INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2010, VOL. 56, NO. 1, PP. 55-62
Manuscript received December 13, 2009; revised March, 2010.
10.2478/v10177-010-0007-2

Random Approach to Optimization of Overlay Public-Resource Computing Systems

Grzegorz Chmaj, Krzysztof Walkowiak

***Abstract*— The growing need for computationally demanding systems triggers the development of various network-oriented computing systems organized in a distributed manner. In this work we concentrate on one kind of such systems, i.e. public-resource computing systems. The considered system works on the top of an overlay network and uses personal computers and other relatively simple electronic equipment instead of supercomputers. We assume that two kinds of network flows are used to distribute the data in the public-resource computing systems: unicast and peer-to-peer. We formulate an optimization model of the system. After that we propose random algorithms that optimize jointly the allocation of computational tasks and the distribution of the output data. To evaluate the algorithms we run numerical experiments and present results showing the comparison of the random approach against optimal solutions provided by the CPLEX solver.**

***Keywords* — Computing Systems, Overlay, P2P, Unicast, Optimization**

I. Introduction

In recent years we can observe the advent of new architectures that provides powerful capabilities for creating advanced information technology services. Both academia and industry need effective computing systems to address various research problems e.g. data analysis, protein folding, experimental data acquisition, financial modeling, earthquake simulation, and climate/weather modeling, astrophysics and many others [14], [19]. Two architectures of computing systems are widely used to meet the growing need for computational power: public-resource computing systems and Grids. Public-resource computing systems also known as global computing or peer-to-peer computing are focused on the application of personal computers and other relatively simple electronic equipment instead of supercomputers and clusters [1], [12]. As an example of the public-resource computing project we can enumerate SETI@home started in the 1999 [1]. SETI@home has been developed using BOINC (Berkeley Open Infrastructure for Network Computing) software [1]. Even though both public-resource computing and Grid computing has the same goal of better utilizing various computing resources, there are differences between them. Grid computing uses more formal organization – elements of the grid (supercomputers, clusters, research labs, companies) are centrally managed, permanently available online, connected by high bandwidth network links. In contrast, participants of public-resource computing projects are individuals with PCs running Windows, Macintosh or Linux operating systems connected to the Internet by DSL access links.

Since most of current computing systems including Grids and public-resource computing systems are mostly implemented in a distributed manner, the network has an important role [14]. Most of previous research on scheduling and resource management of Grid systems do not consider comprehensively the network aspects – usually the simplest unicast transmission is applied and very few constraints on the network layer are considered [14]. Consequently, in this paper we focus mainly on the problem of data distribution in network computing systems with a special focus on public-resource computing. Since many distributed systems based on the Peer-to-Peer (P2P) approach use some kinds of randomness [6], [18], the main goal and contribution of the paper is the evaluation of random algorithms in comparison against optimal results and other heuristics.

The remainder of the paper is organized in the following way. In Section II we formulate and motivate the optimization model of a public-resource computing system. Section III includes description of random algorithms. In Section IV we show results of experiments. Section V contains the related work. Finally, the last section concludes this paper.

II. Models of Overlay Public-Resource Computing Systems

In this chapter we will formulate optimization models of overlay public-resource computing systems. The model assumptions are based mostly on the most popular public-resource computing system, i.e. the BOINC system [1] and recommendations of earlier authors included in [6-8], [10-14] [16-18] [19-24]. In our research we focus on the problem of data distribution, therefore we do not deal in detail with a number of issues related to network computing systems such as: management, security, diverse resources. Nevertheless, due to the layered architecture of both: computer networks (e.g. ISO/OSI, TCP/IP, overlays) and computing systems (e.g. Globus Toolkit) the proposed models can be used in many scenarios independent of protocols and technologies related to computer networks and computing systems.

Nodes (vertices) of the public-resource computing system (e.g. PCs or other computers) representing peers are denoted using index $v = 1,2,\ldots,V$. Each vertex v is connected to the overlay network using an access link with limited download rate (d_v) and upload rate (u_v). In addition, each vertex v has a

This work was supported by The Polish Ministry of Science and Higher Education under the grant which is being realized in years 2008-2011.

Grzegorz Chmaj is with Chair of Systems and Computer Networks, Wroclaw University of Technology (e-mail: grzegorz@chmaj.net).

Krzysztof Walkowiak is with Chair of Systems and Computer Networks, Wroclaw University of Technology
(e-mail: krzysztof.walkowiak@pwr.wroc.pl).

limited processing power p_v (e.g. CPUs, FLOPS) that denotes how many uniform jobs can be calculated on v in a particular time.

We assume that the computational project to be processed (calculated) in the public-resource computing system is divided into uniform tasks (jobs) having the same processing power requirement (e.g. CPUs, FLOPS). Each task is represented in the overlay system as a block, which is the data that is generated due to the processing of the particular task. Since jobs are uniform, every block is of the same size and is denoted using index $b = 1,2,\ldots,B$. Note, that for the sake of simplicity, in the remainder of the paper we use the term block in two senses: computational job and data block.

In this paper we concentrate on two kinds of flows: unicast and P2P. There are numerous papers on network optimization that use unicast flows. Modeling of flows in Peer-to-Peer systems is much more difficult that in the case of unicast flows. One of the most challenging problems encountered in modeling of P2P is the time scale. As in [8], [10], [13], [20-22], [24] we propose to divide the time scale of the system into time slots of the same length, that can be interpreted also as subsequent iterations of the systems. We use index $t = 1,2,\ldots,T$ to denote subsequent time slots. In each iteration t, vertices may transfer blocks between them. After each iteration the information on blocks' availability is updated. Using the P2P approach, each block b may be uploaded in iteration t only from nodes, which posses that block at the start of iteration t.

Each block (task) must be assigned to exactly one vertex for processing. We use the decision binary variable x_{bv} to denote the assignment (scheduling) of block b to vertex v for processing. The second decision variable – y_{bwvt} – is associated with blocks' transfer and equals 1 if block b is transferred from node w to node v in iteration t; 0 otherwise. Note that both variables are coupled – scheduling of blocks influences the transfer process. The computational project is collaborative – each peer of the public-resource computing system (represented by the vertex) wants to receive the whole output of processing. For the sake of fairness of the system, we assume that each vertex participating in the system must be assigned with at least one block (job) for processing.

To enable rational comparison of unicast against P2P, in the unicast model we must use concepts and assumptions developed in the context of the P2P approach. Therefore, the same kind of modeling of the time scale is applied in the unicast model. The main difference in unicast model – comparing against P2P – is that the block b can be downloaded only from the node that computed that block, i.e. node v for which $x_{bv} = 1$. Fig. 1 and 2 show the unicast model and the P2P model, respectively.

We assume that input data of each computational task is delivered prior to initiation of the computing system. Consequently, we do not model transmitting of the input data for processing. So, the time scale of the system starts when all source blocks are calculated on nodes. This assumption follows from the fact that usually source data is offloaded from one network node. If we assume that the size of input and output data is the same, then to transmit input blocks we need at most B (number of all blocks) transfers in the overlay network, because each block must be delivered to exactly one vertex. To transmit the output data to all participants we need $B(V-1)$ transfers, where V is the number of all vertices. From this simple example we can see that if input and output data is of comparable size, much more network traffic is issued in the output data delivery. Moreover, the cost of the input data delivery is included in the cost of processing block b on node v. However, models presented below can be easily modified to incorporate also source data delivery. For the sake of simplicity, we assume that the download and upload rates of vertices are expressed in blocks per time slot – but can simply change the model to use b/s.

The cost function denoting the cost of the whole system includes two elements: processing cost of block b in vertex v denoted as c_v and the cost of transfer from source vertex w to

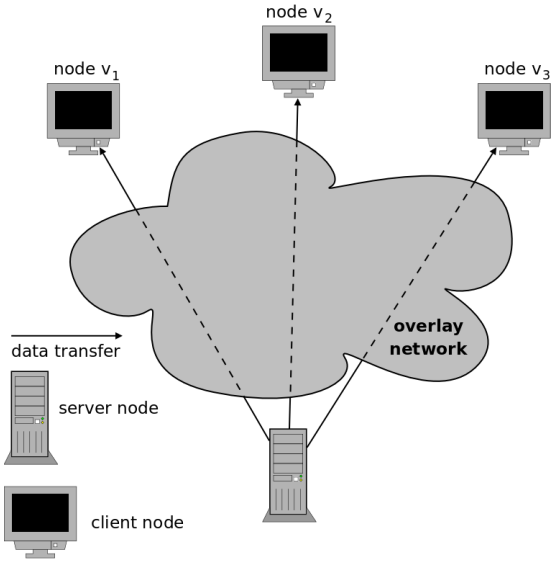

Fig. 1. Unicast flows for the output data distribution in public-resource computing systems.

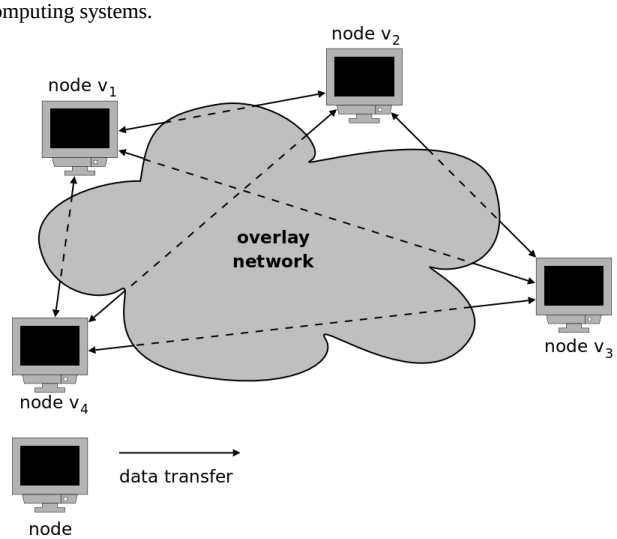

Fig. 2. P2P flows for the output data distribution in public-resource computing systems.

destination vertex v denoted as k_{wv}. The processing cost denotes all aspects of IT infrastructure (energy, maintenance, hardware amortization etc.). Various issues of grid economics can be found in [14]. The second part of the objective function is associated with the transmission cost k_{wv} between vertices w and v. Constant k_{wv} can be interpreted in several ways, e.g. economical, network delay, number of hops, RTT. For a good survey on participating costs in a P2P network refer to [5].

First, we present the problem using unicast flows. We use the notation as in [15].

indices

$b = 1,2,\ldots,B$ blocks (jobs) to be processed (computed) and transferred

$t = 1,2,\ldots,T$ time slots (iterations)

$v,w = 1,2,\ldots,V$ vertices (peers, overlay network nodes)

constants

c_v cost of block processing in node v

k_{wv} cost of block transfer from node w to node v

p_v maximum processing rate of node v

d_v maximum download rate of node v

u_v maximum upload rate of node v

M large number

variables

x_{bv} = 1 if block with index b is processed in node v; 0 otherwise (binary)

y_{bwvt} = 1 if block b is transferred to node v from w in iteration t; 0 otherwise (binary)

objective

$$\text{minimize} \quad F = \sum_b \sum_v x_{bv} c_v + \sum_b \sum_v \sum_w \sum_t y_{bwvt} k_{wv} \tag{1}$$

subject to

$$\sum_b x_{bv} \geq 1 \quad v = 1,2,\ldots,V \tag{2}$$

$$\sum_v x_{bv} = 1 \quad b = 1,2,\ldots,V \tag{3}$$

$$\sum_b x_{bv} \leq p_v \quad v = 1,2,\ldots,V \tag{4}$$

$$x_{bv} + \sum_w \sum_t y_{bwvt} = 1 \quad b = 1,2,\ldots,B \;\; v = 1,2,\ldots,V \tag{5}$$

$$\sum_b \sum_v y_{bwvt} \leq u_w \quad w = 1,2,\ldots,V \;\; t = 1,2,\ldots,T \tag{6}$$

$$\sum_b \sum_w y_{bwvt} \leq d_v \quad v = 1,2,\ldots,V \;\; t = 1,2,\ldots,T \tag{7}$$

$$\sum_v \sum_t y_{bwvt} \leq M x_{bw} \quad b = 1,2,\ldots,B \;\; w = 1,2,\ldots,V \tag{8}$$

The objective is the cost of the computing system including both: processing cost and transmission const. Constraint (2) guarantees that each vertex must process at least one block. (3) assures that each block is assigned to only one vertex. Constraint (4) is the limit on processing power. To meet the requirement that each vertex (peer) must receive all blocks we introduce condition (5). Notice that block b can be assigned to node v for processing ($x_{bv} = 1$) or block b is transferred to node v in one of iterations ($y_{bwvt} = 1$). (6) and (7) are upload and download capacity constraints, respectively. Since we consider only unicast flows, the blocks can be downloaded only from the nodes that calculated these blocks, i.e. $x_{bv} = 1$. Therefore, we use condition (8) to denote the constraint.

In the case of P2P flows, the model is the same as (1)-(8) except condition (8) which is substituted by the following constraint

$$\sum_v y_{bwvt} \leq M(x_{bw} + \sum_{i<t} \sum_s y_{bswi}) \quad b = 1,2,\ldots,B \;\; w = 1,2,\ldots,V \;\; t = 1,2,\ldots,T \tag{9}$$

Note that (9) is specific for P2P systems and guarantees that block b can be sent from peer w to peer v only if w keeps block b in time slot t.

Both unicast and P2P models are Integer Programming problems with binary variables. To solve them in optimal way we can use exact methods like branch-and-bound or branch-and-cut algorithms. However, these methods can provide results only for relatively small systems (in terms of the number of nodes, blocks and iterations). Thus, to obtain results for larger systems we must use some heuristics.

III. Random Algorithms

In this section we will present random algorithms for the models presented above. The motivation to use random strategies in optimization of public resource computing systems comes from the fact that many distributed systems based on the P2P approach applies random strategies in some extent. The most famous example is the BitTorrent system [6].

First we describe the algorithm used for the case of unicast flows. The proposed algorithm – called Unicast Random Algorithm (URA) – uses randomness in the process of allocation of blocks to nodes. The random allocation is performed by UR1 sub-algorithm, which works as follows. At the beginning the UR1 sub-algorithm allocates one block to each node. Remaining blocks are allocated to random nodes with regard of computation limit (p_v) and other model constraints. Random trials are limited to V^2B^2 attempts, also attempt which does not result in allocation (e.g. randomized node has already p_v blocks allocated) counts to the V^2B^2 limit. If V^2B^2 limit is reached, UR1 sub-algorithm quits, no matter if allocation is completed. When allocation performed by UR1 is completed, blocks are computed and distributed among all nodes using the heuristic sub-algorithm called UH2, which is described below. At the initial step, two lists are created: L_v containing all nodes, and L_b containing all blocks. Let's introduce two variables: f_v – indicator of element in L_v list, and f_b – indicator of element in L_b list. These two indicators may be increased – then they point to the next element on the list, but when last element of the list is reached – increasing of the indicator causes it to be set to point to first element. UH2 starts distribution in iteration $t = 1$ and sets lists indicators f_v and f_b to point to the first element on each list. Then it performs the following procedure for each subsequent node from L_v: if node v pointed by f_v is not able to do download (node v made more downloads in iteration t than $d_v - 1$), then increase f_v; otherwise select block to download. The block selection procedure is as follows: if block b indicated by f_b in L_b is not present on v, and node w that computed block b is able to do upload (due to u_w limit) – then selection is successful. In that case UH2 sends block b to node v. If mentioned conditions are not satisfied – f_b is increased by 1 to point to next block on the list, and conditions are checked for new block. Blocks on the list are

checked until block to transfer is found, or until all blocks are examined for the same node v. Then f_v is set to point the next node in list L_v and above procedures are repeated. Indicator f_v is increased until every node was examined by above procedures and there was no transfer made during last V checks. Then, if all nodes have complete set of blocks (5) UH2 exits, otherwise steps to next iteration: $t = t + 1$. If $t > T$ UH2 exits, otherwise sets f_v and f_b to first positions and starts node and block selection procedures again as described above.

As we have UR1 and UH2 defined, we can now simply describe URA algorithm:

Algorithm URA
Step 1. Execute UR1 to assign source blocks to nodes.
Step 2. Perform blocks' computation.
Step 3. Execute UH2 to distribute result blocks to nodes.

Next we will focus on the P2P case. Again the optimization process is divided into two parts: allocation of blocks and distribution of blocks. For the first part we use either random allocation sub-algorithm UR1 (the same as in the case of unicast flows) or an heuristic approach named PH1. The idea of PH1 is as follows.

Blocks are assigned to nodes regarding constraints (2)-(7), (9) according to a special metric defined for each node. First, PH1 allocates a_v nodes to each node v, basing on the following formula

$$a_v = \begin{cases} B - d_v T & \text{if} \quad B - d_v T > 0 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

The idea behind formula (10) is as follows. Recall that (7) indicates that each node v during all T iterations can maximally download d_vT blocks. Constraint (5) denotes that each vertex v must download all blocks that are not allocated to v for processing. Thus, if $d_vT < B$ the number of blocks allocated to v must be $(B - d_vT)$. Otherwise, if $d_vT \geq B$ node v is assigned with one block, which follows from (3).

If $\sum_v a_v < B$, (there are some blocks, which are not allocated for processing), PH1 performs the second phase of allocation. For each node, the metric is computed taking into account both the cost of computation and the cost of distribution. A special coefficient m is used to adjust the importance of blocks' computation cost as part of total processing cost. Three values of m are used: $m_1 = 1$, $m_2 = \left\lceil \frac{B}{V} \right\rceil$, $m_3 = \left\lceil \frac{B}{2} \right\rceil$. The value of m is set to particular value set m_1, m_2, m_3 for which the total cost (1) was the smallest. Blocks are allocated subsequently to the most attractive nodes (regarding limits on processing rate (4)) unless all blocks are allocated. Then blocks are computed, what produces result blocks ready for distribution.

The process of blocks' distribution also can be done either at random way (sub-algorithm UR2) or using an heuristic procedure (sub-algorithm PH2). The sub-algorithm R2 is based on randomizing all parameters of transfer that is to be performed. There are maximum V^2B^2 random attempts allowed in each iteration. One attempt consists of random selection of: source node v, target node w (different than source node) and block b. Then the following checks are made: if source node v is able to send block (upload limit is not exceeded), target node w is able to receive block (download limit is not exceeded), source node v has block b available to send and target node w does not have block b. If all checks are satisfied, then transfer between nodes v and w is set ($y_{bwvt} = 1$). If the random attempt results in transfer that is acceptable in scope of the above checks, but would not be correct regarding any of model constraint, transfer is not set. After $\sum_v u_v$ transfers or V^2B^2 random attempts performed in a iteration, PR2 proceeds to the next iteration (if current iteration is not the last one) or quits (if current iteration is the last one). The sub-algorithm PR2 quits also when all nodes received all blocks.

The second sub-algorithm PH2 is defined in the following way. Distribution of blocks is the process performed in T iterations to saturate the network. Let q denote the maximum number of allowed transfers in one iteration. In the beginning, the PH2 sub-algorithm creates the list of network connections, sorted by cost ascending. Each element of the list contains the source node, the destination node, and the elementary cost of transfer between them. In each iteration, the following steps are made. For top-most positions of list (the cheapest cost), PH2 checks if there are blocks available to send between nodes assigned with this position, and if such transfer is possible (regarding download and upload limits). If these conditions are satisfied, this transfer is performed. PH2 analyses top-most elements of the list. If the transfer for a particular list element is not possible, then next, more expensive element from the list is considered. The iteration is finished, either if number of transfers equals q or $\sum_v u_v$, or if there is no element on the list, for which transfer would be possible to make. The q limit is not used in last iteration $t = T$. In this iteration, when all transfers originating from connection list are performed, PH2 checks if the network is saturated. If it is not, the sub-algorithm tries to transfer missing blocks.

Now we can define the three random algorithms proposed for the P2P flows.

Algorithm PRA
Step 1. Execute UR1 to assign source blocks to nodes.
Step 2. Perform blocks' computation.
Step 3. Execute PH2 to distribute result blocks to nodes.

Algorithm PRB
Step 1. Execute PH1 to assign source blocks to nodes.
Step 2. Perform blocks' computation.
Step 3. Execute PR2 to distribute result blocks to nodes.

Algorithm PRC
Step 1. Execute UR1 to assign source blocks to nodes.
Step 2. Perform blocks' computation.
Step 3. Execute PR2 to distribute result blocks to nodes.

IV. Results

The presented random algorithms were implemented in C++ and extensive experiments were run to evaluate their performance. The major goal was to compare results of

random heuristics against optimal results obtained using CPLEX 11.0 solver [9]. To measure the percentage difference between two results, the following indicator is used

$$D_{\mathrm{ALG2}}^{\mathrm{ALG1}} = \frac{F_{\mathrm{ALG2}} - F_{\mathrm{ALG1}}}{F_{\mathrm{ALG2}}} \cdot 100\% \qquad (11)$$

where F_{ALG1} indicates the value of criterion function returned by algorithm ALG1.

Optimal comparison was made only for 116 small networks whose parameters are shown in Table I, because for larger problem size the CPLEX optimizer is not able to return optimal solution in a reasonable time. Let us denote algorithms that provide optimal results as UOA (for unicast) and POA (for P2P). These algorithms are implemented in CPLEX optimization package which internally uses branch and cut method. Comparison between optimal results and random results are presented as histogram on Fig. 3. For unicast flow (denoted as UOA-URA), most of results ranged between 0-20%, because this kind of flow does not allow much flexibility in the scope of data allocation. Value of *D* indicator ranged between 0-33% with the average value equal to 7,6%. In the case of Peer-to-Peer flows, three random algorithms were examined. For PRA algorithm (random allocation) most of *D* values were in the range 10-30%, there were also several results in the range 0-10%. This gives the conclusion, that random allocation does not have very significant influence on final cost both in unicast and P2P flows. For many networks, the number of blocks could not be much bigger than the number of nodes (because of problem size), what also causes that there were not many blocks available for random allocation. The percentage difference between POA and PRA (denoted on the figure as POA-PRA) was in range 0-45%, the average value was 14%. PRB algorithm (random transfers) had the difference mostly ranged between 10-40%, there were also several results with *D* indicator bigger than 40% and few results in the range 0-10%. Values of the percentage difference for PRA were in the range 1-60% with the average value 26%. Results of PRC algorithm were similar to PRB, but in this case we got most results with *D* difference >40%. Average value of *D* indicator for PRC value was 30% within range 2-62%. We conclude that random transfers have much more influence on the final cost than random allocation.

The next step was to make the comparison between random strategies and other heuristics introduced in [4]. In this case much larger networks were considered – the experiments were made for 20 networks having parameters shown in Table II. Results are presented in Fig. 4. In all cases the algorithm UHA outperformed the algorithm URA, the percentage (denoted as UHA-URA) was in the range 7-12% and the average value was 9%. In the case of P2P flows and PRA-PHA comparison, there were no *D* values in 0-10% range, all values were in the range 10-30%. The average value of the percentage difference was 21% within the range 17-24%. Far more different results appeared for PRB and PRC algorithms. For PRB algorithm all *D* values were in the range precisely 72-78% with the average value equal to 75%. In the case of PRC algorithm, the average percentage difference was 78% within the range 74-79%. For the two largest networks PRC algorithm was not able to return proper result. Experimentation results for random algorithms lead us to the following conclusions. Constructive heuristics proposed in [4] outperforms random strategies. Overall, differences for the random-optimal case were smaller than in the case of random-heuristic differences, because to make the problem feasible in most optimal cases the number of blocks had to be close to number of nodes. Thus, there was smaller share of random choices than for larger networks and heuristic algorithms, where the problem included many blocks and random algorithms made much more random choices. The procedure of random transfers in P2P flows may cause that random algorithm is not able to provide the correct solution.

TABLE I
PARAMETERS VALUES OF NETWORKS USED TO OBTAIN OPTIMAL SOLUTIONS

parameter	range of value
number of nodes	3-8
number of blocks	3-39
number of iterations	3-5

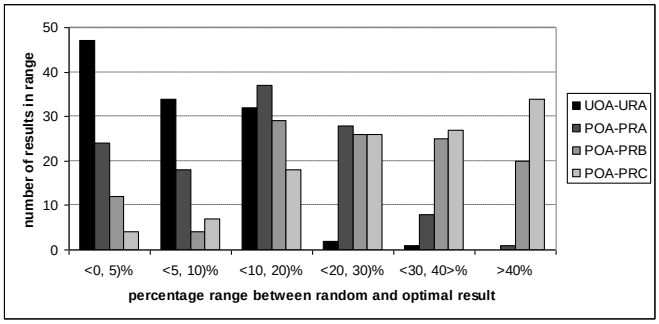

Fig. 3. Comparison between optimal and random results.

V. RELATED WORK

The last section of [17] describes a new concept for content delivery services by linking capabilities of grid computing and peer-to-peer (P2P) computing. The system has the goal to

TABLE II
PARAMETERS VALUES OF NETWORKS USED TO OBTAIN HEURISTIC SOLUTIONS

parameter	range of value
number of nodes	100-200
number of blocks	200-314
number of iterations	15

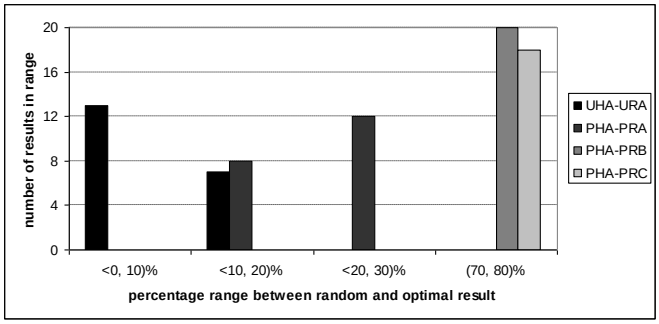

Fig. 4. Comparison between heuristic and random results.

design a secure, reliable, and scalable system for efficient and fast delivery of content. The proposed approach is a combination of nondedicated servers and peers. The IBM Download Grid (IDG) – an IBM internal prototype built based on the proposed approach – is described and discussed.

The authors of [16] present algorithms of parallel rendering with inexpensive commodity components based on multiple PCs connected by network are presented. The idea of k-way replication is applied to distribute a large scene database across components. Experimental results presenting performance of load balancing object assignment algorithms are included and discussed.

An interesting example of a P2P-based file distribution system is BitTorrent protocol [6]. The BitTorrent uses a centralized software called *tracker* that stores information which peers have a particular file. To facilitate the process of downloading, each file is divided into smaller pieces (e.g. 256 KB). A peer that wants to download a file can receive from the tracker a random list of peers that have got the file. A peer that has got the complete file is called *seed*. Then, the downloader requests pieces from all the peers it is connected to. Next, when a peer downloads some pieces, it can upload them to other peers. Since the main objective of the system is effective file sharing, peers are encouraged not only to download but also to upload files.

In [10] the overlay network content distribution problem is considered. All content is organized as set of unit-sized tokens – files can be represented as sets of tokens. The distributed schedule of tokens proceeds as a sequence of timesteps. There is a capacity constraint on each overlay arc, i.e. only a limited number of tokens can be assigned to an arc for each timestep. Two optimization problems are formulated: Fast Overlay Content Distribution (FOCD) and Efficient Overlay Content Distribution (EOCD). The goal of the former problem is to provide a satisfying distribution schedule of minimum number of timesteps. The latter problem aims at minimizing the number of tokens' moves. Both problems are proved to be NP-complete. An Integer Program formulation of EOCD is presented. Various online approximation algorithms for distributed version of overlay content distribution problem are proposed and tested.

The authors of [23] develop several protocols for P2P based file distribution. A centrally scheduled file distribution (CSFD) protocol, to minimize the total elapsed time of a one-sender-multiple-receiver file distribution task is proposed. A discrete-event simulator for the problem is applied to study the performance of CSFD and other approaches (e.g. BitTorrent).

The paper [8] concentrates on the problem how to disseminate a large volume of data to a set of clients in the shortest possible time. A cooperative scenario under a simple bandwidth model is solved in an optimal solution involving communication on a hypercube-like overlay network. Moreover, different randomized algorithms are analyzed. Finally, noncooperative scenarios based on the principle of barter are discussed.

Arthur and Panigrahy show several routing algorithms designed to distribute data blocks on a network with limited diameter and maximum degree [3]. The time scale of the system is divided into steps. A special attention is put on upload policy – a randomized approach is proposed and examined.

The problem we addressed in this work is related to the resource-constrained project scheduling problem (RCPSP). In Chapter 19 of [14] a multi-mode RCSPS is formulated to model the workflow in Grid systems. A detailed description and 0-1 linear programming formulation are presented. Various metaheuristics (e.g. local search, simulated annealing, tabu search and genetic algorithm) are proposed as solution approaches.

In the literature there are many network optimization problems related to unicast flows. For a good survey on these problems see [15] and references therein. Predominant number of these problems assume that network flows are constant in time and are given in bits per second. However, due to dynamics of P2P systems, modeling of flows of P2P systems need other approaches.

In our previous works we proposed and examined a model of flows in P2P systems [20]. In [21] we introduced four optimization models related to the problem of data distribution in public-resource computing system applying the following approaches: unicast, anycast, multicast and P2P. Next, in [4] we presented a heuristic algorithm for the P2P flows optimization in public-resource computing system.

For other issues on P2P systems, networks, Grids and public resource computing refer to [1-8], [10-24].

VI. Concluding Remarks

In this paper we have addressed the problem public-resource computing systems optimization. The objective was to minimize the cost of the public-resource computing system including both the processing cost and the transfer cost. To solve the formulated problem we have applied random strategies following from real P2P systems. The results of the numerical experiments show that random algorithms yield results much worse than optimal solutions for small networks and other heuristics for larger networks. The main conclusion is that the application of the random approach can provide some feasible solutions, but the quality of these results – expressed as the cost function – is not satisfactory.

References

[1] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage," In Proc. of the Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, 2004, pp. 4-10.

[2] N. Andrade, E. Santos-Neto and F. Brasileiro, "Scalable Resource Annotation in Peer-to-Peer Grids," In Proc. Of 8th International Conference on Peer-to-Peer Computing, Aachen, 2008, pp. 231-234.

[3] D. Arthur and R. Paningrahy, „Analyzing BitTorrent and Related Peer-to-Peer Networks," In Proc. of the 17th ACM-SIAM symposium on Discrete algorithm, Miami, 2006, pp. 961-969.

[4] G. Chmaj and K. Walkowiak, "Peer-to-Peer versus Unicast: Two Approaches to Data Transmission in Overlay Public-Resource Computing System," In Proc. of the 4th International Conference on Broadband Communication, Information Technology & Biomedical Application BroadBandCom, 2009.

[5] N. Christin and J. Chuang, "On the Cost of Participating in a Peer-to-Peer Network," Lecture Notes in Computer Science, Vol. 3279, 2004, pp. 22-32.

[6] B. Cohen, "Incentives Build Robustness in BitTorrent," http://www.bittorrent.org/bittorrentecon.pdf, 2003.

[7] I. Foster, A. Iamnitchi, "On Death, Taxes and Convergence of Peer-to-Peer and Grid Computing," Lecture Notes in Computer Science, vol. 2735, 2003, pp. 118-128.
[8] P. Ganesan and M. Seshadri, "On Cooperative Content Distribution and the Price of Barter," In Proc. of the 25th IEEE Intern. Conf. on Distributed Computing Systems, Columbus, 2005, pp. 81-90.
[9] ILOG CPLEX 11.0 User's Manual, France, 2007.
[10] C. Killian M. Vrable, A. Snoeren, A. Vahdat and J. Pasquale, "The Overlay Network Content Distribution Problem," UCSD/CSE Tech. Report CS2005-0824, 2005.
[11] K. Krauter, R. Buyya and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Mangement Systems for Disitributed Computing," Software - Practice and Experience, vol. 32, No. 2, 2002, pp. 135-164.
[12] D. Milojicic and others, "Peer to Peer computing," HP Laboratories Palo Alto, Technical Report HPL-2002-57, 2002.
[13] J. Munidger and R. Weber, "Efficient File Dissemination using Peer-to-Peer Technology," Technical Report 2004--01, Statistical Laboratory Research Reports, 2004.
[14] J. Nabrzyski, J. Schopf and J. Węglarz, (eds.), Grid resource management :state of the art and future trends, Kluwer Academic Publishers: Boston, 2004.
[15] M. Pioro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers, 2004.
[16] R. Samanta, T. Funkhouser and K., Li, "Parallel Rendering with K-Way Replication," In Proc. of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, San Diego, 2001, pp. 75-84.
[17] R. Subramanian and B. Goodman, Peer to Peer Computing: The Evolution Of A Disruptive Technology, Idea Group Publishing, 2005.
[18] R. Steinmetz and K. Wehrle (eds.), Peer-to-Peer Systems and Applications, Lecture Notes in Computer Science, vol. 3485, Springer Verlag, 2005.
[19] I. Taylor, From P2P to Web services and grids : peers in a client/server world, Springer-Verlag, 2005.
[20] K. Walkowiak, "Offline Approach to Modeling and Optimization of Flows in Peer-to-Peer Systems," In Proc. Of the 2nd International Conference on New Technologies, Mobility and Security, Tanger, 2008, pp. 352-356.
[21] K. Walkowiak and G. Chmaj, "Data Distribution in Public-Resource Computing: Modeling and Optimization," Polish Journal of Environmental Studies, vol. 17, no. 2B, 2008, pp. 11-20.
[22] C. Wu and B. Li, "On Meeting P2P Streaming Bandwidth Demand with Limited Supplies," In Proc. of the Fifteenth Annual SPIE/ACM International Conference on Multimedia Computing and Networking, San Jose, 2008.
[23] G. Wu and C., Tzi-cker, "Peer to Peer File Download and Streaming," RPE report, TR-185, 2005.
[24] X. Yang and G. De Veciana, "Service Capacity of Peer to Peer Networks," In Proc. of INFOCOM'04, Hong Kong, 2004, pp. 2242-2252.