RC24208 (W0703-029) March 7, 2007
Computer Science

IBM Research Report

A Self-regulating System for Resource Sharing in Multi-site Grid Environments

Pawel Garbacki
Delft Universityof Technology
Delft, The Netherlands

Vijay K. Naik
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Self-regulating System for Resource Sharing in Multi-site Grid Environments

Paweł Garbacki
Delft University of Technology
Delft, The Netherlands
p.j.garbacki@tudelft.nl

Vijay K. Naik*
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
vkn@us.ibm.com

Abstract

We describe a self-adaptive and self-regulating system for fair sharing of resources in a multi-site grid environment. In such an environment, each site has a set of resources and has internal demand which it manages using internal policies and rules. To manage periodic fluctuations in the local demand and potential shortfall of resources to meet the local demand, the participants borrow and lend resources to one another. This resource sharing among the participants is regulated by a distributed, peer-to-peer resource brokering algorithm that is scalable and fair in its allocation of the shared resources. The self-regulating mechanism enables discovery of shared resources and the possible extent of the donated or borrowed resources. The regulating mechanism automatically clusters participants with similar borrowing and donating properties. We present both analytical and empirical evidence to prove the properties of such a system. In particular, we prove that the system is fair – that is, participants who contribute more resources to the shared pool get to borrow proportionately more resources from others. Such a resource brokering mechanism allows participants in a multi-site grid environment to meet their peak demands by provisioning resources only for average demand.

1 Introduction

Fluctuations in workload and resource demand are common to data center and other IT environments. Neither provisioning for peak demand nor for average demand are desirable solutions. Today this dilemma is handled mostly by provisioning more resources than required to meet the average workload, but not provisioning enough to meet the peak workload. However, any such compromise is far from being satisfactory. Ideally, organizations like to provision and pay for just enough resources to meet the demand and, at the same time, to remain competitive they like to have access to resources on demand to meet any business situation.

Using the emerging concepts such as utility computing, it is possible for organizations to rent external resources when local demand exceeds locally provisioned resources. Apart from the premium that an organization has to pay to acquire resources on demand from the utility, this approach does not address the resource under utilization problem. Grid computing is another emerging approach to harness and share a large number of resources among commercial organizations [1]. However, traditional grid based approaches are centralized, not scalable, and are not suitable for on-demand acquisition of resources [3].

In this paper, we describe a grid based system that enables organizations to participate and exchange resources with one another to meet their internal demands and meaningfully increase the resource utilization when their internal demands are declining. A novelty of the system described here is that, unlike traditional grid systems, the resource brokering mechanisms are based on peer-to-peer interactions among the participants and are completely distributed. Grid participants achieve resource flexibility by borrowing resources when needed from external sources and by lending the excess capacity to other organizations when they are not needed locally.

Resource exchanges can be of two flavors: (a) exchange one type of resource for another at the same time and (b) exchange similar resources at different time periods; i.e., for borrowing a resource now, a similar resource is lent out to others for a similar period of time in the future. The system we describe here accommodates both types of exchanges. In this paper, we concentrate on the latter type of exchange. A resource exchange, such as the one outlined above, needs to address the following issues: finding the matches between borrowers and lenders, ensuring fairness, and ensuring that participants fulfill their obligations when they enter into a barter.

In the absence of independent monitoring and enforcement, for the grid system to succeed, *fairness* in resource sharing – defined as the measure of balance between the

*Corresponding author.

contributions of own resources and the consumptions of the resources of the others – needs to be preserved. The system must provide built-in fairness enforcement mechanisms to prevent possibilities of system abuses by non-cooperative parties, so called *freeriders* [5].

The grid system described here differs from other traditional grid systems in another important aspect. In this system, the participants play the role of both the resource providers and resource consumers. In traditional grid systems, typically these roles not interchangeable; a grid user does not normally contribute resources to the grid. In the system described here, the resources are provided and consumed by the participants. The grid system only enables the resource exchange across organizational boundaries such that the net flow in and out of any one organization is close to zero over a long enough period of time.

The main contributions of our work are: (a) Formulation of a new resource sharing concept for grid environments that allows for resource exchanges among anonymous organizations, while preserving fairness in their resource contributions. (b) Design of an intuitive and formally provable resource sharing mechanism based on the new concept. Organizations using this sharing mechanism make all decisions locally and independent of each other. No central components are required. (c) Experimental evaluation of the resource sharing mechanism using a real-world performance data.

The rest of the paper is organized as follows. In the next section, we describe the design of the system for enabling multi-site resource sharing and the neighborhood forming algorithm used by the participants. In Section 3, we analyze the sharing mechanisms and, using this analysis, characterize the system. In Section 4, we describe an experimental setup and evaluate the sharing mechanisms described in this paper. The results of the evaluation are also presented in that section. Finally we conclude the paper in Section 5.

2 Sharing mechanism design

In this section we present design of a resource sharing mechanism in grid environments. We start with introducing some basic terminology.

2.1 Terminology

A party involved in the resource sharing is called an *organization*. We shall denote the set of all organizations in the system by D. Organizations in the system maintain limited-size sets of *neighbors*, i.e., other organizations they share resources with. Neighbor set of organization $d \in D$ is denoted by N_d. Neighborhood is an asymmetric relation meaning that organization d can be a neighbor of organization $\hat{d}$ even if $\hat{d}$ is not a neighbor of d. Organizations give their neighbors access to their currently idle resources. The set of neighbors is not fixed but it evolves to reflect the changes in the demand and local resource sharing policies. The allowed size of the neighbor set is, however, bounded by a constant n.

In addition to the neighbor set, each organization $d \in D$ maintains a *local view* of the system denoted V_d. New neighbors are selected from among the organizations in the local view. It is important that the size of the view is sufficiently large to guarantee enough selection options. Moreover, local views should be updated frequently in order to efficiently propagate information about organizations that recently joined the system. Details of the local view update process are presented further in this paper.

2.2 Neighborhood formation

In this section, we describe the neighborhood formation algorithm that selects for each organization a set of neighbors optimal in its local context. The presented algorithm is fully distributed and it is executed independently by each organization.

Following a rational strategy of locally optimal choices, each organization tries to find a set of neighbors that offer the most capacity during time periods when the organization is in demand. The neighborhood formation algorithm, presented in Algorithm 1, consists of two phases: the exploration phase and the selection phase, repeated periodically, but independent of each other.

Algorithm 1: Neighborhood formation algorithm.

```
1  explore(d : organization):
   begin
2      while |N_d| < n and V_d \ N_d ≠ ∅ do
3          select randomly an organization d̂ from V_d \ N_d
4          add d̂ to N_d
5          if |N_d̂| < n then
6              add d to N_d̂
           end
7          V_d ← V_d ∪ V_d̂
       end
   end

8  select(d : organization):
   begin
9      sort organizations in N_d according to their
       decreasing contributions
10     while |N_d| > n − r do
11         remove from N_d next organization with the
           lowest contribution rank
       end
   end
```

During the *exploration phase* (line 1), organization d expands its neighbor set up to the predefined capacity limit (line 2). New neighbors are selected randomly from the local view of organization d, excluding organizations that have been already selected as neighbors (line 3). If d adds organization $\hat{d}$ to its neighbor set (line 4), and the neighbor set of $\hat{d}$ has not yet reached its capacity limit (line 5), then also $\hat{d}$ adds d as a neighbor (line 6). New neighbors update their local views by merging their content (line 7).

The *selection phase* (line 8) starts with sorting of neighbors in decreasing order of their resource contribution (line 9). The selection algorithm removes up to r lowest ranked resource contributors from the neighbor set (lines 10 and 11). Note that the neighbor list could be sorted using a variety of criteria. Some example criteria are: (i) amount of resources made available for borrowing by a neighbor, (ii) the quality of the resources made available, (iii) the proximity of the resources, (iv) resource availability weighted by the time when they are available (e.g., availability at peak demand is ranked higher than availability at low demands), an so on.

The impact of parameter r on the neighborhood formation process is twofold. First, parameter r controls the dynamism of the neighbor set modification. Larger r will result in more neighbors removed in the selection phase and replaced with randomly chosen organizations in the exploration phase. In an environment where organizations frequently change their sharing behavior, higher level of neighbor set modification dynamism will result in faster adjustment to the changing conditions. On the other hand, larger r implies also that the neighbor set will contain more organizations that have just been added to the neighbor set and have not been evaluated yet.

Second, parameter r impacts the probability that a bidirectional neighborhood relation will be formed between two organizations d and $\hat{d}$, i.e., d will add $\hat{d}$ as its neighbor and also $\hat{d}$ will accept d as a neighbor. The larger the value of r the higher the chance that the conditional statement in line 5 will evaluate to true forming a bidirectional neighborhood relation between d and $\hat{d}$.

Note that the exploration and selection phases are complementary in a sense that while exploration extends the neighbor set opening opportunities for resource exchanges between anonymous organizations and the selection operation contracts the neighbor set by removing the least promising selections.

The randomness involved in the exploration phase gives all organizations equal chance of becoming neighbors. This property is very important for bootstrapping as it allows newcomers to establish relationships with the existing organizations.

2.3 Sharing resources between neighbors

The mechanism of resource sharing between neighbors is independent of the neighborhood formation protocol. In particular, advanced techniques of policy-based resource management [2, 3] can be employed to compute the resource sharing strategy among neighbors that is optimal according to a set of customizable policies. As it has been shown in [2], computing of an optimal sharing strategy is an expensive operation. Restricting the sharing to a small set of neighbors resolves the scalability limitations of such systems, allowing them to be deployed on a wider scale.

Complex resource sharing strategies between neighbors render additional cost that has to be taken into account while deciding on the properties of the neighbor set. E.g., the resource sharing mechanisms presented in [2] and [3] incur a sharing strategy computation cost that grows exponentially with the size of the neighbor set. The complexity of the sharing strategy computation implies, thus, restrictions on the neighbor set size, which in turn limits the number of possibilities for resource exchange.

The properties of the neighborhood formation protocol that forms the neighbor set from organizations contributing most resources allows to employ simple strategies for sharing resources between neighbors. The sharing strategies may either give each neighbor an equal fraction of the shared resource or divide the shared resource among the neighbors proportionally to their contributions. Some examples of strategies that treats all neighbors equally are: (i) allocate for each of the neighbors currently in demand an equal fraction of resource, (ii) allocate the resource exclusively to one of the neighbors in demand selected according to a fair strategy, e.g., round-robin, for a certain amount of time. Strategies that take into account neighbor contributions would determine the amount of shared capacity assigned to the neighbor or the allocation time based on the amount of contributed resources.

3 Analysis of the sharing mechanism

In this section we study analytically the properties of the resource sharing mechanism presented in Section 2. After introducing a model of the grid environment, we use this model to investigate the structure of the neighbor set found by Algorithm 1.

3.1 System model

We model a grid system that consists of organizations sharing resources of certain capacities. The capacity of the shared resource can differ between organizations.

For the purpose of the analysis we assume that the sharing strategy treats every neighbor in demand equally when

disposing the shared resource. Namely, the shared capacity is distributed evenly among the neighbors currently in demand. Furthermore, we assume that an organization in demand is able to consume any amount of the shared capacity.

We assume that the frequency of entering the demand and the duration of the demand phase are the same for all organizations. Organizations demand resources according to a Poisson process with rate λ. The durations of the demand phases are distributed exponentially. We denote by μ the inverse of the average duration of the demand phase.

According to Algorithm 1, neighbors are selected based on the incomplete information maintained by each organization in the form of local view and the record of the resource contributions. To simplify the analysis, in our system model we assume that every organization has a complete information about the resources shared by other organizations as well as the sets of their neighbors. We believe that this simplification does not affect the credibility of our analysis because the continuous exchanges of the local views between interacting organizations and the random selection of new neighbors will let each organization to finally discover properties of all other organizations.

3.2 Neighbor set formation

In this section we investigate the properties of the neighbor set formed by Algorithm 1.

We denote by c_d the *contribution* of organization d defined as the average amount of resources owned by d consumed by a single neighbor. Combining the assumption that the shared capacity is distributed evenly among neighbors in demand with the fact that all organizations demand resources with the same frequency and for (on average) equal periods of time leads to the observation that the amount of consumed shared capacity is (on average) equal across the neighbors.

Let all organizations be ordered according to decreasing values of their contributions. Organizations that contribute the same amount of resources are ordered randomly among themselves. We further use notation d_i to indicate that organization $d \in D$ is assigned position i in this order.

Under the assumption of global system knowledge, we can rewrite neighborhood formation algorithm in a form presented in Algorithm 2. The algorithm executes in iterations. In iteration i organization d_i selects its neighbors (line 1). Neighbors selection is performed by organizations in the order determined by their contributions. Organizations contributing more resources are always preferred as neighbors, and thus they can be given priority while constructing the neighbor sets. The neighbor set of d_i is expanded until it reaches maximal allowed size (line 2). In a single iteration of the loop in line 2, a new neighbor is added to the neighbor set of d_i. Selection of the new neighbor is performed in three steps. In the first step, set D_1 of organizations that can still accept new neighbors and are not already present in neighbor set of d_i is identified (line 3). If D_1 is empty, the algorithm continues with the next iteration of the outer loop (line 5). In the second step, subset D_2 of organizations with maximal contributions is selected from among the organizations in D_1 (line 6). This step guarantees that the neighbor selection objective of Algorithm 1 is preserved. According to this selection objective, any organizations in D_2 could be potentially selected as a neighbor of d_i. However, to simplify further analysis we assume that d_i selects an organization d_j from D_2 with the smallest number of neighbors (line 7). The neighborhood relation between d_i and d_j is formed by adding d_i to neighbor set of d_j (line 8) and d_j to the neighbor set of d_i (line 9).

Algorithm 2: Equivalent of the neighborhood formation algorithm assuming that each organization has a global system knowledge.

```
1 for i in 1, ..., |D| do
2     while |N_{d_i}| < n do
3         D_1 ← {d ∈ D : |N_d| < n and d ∉ N_{d_i}}
4         if D_1 = ∅ then
5             continue the for loop
          end
6         D_2 ← {d ∈ D_1 : c_d is maximal among
          organizations in D_1}
7         select from D_2 an organization d_j with the
          minimal value of |N_{d_j}|; if many such
          organizations exist, select one of them
          randomly
8         add d_i to N_{d_j}
9         add d_j to N_{d_i}
      end
  end
```

According to Algorithm 2, it is possible that after the algorithm finishes, there are some organizations with neighbor sets of size lower than n. This is due to the fact that set D_1 computed in line 3 can be empty. Let k be the first iteration that results in empty set D_1. The way set D_1 is defined implies that in iteration k there are less than n organizations with neighbor set sizes lower than n. Since during consecutive iterations the neighbor set of any organization can be only expanded, at the end of the algorithm execution there are at most n organizations with neighbor set sizes lower than n. In Section 3.4 we motivate why n should be kept small. Following this motivation, we can simply ignore the few possible outliers and assume that at the end of Algorithm 2 each organization is assigned exactly n neighbors.

One more subtlety related to the neighbor set size differentiates Algorithm 1 from Algorithm 2. Algorithm 1 reserves r positions in the neighbor set for random exploration. We could introduce r in Algorithm 2 by assuming that the size of the neighbor set considered by Algorithm 2 equals $n - r$ instead of n. However, since r is significantly smaller than n and the conversion between $n - r$ and r is straightforward, introducing r in Algorithm 2 would only make the notation more complex without bringing a clear benefit. Therefore, we will continue to use symbol n to denote the neighbor set size in Algorithm 2.

3.3 Incentives for resource contributions

We will provide evidence that organizations in our system are given incentives to contribute resources by showing that the amount of resources acquired from the neighbors depends on the amount of the contributed resources.

Note, that all neighborhood relationships formed in Algorithm 2 are bidirectional — organization d_i adds d_j to its neighbor set only if d_j accepts d_i as a neighbor. In other words, organization d_i will not share its resources with d_j, unless d_j gives d_i access to its own resources.

Let g_{d_i} denote the sharing *gain* of organization d_i, defined as the aggregated amount of resources acquired by d_i from its neighbors, or using a symbolic expression g_{d_i} equals $\sum_{d \in N_{d_i}} c_d$. The following observation characterizes the correlation between the amount of resources contributed to and gained from the neighbors.

Observation 3.1. *If $c_{d_{i-1}} > c_{d_i} = c_{d_{i+1}} = \ldots = c_{d_j} > c_{d_{j+1}}$ and $j - i > n + 1$ then $g_{d_i} > g_{d_{j+1}}$ and $g_{d_k} \geq g_{d_{j+1}}$ for $i \leq k \leq j$.*

Proof. If organization d_i has been selected as a neighbor by n organizations prior to iteration i of Algorithm 2, then $g_{d_i} > nc_{d_i}$. Otherwise, d_i will select remaining neighbors from among organizations $d_{i+1}, \ldots, d_j$. In both cases $g_{d_i} \geq nc_{d_i}$.

Let A be the set $\{d_i, d_{i+1}, \ldots, d_j\}$. If no organization from A selects d_{j+1} as a neighbor then obviously $g_{d_i} > g_{d_{j+1}}$. Otherwise, let's assume that d_l, where $i \leq l \leq j$, is the first organization that selects d_{j+1}. Let's assume that d_l was selected as a neighbor by m organizations prior to iteration l of Algorithm 2. Let's denote by B the organizations from A that are assigned less than n neighbors prior to step l. Organization d_l will select d_{j+1} as a neighbor only if $|B|$ is lower than or equal to $n - m$ (there is not enough organizations in A that can be selected as neighbors). Since among the organizations in A only the organizations in B can select d_{j+1} as a neighbor, d_{j+1} will be selected by at most $n - m$ organizations in A. If m is greater than zero then d_{j+1} can be selected by at most $n - 1$ organizations from A. If m equals to zero the properties of Algorithm 2 guarantee that at least one organizations $d_k \in B$ will be assigned all neighbors prior to iteration k. Organization d_k will then not select d_{j+1} as a neighbor, which also implies that d_{j+1} can be selected by at most $n - 1$ organizations in A. Consequently, $g_{d_{j+1}} \leq (n-1)c_{d_i} + c_{d_{j+1}} < nc_{d_i} \leq g_{d_i}$.

The inequality $g_{d_k} \geq g_{d_{j+1}}$ for $i \leq k \leq j$ can be shown using an analogous argumentation. Organization d_{j+1} will be selected as a neighbor by organization d_l, $l > j + 1$ only if there are no organizations in A that can be selected. Since organizations in A are preferred as neighbors, d_k will have neighbors contributing at least as much resources as the neighbors of d_{j+1}. □

Observation 3.1 shows a certain property of a group of organizations contributing the same amount of resources to their neighbors assuming a certain minimal size of this group. In a general case, each organizations could contribute a different amount of resources, preventing such groups from forming. We should, however, take into account that in the real environment the amounts of resources contributed by the neighbors of an organization are estimated based on the past experiences of this organization. Since the estimates can be inaccurate, it seems to be reasonable to divide the organizations into a number of groups based on the observed resource contributions. Groups should be defined such that the contributions of the organizations inside a group do not vary much. Estimating organization contribution by taking an average of contributions in its group can be used to mask the estimate inaccuracies.

The neighbor sets computed by Algorithm 2 and consequently also the value of the sharing gain depend not only on the amount of resources contributed to the neighbors but also on the ordering among organizations with the same contributions. To eliminate this equivocalness, instead of analyzing the sharing gain of a single organization we rather study the average sharing gain of all organizations with the same contributions. If $d_i, \ldots, d_j$ are all organizations contributing a certain amount of resources then the *average sharing gain* of $d_i, \ldots, d_j$ is defined as $(g_{d_i} + \ldots + g_{d_j})/(j - i + 1)$. We make the following observation.

Observation 3.2. *If $c_{d_{i-1}} > c_{d_i} = c_{d_{i+1}} = \ldots = c_{d_j} > c_{d_{j+1}}$ and $j - i > n + 1$ then average sharing gain of $d_i, \ldots, d_j$ is higher than $g_{d_{j+1}}$.*

Proof. Direct conclusion from the definition of the average sharing gain and Observation 3.1. □

Observation 3.2 says that by contributing more resources organizations can increase their gain. We conclude that the neighborhood formation algorithm provides incentives for resource contributions. Organizations that contribute more experience higher payoffs.

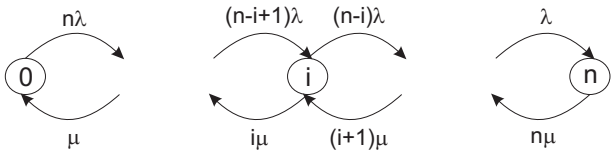

Figure 1. Evolution of the number of neighbors in demand.

3.4 Neighbor set size

The size of the neighbor set is an important parameter of our resource sharing mechanism. In this section we investigate the impact of the neighbor set size on the sharing performance.

We have shown in Section 3.3 that the gain of sharing expressed as the amount of resources acquired from the others depends on the own contribution of each organization. The information about the amount of resources contributed by an organization is computed by each neighbor based on the amount of consumed resources. In this respect, the total amount of resources consumed by all neighbors, rather than the amount of local resources intended for sharing determines the value of the sharing gain. It is, thus, in the best interest of an organization to maximize the utilization of the shared resources. The *utilization* of the shared resource is defined as the fraction of time when the resource is used by at least one neighbor. It is obvious that the utilization depends on the size of the neighbor set. We denote by u_n the value of utilization for the neighbor set size n.

We will investigate how the value of the neighbor set impacts the shared resource utilization. Resource utilization depends on the probability that a certain number of neighbors are in demand at a given time. The evolution of the number of neighbors in demand can be modeled as a birth-death process represented by a Markov chain, as depicted in Figure 1. The state of this process represents the number of neighbors that request the shared resource.

The transition rates from state i to j, $q_{i,j}$, of the birth-death process are given by the following formulas

$$q_{i,j} = \begin{cases} (n-i)\lambda & \text{when } j = i+1, \\ i\mu & \text{when } j = i-1, \\ 0 & \text{otherwise.} \end{cases}$$

In order to compute the equilibrium probabilities of the state transitions we use the method of a cut by applying the global balance condition to the set of states $0, \ldots, n$. In equilibrium the probability flows across the cut are balanced. If π_i is the probability of being in state i then the balanced cut condition translates to

$$(n-i)\lambda\pi_i = (i+1)\mu\pi_{i+1}.$$

We obtain the recursion

$$\pi_{i+1} = \frac{(n-i)\lambda}{(i+1)\mu}\pi_i.$$

Solving the recursion, all the state probabilities can be expressed in terms of that of the state 0, π_0

$$\pi_i = \binom{n}{i}\left(\frac{\lambda}{\mu}\right)^i \pi_0.$$

The probability π_0 is determined by the normalization condition $\sum_{i=0}^{n} \pi_i = 1$

$$\pi_0 = \left(1 + \frac{\lambda}{\mu}\right)^{-n}. \quad (1)$$

Since we assume that the resource is fully utilized as long as there is at least one neighbor in demand, the resource utilization is determined by the probability π_0 of none of the neighbors being in the demand state. The resource utilization u_n can be, thus, expressed as

$$u_n = (1 - \pi_0).$$

Substituting for π_0 the value computed in Eq. 1 we get

$$u_n = 1 - \left(1 + \frac{\lambda}{\mu}\right)^{-n}. \quad (2)$$

According to Eq. 2, the shared resource utilization converges to 1 exponentially with n. Hence, even small values of n result in high utilization. Keeping the value of n small has also a practical reason. Every new neighbor increases the overhead of resource sharing. The sharing overhead comes from several sources. First, the total amount of information that has to be maintained by each organization depends on n. Second, execution of multiple workloads at the same set of resources requires some mechanisms to match resources with workloads and isolate workloads from interfering with each other, e.g., in a form of virtual machines. The cost of resource matching and isolation increases with the number of workloads [8, 9, 10].

4 Experimental evaluation

4.1 Experimental setup

We evaluate our resource sharing mechanism on a resource model derived from real-world traces. We have collected statistics about the server resources used by IBM customers. These statistics have been extracted from the Server Resource Management (SRM) [11] system that reports historical and near real time trends of resources serviced by IBM.

The SRM data set provides for each resource information about the organization that owns the resource and average resource utilization in a designated time period. The data used in our experiments contains resource statistics for 1153 organizations collected during the period from 21 September 2006 until 28 September 2006.

We perform a discrete time simulation of a system composed of organizations sharing resources with characteristics described by the SRM traces. The particular type of resource that we are looking at is the CPU power measured in MHz. The functionality offered by the SRM system is limited to generation of reports summarizing resource usage over a time period of at least one day. SRM does not give access to fine grained statistics which are required to extract characteristics of individual workloads determining organization's demand periods. In our experiments we assume that an organization arrives to the demand state according to a Poisson process with a rate equal to the utilization of the resources owned by the organization. The demand durations are distributed exponentially with average equal to a single time unit.

In one of the experiments, we investigate the impact of the neighbor set size on the sharing performance. In the remaining experiments the size of the neighbor set is equal to 10. The value of parameter r of the neighborhood formation algorithm is set to 1. Initially, the local view of each organization contains 10 randomly selected organizations.

As a measure of the duration of the simulation, we use the average number of the neighborhood formation algorithm invocations. The exploration and selection phases of Algorithm 1 are executed one after another in constant time intervals. The duration of the time interval between subsequent invocations of the neighborhood formation algorithm in all experiments is equal to ten times the average interval between consecutive arrivals of an organization to the demand state. We stop the simulation after each organization invoked the neighborhood formation algorithm at least 100 times. Selection of this particular number is based on the fact that we did not observe a significant change in the obtained statistics for larger number of iterations of the neighborhood formation algorithm.

4.2 Results and discussion

4.2.1 Fairness

The first experiment targets at investigating the preservation of the fairness in resource contributions by our resource sharing algorithm. Figure 2 illustrates the correlation between the average amount of external resources consumed by an organization when it is in demand and the average amount of resources it contributes to its neighbors. Organizations in Figure 2 are ordered according to the increasing values of their contributions. In that figure, for each organization, two data points are shown: consumed resources (plus sign) and contributed resources (triangles). The dark curve is formed by plotting of many data points within a small region indicating the predominant trend.

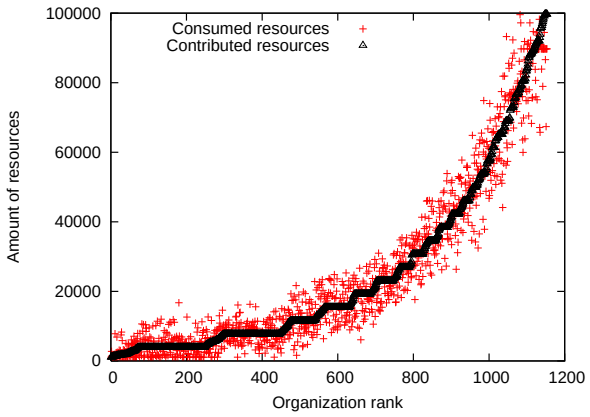

Figure 2. Average amount of resources consumed by each organization during the demand phase compared with the contribution. Organizations are ranked according to increasing contributions.

The results presented in Figure 2 allow us to assess the fairness of the proposed resource sharing mechanism. Analysis of the experimental results indicate that, on an average, the absolute difference between the amount of consumed and contributed resources equals 4149, which is around 16% of the average amount of the shared resources (24896). This result indicates that those contributing less get to consume less resources and those contributing more get to consume correspondingly higher amounts of resources. Thus, the system provides self regulation and prevents formation of freeriders.

4.2.2 Contributing neighbors

In the second experiment, we measure the fraction of bidirectional sharing relationships between organizations. This is a measure of cluster formation amoung organizations of similar resource lending and borrowing characteristics. In Figure 3, for each organization, we present the fraction of neighbors that are sharing resources with this organization (have this organization in their neighbor sets). The organizations in Figure 3 are ordered according to the increasing number of the contributing neighbors.

The maximal size of the neighbor set is equal to 10. Hence, the number of contributing neighbors vary between 0 and 10. The average number of contributing neighbors among all organizations is 7.9. Thus, on an average, 79bor-

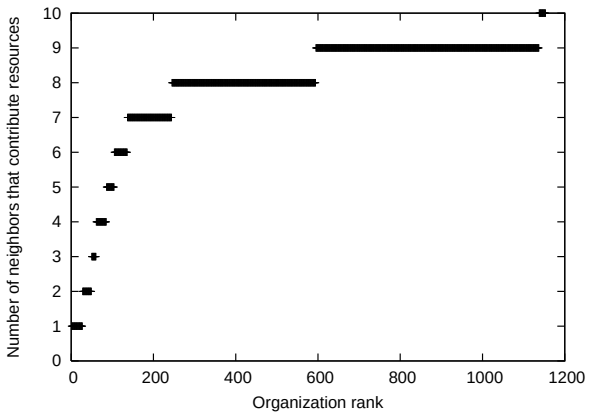

Figure 3. The number of neighbors that also contribute resources for the maximal neighbor set size equal to 10. Organizations are ranked according to the increasing number of contributing neighbors.

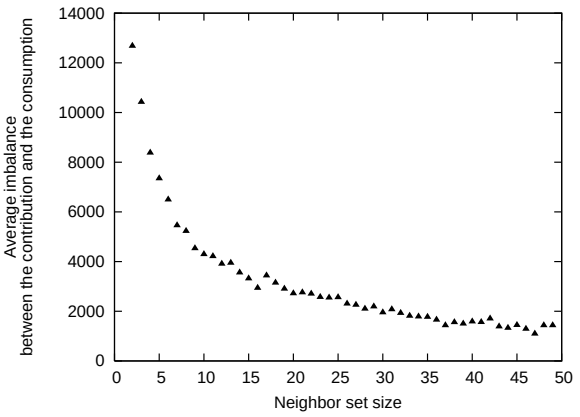

Figure 4. The correlation between the size of the neighbor set and the average imbalance (absolute difference) between the contribution and the consumption.

rowers) of a given organization are also lenders of resources to that organization. In other words, large fraction of organizations have borrower and lender relations among each other. This shows that bidirectional interactions are very strong among organizations and clustering of similar organizations is taking place. Note also that organizations contributing to a small number of other organizations tend to have fewer "friends in need".

The exceptionally small number of organizations with all 10 neighbors contributing resources comes from the fact that r (in our case equal to 1) neighbors are selected randomly regardless of their contributions.

4.2.3 Neighbor set size

In the third experiment we evaluate the impact of the neighbor set size on the average imbalance between the contributions and the consumptions of an organization. The imbalance is formally defined as the absolute difference between the amount of contributed and consumed resources. Figure 4 presents the value of the imbalance averaged over all organizations as it changes with the size of the neighbor set.

This set of experiments indicate that, as the neighbor set size (n) is increased, an organization has a better chance of recovering (i.e., borrowing when it is in demand) all the resources it shared with other organizations. This is less likely at small values of n. However, since there is a cost associated with the size of the neighbor set, there is size beyond which the diminishing returns makes it unattractive to use such large neighbor sets. In fact, the sweet spot is at the knee of the curve. This motivated our choice of n being around 10 for other experiments.

The experimental results confirm the conclusions of the analytical study presented in Section 3.4. Note from Figure 4 that at the knee of the curve, the utilization of the shared resources approaches its maximum, increasing the chances of an organization to find neighbors with similar contributions.

5 Conclusions

In this paper, we describe strategies suitable for multi-site resource sharing among autonomous organizations. The strategies presented here allow participants to exchange resources with one another, with the assurance that when an organization shares a local resource with other organizations in the grid, it stands to gain back use of a similar resource when it is in need of such a resource in the future. We describe mechanisms enabling these sharing strategies and present analytical and experimental results to show that our strategies ensure fairness (i.e., amount of resources contributed by an organization roughly equals the resources borrowed by that organization), scalability, and low overheads.

The approach presented in this paper tries to address the perennial resource management problem experienced by many organizations: the problem of determining the right level of investment in IT resources so that the organization is neither over-provisioning nor under-serving the local demand. Using the approach presented in this paper, organizations can invest and provision resources to meet the average demand over a time period. The peaks and troughs in demand around the average are handled by a grid based resource sharing mechanism. This grid based system allows

participants to borrow resources from others, when they are experiencing higher than average local demands and lend local resources to others, when they are experiencing a decline in local demand. The system described here ensures fairness so that a participant roughly gets to borrow what it lends to others over a period of time. The system described here is self-adaptive and scalable. It does not incorporate any centralized resource broker which makes it robust and extendable to thousands of participants.

Finally, we note that efficient and secure use of shared resources across organizations require further architectural and design considerations. First of all, today IT environments exhibit a high level of heterogeneity in both the hardware and software configurations. At the same time, grid applications usually lack the sophistication required to dynamically reconfigure to a particular execution environment. However, the problem of heterogeneity can be overcome by introducing a layer of indirection between the original resource configuration and the application execution environment, in the form of virtual machines (VMs).

Resource virtualization masks the heterogeneity of the resource instance specific configuration allowing the service to execute on practically any resource. Virtualization has also implications on security aspects of the resource sharing in untrusted environments. In addition, VMs are easier to control and monitor for policy conformance than arbitrary services running directly on the resources. E.g., virtualization technology offers out-of-the-box support for service migration and checkpointing. Disjoining the execution environment of different services by placing them inside separate VMs prevents the services from interfering with each other. We have developed a realization of VM based concepts in the Harmony architecture — a platform for delivery of customized services on grid resources [10, 12]. By adopting a system such as Harmony for their local computing environment, organizations can efficiently share underlying resources with one another using the mechanisms described in this paper.

References

[1] A. Leff, J. T. Rayfield, and D. Dias, "Meeting service level agreements in a commercial grid," *IEEE Internet Computing (Special issue on Grid Computing)*, July 2003.

[2] M. Srivatsa, N. Rajamani, and M. Devarakonda, "Combating state space explosion problem in policy-based performance evaluation," Tech. Rep., 2004.

[3] M. Devarakonda, V. K. Naik, and N. Rajamanim, "Policy-based multi-datacenter resource management," in *6th IEEE International Workshop on Policies for Distributed Systems and Networks*, June 2005.

[4] A. Li Mow Ching, L. Sacks, and P. Mckee, "Sla management and resource modelling for grid computing," in *London Communications Symposium*, London UK, September 2003.

[5] E. Adar and B. Huberman, "Free riding on gnutella," Xerox PARC, Tech. Rep., 2000. [Online]. Available: http://citeseer.ist.psu.edu/adar00free.html

[6] "Seti@home project page," http://setiathome.ssl.berkeley.edu/.

[7] J. Shneidman and D. Parkes, "Rationality and self-interest in peer to peer networks," in *2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)*, February 2003. [Online]. Available: http://www.eecs.harvard.edu/~parkes/pubs/iptps.pdf

[8] V. Naik, P. Garbacki, K. Kummamuru, and Y. Zhao, "On-line evolutionary resource matching for job scheduling in heterogeneous grid environments." in *Proceedings of International Conference on Parallel and Distributed Systems 2006 (ICPADS'06)*, Chicago, IL, July 2006.

[9] V. K. Naik, C. Liu, L. Yang, and J. Wagner, "On-line resource matching for heterogeneous grid environments." in *CCGRID'05*, Cardiff, UK, May 2005.

[10] V. K. Naik, P. Garbacki, and A. Mohindra, "Architecture for service request driven solution delivery using grid systems," in *IEEE International Conference of Services Computing (SCC'06)*, Chicago, IL, USA, September 2006.

[11] "SRM page." https://srm.raleigh.ibm.com.

[12] V. K. Naik, S. Sivasubramanian, and S. Krishnan, "Adaptive resource sharing in a web services environment," in *Middleware'04*, Toronto, Canada, October 2004.