VIRTUAL CLUSTERS: A DYNAMIC RESOURCE COALLOCATION STRATEGY FOR COMPUTING UTILITIES

Balasubramaneyam Maniymaran
Dept. of Electrical and Computer Engineering,
McGill University,
Montreal, QC, Canada.
Email: bmaniy@cs.mcgill.ca

Muthucumaru Maheswaran,
School of Computer Science,
McGill University,
Montreal, QC, Canada
Email: maheswar@cs.mcgill.ca

Abstract

This paper presents a concept called virtual clusters (VCs) to allocate the required resources for an application from a computing utility that can have a geographically distributed resource base. The VC creation process is modeled as a facility location problem and an efficient heuristic is devised to solve it. We extend the model to include an "overload partition" to a VC that can efficiently handle the loads surges. Extensive simulations have been conducted to examine the performance of VCs under different scenarios and to compare it with a fully dynamic scheme called the Service Grid. The results indicate that VC is more cost-effective and robust than Service Grid.

1 Introduction

Constant improvement in computer communication and microprocessor technology are finally making *computing utilities* (CUs) a reality. The core idea of a CU is to connect resources belonging to various organizations into a single system such that clients can get necessary resources for their computing needs in an on-demand basis. The major motivation for CUs is the opportunity it offers to the clients to outsource their computing and data processing activities to the utility provider. Several CUs have been created in the recent past using the Grid technology [1] for both research and commercial purposes. Since the CUs pool resources belonging to multiple geographically distributed organizations, clients of the CUs can acquire resources placed at different network localities. For example, a data archiving application might want to allocate storage from different geographical areas to prevent data losses due to natural disasters; Or a content service provider might want to allocate storage and processing capacities at different network locations so that data could be staged closer to the consumers.

In this paper, we assume that a CU already exists and it provides primitive mechanisms to allocate resources one-at-a-time. Assuming such a CU, we develop a mechanism to allocate resource *collections* for an application. This mechanism is referred to as the *Virtual Cluster* (VC). The VC is a co-allocated set of resources that meets some minimum performance expectations of a client. We model the VC resource allocation process as a *facility location problem* which optimizes the resource allocation (minimize the allocation cost) while satisfying the performance requirement. The dynamic variations in the demand for the service hosted on a VC poses a serious challenge to the idea of pre-allocating a set of resources to the VC. To address this issue, the concept of *overload partition* (OLP) is introduced. This is another optimized collection of resources that is shared among the different VCs such that those VCs can use these resources to handle the above normal demands. An extensive simulation study was carried out to examine the performance of VCs with and without the OLPs. The performance of the VCs were compared with a fully dynamic resource management scheme called the *Service Grid*.

Section 2 describes the VC concept in detail. In Section 3, the VC creation problem is formulated as a *capacitated fixed charged facility location problem* and its practical validity is discussed. Section 4 examines the ideas for solving the model and presents a heuristic. The details of a simulation study and the results are presented in Section 5. Section 6 provides some related works and Section 7 concludes this paper.

2 Virtual Clusters

Clusters are a popular model for high-performance computation and other variety of applications including web hosting and multi-media streaming. Nevertheless, one of their drawbacks is that they should be designed for worst-case demand conditions that leads to significant resource under utilization during off-peak hours. The VC concept presents the idea of a dynamic cluster where the resources are allocated on-demand from a base pool. It has a significant benefit over the traditional cluster as VCs can be resized dynamically or relocated to a different network locality in response to migrating service demands, thus maximizing resource utilization.

Figure 1 illustrates the steps involved in creating a VC. The VC creation process begins when a *service originator* (SO) makes a request to the CU manager. SOs are the clients of CU that lease resources from CU to host their services so that the ultimate *end users* (i.e., clients of the SOs) can easily access them. The request for a VC initiates a negotiation between the SO and CU manager that results in
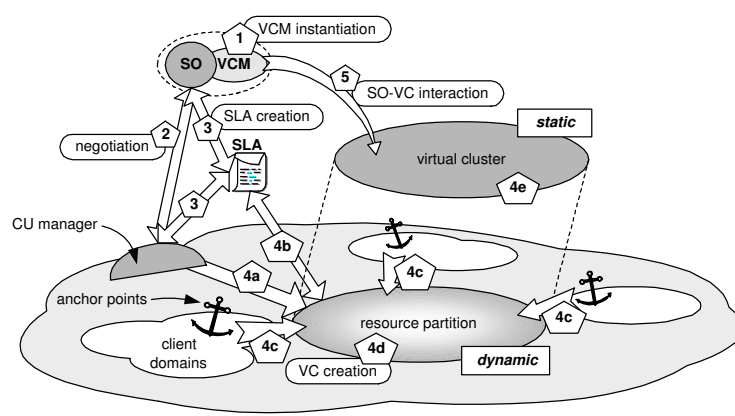
Figure 1. Example VC creation process on a PCU system.

a *service level agreement* (SLA). As introduced in [2], this SLA describes the agreed performance level, tolerance, implementation cost, and other metrics of performance. The CU manager creates a VC based on this SLA. This creation is constrained by both the SLA and the *anchor points* (APs) specified by the SO. APs are the "centroids" of the end-user demand concentration. For example, the gateway of an university network can be considered as the AP for the end-users in the university. Each AP is defined with a location and a load. The resources in the VCs are placed close to the APs to enable edge-delivery. The SO is provided with a virtual view of the VC that hides the dynamism and makes the VC look like a traditional cluster. It simplifies SO-level VC management. A software agent called *virtual cluster manager* (VCM) instantiated at the SO is used to manage this view. Depending on the SLA the VC is subjected to reallocations that can be triggered periodically or as demand levels change by predefined thresholds.

The VC creation is optimized for selecting a minimal set of resources that can support the expected demand for the services to be hosted. Therefore a VC is susceptible to overload conditions. Even though the a VC can dynamically change size to cater increasing demand, such reconfigurations are slow to handle demand spikes and can incur significant overhead when launched frequently. We introduce *overload partitions* (OLPs) to handle this situation. An OLP is a superset of resources that includes the resources currently in the VC. It is created with the same VC creation mechanism, but for a hypothetically higher demand level. While the resources in a VC are dedicated to the VC, the resources in the OLP are shared among multiple VCs. This is to reduce the potential resource underutilization problem.

3 Mathematical Model

We model the VC creation problem as a *facility location problem* (FLP) [3]. FLP is an optimization problem that tries to allocate facilities to demand points minimizing the total facility–demand allocation cost. In VC creation resources are considered as facilities, APs as demand points, and the QoS metric (for example, network delay) between resources and APs as the facility–demand allocation cost. There are many variations of FLPs and the VC creation algorithm uses the *capacitated fixed-charge location problem* (CFCLP). The CFCLP assumes a fixed cost for using each facility and each has limited capacity towards covering the demands. This model exactly matches the VC problem as allocation of resources incur a fixed cost (resource rental) and capacity of each resource is always limited.

We model the CU system as a graph with the servers and APs as the nodes and the network links as the edges. Let V_c be the set of potential server nodes and V_a be the set of APs. In addition, we define the following parameters: (a) d_{ij}: network delay between nodes i and j, (b) b_{ij}: bandwidth between nodes i and j, (c) m_j: capability of server j denoted by attributes such as processor architecture, (d) k_j: capacity of server j that denotes the maximum number of concurrent requests it can handle, (e) c_{ij}: cost of covering node i in V_a by node j in V_c as defined as below, (f) f_j: fixed cost for allocating node j as a facility, (g) h_i: demand at node i, (h) M: required capability of a suitable resource, and (i) B: intra-VC bandwidth. The covering cost c_{ij} is defined as $c_{ij} = \left(\alpha d_{ij} + \frac{\beta}{b_{ij}}\right)$ where α and β are VC specific and are given by the SO. For example, by making $\alpha = 0$, we can ignore the inter-node delay and just consider inter-node bandwidth. The covering cost c_{ij} can be considered as the inverse of the delivered QoS. The decision variables are defined as follows:

$$s_j = \begin{cases} 1 & \text{if node } j \text{ is part of the VC} \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{if node } j \in V_c \text{ covers node } i \in V_a \\ 0 & \text{otherwise} \end{cases}$$

With these parameters, the optimization problem for creating a VC becomes (with $i \in V_a$ and $j, k \in V_c$):

$$\text{minimize} \quad \sum_j f_j s_j + \sum_i \sum_j h_i c_{ij} a_{ij} \quad (1)$$

Subject to

$$m_j \geq Ms_j \quad \forall j \tag{2}$$

$$s_j + s_k \leq 1 \quad \forall\{j,k|b_{jk} < B\} \tag{3}$$

$$a_{ij} \leq s_j \quad \forall i,j \tag{4}$$

$$\sum_j a_{ij} \geq 1 \quad \forall i \tag{5}$$

$$\sum_i h_i a_{ij} \leq k_j s_j \quad \forall j \tag{6}$$

$$s_j = 0,1 \tag{7}$$

$$a_{ij} = 0,1 \tag{8}$$

The objective function given in (1) tries to minimize the number of servers (with s_j) in the VC while maximizing the delivered QoS (with c_{ij}). Using demand-weighted cost ($h_i c_{ij}$) places the server closer to the APs with high demand.

The Constraint (2) restricts the VC membership to nodes with given capabilities. This constraint is satisfied by the CU's resource discovery mechanism and hence need not be considered in the FLP solution procedure. The intra-VC bandwidth requirement is defined by the Constraint (3). The Constraint (4) denotes that a server can cover an AP only if it is part of VC. The Constraint (5) makes sure that every AP is covered. Finally, the Constraint (6) ensures that each server is serving within its capacity.

In this formulation, we assume that the QoS of a service is decided by the network bandwidth and the delay. This assumption is valid in most cases where the the response time and the throughput represents the QoS. Even in other cases, this formulation can be still used with c_{ij} redefined to suit the given scenario. The above formulation maximizes the delivered QoS, but does not guarantee a given QoS level. This becomes an issue when a value for QoS is agreed upon on the SLA. Nevertheless, it can be addressed by preprocessing the c_{ij} matrix such that, the c_{ij} elements that fail to meet the required QoS bound are replaced by a very large number. This transformation will make the model to converge to a solution that satisfies the stipulated bound.

In general, a FLP is formulated with $a_{ij} \geq 0$ instead of the Constraint (8). When the integer constraint is relaxed as such, the demand at an AP becomes *splittable* and can be served by multiple servers. It generally results in a tighter solution. However, with the integer constraint, each AP is provided with a "home server" and the unused capacity on the home servers naturally provides a safety margin in the system.

4 VC Allocation Heuristic

Solving CFCLPs is known to be NP-hard [4, 3]. Several algorithms have been proposed in the literature including local search and Lagrangian relaxation that provide near-optimal solutions [4, 3]. However, we need a solution technique that need not result in well optimized solution

but provides a solution fast. Although a VC is relatively long-lived, a fast VC creation procedure is necessary because a CU can be creating, destroying, and modifying VCs in a continuous manner in response to varying demand. This paper presents a centralized heuristic algorithm that is based on the *drop heuristic* given in [3], using a local search technique [4].

Our heuristic works in two phases. In the first phase, a feasible configuration is found for a VC that includes all available servers satisfying the Constraint 2 and each AP is greedily assigned to a server minimizing the covering cost c_{ij}. Once the feasible allocation is found, the second phase is initiated to drop excess serving capacity and reduce the cost of the VC. The dropping phase considers dropping each of the already selected servers and computes the resultant change in the objective function value. The value can increase or decrease. It may decrease because as servers are dropped, the first part of the objective function (1) decreases. On the other hand, it may increase because the covering cost making up the second part of the objective function can increase. Once the changes in the objective function value for dropping each server are calculated, the server producing the largest reduction is marked for removal from the VC. However, before this server is removed, the APs earlier covered by the leaving server need to be reallocated to other servers. The result of this reallocation has three possibilities: (1) reallocation is infeasible due to the capacity overflow a the other servers; (2) it is feasible, but the resulting objective function value is larger than the previous one due to the increase in the covering cost; or (3) it is feasible with a reduced objective function value. For the first two cases the algorithm terminates without removing the selected server; for the third case, the selected server is removed and the algorithm goes into another drop phase.

The intra-bandwidth constraint in Constraint (3) is applied prior to the drop heuristic to produce a connected set of resources on which the heuristic can be applied. The application of this constraint can yield several connected components and the minimum cost VC produced among these connected components is taken as the final VC.

5 Performance Evaluation

5.1 Simulation Setup

This simulation study compares the performances of the VC configurations (with and without OLP) with another fully dynamic scheme called Service Grid [5]. A web content serving application is assumed as the hosted service where the demand is posed in the form of document requests from various network localities. Sufficient bandwidth between the resources and clients is assumed so that no performance bottlenecks are encountered in the network. The requests arrivals at the APs are assumed to follow Poisson distributions and the request lengths are Gamma distributed. The request length is defined as the

time a server will be busy with that request. The average demand for the service is defined as the mean request length divided by the mean request interval. The mean request intervals and lengths are varied in the range of $1 \ldots 10$ and $100 \ldots 500$ seconds respectively. Because the Service Grid does not allow specifying internal connectivity constraints, the Constraint 3 (Section 3) is neglected. The simulations are carried out for a fixed duration of time.

The network topology for the simulation is generated using the *Tiers* [6] Internet topology generator. It creates a network with WAN-MAN-LAN topology. The APs are placed at randomly chosen LAN nodes and the rest of the nodes are assumed to be potential VC servers. The created topology had 56 nodes with 29 APs. The created VC had 8 servers in the dedicated partition and 4 more in the OLP. Once the VC configuration is determined, a discrete event simulator that simulate the arrival and processing events is used to measure the performance metrics. Currently, our simulator handles only discrete document transfers. A future study will extend the simulator to handle persistent connections that represent streaming media transfers.

5.2 Service Grid

The resource management in Service Grid is two-tiered with a set of *group managers* (GMs) and a *resource manager* (RM). RM assigns resources to GMs from the global pool based in response to the requests from GMs. Each GM maintains a set of servers to serve the requests originate within its locale. Each request is assigned to a server that gives the minimum $h_i c_{ij}$ value. The GM monitors these request bindings for capacity and/or QoS breaches. When these breaches exceed a predefined threshold, the GM requests an additional server from the RM. Similarly, the GM releases a server if it idles for more than a predefined threshold. The threshold values are set such that there are no oscillations. This one-by-one server addition/removal makes the Service Grid more dynamic than the VC.

5.3 Results and Discussions

The performance metric *response time* (T_r) is defined as the time taken from the origination of the request to the receipt of the service completion acknowledgment at the client. The response time is comprised of (a) *binding time*: the time taken for a request to be assigned to a server; (b) *communication time* (T_c): the time spent on transport and wait; and (c) *service time* = request length. To isolate the effect of service time, T_c is considered as a performance metric as well. When a request is assigned to a server that is over its capacity, the request is queued at the server increasing the communication time and thus the response time. Even though the request is assigned to a server with a satisfying $h_i c_{ij}$ value, the actual QoS might be poor due to the queuing. Therefore, the *percentage of queued requests* (P_q) is considered as another performance metric. Additionally,

the *percentage of QoS failed requests* (P_{QoS}) is used as another performance metric in Service Grid as there is always a possibility of some requests served under acceptable QoS. The maximum $h_i c_{ij}$ among all the AP–home server assignments in the corresponding VC is considered as the minimum acceptable QoS. Further, because the Service Grid, not like the VC, utilizes as many resources as needed and available, the number of servers used is also used to compare the performances. All the values presented in the rows of the tables are averages of at least five runs.

Tables 1 and 2 show the performances of the Service Grid and the VC (without OLP) with varying normalized mean request intervals. The normalized request interval of 1 denotes the nominal load condition for which the VC was created. The smaller the mean request interval, the higher the load as tabulated in the normalized load column. From the tables we can make following observations: (a) the performance of the Service Grid in terms of P_{QoS} and P_q is fairly steady; (b) the VC outperforms Service Grid at underload or nominal conditions, while the reverse is true at overload conditions; (c) Service Grid consumes more than double the number of serving resources compared to VC; and (d) even with that many resources consumed, a fraction of requests are always served with poor QoS. A similar performance is observed with the varying load in terms of request lengths.

Table 2 also shows the vulnerability of the VC to overload conditions. Because a VC operates with a limited set of resources, when it is overloaded even by a small amount, the requests quickly start queuing at the resources, dramatically degrading the performance. Table 3 shows the performance improvement when an OLP is added to the VC. The OLP is created for a hypothetical 1.5x overload condition. Even with the OLP, the VC uses only 12 servers in total. Therefore, the performance improvement comes without a dramatic increase in the cost of creating and maintaining the VC.

Tables 4 and 5 shows the performances when the demands are not uniform across the APs. Here the load injected by each AP is randomly assigned from a range. The width and the center of this range is varied and the results are shown in the tables. The results clearly shows that the VC performs much better in this real-world like conditions.

The Figure 2 shows the variation of total unused capacity in both Service Grid and VC with time. The lesser the unused capacity the better the resource utilization. From the figure it is clear that VC without OLP has the best resource utilization. The resource utilization in VC with OLP is comparatively low, but it still outperforms the Service Grid. In VC with OLP, the utilization of the dedicated VC resources is higher than the OLP resources. It is an outcome of the design, as the OLP resources are to be shared among multiple VCs.

The systems were compared with another metric called *unit utilization cost*. It is defined as total resource acquisition cost divided by used capacity of the resources. Figure 3 compares the systems based on this metric. Opti-

Norm. request interval	Norm. load	T_r (sec.)	T_c (sec.)	No. of requests	P_{QoS} (%)	P_q (%)	Total servers used	Max. servers at a time
1.5	0.7	462	1.319	283 972	0.01	10.9	15	12
1.1	0.9	461	1.350	398 176	0.01	13.4	21	18
1	1	462	1.359	404 119	0.01	13.6	21	19
0.9	1.1	463	1.914	528 964	0.01	28.4	26	26
0.7	1.5	462	2.091	670 002	0.05	34.1	26	26

Table 1. Performance of the Service Grid with mean request interval.

Norm. request interval	Norm. load	T_r (sec.)	T_c (sec.)	No. of requests	P_q (%)
1.5	0.7	461	0.643	267 996	0
1.1	0.9	461	0.655	370 873	0
1	1	460	0.655	376 761	0.00
0.9	1.1	9 452	8 991	486 472	99.4
0.7	1.5	20 058	23 597	583 872	99.6

Table 2. Performance of the VC without OLP with mean request interval.

Norm. request interval	Norm. load	T_r (sec.)	T_c (sec.)	No. of requests	P_q (%)
1	1	461	0.666	376 967	0
0.8	1.2	461	0.676	454 031	0
0.7	1.5	461	0.697	564 569	0.00
0.6	1.6	461	0.704	602 798	0.00
0.5	2	5 700	5 240	753 214	96.4

Table 3. Performance of the VC with OLP with mean request interval.

Norm. range size	Norm. range center	T_r (sec.)	T_c (sec.)	No. of requests	P_{QoS} (%)	P_q (%)	Total servers used	Max. servers at a time
0	1	922	2.940	212 930	0.01	27.4	20	16
1	1	889	2.893	213 217	0.01	25.2	19	15
1.5	1.25	1 038	4.478	213 241	0.02	41.1	21	17

Table 4. Performance of the Service Grid for nonuniform loading.

Norm. range size	Norm. range center	T_r (sec.)	T_c (sec.)	No. of requests	P_q (%)
0	1	461	0.666	377 042	0
1	1	513	0.672	376 769	0
1.5	1.25	729	86.5	376 260	16.2

Table 5. Performance of the VC with OLP for nonuniform loading.

mized resource acquisition in VC both cuts down the system cost and increases the utilization and therefore outperforms the Service Grid. Also it can be observed that the VC configurations hold this metric steady while the Service Grid shows significant oscillations due to the continuous inclusion and deletion of serving resources.

6 Related Work

As described in Section 5.2, Service Grid [5] is a fully dynamic resource allocation scheme that allocates resources with required QoS on a per request basis to meet the demand conditions. Although its performance is found to be stable over a range of loading condition, our study shows that it is less cost-effective.

Océano [2] is a resource management system for a utility that manages resources at a single site. Océano has two types of servers allocated for a service: a fixed number of "whale" servers and a varying number of "dolphin" servers. Because the VC spans multiple geographical locations, its FLP based allocation process is different from that of Océano which uses a simple event driven system to decide when it is appropriate or essential to add or delete a server from a service. Océano also is aware of QoS constraints.

Cluster on demand (COD) [7] is an architecture that shares our views of the global resource pool and service specific resource collections. However, the *constraint satisfaction problem* algorithm used by COD for resource scheduling considers only the capabilities of the resources. In COD, it is the SO's responsibility to determine the number and locations of the resources needed to satisfy the QoS requirements.

Cluster reserves [8] is a scheme for providing performance isolation among wide area services. Unlike the other
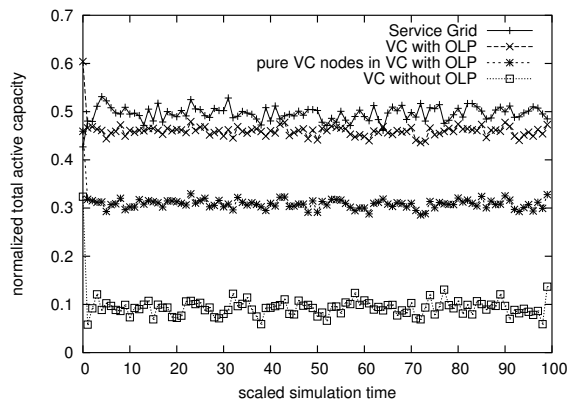
Figure 2. Resource utilization within a cluster.

Figure 3. Unit utilization cost of a cluster.

architectures where the granularity of resources allocations is at a resource level, granularity in cluster reserves is finer allowing multiple services to share a single resource. It uses collections of OS level *resource containers* to provide performance isolation. Optimization algorithms are used to determine the portion of a node that a service should occupy to maximize the total resource utilization.

There are number of projects in the literature [9, 10, 11] that approach the placement of web server replicas and caches as facility location problems. In general, these approaches consider hop distance as the parameter of performance. Their formulations differ from ours in a number of ways, since our problem has to deal with capabilities, capacities, and different QoS metrics. Further, the concept of anchor point is unique in our work.

7   Conclusion

This paper presents a new mechanism called *Virtual cluster* (VC) for "soft" wiring a cluster from pool of resources maintained by computing utilities. Generally clusters are designed for maximum demand conditions resulting in poor overall resource utilization. Here, we present an alternative approach, where each VC has a dedicated and a shared set of resources. The dedicated portion guarantees a minimal level of service with the shared partition adding robustness to withstand the demand variations. Also this paper presents an optimization based algorithm for VC creation that produces minimum cost resource collection that satisfies the performance requirements.

Our simulation study compared the VC approach with a fully dynamic approach called Service Grid. The VC was able to improve the performance of Service Grid by about 40% while consuming 33% fewer serving resources. In a CU environment this is very significant because VCs will be able to guarantee a minimal level of service while consuming less resources. Conversely, the CU will be able to host more applications and increase its revenue for a fixed global capacity.
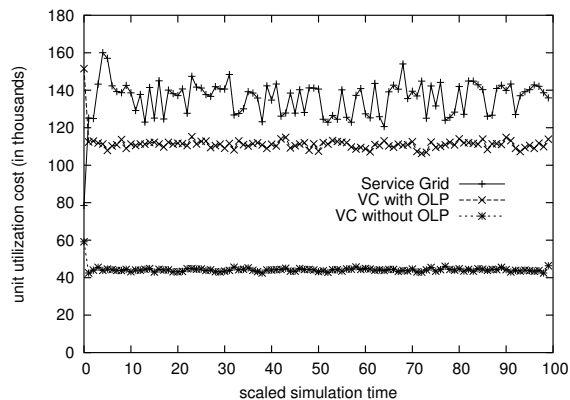
References

[1] I. Foster, C. Kesselman, and S. Tuecke,   "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal on Supercomputer Applications*, 2001.

[2] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, and et al., "Océano – SLA based management of a computing utility," in *IEEE/IFIP International Symposium on Integrated Network Management*, May 2001.

[3] M.S. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications*,   John Wiley & Sons, Inc., New York, NY, 1995.

[4] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a local search heuristic for facility location problems," in *Journal of Algorithms*, 2000.

[5] Byoung Lee and Jon B. Weissman, "Dynamic replica management in the Service Grid," in *IEEE 2nd International Workshop on Grid Computing*, Nov. 2001.

[6] M. B. Doar, "A better model for generating test networks," in *IEEE Globecom*, Nov. 1996, pp. 86–93.

[7] Justin Moore and Jeff Chase, "Technical report: Cluster on demand," Tech. Rep., Department of Computer Science, Duke University, May 2002.

[8] Mohit Aron, Peter Druschel, and Willy Zwaenepoel, "Cluster reserves: A mechanism for resource management in cluster-based network servers," in *ACM Sigmetrics 2000 International Conference on Measurement and Modeling of Computer Systems*, June 2000.

[9] Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz, and Yuval Shavitt, "Constrained mirror placement on the Internet," in *INFOCOM*, 2001, pp. 31–40.

[10] Israel Cidon, Shay Kutten, and Ran Soffer, "Optimal allocation of electronic content," in *INFOCOM*, 2001, pp. 1773–1780.

[11] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker, "On the placement of web server replicas," in *INFOCOM*, 2001, pp. 1587–1596.