Heterogeneous Resource Allocation under Degree Constraints in Peer-to-Peer Networks

Chapram Sudhakar
Department of CSE,
National Institute of Technology,
Warangal, INDIA.
chapram@nitw.ac.in

Vatsal Rathod
Department of CSE,
National Institute of Technology,
Warangal, INDIA,
vatsaljrathod@gmail.com

T. Ramesh
Department of CSE,
National Institute of Technology,
Warangal, INDIA,
rmesht@nitw.ac.in

Abstract—**Peer-to-peer model is one of the commonly used model for distributed computing. Some of the peers are having demand for certain resources some others may be having additional capacity of resources. Peers may have limitations on the number of concurrent connections (degree). In the present work allocation problem of peers having demand, capacity and degree is considered. The problem is to find an allocation of peers such that the number of peers allocated to a particular peer *P* should not exceed the degree of *P* and total demand of allocated peers should not exceed the capacity of *P*, while maximizing the overall throughput. Two versions namely Offline (when peers are known in advance) and Online (when peers can join and leave the network at any time) versions of the problem are considered. By introducing degree constraints the problem becomes NP-complete. Resource augmentation based three approaches are proposed to solve this problem. The performance (in terms of throughput) and the cost (in terms of disconnections and reconnections) of the proposed approaches is compared through a set of extensive simulations. The observed results are impressive.**

Keywords—Peer-to-Peer Networks, Heterogeneous Resource Allocation, Resource Augmentation.

I. Introduction

In Peer-to-Peer network, where individual node is called peer, each peer acts as supplier, as well as consumer of resources. Each peer has three properties: capacity, degree and demand. Total amount of resources available at a peer is called as its capacity. Number of connections handled by a peer simultaneously is called as degree of that peer. Demand is the need (in terms of resources) of peer, which may be satisfied by other peer's resources. Each property has a constraint associated with it. These constraints are capacity constraint, demand constraint and degree constraint. The usage of the terms capacity, degree and demand is always depends upon the context [1]. For example, capacity may be the quantity of data that it can send, or the number of flops that it can process during one time unit. Degree will be maximal number of open TCP connections handled by peer simultaneously. Demand will be the number of tasks that a peer need to process during one time unit, or its computational demand per time unit [1].

Peer in the network is denoted by P_i. Each Peer P_i has the capacity b_i, degree d_i and demand w_i. Capacity allocated by peer P_j to peer P_i is denoted as w_i^j. The three constraints associated with each peer are given below.

- Capacity constraint: Sum of the demand of other peers satisfied by a particular peer would be less than or equal to its capacity.

$$\forall j \sum_{i=1}^{n} w_i^j \leq b_j$$

- Degree constraint: For each peer number of peers allocated to it would be less than or equal to the degree of the peer.

$$\forall j \; Card\{\, i : w_i^j > 0 \,\} \leq \; d_j$$

- Demand constraint: For each peer, sum of the demand satisfied by various other peers would be less than or equal to its demand.

$$\forall i \sum_{j=1}^{m} w_i^j \leq w_i$$

The problem is to find an allocation of peers such that the number of peers allocated to a particular peer is less than the peer's degree and their overall demand is less than the peer's capacity, while maximizing the overall throughput [9]. In literature this problem is termed as Maximized Throughput Bounded Degree (MTBD) problem[1, 3], which can be formally specified as:

Maximize $\sum_{j=1}^{n}\sum_{i=1}^{n} w_i^j$ under the capacity, degree and demand constraints.

In the corresponding decision problem, Throughput-Bounded-Degree-Decision, the goal is to decide whether the throughput k is achieved, given a set of peers. This is NP-Complete problem and can be reduced to 3-partition problem [5].

There are two possible versions of this problem, namely offline and online versions. In the offline version static set of peers are considered and in the online version peers can leave and join the system dynamically. For comparison of performance, the total throughput achieved is the key parameter for the offline version. Whereas for the online version, average cost per peer, i.e., number of changes in the allocation caused by a peer arrival or departure, is the key parameter for comparison. In the present work, offline and online versions of solution to the problem is developed using resource augmentation approach.

Rest of the paper is organized as follows. In Section II, related work is presented. In Section III, offline and online algorithms are described. Experimental results are discussed in Section IV and conclusions are given in Section V.

II. Related Work

Resource allocation in existing peer to peer system can be roughly classified into the following four categories: Best Fit, Random Selection, Preemption Based and Reputation or Rank Based algorithms [4].

Another closely related problem is virtual machine placement, i.e., the process of mapping virtual machines to physical machines. Placement goal can either be maximizing the usage of available resources or saving the power by shutting down of some servers [6]. Placement algorithms can be classified into two categories on the basis of their placement goal: Power Based Approach or Application QOS Based Approach

For client/server architecture Beaumont et al., proposed the heterogeneous resource allocation model, under degree constraints [1,3]. In this model of computing platform, servers have heterogeneous capacity and degree, and clients have heterogeneous demand. Clients generate the tasks that are transferred and processed by the servers. This has been formulated as MTBD problem.

Peer-to-peer networks have heterogeneity of resources, demand, capacity and degree constraints. This also can be formulated as a MTBD problem and good results are observed in the current work.

III. Proposed Algorithm

In this section the proposed two algorithms, one for offline (static) problem and another for online (dynamic) problem are presented.

A. *Offline Algorithm*

Let $P : \{P_i\}_{i=1}^{n}$ is the set of Peers, b_i is the capacity of peer P_i and d_i is the maximal number of peers that P_i can handle simultaneously. The demand of a peer P_i is denoted by w_i. Let w_i^j denotes the capacity allocated by peer P_j to peer P_i

LP : Ordered list of Peers with respect to some resource attribute

$A : \{A_i\}_{i=1}^{n}$ Allocation set for each peer P_i,

$LP(l,k) = \sum_{i=l}^{k} w_i$: Sum of demand of the sorted list of peers from $P_l, P_{l+1}, \dots, P_{l+k}$.

```
1.  $P : \{P_i\}_{i=1}^{n}$ Set of Peers
2.  $A : \{A_i = \{\emptyset\}\}_{i=1}^{n}$ Initial allocation
3.  $LP$ : Ordered list of Peers
4.  For each peer, $i$=1 to $n$ do,
5.     If $\exists l$ such that $LP(l, l+d_i-1) < b_i$ and $LP(l, l+d_i) \geq b_i$ then
6.        Pick $l$ s.t. $LP(l, l+d-1) < b$ and $LP(l+1, l+d) \geq b$
7.        Split the $P_{l+d_i}$ in $P'_{l+d_i}$ and $P''_{l+d_i}$ with $w_{l+d_i} = w'_{l+d_i} + w''_{l+d_i}$
8.        $A_i = \{P_l, P_{l+1}, \dots, P_{l+d_i-1}, P'_{l+d_i}\}$
9.        Remove the $P_l, P_{l+1}, \dots, P_{l+d_i-1}, P_{l+d_i}$ from $LP$ and insert $P'_{l+d_i}$ in $LP$
10.    End If.
11.    If $LP(1, d_i) \geq b_i$ then
12.       Search smallest $k$ such that $LP(1,k) \geq b_i$
13.       Split the $P_k$ into $P'_k$ and $P''_k$ with $w_k = w'_k + w''_k$
14.       Set $A_i = \{P_1, P_2, \dots, P_{k-1}, P''_k\}$
15.       Remove $P_1, P_2, \dots, P_k$ and insert $P'_k$ in $LP$
16.    End If
17.    If $LP(n-d_i, n) < b_i$ then
18.       $A_i = \{P_{n-d_i+1}, P_{n-d_i+2}, \dots, P_n\}$
19.       Remove $P_{n-d_i+1}, P_{n-d_i+2}, \dots, P_n$ from $LP$
20.    End If
21. End For
22. Return $A = \{A_i\}_{i=1}^{n}$
```

Algorithm 1: Offline Algorithm

Throughout the computation, offline algorithm shown in Algorithm 1., maintains an ordered list of remaining peers called free list. At each step, it picks up a peer P_i from the peer list and goes through the list to find a suitable set of peers for P_i. A suitable set of peers is a set of d_i+1 consecutive peers in the ordered list, called an interval of length d_i+1, with a condition of their total capacity being at least b_i, and sum of the capacities of the first d_i peers being less than b_i.

These constraints ensure that the whole capacity and the maximum degree of the peer are used. If such interval (l, l+d_i) exists offline algorithm select the rightmost one such that LP(l, l+d_i-1) < b_i and LP(l+1, l+d_i) ≥ b_i. It may happen that no suitable interval exists for any server. This can happen because of two reasons. The first reason is that any set of d_i+1 peers has not enough demand to use all the capacity b_i (i.e., the overall capacity of the d_i+1 largest peers is not big enough). In this case, offline algorithm allocates to peer P_i the d_i largest peers (the last d_i peers in the ordered list). Second, if any set of d_i peers has overall capacity larger than b_i (i.e., the overall capacity of the d_i smallest peers is already too large), then the algorithm simply allocates the k smallest peers, where k is the smallest index such that LP(1, k) ≥ b_i. In this case also, the last peer may be split and assigned, which will result in a left over demand of LP(1, k) – b_i. An example of the execution of one such general step of the offline algorithm is given in Fig. 1. In that diagram a peer with a degree of 2 and capacity of 30 is considered. The interval 2-4 is satisfying the first condition and fourth request is split into two requests and assigned to the peer under consideration. The left over request in the list is with a demand of 27.

Throughput: Total throughput achieved by the algorithm is the ratio of sum of all the demand satisfied by actual demand of the peer to peer network.

Throughput = $\frac{\Sigma w_i'}{\Sigma w_i}$

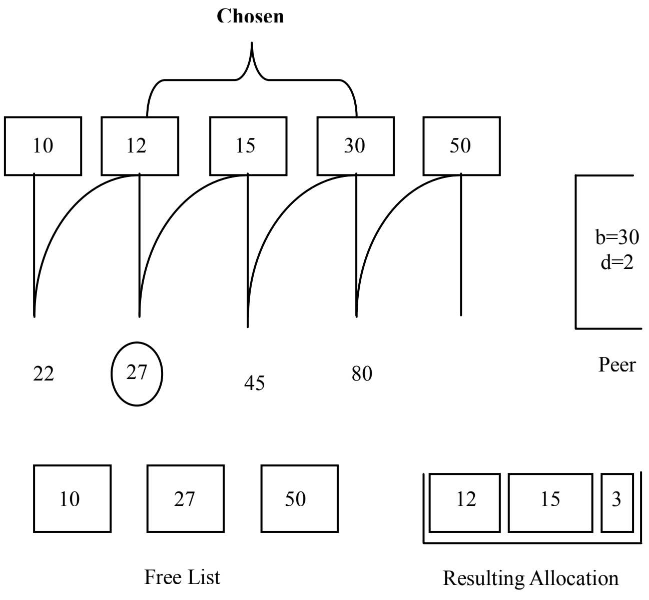

Fig. 1: Allocation Policy

B. Online Algorithm

In real time environment peers join and leave the network dynamically. Therefore to develop an online version of the above algorithm a new term "round" is introduced. Whenever a peer joins the network or leaves the network it would be considered as starting point of a new round. For each round the algorithm maintains free list of peers. Offline algorithm is applied on this instance and allocation for this instance is obtained. This process is repeated whenever a new round starts.

Let LP^t denotes a set of peers present at round t in sorted order and $w_i^j(t)$ is allocated capacity value by peer P_j to peer P_i at round t. Peer P_i connects to peer P_j at round t if $w_i^j(t-1) = 0$ and $w_i^j(t) > 0$. Symmetrically peer P_i disconnects from peer P_j at round t if $w_i^j(t-1) > 0$ and $w_i^j(t) = 0$. If $w_i^j(t-1) > 0$ and $w_i^j(t) > 0$, it indicates no change. Let $\mathcal{N}_i^t = \left|\{j, w_j^i(t-1) \neq w_j^i(t)\}\right|$ denotes the number of changes occurring at peer P_i at round t. Online version of the algorithm is shown in Algorithm 2.

```
1. Upon start of a new Round
2.     Set P: List of peers
3.     LP: Ordered list of peers available for current round.
4.     Apply Offline algorithm to this Instance.
5.     Return Allocation of the current round.
```

Algorithm 2: Online Algorithm

If the algorithm includes l changes per peer in the connections, in the peer-to-peer network during r rounds, then the average cost is defined as:

Average cost per peer = $l = \frac{\sum_{t=1}^{r}\sum_{i=1}^{n}\mathcal{N}_i^t}{N}$

Here N is the maximum number of peers available in all the rounds.

C. Sorting Criteria

Proposed algorithm finds the suitable set of peers (interval) from the ordered list of peers (free list). At each iteration algorithm maintains an ordered list of peers. Sorting is done based on three different parameters.

1. Based on remaining demand (Type 1):

Sort the list of peers based on peers' remaining demand. Interval chosen by the algorithm would have largest demand that satisfies all the constraints of host.

2. Based on remaining degree (Type 2):

Sort the list of peers based on peer's remaining degree. Interval chosen by the algorithm would have largest degree that satisfies all the constraints of host.

3. Based on remaining demand per degree (Type 3):

Sort the list of peers based on peer's remaining demand per connection. Interval chosen by the algorithm would have largest degree that satisfies all the constraints of host.

IV. Experimental Evaluation

In this section test data generation, experiments conducted are presented.

A. Test Data Generation

An instance of set of peers is generated randomly, with focus on the realistic scenario [11]. Allocation is more difficult when the sum of the peer's capacity ($\sum_i b_i$) is roughly equal to sum of the peers' demand ($\sum_i w_i$). Degree of peer depends upon the capacity of peer. If capacity increases, degree will also increase and if capacity decreases, degree will be decreased. Randomly generated instances follow all these rules. In the experimentation maximum degree of a peer is varied from 5 to 25 and the numbers of peers are varied from 50 to 1000.

B. Simulation of Offline Algorithm

Figure 2 and 3 shows the graph of utilization parameter for offline algorithm vs. number of peers in network with varying degrees. It is clear from the graph, that Type 2, which sorts the peer list by remaining degree gives the best throughput among the three methods. In most of the cases performance of Type 1 and Type 3 are close to each other. For a particular degree, it is observed that, if number of peers in the peer-to-peer network increases, total throughput achieved by the algorithm remains almost same in case of all three approaches. When the number of peers is increased for the maximum degree 25, total throughput achieved by the algorithm is near to maximum. This observation is true for all three different approaches of the algorithm.

Figure 4 and 5 shows utilization of offline algorithm vs. max degree in network with varying numbers of peers. In this case also Type 2 gives the best throughput among the three. In most of the cases performance of Type 1 and Type 3 are close to each other. It is clearly observed from the graph that, as the maximum degree available in peer-to-peer network increases, total throughput achieved is also increasing for all three approaches. As the maximum degree available in the peer-to-peer network exceeds some point, total throughput of all three approaches is close to maximum.

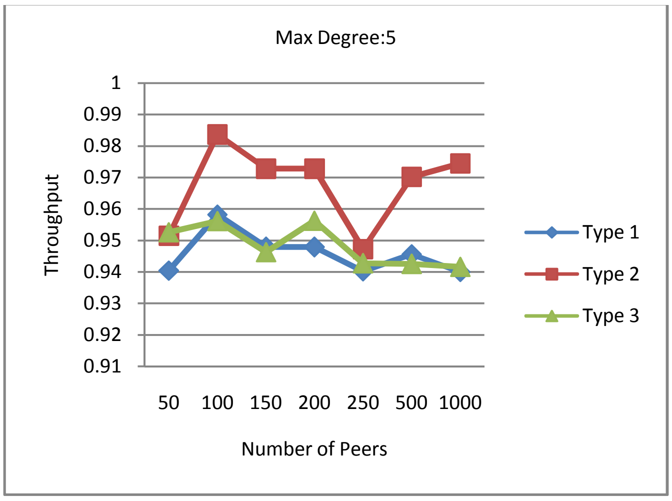

Fig. 2: Throughput with Max Degree=5

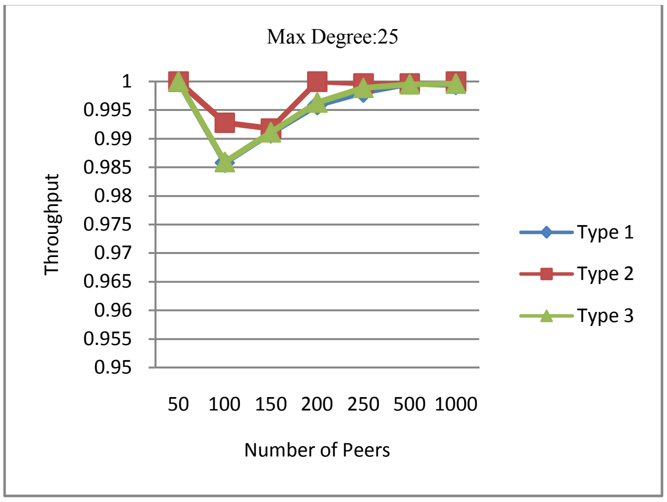

Fig. 3: Throughput with Max Degree=25

C. *Simulation of Online Algorithm*

In the online approach peers can enter and exit the network at any time. Change in the connection from the previous round to the next round would be the cost of the peer for that particular round. Average cost of peer vs. number of peers in network with change in maximum degree of peers' in network is shown in figure 6 and 7. Type 1 has higher average cost per peer than Type 2 and Type 3. For a low value of maximum degree, average cost for Type 2 and Type 3 are close each other. Here Type 3 is exhibiting best performance among all the three approaches.

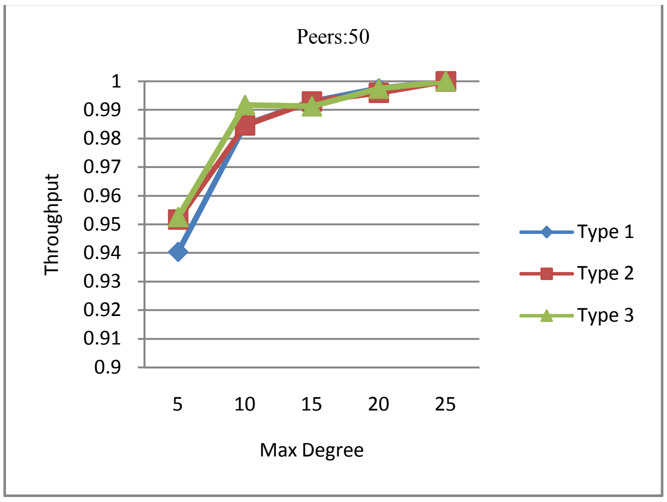

Fig.4: Throughput with peers=50

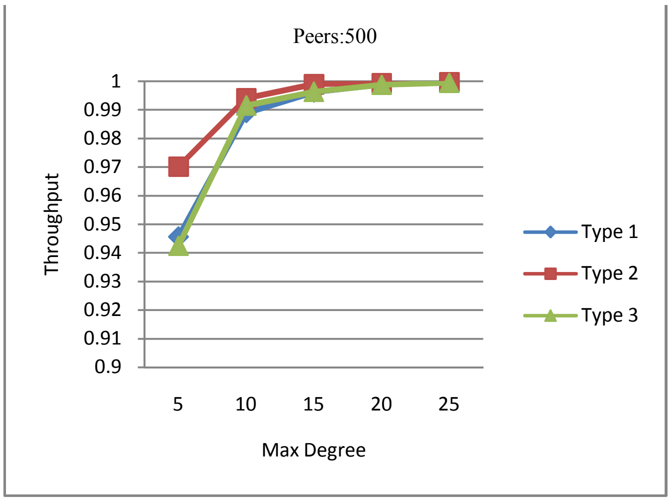

Fig. 5: Throughput with peers=500

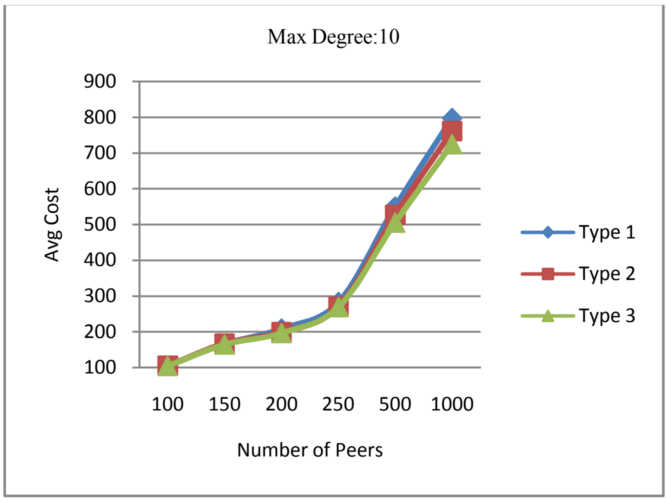

Fig.6: Avg. Cost with Max degree=10

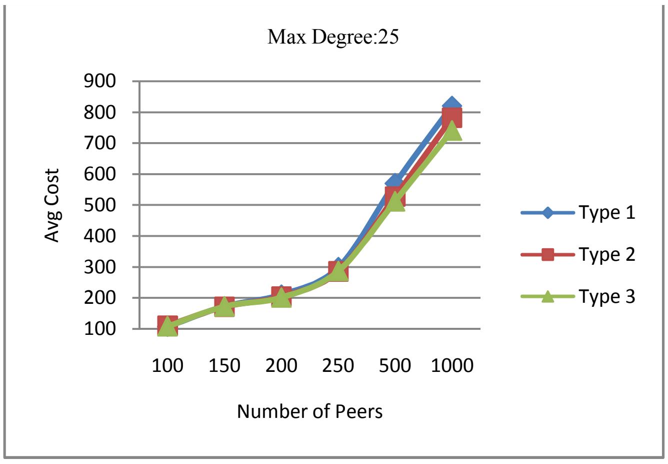

Fig. 7: Avg. Cost with Max Degree=25

V. CONCLUSIONS

In this work, the resource allocation problem is considered for a peer-to-peer network. Main contribution of this work is to introduce the degree constraint for each peer of a peer-to-peer network. Even if this additional constraint makes the resource allocation problem NP-Complete, only a very small resource augmentation on the peers' degree is sufficient to solve this problem approximately. Online version of the solution is also analyzed where the peers can change during the execution.

For ordering the peers, three different approaches are followed. Those are based on remaining demand of peers, remaining degree of peers and ratio of remaining demand to degree, which are referred as Type 1, Type 2 and Type 3 respectively. Simulation shows that Type 2 has a best utilization ratio among the three approaches. In the context of average cost per peer, Type 3 gives the best performance.

Present resource allocation model is developed for peer-to-peer network. This work can be extended to consider more complex virtual topologies (overlay networks) to organize participating nodes.

REFERENCES

[1] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves, "Heterogeneous Resource Allocation under Degree Constraints", Proc. IEEE Int'l Conf. Parallel and Distributed Systems, 2013

[2] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves, "On- Line Allocation of Clients to Multiple Servers on Large Scale Heterogeneous Systems", Proc. 18th Euromicro Conf. Parallel, Distributed and Network-Based Processing (PDP),2010.

[3] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves, "Allocation of Clients to Multiple Servers on Large Scale Heterogeneous Platforms", Proc. IEEE 15th Int'l Conf. Parallel and Distributed Systems 2009.

[4] Dongyu Liu, Fei Li, Songqing Chen, "Towards Optimal Resource Utilization in Heterogeneous p2p Streaming", ICDCS'09, 29th IEEE International Conference on Distributed Computing Systems, 2009.

[5] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness.", WH Freeman, San Francisco, 1979.

[6] Anjana Shankar, Prof.Umesh Bellur, "Virtual Machine Placement in Computing Clouds", Department of Computer Science and Engineering, Indian Institute of Technology Bombay, 2010.

[7] Yonghe Yan, Adel El-Atawy and Ehab Al-Shaer, "Ranking-based Optimal Resource Allocation in Peer-to-Peer Networks", 26th IEEE International Conference on Computer Communications, INFOCOM 2007.

[8] Clark, Dave, Bill Lehr, Steve Bauer, Peyman Faratin, Rahul Sami, and John Wroclawski, "Overlay Networks and the Future of the Internet", Communications and Strategies 63 (2006): 109.

[9] B. Hong and V.K. Prasanna, "Distributed Adaptive Task Allocation in Heterogeneous Computing Environments to Maximize Throughput", Proc. 18th Int'l Parallel and Distributed Processing Symp., 2004.

[10] Vinothina, V., Dr R. Sridaran, and Dr PadmavathiGanapathi, "A survey on resource allocation strategies in cloud computing", International Journal of Advanced Computer Science and Applications 3, no. 6, 2012.

[11] D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage", Proc. IEEE/ACM Fifth Int'l Workshop Grid Computing, pp. 365-372, 2004.

[12] M. Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing", Technical Report UCB/EECS-2009-28, EECS Dept., Univ. of California, Berkeley, 2009.

[13] C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert, "Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms", IEEE Trans. Parallel and Distributed Systems, vol. 15, no. 4, pp. 319-330, Apr. 2004.

[14] A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers", Proc. IEEE/ACM 10th Int'l Conf. Cluster, Cloud and Grid Computing, pp. 577-578, 2010.

[15] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M.Q. Dang, and K. Pentikousis, "Energy-Efficient Cloud Computing", The Computer J., vol. 53, no. 7, p. 1045, 2010.

[16] D. Bertsimas and D. Gamarnik, "Asymptotically Optimal Algorithm for Job Shop Scheduling and Packet Routing", J. Algorithms, vol. 33, no. 2, pp. 296-318, 1999.

[17] Vmware, http://www.vmware.com/virtualization/, 2012.

[18] Xen, http://www.xen.org/, 2012.

[19] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of the-Art and Research Challenges", J. Internet Services and Applications, vol. 1, no. 1, pp. 7-18, 2010.

[20] C.A. Phillips, "Optimal Time-Critical Scheduling via Resource Augmentation", Algorithmica, vol. 32, no. 2, pp. 163-200, 2008.