V.Siva Kumar, Dr. G.Prakash Babu / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp.299-305

Two Level Resource Discovery And Replication For Secure Utilization In Distributed Computing Environments

V.Siva Kumar, Dr. G.Prakash Babu, Ph.D.
Dept of CSE, Intell Engineering College, JNTU, Anantapur, Andhra Pradesh, India-515001
Associate Professor, Dept of CSE, Intell Engineering College, JNTU, Anantapur, Andhra Pradesh, India-515001

Abstract

This paper presents the details of a novel method for passive resource discovery in cluster grid environments, where resources constantly utilize inter node communication. This method offers the ability to non-intrusively identify resources that have available CPU cycles; this is critical for lowering queue wait times in large cluster grid networks. The benefits include: 1) low message complexity, which facilitates low latency in distributed networks, 2) scalability, which provides support for very large networks, and 3) low maintainability, since no additional software is needed on compute resources. Using a 50-node (multicore) test bed (DETER lab), this paper demonstrate the feasibility of the method with experiments utilizing TCP, UDP, and ICMP network traffic. I use a simple but powerful technique that monitors the frequency of network packets emitted from the Network Interface Card (NIC) of local resources. I observed the correlation between CPU load and the timely response of network traffic. A highly utilized CPU will have numerous, active processes which require context switching. The latency associated with numerous context switches manifests as a delay signature within the packet transmission process. This method detects that delay signature to determine the utilization of network resources. Results show that the method can consistently and accurately identify nodes with available CPU cycles (<70 percent CPU utilization) through analysis of existing network traffic, including network traffic that has passed through a switch (non-congested). Also, institutions where there is no existing network traffic for nodes, ICMP ping replies can be used to ascertain this resource information.

I. Introduction

With the advances in network technologies, applications are all moving toward serving widely distributed users. However, today's Internet still cannot guarantee quality of services and potential congestions can result in prolonged delays and leave unsatisfied customers. The problem can be more severe when the accesses involve a large amount of data. Replication techniques have been commonly used to minimize the communication latency by bringing the data close to the clients. Web caching is a successful example of the technique. However, when the data may be updated, the problem is more complicated. The more models in the system, the higher the update cost will be. Thus, data needs to be carefully placed to avoid unnecessary overhead.

There have been a lot of research works addressing the data model placement issues in Computational Clusters [1, 4, 7, 14, 15]. Generally, the access pattern is used to guide the placement decision. Several considerations like static [1] and dynamic [2, 7, 15] placement decisions and full and partial replication schemes have also been investigated. Partial replication take into consideration reproduce subsets of resources and is generally required for environments showing strong access locality [8, 11]. Almost all the model placement algorithms do not specifically presume full or partial replication [5, 15]. In reality, most of the placement decision algorithms can be applied to both cases by managing the granularity of the resources that are considered. But, each of these algorithms presumes that resources are not dependent on each other. In several applications, each and every request/transaction may have access to multiple resources and, so, bringing correlation among the resources. Going by example, for a read transaction accessing multiple resources, all of these resources at a local site will be able to decline communication overhead. But, in the case of only a part of the data set that is being accessed is modeled at a local site, then the replication will not bring much advantage. This is because the read transaction still requires to be forwarded in order to recover the resources that are left out. The same is applied to an update transaction that is accessing multiple resources. In the case of only a part of the data set that is being updated is modeled, a message is required for sending the update request. So, for getting better model allocation, it is necessary to take into consideration the access correlation among resources. In [12], we have dealt about the model placement issues of correlated resources in mobile environments considering mobile networks that are having a star topology. Further, it presumes that every time the base station node holds the total data set (comprising all resources that may be required by the mobile nodes). This is required for minimizing the mobile unit connection time for the accesses.

Whenever the general Internet environment is taken into consideration, several issues are to be considered. The first one is that there is no need of considering the connection time for Internet based clients and partial replication is needed.

Moreover, a greater complex topology requires consideration for Internet accesses. So, a more sophisticated placement algorithm for correlated resources should be evolved. A crucial issue in model placement algorithms in such kind of environment is that who executes the intensive placement computation. Some of the placement algorithms are centralized [6, 14] that makes the central decision-making site to be overloaded severely. Distributed model placement algorithms, like [10, 15], grant each model site to make localized decisions. They will be able to react to changes in access patterns and naturally divide the computation. In this paper, we construct a dynamic model placement algorithm, which (1) takes into consideration a tree topology network, (2) considers the correlation among resources whenever accessed by the same requests, and (3) gives distributed algorithm that does localized analysis for declining the computation cost. Moreover, our algorithm is adaptive. It takes into consideration 2 schemes. To get optimal solutions, a distributed branch and bound algorithm is utilized to calculate the optimal set of resources models to be designated on the computational clusters. Remember that in commercial applications, transaction access patterns follow the "80/20" rule [3], i.e., just 20% of the transaction types accounts for 80% of the total amount of the transactions. Whenever the transaction access pattern is stabled, the set of resources associated is considerably small. Lanier Watkins et al[] discussed a Passive Solution to the Memory Resource Discovery Problem in Computational Clusters. When compared to existing models this solution is more robust secure and scalable. The prime benefits of this model are low message complexity, scalability, load balancing, and low maintainability but limited their solution to deal with resource requirements of the clusters.

So, the optimal algorithm is obtainable in this case. For dealing with the access patterns that are frequently changed, we suggest resource discovery, replication and utilization algorithms for getting near optimal solutions, which is motivated from the work carried out by Lanier Watkins et al[]. The remaining part of this paper is distributed as follows. Section 2 explains our system model and problem definition, depending on the system model that is defined in [12]. Section 3 delves about the transaction based cost model in a distributed environment. Section 4 gives a distributed partial replication algorithm (ICRDU). Section 5 deals about a intra cluster resource replication and utilization partial replication algorithm (ICRRU). Section 6 is about the experimental study results and Section 7 concludes the paper.

II. Resource allocation System Format

Studies reveal that the networks can be disintegrated into connected autonomous systems, which are under separate administrative control [9]. These autonomous systems are generally treated as clusters. By this way the underlying topology of the widely distributed system as a cluster based general graph is modeled by us. For scaling the system, we presume that there is a data server in each and every cluster. In this research, we take into consideration only the data model placement on the computational clusters, and the model assignment inside each cluster can be treated separately. We presume that the actual copy of the entire set of N resources that are to be accessed by users is put up at a primary data server that is indicated as O. Let D_O indicate the N resources on the data server O, then, $D_O = \{d_1, d_2, ..., d_N\}$ and $|D_O|$ = N. Whenever applications use data saved in this primary sever, they generally follow a shortest path tree routing to the data source that is positioned at the primary server O [9]. Study reveals that almost all routings are stable in days or weeks and so the routing paths can be looked upon as a tree topology that is rooted at the primary server O. So, we take into consideration replication the widely distributed system as a tree graph, indicated as T, rooted at the primary server O. To expedite efficient data accesses by users in the Internet, a subset of resources in D_O are modeled on computational clusters in T.

Let D_x indicate the resources on a data server x, then $D_x \subseteq D_O$, and $|D_x|$ is the number of modeled resources in D_x. Take a set of resources Ω, let R(Ω) indicate the resident set of Ω that is the set of computational clusters hold Ω or a superset of Ω, i.e. R(Ω) = {x∈T | Ω ⊆ D_x }.The divided system bolsters both read and update requests and each request can have access to an random number of resources in D_O. For each and every data server x in T, there are a number of clients that are connected to it. The requests that are given by these clients can be seen as requests from the data server x. Presuming that clients can give both read and update requests. Let t indicate a transaction, it can be a read transaction or an update transaction. Let D (t) indicate the set of resources read or updated by t, D (t) ⊆ D_O, and D (t) is designated as a transaction-data set of t. For a transaction t accessing D (t) given by a data server v, if D (t) ⊆ D_V, then t is served

locally. Contra wise, t is send to the closest data server, which can serve t.

For an update transaction t, the data server that performs t requires to send the update to other computational clusters through the edges in a tree that only possess of all those computational clusters *x in T, where D (t) ∩ D_{*x} ≠ φ. Our aim is to optimally allocate the models resources in D_O to computational clusters in T in such a way that aggregate access cost is minimized for a given client access pattern. We define a model placement that is having the minimal cost as an optimal model placement of D_O (OptPl (D_O)). Observe that optimal placement solutions may not be unique. Take a data set Ω, R (Ω) in an OptPl (D_O) is an optimal resident set of Ω. During this research, we don't take into consideration node capacity constraint, message losses, node failures, and the consistency maintenance issues. The communication cost that is brought up by model placement and de-allocation is also not taken into consideration.

III. Operation Based limited Replication rate format

During this section, we make definition of operation based limited replication rate formats for correlated resources in a tree graph, depending on the cost models that are defined in [Tum06a]. Take 2 computational clusters x and y, where data server y is the parent data server of x. Here, we only emphasize model allocation decisions at y and model de-allocation decision at x. This is because model allocation and de-allocation models and approaches can be applied in a similar fashion to other computational clusters in tree T. Take a set of resources Ω, Ω ⊆ D_O . If Ω is modeled to x, a read transaction t, D (t) ⊆ Ω, can be served locally and one message can be saved. We take into consideration this as one unit of advantage for modeling Ω on x. But, because of the replication of Ω on x an update transaction t, D (t) ∩ Ω ≠ φ, requires to be send to x. We regard this as one unit of cost put up by modeling Ω on x.

We define update_a (Ω, x) as the cost if Ω is modeled from x's parent data server (y) to x, where update_a(Ω, x) = Σ|W(S)|, where (S ∩ Ω ≠ φ) ∧ (S ∩ D_x = φ)

We define read_a(Ω,x) as the additional advantage if R is modeled to x, where

Read_a(Ω, x) = Σ|Q(S)|, where (S ⊆ Ω ∪ D_x) ∧ (S ∩ Ω ≠ φ)

Now, we define cost_a(Ω, x) as the data object access cost for modeling Ω at x from y, where

cost_a(Ω, x) = update_a(Ω, x) – read_a(Ω, x)

Let S^a_{opt} indicates the set which minimizes cost_a(Ω, x),

i.e. S^a_{opt} = arg min (cost_a(Ω, x)). The transaction based partial replication models S^a_{opt} to x if cost_a (S^a_{opt}, x) < 0.

In ICRDU, a node x can only de-allocate a data set Ω if and only if x is the leaf node of R(Ω) depending on the knowledge of the model on its child nodes, and x has held the model of Ω for since the end of last time period. We indicate Dd_x as the data set on x in such a way that x is the leaf node of R(Dd_x) and x has held the model of Dd_x since the end of last time period. A set of resources, Ω ⊆ Ddx, may require de-allocation from x if modeling Ω is not going to benefit in terms of communication cost. Let update_d(Ω, x) indicate the de-allocation advantage for de-allocating Ω. In essence, the de-allocation advantage of Ω comprises all update transactions from y, accessing a subset of resources in Ω. update_d(Ω, x) = Σ|W(S)|, where (S ⊆ Ω) ∧ (Ω ⊆ Dd_x) ∧ (S ≠ φ)Let read_d(Ω, x) indicate the de-allocation cost. In essence, the de-allocation cost of Ω comprises all read transactions that are issued by x, accessing a subset of Dd_x and some resources in Ω. read_d(Ω, x) = Σ|Q(S)|, where (S ∩ Ω ≠ φ) ∧ (S ⊆ D_x) ∧ (Ω ⊆ Dd_x). The model de-allocation access cost_d(Ω, x), is the dissimilarity between read cost and update advantage for de-allocating Ω from x.cost_d(Ω, x) = read _d(Ω, x) – update_d(Ω, x)Let S^d_{opt} indicate the set taht minimizes cost_d(Ω, x), i.e. S^d_{opt} = arg min(cost_d(Ω, x)). The transaction based partial replication de-allocates S^d_{opt} from x if cost_d (S^a_{opt}, x) < 0.

IV. ICRDU Algorithm

From [13], the model placement problem in T can be resolved by designating models in each subtree respectively, and the placement of models on each data server in T can also be done independently to its neighboring computational clusters in T. Moreover, the set of resources on v, D_V, is a subset of D_w, where w is the parent data server of v in T. The model placement on v can be treated only dependent on w and they are not dependent on other computational clusters. By this way, the model placement problem can be resolved by a distributed optimal partial replication (ICRDU) algorithm (including ICRDUA and ICRDUD, in Fig. 1). At the end of time period,τi, parent data server y creates

model allocation decision for its child data server x by utilizing ICRDUA, depending on its knowledge of the models on its child data server x that is got from the end of last time period τ_i -1.At the time of getting the models that are allocated by y, data server x creates model de-allocation decision for itself by utilizing ICRDUD. In ICRDUD, data server x can only de-allocate a set of resources Ω from x only if x is a leaf data server of R(Ω), and it has hold the model of resources in Ω since the end of time periodτi-1. Let S^a_{opt} (y, x, τ_i) indicate the data set calculated by ICRDUA by data server y for its child x at the end of τ_i. Let S^d_{opt} (x, τ_i) indicate the data set calculated by ICRDUD by data server x for itself at the end of τ_i. See that in for averting replication oscillation, ICRDU need that model de-allocation decision must be made for x after x has obtained model from its parent data server y in the same time period τ_i. Moreover, in ICRDUA and ICRDUD, we require calculating the lower bound cost for each new transaction-data union $S_{TranListIndex} \cup S$. One easy way to calculate the lower bound cost is read_a (S_{super}, x) – update_a (($S_{TranListIndex} \cup S$), x), where $S_{super} = S \cup \bigcup_{i=TranListIndex}^{TranList.Length-1} S_i$, and Si is the data set accessed by TranList[i], TranListIndex ≤ i ≤ n.

Min Cost: cost_a($S^a_{opr}(y, x, \tau_i), x$), initialized to +∞;

Min Cost_d: cost_d($S^a_{opr}(y, x, \tau_i), x$), initialized to +∞;

S: the contained set during the search, initialized to ϕ;

Lower Bound Cost(S_{new})& Lower Bound Cost_d(S_{new}):

Defined following the algorithm

Tran List: the set of distinct transactions that access data;

Tran List Index: initialized to 0;

If(TranListIndex<TranList.Length){

$S_{TranListIndex}$ =TranList;

[TranlistInded].getDataObjectSet()

$S_{new} = S \cup S_{TranListIndex}$;

If(LowerBoundCost(S_{new})<MinCost)

ICRRU-A($S_{new,}$ TransListIndex+1);}

ICRRU-A (S, TranListIndex+1);}

else if(cost a(S, x)<Min Cost) {

MinCost=cost a(S, x);

$S_{opt^a}(y, x, \tau_i)$ = S - D_{Mc}; }

if (TranListIndex<TranList.Length) {

$S_{TranListIndex}$ = TranList;

[TranListIndex].getDataObjectSet();

$S_{new} = S \cup S_{TranListIndex}$;

If(LowerBoundCost_d(S_{new}) ≤ MinCost_d){

ICRRU-A ($S_{new,}$ TransListIndex+1);}

ICRRU-A (S, TranListIndex+1);}

Else if(cost_d(S,x) ≤ MinCost_d) {

MinCost_d=cost_d(S,x);

$S_{opt^a}(x, \tau_i)$ = S; }

Fig 1: ICRDU Algorithm

Theorem 1 Let L indicate the tree level. ICRDU stabilizes and converges to optimal in 2L time periods. Proof: The proof is removed because of space limitation. Kindly refer [13] for the full proof.

V. ICRRU Algorithm

ICRRU since the run time of OPR algorithms develops exponentially with the number of transactions; we construct a intra cluster resource replication and utilization replication algorithms ICRRU, in which the heuristic Expansion-Shrinking algorithms (discussed in [12]) is utilized.

S^a_{heu}, * S^a_{heu} ** S^a_{heu} .S: data sets and initialized to ϕ;

* * L_Y ; a record log vector and initialized to ϕ;

Make a copy of L_Y for cost computing;

While* $L_Y \neq \phi$

for all data set recorded in * L_Y choose the data set ***S*** such that $|S - S^a_{heu}|$ is minimal:

if $(S \cap ** S^a_{heu} \neq \phi)$ delete the record with ***S*** from* L_Y.

else if ***cost a(S, x)*** < ***0 &&*** $S \cap ** S^a_{heu} \neq \phi$

then $S^a_{heu} = S^a_{heu} \cup S$;

else if ***(cost a(S, x)*** >= ***0***

&& $S \cap S^a_{heu} \neq \phi) \| (S \subseteq S^a_{heu})$

then ***S^{a}ht$_U$ = S";***w ***S;***

else if (***cost_a(S, x)*** >= 0 && S n = (S c

then S^a_{heu} remains unchanged;

if the size of S^a_{heu} has been increased,

then delete the record with S from * L_Y , and

move all records from** L_Y to* L_Y .

Else move the record with S from * L_Y to ** L_Y .

* $S^a_{heu} = S^a_{heu}$;

move all records from** L_Y to * L_Y , and $S^a_{heu} \neq \phi$;

Assume that* S^a_{heu} has been allocated

to $(*D_x = D_x \cup *S^a_{heu})$, and re-do log reduction

in* L_Y .

while* $L_Y \neq \phi$

for all data set recorded in* L_Y

choose the data set S with $|S - S^a_{heu}|$ is minimal;

if cost_a $(D_y - *D_x - S - S^a_{heu}x) < \cos t\_a(D_y - *D_x - S - S^a_{heu}x)$

$S^a_{heu} = S^a_{heu} \cup S$;

if the size of has been increased,

then delete the record with ***S*** from* L_Y

and move all records from* L_Y to* L_Y .

else move the record with S from* L_Y to** L_Y .

** $S^a_{heu} = D_y - *D_x - S^a_{heu}$;

If (cost(** S^a_{heu} , x)<0) ** $S^a_{heu} = \phi$; ;

S^a_{heu} =** $S^a_{heu} \cup *S^a_{heu}$:

Fig 2: ICRRU algorithm for Model allotment(ICRRU-A)

The heuristic Expansion-Shrinking algorithms now utilize the cost functions that are defined in Section 3. Moreover, in order to make ICRRU stabilize, we combine the Set-Expansion and Set-Shrinking algorithms. In both algorithms, the data server first performs the Set-Expansion algorithm and calculates a data set, which is a subset of the optimal data set that is calculated by the optimal algorithms. Now, it performs the Set-Shrinking algorithm presuming that the data set calculated by the Set-Expansion algorithm is already designated to the child data server or de-allocated from itself. Finally, we merge the data sets calculated by the two algorithms as the final data set to be designated from y to x or de-allocated from x. The model allotment process of ICRRU algorithm (ICRRU-A) and the model withhold for ICRRU algorithm (ICRRU-W) are indicated in Fig. 2 and 3, respectively. ICRRU is defined as given below. At the end of time period, τi, parent data server y creates model allocation decision for its child data server x by performing ESRA, depending on its knowledge of the models on its child data server x got from the end of last time period τ_{i-1}. After getting the models from y, data server x performs ESRD and makes model de-allocation decision for itself. At this point, data server x can only de-allocate a set of resources Ω from x only if x is a leaf data server of R(Ω), and it has held the model of resources in Ω since the end of time period τ_{i-1}. Observe that for averting replication oscillation, ICRRU needs that model de-allocation decision must be made for x after x has obtained model from its parent data server y during same time period τ_i .

Theorem 2 Let L indicates the tree level. ICRRU stabilizes in at most 2L time periods.

Proof: The proof is removed because of space confinement. Kindly refer [13] for the full proof.

S^d_{heu}. $*S^d_{heu}$, $** S^d_{heu}$. ***S:*** data sets and initialized to ϕ; ** L_x temporary record log vector and initialized to ϕ;

Make a copy of $\boldsymbol{L_x}$ for cost computing:

while $(*L_x \neq \phi)$

for all data set recorded in $*L_x$ choose

the data set S such that $|S - S^d_{heu}|$ is minimal;

if $(S \cap ** S^d_{heu} \neq \phi)$ delete the record with ***S***

from $*L_x$

else if $\boldsymbol{cost\_d}(S, x) < 0 \,\&\&\, S \cap S^d_{heu} = \phi$

then $S^d_{heu} = S^d_{heu} \cup S$;

else if (COST $\boldsymbol{d}(S^d_{heu} \cup S, x) < \cos t\_a(S^d_{heu}, x)$

then $S^d_{heu} = S^d_{heu} \cup S$;

else if (

$\boldsymbol{cost\_d}(S, x) < 0 \,\&\&\, S \cap S^d_{heu} = \phi) \,\|\, (S \subseteq S^d_{heu})$

then S^d_{heu} remains unchanged,

if the size of has been increased

then delete the record with S from $*L_x$ and

move all records from $** L_x$ to $*L_x$

else move the record with ***S***

from $*L_x$ to $** L_x$

$*S^d_{heu} = S^d_{heu}$;

move all records from $** L_x$ to $*L_x$ and =

$S^d_{heu} = \phi$;

Assume $*S^d_{heu}$ has been de-allocated from ***x***

$(*D_x = D_x - *S^d_{heu})$ and re-do log reduction

in $** L_x$

while $*L_x \neq \phi$

for all data set recorded in $*L_{BC}$,

choose the data set S with $\left|S - S_{heu}^{d}\right|$ minimal:

if

$$cost\_d(*D_x - S - S_{heu}^{d}, x) < cost\_d(*D_x - S_{heu}^{d}, x)$$

$$S_{heu}^{d} = S_{heu}^{d} \cup S;$$

If the size of S_{heu}^{d} has been increased, then

delete the record with S from $*L_x$

and move all records from $**L_x$ to $*L_x$

else move the record with ***S***

from $*L_x$ to $**L_x$.

$$**S_{heu}^{d} = *D_x - S_{heu}^{d};$$

If(cost_d($(**S_{heu}^{d}, x) \geq 0) **S_{heu}^{d} = \phi;$

$$S_{heu}^{d} = **S_{heu}^{d} \cup S_{heu}^{d};$$

Fig 3: ICRRU algorithm for model withhold (ICRRU-W)

VI. Simulation Results

In the simulation, comparison of the ICRRU algorithm with the commonly utilized distributed frequency-based partial replication (DFPR) scheme is done for studying its performance and effectiveness on message saving. Observe that the frequency based algorithm defined in [12] can easily be acclimatized to the distributed solution, DFPR that is fairly direct that each and every node makes local decision depending on the access frequency. The tree network that we chose comprises 20 nodes with height of 6. Initially, the root node O (that indicates cluster HMSC) hosts all the resources in data set D_O. Each and every node x in the tree is haphazardly allocated a partial set of the resources $D_x \subseteq D_O$. The requirement is that $D_x \subseteq D_y$ if y is an ancestor of node x in the tree network. The metric that we take into consideration is the product of the number of messages and the numbers of hops these messages are send in order to process the requests in the tree network. Going by example, if a request that is issued locally can be served locally, then the number of message is counted as 0. In the case of the request being served at its parent, then one message is taken into count.

In this simulation, we utilize one PC platform for simulating 20 server clusters and the period of execution for each and every experiment spans 10 time periods. The simulated distributed algorithm requires retaining the history logs for every request (most of the information is not required in a real system). Because of the excessive I/O, the simulation process becomes very time-consuming (observe that the time is not because of the algorithm itself). Helping to avert prejudiced access patterns, multiple access patterns are produced haphazardly and the data collection process is redone for each and every pattern. In order to balance between the confidence level of the experimental results and the time for the simulation study, we select repeating the procedure 100 times (i.e., producing 100 distinct access patterns). Whenever the number of resources and the number of transactions increase, the system is likely to overload and we have to further decline the repetition to 10 times (i.e., utilizing only 10 distinct access patterns). The final data given in the following subsections are the average of these trials.

6.1 Transaction Generation

We presume that the resources accessed by most of the transactions follow various patterns. Moreover, because of access locality, we presume that the patterns will be stable for some time periods. In this experimental study, we first produce a series of transactions and scrutinize the resources they access. The set of resources accessed by a transaction is defined as a resource bundle. We produce transactions till M distinct resource bundles are identified. Observe that the resource bundles may have overlapping resources but no two resource bundles are identically same. For producing a transaction, the number of resources to be accessed by the transaction, indicated as μ, is first ascertained. μ is surrounded by TS and follows a Zipf distribution. More specially, the chance of a transaction having data set size μ is proportional to $1/\mu^{Z_S}$, where Z_S is the skew parameter of the Zipf distribution [13]. Now, the μ resources are ascertained. Let NS indicate the total number of resources we have taken into consideration in the experimental study.

Each and every data object is ranked. The chance that a data object is accessed by a transaction is proportional to $1/r^{Z_D}$ (also a Zipf distribution), where Z_D is the skew parameter. Lastly, if a transaction is read only or update is ascertained by the ratio, R/W, in a uniform distribution. Here R is the total number of read transactions given by P_{MC} and W is the aggregate number of update transactions given by nodes other than P_{MC}. From the first batch of transactions produced, we get M resource bundles and utilize them as the basis in order to formulate an access distribution. The M resource bundles are separated into 2 clusters, comprising the read cluster having M_1 resource bundles for the read transactions and the write cluster having M_2 resource bundles for the update transactions, where $M_1 + M_2 = M$ and M_1 / M_2

= R/W. For each and every cluster, 0.5 M_1 (or 0.5 M_2) virtual resource bundles are appended and the pre-generated M_1 (or M_2) resource bundles along with the virtual resource bundles are ranked. The transactions are produced in the similar way as talked about in [12]. Regarding simulation, we presume that in a time limit T, T_N transactions are given from each data server in the tree.

In this manner, in each time period, all computational clusters issue T_N transactions that are haphazardly produced depending on the same access pattern. For making the model allocation decision for a child data server, each and every data server records the number of read transactions that are sent from the child data server and the number of update transactions received to the child data server. In a similar fashion recording is also done in order to make the model de-allocation decision for a child data server. In case of a read transaction being issued locally or sent from its children can be served locally, then the transaction is recorded by the data server records in its local log. Apart from that it also records the update transactions spread from its parent in the local transaction log. After the time period, a model allocation decision is done for each and every child data server depending on the transaction log that is recorded for that child data server and resources are reproduced and assigned to the child data server if required. Now, All computational clusters make the de-allocation decision depending on its local transaction log. Performance data are calculated depending on the allocated transactions and the model set at each data server in each time period, just before it gets another model set from its parent. Observe that in this simulation, the number of resources selected is small as the amount of memory needed for recording logs for each data server in each time period. A much larger number of resources can be selected in case the ICRRU is operated in a real system, as indicated in [12].

6.2 The Stabilization

Comparison of the intra cluster resource replication and utilization algorithm with the distributed frequency-based algorithm is done for observing the stability of the two algorithms. Fig. 4 gives the number of message necessary for processing every transaction that is issued in the system for the 2 algorithms. The number of messages needed in the system drops very speedily in the first 4 time periods for both algorithms. The percentage of the number of messages declined at the 1st time period is much greater compared to that of the 2nd time period, which is instead greater than that of the 3rd time period, so on and so forth. After the completion 4th time period, the number of messages that are required in the system remains stable for both algorithms, less than 40% messages are required. At any time period, the ICRRU requires less number of messages than the ICRDU algorithm but the difference is negligible.

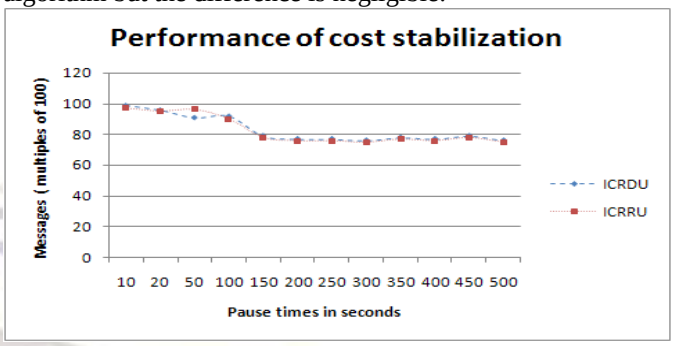

Fig 4: The performance of the cost stabilization by ICRDU and ICRRU

6.3 Scalability in terms of Resources and Transactions proportionality

Comparison of the intra cluster resource replication and utilization algorithm ICRRU with the distributed frequency-based algorithm DFPR is done and calculate the effect of NS (the number of resources in D_O) on their performance. The parameters in this experiment are set as given here: Z_D = 0.2, Z_S = 0.5, R/W = 0.9, T_S = 3, T_N = 50 * N_S. M and NS changes from 10 to 100 (M is adjusted in order to make sure that we have access to almost all resources). Fig. 5 indicates the number of messages necessary for processing all the transactions given by the system for the two algorithms. If N_S increases, the number of messages necessary for the two algorithms increases as the number of transaction also increases. But, the difference of the message necessary for the 2 algorithms will be remained in a stable way (the number of message necessary for resource discovery and replication with ICRRU&ICRDU is 17% of that needed for only resource discovery[16]), even though the variation in the number of messages increases.

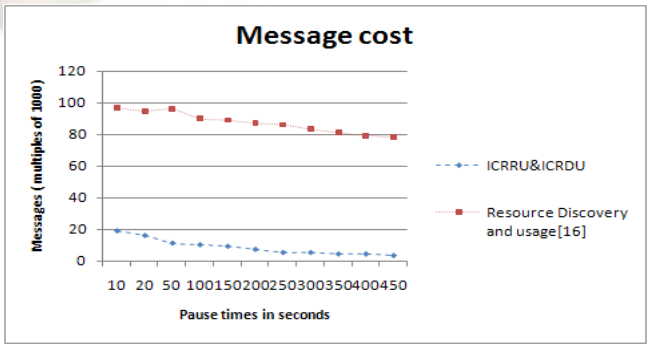

Fig 5: Scalability of resource discovery and usage[16] and ICRRU&ICRDU

VII. Summary

In this paper, we scrutinize the dynamic designation of correlated resources in computational clusters. We model the topology of the network that is formed by the computational clusters in the distributed system as a tree. Initially we show that model designation decisions can be done locally for each and every model site in a tree network, using data access knowledge of its neighbors. Now, we develop a new replication cost model for correlated resources in Internet environment. Depending on the cost model and the algorithms that are used in previous research, we have put in effort to develop a inter cluster resource discovery and utilization algorithm (ICRDU) for correlated data in internet environment. ICRDU has 2 sub algorithms, ICRDUA and ICRDUD, and it is indicated that ICRDU stabilizes and converges in 2L time periods. Here, L is the height of the tree.

Now, an intra cluster resource replication and utilization algorithm (ICRRU) is now developed for making model placement decisions in an efficient manner. The ICRRU has 2 sub algorithms, namely, ICRRU for model allotment and ICRRU for model withhold, and it is indicated that ICRRU stabilize in 2L time periods. The algorithm gets near optimal solutions for the correlated data model and produces significant performance gains. The simulation results indicate that the intra cluster resource replication and utilization allocation algorithm outperforms the general resource discovery and utilization schemes in a significant way.

Reference

[1] P. Apers. Data allocation in distributed database systems. ACM Transactions on Database Systems Vol. 13, No. 3 (Sept.).1988.

[2] A. Bestavros and C. Cunha. Server-initiated document dissemination for the WWW. IEEE Data Engeneering Bulletin 19, 3 (Sept.), 3–11. 1996.

[3] S. Ceri, S. B. Navathe, and G. Wiederhold. Distribution design of logical database schemas. IEEE Transactions on Software Engineering, Vol. SE-9, No. 4. 1983.

[4] D. Dowdy and D. Foster. Comparative models of the file assignment problem. Computing Surveys, 14(2), 1982.

[5] Y. Huang, P. Sistla, and O. Wolfson. Data replication for mobile computers. In Proceeding of 1994 ACM SIGMOD, May 1994.

[6] K. Kalpakis, K. Dasgupta, and O. Wolfson. Optimal placement of models in trees with read, write, and storage costs. IEEE Transactions on Parallel and Computational Clusters. Vol 12, No. 6. 2001.

[7] D. Kossmann. The state of the art in distributed query processing. ACM Computing Surveys (CSUR). Volume 32, Issue 4. December 2000.

[8] Z. Lu and K. S. McKinley. Partial collection replication versus cache for information retrieval systems. In Proceedings of the ACM International Conference on Research and Development in Information Retrieval, Athens, Greece, July 2000.

[9] V. Paxson, End-to-End Routing Behavior in the Internet, IEEE/ACM Transactions Networking, 5(5) (1997) 601-615.

[10] J. Sidell, P. Aoki, A. Sah, C. Staelin, M. Stonebrakeer, and A. Yu. Data replication in Mariposa. In Proceedings IEEE Conference on Data Engineering (New Orleans, LA, Feb.), 485–494. 1996.

[11] A. Sousa, F Pedone, R Oliveira, and F Moura. Partial replication in the database state machine. In Proceedings of the IEEE International Symposium on Network computing and Applications. 2001.

[12] M. Tu, P. Li, L. Xiao I. Yen, and F. Bastani. Model placement algorithms for mobile transaction systems. IEEE Transactions on Knowledge and Data Engineering. Vol. 18, No. 7. 2006.

[13] M. Tu. A data management framework for secure and dependable data grid. Ph.DDissertation, UT Dallas. http://www.utdallas.edu/~tumh2000/ref/Thesis-Tu.pdf. July 2006.

[14] O. Wolfson and A. Milo. The multicast policy and its relationship to modeled data placement. ACM Trans. Database Systems. Vol.16, No.1. 1991

[15] O. Wolfson, S. Jajodia and Y. Huang. An adaptive data replication algorithm. ACM Transactions on database systems. Vol22. No.2 pages 255-314. 1997.

[16] Lanier Watkins, William H. Robinson, Raheem A. Beyah: A Passive Solution to the Memory Resource Discovery Problem in Computational Clusters. IEEE Transactions on Network and Service Management 7(4): 218-230 (2010)