. Because there are a lot of restriction factors, as well as we can't confirm the weight of these factors to performance. In this paper, we propose a method based soft computing to evaluate the performance of applications in distributed environment. We use the concept of Rough Set and history information to predict some target that the traditional methods can't obtain. The results of the experiment show that the use of soft computing can dealing with uncertain problem in distributed computing and obtain better result than traditional methods. It can be applied in widely area, such as predicting computing power, memory size, system throughput and so on.

Key-Words: Performance Evaluation, Distributed computing, soft computing

1 Introduction

Grid computing is employing the resources of many computer nodes in a network to a certain question, usually to a scientific, technical, and commerce problem that requires much computer process power or access to mass data. In concept, Grid computing is a subset of distributed computing; On the other hand, in function, Grid computing is expansion and continuity to distributed computing. Grid emphasize coordination and cooperation between Grid resources[1-5].

It becomes the encouraging trend, because of the following reasons:

(1) Grid computing can effectively make use of the existing resources.

(2) It can condense a large amount of computing capability to solve the problem which can not be solved before grid.

(3) It will build widely distributed computing platform to integrate all kinds of resource including computation power resource, data resource, network resource and so on.

The research of scientist now focus on the resource allcation and task schedulling in Grid computing. It is the key component in Grid sytem. In order to workout the above problem, scientist must estimate the performance of Grid. In this paper, we propose a method based soft computing to evaluate the performance of applications in distributed environment. We use the concept of Rough Set and history information to predict some target that the traditional methods can't obtain. The results of the experiment show that the use of soft computing can dealing with uncertain problem in distributed computing and obtain better result than traditional methods. It can be applied in widely area, such as predicting computing power, memory size, system throughput and so on.

The rest of this paper is organized as followed: We introduce some related Rough Set concept in section 2; and then we propose a novel reduct algorithm in section 3; in section 4, we conduct experiment to evaluate our approach. Finally in section 5, we conclude this paper.

2 Related Concept on Rough Sets Theory

Rough set theory was presented by Zdzis law Pawlak in the early 1980's. It deals with the classificatory analysis of information system. Its main idea is to maintain the ability of the same classification and derive classification rules through reducing knowledge.

Reduct is an very important aspect in rough sets theory. Reduct is an information system with minimal field sets, which remove the redundant data. The method for this idea is to search a certain fields that can represent original system wholly. So searching a reduct is to select some data with characteristic. Rough sets offers a set of method to find out all reduct. In this section, we introduce the principal concepts of rough sets theory related to our feature selection approach. The detail of the theory can be found in [6-10].

2.1 **Information System (IS)**

An information system is represented as a table, where each line represents a record. Every field represents a property. This two-dimensional table is called an information system:

S=(U,A)

where U is a non-empty finite set of records called the universe and A is a non-empty finite set of fields. In rough sets theory, an information system S is denoted as:

S={U,A,V,f}

where U is a finite set of records, U={ $x1$, $x2$,. . . , xn }, A is a finite set of fields. The attributes in A can be classified into two disjoint subsets, condition attribute set C and decision attribute set D:

$A = C \cup D$ and $C \cap D = \phi$

$V = U_{p \in A} V_p$ and V_p is a domain of attribute P.

$f : U \times A \to V$ is a function which $f(x_i, q) \in V_p$ for every $q \in A, x_i \in U$.

A simple IS is shown in Fig.1. The information system, two-dimensional table, this information system is composed of six records and two fields.

	F1	F2
R1	R1F1	R1F2
R2	R2F1	R2F2
R3	R3F1	R3F2
R4	R4F1	R4F2
R5	R5F1	R5F2
R6	R6F1	R6F2

Fig. 1. Information System

2.2 Indiscernibility Relation

An information system presents all the knowledge in related area. This two-dimensional table may be unnecessarily large because it may be superfluous in the two dimensions. The same or indiscernible records may be described several times, or some of the attributes may be redundant.

Let $P \subseteq A$, $xi, xj \in U$. A binary relation IND called indiscernibility relation is defined as follow:

$IND(P) = \{(xi, xj) | (xi, xj) \in U \times U, a \in P, f(xi, a) = f(xj, a)\}$

Let $U/IND(P)$ denote the set of all equivalence classes of the relation IND(P).

2.3 Lower Approximation

Let $R \subseteq C$ and $X \subseteq U$. The lower approximation of X with respect to R is defined as follow:

$RX = \_\{Y \in U/R : Y \subseteq X\}$

RX is the set of all elements of U which can be with certain classified as elements of X, according to knowledge R.

2.4 Positive Region

Given an information system:

$S= (U, A \cup \{d\})$

let $X \subseteq U$ be a set of records and $B \subseteq A$ be a selected set of fields. The lower approximation of X with respect to B is:

$B_*(X)=\{x \in U: [x]_B \subseteq X\}$.

The upper approximation of X with respect to B is:

$B^*(X)=\{x \in U: [x]_B \cap X \neq \Phi\}$.

The positive region of decision d with respect to B is:

$POS_B(d)= \cup\{B_*(X): X \in U/IND(d)\}$

The positive region of decision attribute with respect to B represents approximate quantity of B. Not all fields or records are necessary while describing approximate quantity of original IS, some are redundant. Reduct is the minimal set of fields describing approximate quantity.

2.5 Reduct

An attribute a is dispensable in $B \subseteq A$ if $POS_B(d)= POS_{B-\{a\}}(d)$. A reduct of B is a set of attributes $B' \subseteq B$ such that all attributes $a \in B-B'$ are dispensable, and $POS_B(d)= POS_{B'}(d)$.

A reduct consists of the minimal set of condition attributes that have the same discerning ability as the original IS. In other words, the reduct includes the most significant attributes. All reducts of a dataset can be found by constructing a kind of discernibility function from the dataset and simplifying it. Unfortunately, it has been shown that finding minimal reduct or all reducts are both NP-hard problems.

There are usually many reducts in an information system. In fact, one can show that the number of reducts of an information system may be up to $C^{|A|/2}_{|A|}$. In order to find reducts, discernibility matrix and discernibility function are introduced.

2.6 Discernibility Matrix

The discernibility matrix of an information system is a symmetric matrix:

$|U| \times |U|$

with entries c_{ij} defined as:

$\{a \in A | a(x_i) \neq a(x_j)\}$ if $d(x_i) \neq d(x_j)$, Φ otherwise.

A discernibility function can be constructed from discernibility matrix by or-ing all attributes in c_{ij} and then and-ing all of them together. After simplifying the discernibility function using absorption rule, the set of all prime implicants decides the set of all reducts of the IS.

3 A Novel Reduct Algrithm

The heuristic comes from the fact that intersection of a reduct and every items of discernibility matrix can not be empty. If there are any empty intersections between some item c_{ij} with some reduct, object i and object j would be indiscernible to the reduct. And this contradicts the definition that reduct is the minimal attribute set discerning all objects (assuming the dataset is consistent).

A straightforward algorithm can be constructed based on the heuristic. Let candidate reduct set $R=\Phi$. We examine every entry c_{ij} of discernibility matrix. If their intersection is empty, a random attribute from c_{ij} is picked and inserted in R; skip the entry otherwise. Repeat the procedure until all entries of discernibility matrix are examined. We get the reduct in R.

The algorithm is simple and straightforward. However, in most times what we get is not reduct itself but superset of reduct. For example, there are three entries in the matrix: $\{a_1, a_3\}$, $\{a_2, a_3\}$, $\{a_3\}$. According the algorithm, we get the reduct $\{a_1, a_2, a_3\}$ although it is obvious $\{a_3\}$ is the only reduct. This is because that our heuristic is a necessary but not sufficient condition for a reduct. The reduct must be a minimal one. The above algorithm does not consider this. In order to find reduct, especially shorter reduct in most times, we need more heuristics.

A simple yet powerful method is sort the discernibility matrix according $|c_{ij}|$. As we know, if there is only one element in c_{ij}, it must be a member of reduct. We can image that attributes in shorter and frequent $|c_{ij}|$ contribute more classification power to the reduct. After sorting, we can first pick up more powerful attributes, avoid situations like example mentioned above, and more likely get optimal or sub-optimal reduct.

The sort procedure is like this. First, all the same entries in discernibility matrix are merged and their frequency is recorded. Then the matrix is sorted according to the length of every entry. If two entries have the same length, more frequent entry takes precedence.

When generating the discernibility matrix, frequency of every individual attribute is also counted for later use. The frequencies is used in helping picking up attribute when it is need to pick up one attribute from some entry to insert into reduct. The idea is that more frequent attribute is more likely the member of reduct. The counting process is weighted. Similarly, attributes appeared in shorter entry get higher weight. When a new entry c is computed, the frequency of corresponding attribute $f(a)$ are updated as $f(a)=f(a)+|A|/|c|$, for every $a \in c$; where $|A|$ is total attribute of information system. For example, let $f(a_1)=3$, $f(a_3)=4$, the system have 10 attributes in total, and the new entry is $\{a_1, a_3\}$. Then frequencies after this entry can be computed: $f(a_1)=3+10/2=8$; $f(a_3)=4+10/2=9$.

Input: an information system $(U, A \cup \{d\})$, where $A= \cup a_i, i=1,\ldots,n$.

Output: a reduct *Red*

1. $Red=\Phi$, count(a_i)=0, for $i=1,\ldots n$.
2. Generate discernibility matrix M and count frequency of every attribute count(a_i);
3. Merge and sort discernibility matrix M;
4. For every entry m in M do
5. If ($m \cap Red == \Phi$)
6. select attribute a with maximal *count(a)* in m
7. $Red=Red \cup \{a\}$
8. Endif
9. EndFor
10. Return *Red*

Fig. 1. A Heuristic Reduct Algorithm

Figure 2 is a heuristic reduct algorithm written in pseudo-code. In line 2, when a new entry c of M is computed, $count(a_i)$ is updated. $count(a_i):=count(a_i)+n/|c|$ for every $a_i \in |c|$. In line 3, Same entries are merged and M is sorted according the length and frequency of every entry. Line 4-9 traverses M and generates the reduct.

4 Primary result of experiment

We applied our rough sets approach in estimating the computation times of data-mining tasks. We differentiated the test case from the historical records by removing the runtime information. Thus, a test case consists of all the information specified except the recorded runtime. The runtime information recorded in the test case was the task's actual runtime. The idea was to determine an estimated runtime using our prediction technique and compare it with the task's actual runtime.

We compiled a history of data-mining tasks by running several data-mining algorithms and recording information about the tasks and environment. We executed several runs of data-mining jobs by varying the jobs' parameters such as the mining algorithm, the data sets and the sizes of the data sets. The algorithms we used were

from the Weka package of data-mining algorithms. We generated several data sets of sizes varying from 1 to 20 Mbytes.

The simulated environment is similar to an actual Grid environment, composed of three machines which installed with GT3. Each machine is interconnected by a switched fast Ethernet. Three distributed machines with different physical configurations and operating systems: a Pentium III running Windows 2000 with an 833-MHz processor and 512 Mbytes of memory; a Pentium 4 running Windows 2000 with a 2.0 GHz processor and 1Gbytes of memory; and a Sun Sparc station running Sun OS 5.8 with a 444-Mhz processor and 256 Mbytes of memory. For each data-mining job, we recorded the following information in the history: the algorithm, file name, file size, operating system, operating system version, IP address of the local host on which the job was run, processor speed, amount of memory, bandwidth, and start and end times. We used histories with 100 and 150 records, and as before, each experimental run consisted of 25 tests.

In our experiment, the mean error was 0.23 minutes, and the mean error as a percentage of the actual runtimes was 7.6 percent. This shows that we accurately estimated the runtime for data-mining tasks on Grid. The reduct that our algorithm selected as a similarity template included the bandwidth, algorithm, file size, dimensionality, and available memory attribute. Figure 4 illustrates the actual and estimated runtimes from one of our experimental runs.

5 Conclusions

We have presented a novel rough sets approach to estimating application run times. The approach is based on frequencies of attributes appeared in discernibility matrix. The theoretical foundation of rough sets provides an intuitive solution to the problem of application run time estimation on K-Grid. Our hypothesis that rough sets are suitable for estimating application run time in Grid environment is validated by the experimental results, which demonstrate the good prediction accuracy of our approach. The estimation technique presented in this paper is generic and can be applied to others optimization problems.

References:

[1] M. Cannataro, D. Talia, P. Trunfio, KNOWLEDGE GRID: High Performance Knowledge Discovery Services on the Grid. *Proc. GRID 2001*, LNCS, Springer-Verlag, 2001.

[2] Foster I. and Kesselman C. (eds.) *The Grid: Blueprint for a Future Computing Inf.,* Morgan Kaufmann Publishers, 1999.

[3] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. of Network and Comp. Appl, 2001.

[4] A.B. Downey , "Predicting Queue Times on Space-Sharing Parallel Computers,"*Proc. 11th Int'l ParallelProcessing Symp.* (IPPS 97), IEEE CS Press, 1997

[5] R. Gibbons , "A Historical Application Profiler for Use by Parallel Schedulers,"*Job Scheduling Strategies for Parallel Processing ,* LNCS 1291, Springer-Verlag, 1997

[6] X.Hu, Knowledge discovery in databases: An attribute-oriented rough set approach, Ph.D thesis, Regina university, 1995.

[7] J.Starzyk, D.E.Nelson, K.Sturtz, Reduct generation in information systems, Bulletin of international rough set society, volume 3, 1998.

[8] S.K.Pal, A.Skowron, Rough Fuzzy Hybridization-A new trend in decision-making, Springer, 1999.

[9] Witten,I,H., and Eibe,F., “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, Morgan Kauffman, 1999.

[10] Keyun Hu, lili Diao and Chunyi Shi: A Heuristic Optimal Reduct algorithm. 22nd Intl. Sym. on Intelligent Data Engineering and Automated Learning (IDEAL2000), Hong Kong, (2002)