2011 11th IEEE International Conference on Computer and Information Technology

Proximity-Aware Resource Discovery Architecture in Peer-to-Peer based Volunteer Computing System

Toktam Ghafarian-M.
Ferdowsi University of Mashhad
Department of Computer Engineering,
Khayyam Institute of Higher Education
Mashhad ,Iran
ghafarian@stu-mail.um.ac.ir

Hossein Deldari
Ferdowsi University of Mashhad
Department of Computer Engineering,
Azad University of Mashhad
Mashhad ,Iran
hd@ferdowsi.um.ac.ir

Mohhamad-H. Yaghmaee-M.
Ferdowsi University of Mashhad
Department of Computer Engineering
yaghmaee@ferdowsi.um.ac.ir,yaghmaee@ieee.org

Abstract— **Volunteer computing which benefit from idle cycles of desktop PCs over the Internet can integrate power of hundreds to thousands desktop systems to achieve high computing power. Centralized volunteer computing system has dedicated servers to maintain information about the resources. However, in the decentralized system resource information is distributed in the system. Resource discovery architecture is a key factor for peer-to-peer based volunteer computing system. Usually, there is a complex relationship between the distribution of resource information and performance of a system. The main contribution of this paper is to develop a proximity-aware resource discovery architecture for peer-to-peer based volunteer computing system. This architecture has simplicity of a centralized system and can achieve close performance compared to this system, and it is scalable. Furthermore, proposed architecture distributes jobs among the resources fairly. Two resource discovery algorithm with and without proximity-aware feature are compared. The proximity-aware resource discovery algorithm can gain a better result.**

Keywords- resource discovery; peer- to- peer network; volunteer computing system; proximity-aware; Cycloid; load balance.

I. INTRODUCTION

Desktop grid in which cycles are taken from idle desktop computers is an attractive cost efficient platform for running scientific projects with the significant computational requirement [1]. Some of these projects are SETI@home [11], Folding@home [13], EDGes project [2], Climateprediction.net [3] and World Community Grid [14].

Centralized volunteer computing (VC) system needs dedicated servers which keep resource information in the system. Server provides jobs to a set of volunteer machine distributed across the internet. The server must guarantee the robustness and reliability of the system by keeping the status of all jobs running on unreliable volunteer resources. Some of the volunteer computing systems are BOINC [3, 20, 19], condor-like grid system [26, 27, 29], Entropia [4], XtremeWeb [28], Aneka [21], SZTAKI [20] and QADPZ [15]. In order to build VC system, which relies only on volunteer and non-dedicated resources a decentralized and self-organized VC system needs to be developed. Peer-to-Peer (P2P) based VC systems exhibit self-adjusting and scalable properties, which make them able to deal with the limiting characteristics of non-dedicated resources. These systems distribute resource information on the P2P network so that resource discovery architecture is a key factor for the efficient P2P based VC system. The main idea of this work is to introduce proximity-aware resource discovery architecture in the P2P based VC system. It is named CycloidGrid. The Proximity-aware feature of CycloidGrid can decrease the communication overhead and increase system performance. Since in the VC system, millions of heterogeneous resources are disseminated across geographically distributed nodes, consequently running a job on the node with lower delay time to requester can decrease communication overhead. Other works in this area often use logical proximity derived from the P2P overlay; meanwhile logical proximity doesn't match physical proximity, in reality. CycloidGrid distributes computational jobs on volatile nodes in such a way load balance exists between these nodes. In this architecture, a resource discovery algorithm running on each volatile node chooses a node with shorter round-trip time (RTT) among capable nodes for running a job in the system. The proposed system can acquire a reasonable result compared to the centralized systems. However, it reduces maintenance cost and risk of single point failure of a centralized system and is scalable. It can tolerate against the churn and have better behavior compared to the similar system without proximity-aware feature.

The remainder of this paper is organized as follows: Section 2 presents a literature review. Section 3 discusses proposed P2P based proximity aware resource discovery

978-0-7695-4388-8/11 $26.00 © 2011 IEEE
DOI 10.1109/CIT.2011.37

architecture (CycloidGrid), section 4 represents performance evaluation, and finally section 5 contains a conclusion.

II. LITRITURE REVIEW

In [5] a desktop grid system based on CAN [6] has been introduced. In this system, a structured P2P overlay CAN has been customized in order to handle the complex query of available resources according to the job constraint. Each resource attribute has been associated with a distinct dimension in the CAN. Resource discovery algorithm has been considered as routing problem in the CAN space. It searches a node whose coordinate in all dimensions satisfies or exceeds the job's constraints. All the jobs are independent and are injected to the system from any available nodes in the system. The matchmaking algorithm distributes jobs between capable resources fairly so that load balance exists in the system. The Proximity-aware feature is neglected in this work, and the system can't handle a job with many constraints because the dimension of customized CAN increase. It causes degradation of overall system performance.

PastryGrid [7] has been suggested for management of institutional desktop grid over a fully decentralized P2P network. A distributed application in this system consists of one or more modules with precedence constraints between them. Each application has a rendezvous node in the system which contains modules and necessary data of them. This node initializes the execution phase of application. The resource discovery protocol has based on pastry routing algorithm. When a machine receives one task for execution, it contacts a rendezvous node of its application to receive necessary data. After it completes a task, it can participate in the discovery process for the successor of this task and so on. Resource discovery algorithm only found capable nodes for running a task, but there is not any load balance between nodes in the system. Communication overhead among the nodes is neglected in the resource discovery algorithm, and their resource discovery algorithm is not proximity-aware.

In another system [9] a super-peer based [10] volunteer computing system has been proposed. Nodes have different roles such as job manager, data cache nodes, data sources, super-peer nodes and workers. Resource discovery algorithm consists of two phases: job-assignment and data-download phase. In the job assignment, job manager generates a number of job's advert and sends them to the local super-peer and some of the other super-peers in the system. Workers generate a job query. Then job query travels the network through the super peer interconnections until the time-to-live parameter decreases to zero or job query finds matching job advert. In the data-download phase, the worker sends a data query and downloads the data file from data centre. In this work, data file of each job is downloaded with regarding to distance and available bandwidth, but load balance didn't exist in the system.

Abdullah et al [30] has been suggested a dynamic, self-organizing model for ad hoc grid. In this work, three types of agent named customer, producer and matchmaker are introduced. The whole identifier space has been divided into zones, which have a responsible matchmaker. The matchmaker uses a continuous double auction to perform resource allocation and looks for the matches between the producers and consumers. It is done by matching offers (starting with lowest ask price and moving up) with requests (starting with the highest bid price and moving down). Load balance only exists between matchmakers, not among the producers. Physical distance and bandwidth consideration has not been studied in this work, and it is not proximity-aware algorithm.

As stated in [25] a hybrid of epidemic information dissemination [22, 23] and P2P structured overlay chord [24] has been used. In this work, a regular multicast is applied for resource discovery. Each node that attends in the multicast distribution halves its sending interval after every contact. This distribution scheme causes a spanning tree covering all the nodes in the system. In this work, a resource discovery algorithm has not studied job constraints. They supposed all the resources can run every job. Furthermore, load balance didn't exist in this work among peers. Only logical distances used to determine closeness. Meanwhile logical distance doesn't match physical distances. The Proximity-aware feature has not been considered.

In [16] three main agents have been defined in their system. These agents are worker, client and matchmaker. A worker sends advertisement to the multiple matchmakers in the system. When a client needs resources, it asks matchmaker and matchmaker search between advertises in order to find possible matches. In this work load balance didn't exist in the system between workers. Furthermore, the resource discovery algorithm has not been studied proximity-aware feature. Communication overhead has not been studied in this work.

The proximity-aware feature that is studied in this paper is a novelty of proposed approach comparing to another works in this area. Also the proposed algorithm considers load balancing feature. This feature is studied in a few works in this domain.

III. CYCLOIDGRID:PROXIMITY-AWARE RESOURCE DISCOVERY ARCHITECTURE IN PEER-TO-PEER BASED VOLUNTEER COMPUTING.

A. Basic Framework

CycloidGrid is built on cycloid [12]. Cycloid is a constant-degree structured P2P overlay with $n = d.2^d$ nodes, where d is a dimension. It takes a time complexity $O(d)$ hops for the lookup request with $O(1)$ neighbors per node. Each node in the cycloid is presented by a pair of indices $(k, a_{d-1}a_{d-2}...a_0)$ where k is a cyclic index and $a_{d-1}a_{d-2}...a_0$ is a cubical index. The cyclic index is an integer number from *0* to *d-1* and the cubical index is a binary number ranging from *0* to $2^d - 1$. All nodes are classified into some clusters. These clusters are differentiated by $a_{d-1}a_{d-2}...a_0$ and inside the cluster the nodes are identified by k. However, all clusters ordered by their cubical indices mod 2^d on a large cycle while inside each cluster the nodes are ordered by their cyclic index mod d. Each node keeps a routing table and two leaf sets with seven entries to sustain its connectivity to the

rest of the system. The largest cyclic index in a local cycle is named the primary node of the local cycle. The Cycloid DHT assigns keys onto their ID space by applying consistent hashing function. For a key or node, its cyclic index adjusts to its hash value of key or IP address modulated by *d*, and the cubical index adjusts to hashing value divided by *d*. A key will be assigned to a node whose ID is closest to its ID. In the proposed framework cycloid is chosen for two reasons:

First: Cycloid has small constant size routing table. This routing table doesn't increase with the growing number of peers unlike other P2P overlay such as Chord [24], CAN [6], Pastry [8] and Tapestry [31]. Consequently, a small constant size routing table can help CycloidGrid in consuming disk space of a volunteer node and more space can be used for running a job.

Second: Cycloid classifies peers into some clusters. As it is discussed in the next section, proposed architecture classifies resources into some clusters by using a decision tree. In consequence, cycloid with cluster based structure prefers to other P2P systems in this research.

B. CycloidGrid

CycloidGrid is a proposed architecture for resource discovery in P2P based VC system. It has three types of node. These nodes are called reporting node, host node and client node. Reporting node is a node responsible for keeping resource information in the system. Host node is a node which donates its resource for running a job. Client node has a request for running a distributed application. Each distributed application consists of multiple independent jobs in this research.

Each resource in the system is described by a set of attributes. In this research, these attributes are CPU speed, the amount of RAM size, the amount of hard disk space available, operating system type, and model. The first three attributes are continuous statistical variable; meanwhile the last two ones are discrete variable. A decision tree is made on these attributes to classify resources into some clusters as shown in Fig. 1. A decision tree is a tree-structured plan of a set of attributes to test in order to predict the output. In the decision tree, non leaf nodes are labeled with attributes and the arcs out of a node are tagged with possible attribute values for that attribute. The leaves of the tree are labeled with classification.

In Fig. 1 model and operating system type are discrete statistical variable and exact value of them is tested. Hard disk space, RAM size and CPU speed are continuous variable and divided into four discrete ranges. These attributes are static and don't change during the life time of a resource. Four attribute values are selected in each level. This number of attribute values is a trade off between the number of clusters and cover of various values for these attributes. Consequently, the number of clusters in the DT is $4^5 = 1024$ clusters. Information of all resources with the same attribute values is gathered in the same cluster of DT.

Clusters in the CycloidGrid are grouped into two types of clusters.

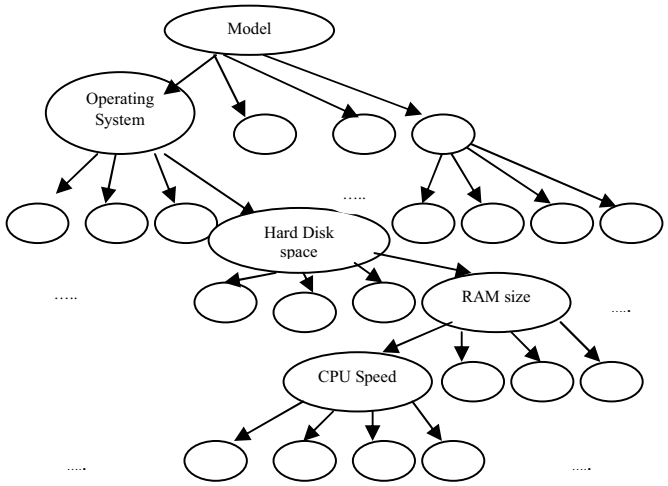

Figure 1. Decision tree for classification of resources

The first one is called reporting clusters, and the second one is called host clusters. Each cluster in the DT of Fig. 1 is assigned to one reporting cluster in CycloidGrid. However, there are 1024 reporting clusters and the other clusters are reserved as host clusters.

Reporting clusters consist of reporting nodes and host clusters contain host/client nodes. Resource information of similar host nodes is kept in the same reporting cluster. Each reporting cluster contains one primary node and some of the replica nodes. Primary reporting node is a node which cyclic index is greater than others and keeps all the resource information of host nodes belongs to this cluster. Replica reporting nodes have a snapshot of resource information from primary node. When a primary node leaves the system one of the replica nodes can get the role of a primary node based on election procedure. The role of primary and replica nodes is discussed in detail in the next section. The organization of clusters in CycloidGrid is shown in Fig. 2.

C. Churn Management in CycloidGrid

When a node joins to the system, it should be determined it is a client node or host/reporting node.

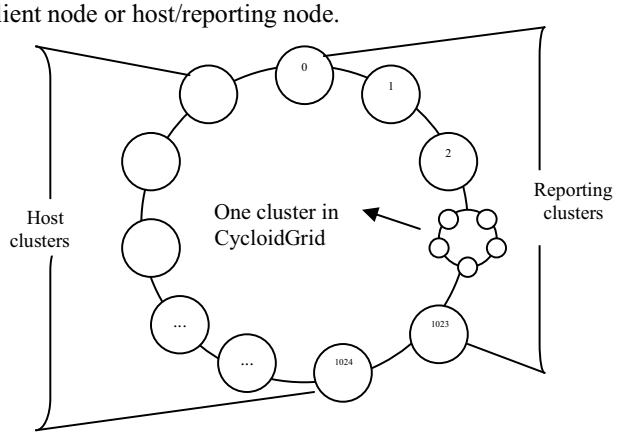

Figure 2. The organization of clusters in the CycloidGrid

If it is a client node, a node identifier will be assigned to it according to the consistent hashing of its IP address. Client node can start the execution of its application by sending lookup requests to active host nodes for each of independent job.

If a node wants to donate its resource it will be a host node or reporting node. At first, resource attribute values of a new node will be imported into the DT. DT determines the reporting cluster in which this node should report resource information. If its reporting cluster is empty or the number of reporting node in this cluster are lower than a replica factor, this node is inserted to the system as reporting node otherwise it is inserted to the system as a host node.

If a node is inserted as reporting node and its reporting cluster is empty, it will be a primary node otherwise it will be a replica node. Primary node periodically sends a state request to the host node belongs to its cluster. Those host nodes respond to this request by sending current queue length. However, primary node modifies resource information with last queue length and deletes unavailable resources from its resource information. This node sends a snapshot of the resource information to the replica nodes after each update.

If a node is inserted as a host node, it will get a node identifier by consistent hashing of its IP address. Then, it should report resource information to the primary node of its reporting cluster.

When a node leaves the system, the behavior of the system is different from host node to reporting node.

If a host node leaves the system, all the jobs in the queue should be rerun by another active host node in the system. Client node sends heartbeat messages to the host nodes responsible for running its jobs. If a host node leaves the system, Client node will recognize it and rerun its job on another node.

If a reporting node leaves the system, the behavior of the system is depending on it is primary node or replica nodes. As we discussed earlier primary node periodically sends a snapshot of its resource information to the replica nodes. When a primary node sends a snapshot, it will receive an acknowledgment message. After the primary node finds the replica node leaves the system, one host node is selected randomly. Host node changes its role from host node to reporting node. All the jobs on its queue will be run on this new reporting node, but this new reporting node doesn't accept any new job from now on. New replica reporting node gets a snapshot of primary node and act as the reporting node from now on.

Replica nodes periodically receive a snapshot from primary node .if they don't receive a snapshot in a constant period of time they understand primary node leaves the system. Then they will start election. Election procedure is discussed in the next section. After the election is finished, the winner of election among the replica nodes accepts the primary node's role. Replica node which changes its role to primary node is replaced by an active host node later.

The system can guarantee reporting clusters always have primary and replica nodes. Because leaving these nodes from the system is replaced by leaving an active host node as soon as it is recognized.

1) Election Procedure

Distributed election runs in each replica reporting node in two steps. In the first step, replica node sends the election start message to other replica nodes in its reporting cluster. Each election start message has timestamp with cyclic index of its replica node. In the second step of distributed election, each replica node receives some election start messages. If the replica node gets the election start message with a timestamp higher than its cyclic index, it leaves the election and waits for election end message from new primary node. However, if the replica node receives the election start messages with lower timestamp of its cyclic index, it will be a winner of election and send an election end message to other replica nodes. In other words, replica nodes with higher cyclic index always win the election.

D. Resource Discovery Architecture of CycloidGrid

A job in the system is a computation to be performed and is modeled by a job's profile and data file. Job's profile includes the client identifier that submitted it and minimum resource requirements of this job. A job constraint can consist of minimum CPU speed, minimum hard disk space requirement, minimum RAM size, model and operating system type. Fig. 3 shows the overall resource discovery architecture of CycloidGrid. The steps of job executions are as follows:

Client enters to the CycloidGrid as a client node. Client node sends a lookup request for each job to the active host nodes in the system randomly. These host nodes are called injection node. Injection node is responsible for finding capable host node for running a received job (step 1).

Injection node uses DT to find which reporting clusters can be useful to search. In this phase reporting cluster with attribute values have minimum or higher resource requirement of the job are found. Host nodes with the minimum resource requirement may be overloaded, while the host nodes with higher values may be under loaded. These reporting clusters are called selected reporting clusters. As it is mentioned, reporting cluster has primary and some replica nodes. Primary or replica nodes can respond to a lookup request from injection node. Injection node computes a prediction of RTT between itself and reporting nodes in each selected reporting cluster. Then a reporting node with the minimum predicted RTT is selected in each selected reporting cluster. Lookup request is sent to these reporting nodes. Prediction of RTT value between two nodes is done by Vivaldi algorithm [32] in this research. Vivaldi is a simple, adaptive, decentralized algorithm, which computes synthetic coordinate for internet host. In this algorithm, an Internet host can compute synthetic coordinate in some coordinate space. Distance between two host's synthetic coordinates predicts the RTT between them in the Internet. In this research Euclidean distance between two synthetics coordinates is used for prediction of RTT between two nodes (step2).

Selected reporting node searches among all resource information which is hosted locally. A ranking algorithm is

done in each selected reporting node. In this ranking algorithm, resources get a rank in the terms of the queue length, CPU speed and ticket parameters. Ticket is a parameter assigned to each resource in the reporting node. As it mentioned earlier, reporting nodes refreshes resource information periodically so that queue length of each resource is refreshed. If a resource is recommended by reporting node to the requested injection node, the ticket value of this resource will be incremented locally. In fact, the role of this parameter is controlling a convergence to the lowest load resource in the time slice between two update. Ticket is reinitialized to zero after each updates. The rank of each resource is computed by weighting function as is shown in "(1)".

$$r = \frac{w_1 l + w_2 s + w_3 t}{w_1 + w_2 + w_3} \quad (1)$$

l is a queue length, s is scaled value of inverse CPU speed and t is ticket value for every resource. w_1, w_2, w_3 are the weighting coefficients and are selected with respect to the importance of that factor. After computing the rank of each resource, a resource with the lowest rank is selected. Since each reporting node searches among resources independently in parallel, so that a distributed search is done in this phase (step3).

Each selected reporting node sends a recommended host node identifier with its rank to the injection node (step 4).

Injection node collects this information. A new rank assigns to recommended host nodes in this phase as it is stated in "(2)".

$$r' = \frac{\alpha r + \beta d}{\alpha + \beta} \quad (2)$$

r is a previous rank sent by selected reporting node and d is a summation of Euclidean distance between synthetic coordinate of itself and recommended host nodes and Euclidean distance between synthetic coordinate of its client node and recommended host nodes. α, β are the weighting coefficients. Consequently, a host node with the minimum predicted RTT from client node and injection node is preferred to other capable host nodes in this phase. Selected host node is called run node. Selection of a host node with the minimum predicted RTT can help to decrease communication overhead and proposed resource discovery architecture is proximity-aware (step 5).

Job's profile is sent to run node. It is added in the run node's queue in FIFO order and waits for execution. Run node sends a job request message to the client node and asks job and its data file. Heartbeat message are exchanged between a client node and run node periodically (step 6).

When the job is finished, a result is returned to the client node (step 7).

IV. PERFORMANCE EVALUATION OF CYCLOIDGRID

A. Experimental setup

CycloidGrid simulator is written in Java to evaluate the proposed resource discovery architecture. This simulator is extended version of Cycloid simulator to emulate the P2P volunteer computing system. Cycloid simulator is developed to store files in the P2P environment. Physical network in CycloidGrid is emulated by brite topology generator [17].

A physical network with n computers which are connected by Waxman model and different link bandwidth is generated by brite. Nodes are distributed in the physical network randomly. Vivaldi algorithm [32] is used to compute synthetic coordinate of each node in the physical network to predict RTT between two nodes in the system. A 2-dimensional Euclidean model with height vectors is used in this research. Predicted RTT between two nodes in the system varies from one millisecond to 1.2 seconds.

In order to emulate resources in the CycloidGrid simulator Xtremlab trace [18] is used. Xtremlab trace is exported from BOINC database, and their information is collected by client or server. To evaluate a performance of the system a workload of mixed job is generated. These jobs are grouped into light and heavy jobs. The number of job constraints is 0 to 2 in light jobs; meanwhile this number is 3 to 5 in heavy jobs. Job's constraints are selected randomly among model, operating system type, minimum CPU speed, minimum hard disk space and minimum RAM size. These constraints are different from one job to another job.

Events in the system include job submission, and node joins/departure. These events are generated by Poisson distribution with an arrival rate of $\frac{1}{\tau}$ (τ is an average event inter-arrival time).

If a job needs the computation time w, w will be scaled on a computer with higher CPU speed.

In this work, two parameters are computed for evaluating performance of the system. The first parameter is job's wait time.

Wait time is considered as time slag between importing a job until the start of its execution. Wait time of the job is computed by "(3)".

$$t_{wait} = \sum(\|v_c - v_i\|)/2 + \max(\sum\|v_i - v_{p_k}\|) + \sum(\|v_i - v_r\|)/2 + \sum_{l=1}^{S} w_l)$$
$$k = 1..R \quad (3)$$

In the above equation v_c, v_i, v_r are a synthetic coordinate of the client node, injection node and run node respectively.

Cycloid P2P overlay has time complexity $O(d)$ for a lookup request where d is a dimension. Consequently, Euclidean distance between two nodes in CycloidGrid equal to a summation of Euclidean distance between nodes in the path between those nodes in the P2P overlay. Sigma operator is used to show this summation.

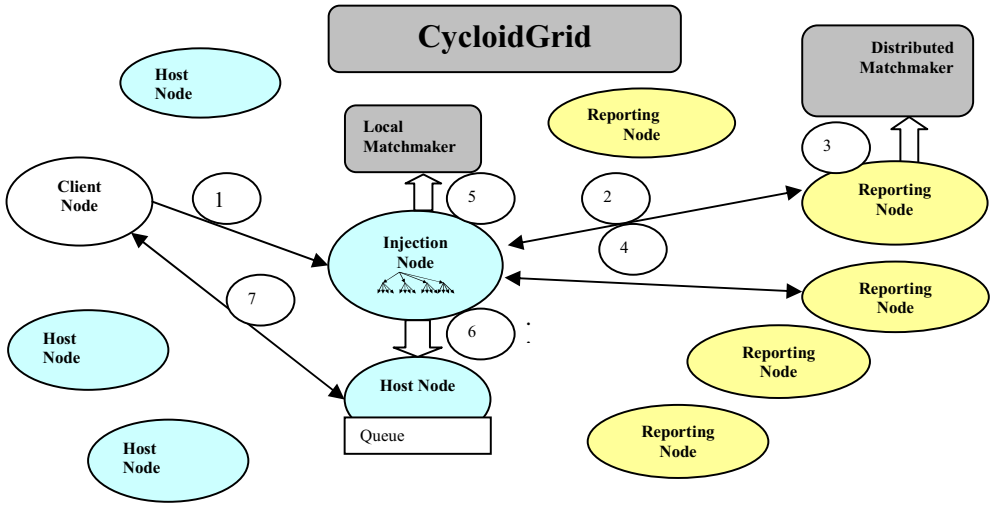

Figure 3. Resource discovery architecture of CycloidGrid

Predicted RTT is divided by 2 for one-way communication. v_{p_k} is a synthetic coordinate of the k^{th} reporting node in the pool of selected reporting nodes. R is the number of selected reporting nodes. The maximum of communication overhead between the injection node and selected reporting nodes is added to wait time of the job. Since, injection node contacts selected reporting nodes in parallel. w_l is a run time of any job in the run node 's queue and s is the number of jobs. The second parameter is a current run node's queue length when a job is inserted in its queue. This parameter can show load balance state in the system.

CycloidGrid is compared with centralized desktop grid. Centralized desktop grid has a database for keeping up-to-date resource information. Resource discovery algorithm searches within this database to find the capable resources. A weighting function is used to ranking resources in the terms of queue length and CPU speed as the same as "(1)" with no ticket parameter. The weighting coefficient is equal to CycloidGrid. CycloidGrid is considered with two options: with the proximity-aware feature (ProxCycloidGrid) and without it (NoProxCycloidGrid). These two systems have some differences as discussed below. Reporting node in NoProxCycloidGrid is selected randomly in step 2 of section 3.4. Furthermore, run node is selected only in the terms of queue length and CPU speed in step 5 of section 3.4.

B. Experimental Results

In the experiment, at first 1000 nodes join to the system, and then 10000 jobs submit to the system with an arrival rate of $\tau = 0.1$ s. The execution times of jobs are selected uniformly at random with 110 seconds on average. Weighing coefficient is initialized to $w_1 = 0.7, w_2 = 0.1, w_3 = 0.2$ in "(1)".

$\alpha = 0.5, \beta = 0.5$ in "(2)" of ProxCycloidGrid; meanwhile $\alpha = 1, \beta = 0$ in NoProxCycloidGrid.

In the first experiment, the system is relatively static and no nodes join or leave during the experiment.

Fig. 4, 5 indicates a cumulative fraction of queue length for heavy and light jobs respectively. The maximum queue length in three systems is different from 5 to 7. Behavior of these three systems is almost similar. It is indicated that although in CycloidGrid decision making on selection of capable run node is distributed, but the functionality of three systems is equal and load balance exists in the system. ProxCycloidGrid is better than NoProxCycloidGrid in these figures. Because local matchmaking in ProxCycloidGrid is based on queue length and distance in each host node, while in the NoProxCycloidGrid it is based only on queue length. Distance parameter in ProxCycloidGrid adds randomness to selection and precludes convergence to the least loaded node in the system. Fig. 6, 7 shows a cumulative fraction of wait time for heavy and light jobs. Wait time is computed by "(3)" in CycloidGrid; meanwhile it is equal to prediction of RTT between server and run node and summation of job's computation time in the run node's queue in the centralized system. It is shown by equation "(4)".

$$t_{wait} = (\|v_s - v_r\|)/2 + \sum_{l=1}^{S} w_l \quad (4)$$

In the above equation v_s, v_r are a synthetic coordinate of a server node and run node.

The overall wait time of CycloidGrid is more than a centralized system because of communication overhead of P2P system. In the P2P system, a message is delivered hand by hand, so that the RTT between two nodes in the system is computed as summation of RTT between all of the nodes in the path between two nodes in the P2P overlay. Cycloid have time complexity *o(d)* , with d is a dimension. It means that a message at most travel on P2P overlay *d* steps to find a destination. It causes performance degradation compared to the centralized system. Meanwhile in the centralized system only direct RTT between two nodes are computed.

Wait time of ProxCycloidGrid is less than NoProxCycloidGrid.

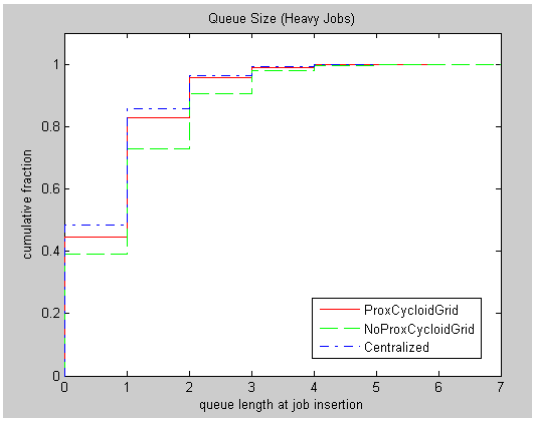

Figure 4. Queue length at job insertion in heavy jobs

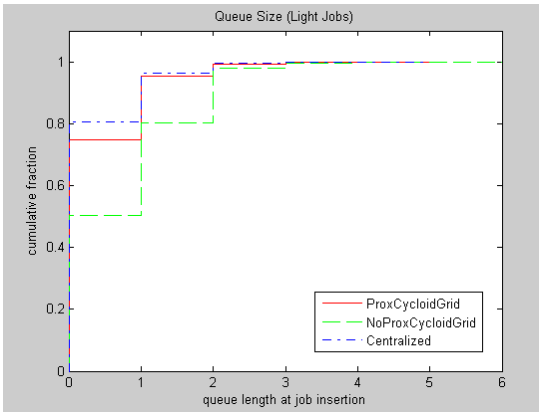

Figure 5. Queue length at job insertion in light jobs

It means that the percentage of jobs with wait time is lower than the specified value in ProxCycloidGrid is higher than NoProxCycloidGrid. As it is discussed earlier, ProxCycloidGrid selects run node based on queue length, CPU speed and minimum RTT. Meanwhile in the NoProxCycloidGrid run node is selected only based on queue length and CPU speed. In the P2P system a message is delivered hand by hand and summation of RTT between all of the nodes along the path is computed, so that selection of minimum RTT is caused to select shorter path and decrease in communication overhead. It also increases the system performance.

In the second experiment, nodes join or depart from the system by Poisson distribution. In this experiment, three light workloads with 10000 jobs are studied. In these dynamic workloads after 1000 nodes initially join to the system, some nodes leave while some nodes join to the system. The departure rate of nodes in these three workloads is between10% until 30% of all nodes in the system, in such a way, Third workload has highest node departure rate.

Since different sets of the node are available in the system in these three workloads; comparison against the workload is not correct so that only average wait time for light jobs is measured and compared with.

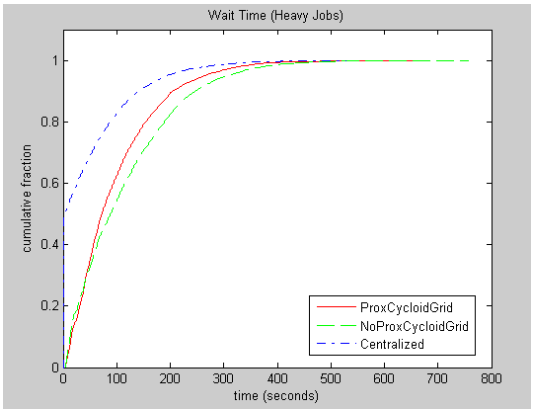

Figure 6. Wait time in heavy jobs

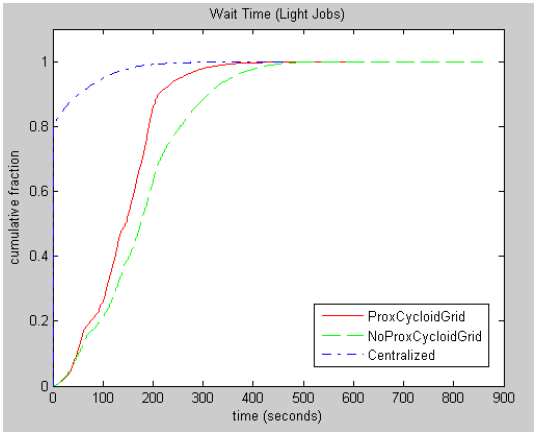

Figure 7. Wait time in light jobs.

In Fig. 8, a comparison of average of wait time is shown in the centralized and two P2P systems. In this experiment, the average of wait time of ProxCycloidGrid is better than NoProxCycloidGrid. In this figure, the more leaving rate increases, the more wait time increases. Some of the jobs should be reassigned increase with leaving rate increasing, and it will influence the wait time.

V. CONCLUSION

In this paper, CycloidGrid is introduced. This system is a proximity-aware architecture for resource discovery in P2P based volunteer computing system. A Proximity-aware feature is considered in this research; meanwhile in most of the works this feature is neglected. CycloidGrid can gain a reasonable result with lower overhead compared with the similar system without proximity-aware feature. The performance of the system is very close to the centralized system and load balance exists in the system. In the future, we will decide to extend our system to run a larger class of distributed application with precedence between tasks. This group of distributed application increases the application of volunteer computing in the scientific applications.

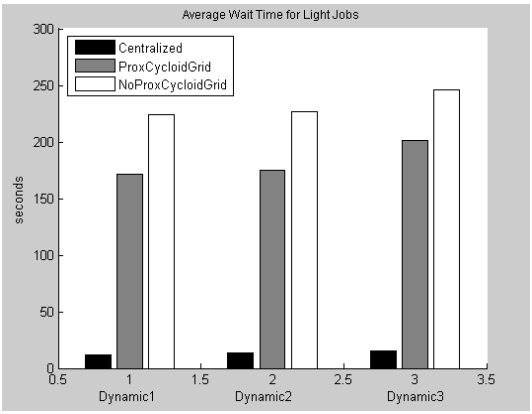

Figure 8. average wait time of three light workload

REFERENCE

[1] D. Kondo, B. Javadi, P. Malecot, F. Cappello, D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," Proc. IEEE International Symposium on Parallel & Distributed Processing (IPDPS 09) , IEEE Press , May 2009 , pp. 1-12, doi: 10.1109/IPDPS.2009.5160911.

[2] http://www.edges-grid.eu/

[3] http://climateprediction.net/

[4] A. Chien, B. Calder, S. Elbert, K. Bhatia," Entropia: architecture and performance of an enterprise desktop grid system," J. Parallel Distrib. Comput., vol. 63, May 2003, pp. 597-610 , doi: 10.1016/S0743-7315(03)00006-6.

[5] J. S. Kim, B. Nam, P. keleher, M. Marsh,B. Bhattacharjee, A. Sussman, "Trade-offs in matchmaking job and balancing load for distributed desktop grids," Future Gener. Comp. Sy. ,vol. 24 , May 2008 , pp. 415-424, doi:10.1016/j.future.2007.07.007.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp,S. Shenker," A scalable content addressable network," Proc. ACM SIGCOMM (SIGCOMM 01),ACM Press, Aug. 2001, pp. 161-172,doi: doi: 10.1145/383059.383072.

[7] H. Abbes, C. cerin, M. Jemni, "PastryGrid: decentralisation of the execution of distributed applications in desktop grid, " Proc. International Workshop on Middleware for Grid Computing (MGC 08), ACM Press, Dec. 2008 , doi: 10.1145/383059.383072.

[8] A. Rowstran, P. Druschel, "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems,"Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01), Springer, Nov. 2001, pp. 329-350, doi: 10.1.1.28.5987.

[9] C. Mastroianni, P. Cozza, D. Talia, I. Kelley, I. Taylor, "A scalable super-peer approach for public scientific computation, "Future Gener. Comp. Sy. ,vol. 25 ,Mar. 2009, pp. 213-223 , doi:10.1016/j.future.2008.08.001.

[10] B.Yang, H. Garcia-Molina, "Designing a super-peer network," Proc. International Conference on Data Engineering (ICDE 03), IEEE Press,Mar. 2003, pp. 49-62 , doi: 10.1109/ICDE.2003.1260781.

[11] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, "SETI@home: an experiment in public-resource computing, " Commun. ACM ,vol. 45 , Nov. 2002, pp. 56-61, .doi: 10.1145/581571.581573.

[12] H. Shen, C. Xu, G. Chen, "Cycloid: a scalable constant-degree p2p overlay network, Perform Evaluation ," vol. 63 ,Nov. 2006 ,pp. 195-216, doi: 10.1016/j.peva.2005.01.004

[13] http://folding.stanford.edu

[14] http://www.worldcommunitygrid.org

[15] M. Vladoiu, Z. Constantinescu, "QADPZ - A desktop grid computing platform," Int. J. of Computers,Communications & Control, Vol. IV , Mar. 2009, pp. 82-91.

[16] D. Lazaro, J. M. Marques, X. Vilajosana," Flexible resource discovery for decentralized P2P and volunteer computing systems, Proc. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises(WETICE 10), IEEE Press, Jun. 2010 ,pp. 235-240, doi: 10.1109/WETICE.2010.44.

[17] Brite topology generator. http://cs-pub.bu.edu/brite.

[18] XtremLab Trace. http://xtremlab.lri.fr/traces.

[19] D. P. Anderson, K. Reed, "Celebrating diversity in volunteer computing,"Proc. the Hawaii International Conference on System Sciences (HICSS 09), ACM Press, Jan. 2009, pp. 1-8, doi: 10.1109/HICSS.2009.105.

[20] A. C. Marosi, G. Gombas, Z. Balaton, P. Kacsuk, T. Kiss, "SZTAKI desktop grid: bulding a scalable , secure platform for desktop grid computing," Making Grids Work,vol. VII, 2008, pp.365-376,doi: 10.1007/978-0-387-78448-9_29.

[21] X. Chu, K. Nadiminti, C. Jin, S.Venugopal, R. Buyya, "Aneka: next-generation enterprise grid platform for e-science and e-business applications, "Proc. IEEE International Conference on e-Science and Grid Computing(eScience 07), ACM Press , Dec. 2007 , pp. 151–159, doi: 10.1109/E-SCIENCE.2007.12 .

[22] M. Portmann, A. Seneviratne, "Cost-effective broadcast for fully decentralized peer-to-peer networks, "Comput. Commun. Vol. 26 , Jul. 2003, pp. 1159–1167, doi: 10.1016/S0140-3664(02)00250-5.

[23] M. Jelasity, M. Preuss, B. Paechter, "A scalable and robust framework for distributed applications,"Proc. IEEE Congress on Evolutionary Computation(CEC 02) , IEEE Press , May 2002, pp. 1540-1545, doi: 10.1109/CEC.2002.1004471.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications,"Proc. ACM SIGCOMM (SIGCOMM 01), ACM Press, Aug. 2001, pp. 149-160, doi:10.1145/964723.383071.

[25] P. Merz, K. Gorunova, "Fault-tolerant resource discovery in peer-to-peer grids," J Grid Computing ,vol. 5 , Sep. 2007 , pp. 319–335 ,doi: 10.1007/s10723-006-9057-1.

[26] D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, J. Pruyne, "A worldwide flock of condors: load sharing among workstation clusters, " Future Gener. Comp. Sy., vol. 12 , May1996, pp. 53-65, doi:10.1016/0167-739X(95)00035-Q.

[27] M. J. Litzkow, M. Livny, M. W. Mutka, "Condor - a hunter of Idle Workstations," Proc. International Conference on Distributed Computing Systems(DCS 98),IEEE Press Jun.1998, pp. 104-111, doi: 10.1109/DCS.1988.12507.

[28] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Neri , O. Lodygensky ,"Computing on large scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid, Future Gener. Comp., vol. 21 , Mar. 2005, pp. 417-437 , doi:10.1016/j.future.2004.04.011

[29] D. Thain, T. Tannenbaum, M. Livny, "Distributed computing in practice: the condor experience," Concurrency Computat. Pract. Exper. ,vol. 17 , Feb. 2005,pp. 323-356, doi: 10.1002/cpe.v17:2/4.

[30] T. Abdullah, L. O. Alima, V. Sokolov, D. Calomme, K. Bertels, "Hybrid resource discovery mechanism in ad hoc grid using structured overlay ," Lect. Notes Comput. Sci.,vol. 5455 ,Mar. 2009, pp. 108-119, doi: 10.1007/978-3-642-00454-4_13.

[31] B. Y. Zhao, J. Kubiatowicz, A. D. Joseph," Tapestry: an infrastructure for fault-tolerant wide-area location and routing, " Technical Report: CSD-01-1141, University of California at Berkeley,ACM Press , 2001.

[32] F. Dabek, R. Cox, F. Kaashoek, R. Morris, " Vivaldi: a decentralized network coordinate system, "Proc. ACM SIGCOMM (SIGCOMM 04), ACM Press, Aug. 2004, pp. 15-26, doi: 10.1145/1030194.1015471.