8-19-2008

Preliminary Study on Optimization of Data Distribution in Resource Sharing Systems

Grzegorz Chmaj
University of Nevada, Las Vegas, chmajg@unlv.nevada.edu

Krzysztof Walkowiak
Wrocław University of Technology, krzysztof.walkowiak@pwr.wroc.pl

Preliminary Study on Optimization of Data Distribution in Resource Sharing Systems

Grzegorz Chmaj, Krzysztof Walkowiak
Chair of Systems and Computer Networks, Wroclaw University of Technology, Poland
Grzegorz@Chmaj.net, Krzysztof.Walkowiak@pwr.wroc.pl

Abstract

Grid structures are increasingly considered as very convergent with peer-to-peer networks. This paper presents a model of network acting both as grid and peer-to-peer network, used for data computation and distribution. Presented PPLC algorithm is a complete solution for both grid and peer-to-peer points of view. Problem formulation is presented, as well as solution heuristic algorithm and research results.

1. Introduction

Peer-to-peer (p2p) networks are structures containing many nodes having the same privileges and performing the same tasks [2], [4]. The one paradigm of p2p network is that each node has the same role in such network (server and client). However, some peer-to-peer systems (e.g. BitTorrent) use many roles – some nodes may perform additional management tasks. Grid networks are built to achieve processing power which is spread across computation centers. Each grid participant offers computation resources – the computation problem is divided into fragments and each fragment is independently processed. Then results can be combined back into one product [3], [6]. By source data we mean the input data which requires large computation power to be processed, so it cannot be done at one computing unit. Most known grid projects are Seti@home (searching for extra terrestrial intelligence), Folding@home (protein folding and other molecular dynamics). Many components were created to support grid development: APIs (OGSA, OGSI, SAGA, CORBA, DRMAA, GSI) and software implementations (BOINC, Globus Toolkit).

Grids and p2p are converged in many aspects [1], one common architecture could be used (or interpreted) as grid or p2p network [5]. This paper describes joint of grid and peer-to-peer network, used to compute data and distribute results to all participants. Initial ("source") data is divided into blocks and distributed among nodes of network (blocks represent network data units). Network – acting as grid structure – performs computation of blocks. Each node process blocks it has assigned to itself, computation cost may differ among nodes. To become project's participant, such node has to process at least one source data block. Computation process transforms blocks into new form, considered as "result" data. Each node computes blocks autonomously, without interaction with other nodes. Next, result data is distributed to all participating nodes (as all blocks have to be distributed to all nodes) using peer-to-peer mechanisms. We model peer-to-peer transfer as set of iterations which may be considered as time slots. Each of iteration allows performing some actions, with results visible in the next time slot. Both computation of source block and data transfer between two nodes of network introduces a cost which is the objective to be minimized. Whole process has also time limit, determined by iterations.

We present a network model covering all above aspects, suitable for systems performing data processing and result sharing. Both optimal and heuristic solutions with results are presented. This kind of system can be applied for example to geographic maps processing performed by set of collaborators, which need to have all results visible locally. In that case we would have the map structure stored in each processing cluster. Map would be divided into fragments independently processed by various units.

This paper is organized as follows: section 2 introduces nomenclature used in further considerations. Section 3 contains problem formulation. PPLC algorithm is described in section 4, results of experiments are showed in section 5. Conclusions are formulated in section 6.

2. Nomenclature

Following terms will be used to comprehensively describe the problem and its solution: **block** – represents data fragment denoted as *b*, that can be processed on network node and sent between network nodes; **vertex** (network node) – denoted as *v*, *w*, *s* - element of the network, that is able to process data blocks, send them and fetch to/from other nodes; **network –** set of *V* nodes connected with each other. Vertices that belong to one network may share information between each other. **iteration** (time slot) – denoted as *i*. We consider network processing as the set of iterations. In each of iteration, nodes may transfer blocks between them, but information about assignment of blocks to nodes is updated at the beginning of next iteration. More ways of network modeling you can find in [7]; **processing** – block *b* can be computed (processed) on network node. Resulting data replaces original (source) data. Block *b* can be processed only by vertex which has block *b* assigned to; **block to node assignment** – block *b* is assigned to node, when it is stored physically on given node *v* – then, all other nodes may fetch block *b* from node *v*; **transfer** – the process of sending data (blocks) from source node to destination node (then block transferred to node *v* becomes assigned to node *v*)*;* **peer-to-peer network** (abbreviated to p2p network) – network in which all nodes act both as clients and servers [2]; **grid** – set of nodes, considered as one body that is able to perform common tasks; **saturated network –** network is saturated, when every node in the network has all blocks assigned to itself (4).

3. Problem formulation

To mathematically represent the problem we introduce the following notations:

indices

$b = 1,\ldots,B$ blocks to be transferred
$t,i = 1,\ldots,T$ time slots (iterations)
$v,w,s = 1,\ldots,V$ vertices (network nodes)

constants

c_v cost of processing block in node *v*
k_{wv} cost of block transfer from node *w* to node *v*
M large number

variables

x_{bv} =1 if block with index *b* is processed (calculated) in node *v*; 0 otherwise (binary)
y_{bwvt} =1 if block *b* is transferred to node *v* from node *w* in iteration *t*; 0 otherwise (binary)

objective

$$\min: F=\sum_b\sum_v x_{bv}\,c_v+\sum_b\sum_v\sum_w\sum_t y_{bwvt}\,k_{wv} \quad (1)$$

constraints

$$\sum_b x_{bv} \geq 1 \quad v = 1,\ldots,V \quad (2)$$
$$\sum_v x_{bv} = 1 \quad b = 1,\ldots,B \quad (3)$$
$$x_{bv} + \sum_w\sum_t y_{bwvt} = 1 \quad b = 1,\ldots,B \;\; v = 1,\ldots,V \quad (4)$$
$$\sum_v y_{bwvt} \leq M\left(x_{bw} + \sum_{i<t}\sum_s y_{bswi}\right) \quad b = 1,\ldots,B \quad (5)$$
$$w = 1,\ldots,V \;\; t = 1,\ldots,T$$

The objective function (1) is the cost of processing of all blocks (the first term) and the cost of blocks' transfer (the second term). Condition (2) states that each vertex must process at least one block. Constraint (3) assures that each block is assigned to only one network vertex. To meet the requirement that each processed block must be transported to each network node we introduce the condition (4). (5) defines the source node from which block *b* may be transferred. *M* variable should be larger than *VB* to guarantee that block *b* resides on node *w*.

4. PPLC Algorithm

In this section we describe a heuristic algorithm PPLC (*Pre-Post Limited Cost*) that solves the problem (1-5). The PPLC algorithm (see Fig. 1) consists of two phases. The first phase is aimed to assign blocks to vertices for processing, i.e. values of variables x_{bv} are selected. The second part of PPLC is responsible for the P2P-based block transfer, i.e. values of variables y_{bwvt} are found. The overall objective of PPLC is to find a solution that satisfies constraints (2-5) and minimizes the cost function given by (1). Whole processing has to be completed in *T* iterations.

4.1. Assignment of blocks

In this subsection we will present how the PPLC algorithm assigns blocks to nodes for computation. General allocation schema is presented on Fig. 2.

Notice that the block *b* can be assigned to node *v* for processing (variable x_{bv}=1) or block *b* is transferred to node *v* in one of iterations (variable y_{bwvi}=1). At the beginning one block is allocated to every node – to satisfy formula (2) (*V* blocks are allocated). Then, if *V*<*B*, PPLC performs II level allocation. For every node $v=1,\ldots,V$ score is calculated using formula (6), and maximum value among score values is chosen (7). For every node $v=1,\ldots,V$ score gap g_v is computed (8), and all g_v values are put into a *G* array which is then

```
1  allocate blocks to vertices, according to (2) and (3)
2  process blocks' computation
3  compute m value.
4  find sets of nodes having distance ≥ m (CVGs). If found
   more than 1 CVGs, then reserve first iteration for TR0
   and transport blocks between all CVGs (TR0)
8  if f_T=m and f_{T-1}≠m and exist at least one node v which
   transport costs to all other nodes are greater or equal
   m, then reserve last iteration for TR3
9  perform first main (not reserved) iteration: compute
   f_i, perform all possible transfers having cost ≤ f_i
10 perform TR1 phase (see Fig. 3)
11 for all not-reserved iterations i: compute f_i and perform
   all possible transfers having cost less or equal f_i
12 after all transfers in last main iteration perform TR2
   (see Fig. 4)
13 if last iteration is not reserved, perform TR3 phase,
   otherwise go to last iteration and then perform TR3 phase

TR3: for each node v, transfer blocks b not existing on node
v from node w closest by distance to node v
```

Fig. 1 Pseudocode of PPLC algorithm

```
1 allocate one block to each node
2 if all blocks are allocated – end of allocation.
  Otherwise go to step 3.
3 compute score e_v for each node v
4 compute score gap g_v for each node v
5 allocate blocks to nodes having highest g_v unless all
  blocks are allocated
```

Fig. 2 Pseudocode of blocks' allocation

sorted by value descending. Let us notice, that element g of array G is the value of gap (g_v) and we also have the information to which node this particular gap value is assigned. Let *first*(G) return the first element of G. $G=G-\{g\}$ denotes deletion of element g from array G, so then next element becomes the first one.

$$e_v=c_v+\sum_w k_{vw} \tag{6}$$

$$e_{\max} = \max_{v=1,\ldots,V}(e_v) \tag{7}$$

$$g_v = \frac{e_v - e_{\max}}{e_v} \tag{8}$$

Having array of g values, PPLC allocates blocks to nodes using following schema: Let a_{sum} be the number of blocks allocated in II level of allocation, and $a_{sum}=0$. Allocate a_v (9) blocks to node associated with g.

$$a_v = \begin{cases} g\cdot(B-V) \text{ if } (B-V)(1-g)-a_{sum}\geq 0 \\ B-a_{sum} \quad \text{otherwise} \end{cases} \tag{9}$$

While there are still unallocated blocks, set $G=G-\{g\}$, $v=first(G)$ and a_{sum} is increased by a_v and allocation is performed again based on (9). This process is repeated unless all blocks are allocated. When all blocks are assigned to nodes, computation process is performed.

4.2. Transfer of blocks

In this subsection we will explain how the PPLC algorithm distributes the blocks to all vertices. Recall that according to our model (1-5) the P2P approach is applied for distribution of blocks. First, we introduce parameter m defined as follows

$$m_v=\min(k_{vw}) \quad v,w=1,\ldots,V \quad v\neq w \tag{10}$$

$$m=\max(m_v) \quad v=1,\ldots,V \tag{11}$$

Value m (11) is the coefficient used to scale cost limit. It has constant value during processing and is established at the beginning of algorithm, acting as general constant parameter.

PPLC limits the maximal cost in each operation i to settled value f_i (12), which is computed for each of iteration separately.

$$f_i = \left\lceil m\frac{i}{T} \right\rceil \tag{12}$$

High values of m would let blocks to be transferred at higher costs, what is not favorable according to objective (1). Too low value of m could cause such situation, that many blocks are transferred in *TR*3 phase (what implies high costs). Value of m is the maximum cost of acceptable transfer during iterations, except *TR*0 - and *TR*3.

Now we define *CVG* (*closed vertices group*). A *CVG* contains all vertices that are within distance limited by m. More precisely, any two nodes w and v belong to the same *CVG* denoted as Q if $k_{vw}\leq m$. If for any two nodes w and v the following condition holds $k_{vw}>m$, it means that w and v do not belong to the same *CVG*. If (13) and (14) are satisfied – then Q_1 constitutes *CVG*.

$$k_{vw}\leq m \quad v\in Q_1 \quad w\in Q_1 \tag{13}$$

$$k_{vw}>m \quad w\neq v \quad v\in Q_1 \quad w\notin Q_1 \tag{14}$$

The PPLC algorithm finds all *CVGs*. When more than one such group is found then the first iteration is purposed only for blocks' transport between *CVGs*, which is performed in the following way. Firstly, PPLC creates a set K of containing pairs of indices identifying *CVGs*. All possible combinations of *CVG* pairs are put into K. Value of f_1 is computed using the following formula (15)

$$f_1 = \max_{v,w} k_{vw} \quad v,w=1,\ldots,V \tag{15}$$

We assume that the operator *first*(K) returns the first pair of *CVGs* from K. Let (Q_1,Q_2) = *first*(K).

Let D_v denote the set of blocks which are allocated to node v (D_w and D_s – by analogy). PPLC performs transfer from Q_1 to Q_2 (16) and from Q_2 to Q_1 (17).

$$y_{bvwt}=1 \quad v\in Q_1 \quad w\in Q_2 \quad b\in D_v \quad t=1 \tag{16}$$

$$y_{bwvt}=1 \quad w\in Q_2 \quad v\in Q_1 \quad b\in D_w \quad t=1 \tag{17}$$

Destination nodes: w for transfer Q_1 to Q_2, and v for transfer Q_2 to Q_1 should satisfy conditions (18) (for node v) and (19) (for node w).

$$k_{vs}\leq m \quad v\in Q_1 \quad s\in Q_1 \tag{18}$$

$$k_{ws}\leq m \quad w\in Q_2 \quad s\in Q_2 \tag{19}$$

```
1 compute f_{T-1}
2 for every node s=1,…,V do:
3  if transport cost from s is less or equal m to less than
   3 nodes
   or: transport cost from s to other node w is less or
   equal f_{T-1} and transport cost from s to all nodes other
   than w is greater than f_{T-1}
   then: transfer all block available on s to node v
```

Fig. 3 Pseudocode of *TR*1

```
1 for every node v=1,…,V do:
2 if transport cost from v is less or equal m to less than 3
  nodes then find node s which has smallest transfer cost to
  node v
3 for every block b absent on v and absent on s:
4   tcost1=cost of sending b to v in current iteration
5   tcost2=cost of sending b to s in current iteration
6   tcost3=cost of sending b to v in last iteration
7   tcost4=cost of sending b to v in last iteration
8   if tcst3+tcst4>tcst1+tcst2 then transfer b to s
```

Fig. 4 Pseudocode of *TR*2

If these conditions cannot be satisfied due to network topology, nodes v and w are chosen in a way: that for transport Q_1 to Q_2, node v is the one that has the biggest quantity of transfer costs to other nodes belonging to Q_2; Q_2 to Q_1, node w is the one that has most quantity of transfer costs to other nodes belonging to Q_1.

Transfers between *CVGs* are performed for all elements from K array, so transfers between all possible pairs of *CVGs* are done. If there were more than one *CVG* groups found, then first operation is purposed only for *TR*0. At this point, PPLC decides if last iteration should be purposed for transfers that cost more than m (iteration $i=T$ is purposed for *TR*3 phase only). If $f_T=m$ and $f_{T-1}\neq m$ (12) and there exists at least one node v having all transfers costs to other nodes greater or equal m (20), then last iteration is reserved for over m cost transfers. Let us name not-reserved iterations as *main iterations*. The next step is to transfer blocks using all iterations except reserved earlier.

$$k_{vw}\geq m \quad w=1,\ldots,V \quad w\neq v \tag{20}$$

For each iteration i (except iterations reserved for *TR*0 and *TR*3 if such reservation was made), value of f_i is computed using (12). Then, all possible transfers having cost less or equal m are performed (21) – nodes fetch only blocks, which they do not possess yet.

$$y_{bwvi}=1 \text{ for } b=1,\ldots,B \; w,v=1,\ldots,V, \quad w\neq v,\; k_{wv}\leq f_i \text{ and } x_{bv}+\textstyle\sum_{t\leq i}\sum_s y_{bsvt}=0 \tag{21}$$

If this is first main iteration, then *TR*1 transfer is performed, according to schema presented on Fig. 3.

First step of *TR*1 is to compute the cost limit for iteration T-1 (12). PPLC finds the most attractive node v – node having the biggest quantity of transfer costs less or equal m (24).

$$h_{vs}=\begin{cases}1 & if\ k_{vs}\leq m\\ 0 & otherwise\end{cases} \tag{22}$$

$$H_v=\textstyle\sum_s h_{vs} \tag{23}$$

$$v=\max_s H_s \quad s=1,\ldots,V \tag{24}$$

Then, for each node s following checks are made: a) if number of network connections having transfer costs less or equal m from node s is less than 3; b) node s has the following transport costs' structure: there is one network node, to which s has transport cost less or equal m, and transport cost to all other nodes is greater than m. If a) or b) is true, then all blocks existing on s and not existing on node v are transferred from s to v. After *TR*1 phase, PPLC algorithm continues with main iterations. At the end of last main iteration, phase *TR*2 is performed, according to Fig. 4: for each node v, following steps are performed: if number of network connections having transfer costs less or equal m from node v is less than 3, then node s is found. Node s is selected based on criterion: node s is the one that has minimal transfer cost to node v. Then, for every block b that does not exist on v (25) and does not exist on node s (26): if $tcost_3+tcost_4>tcost_1+tcost_2$ (please see (27), (28), (29), (30)) then PPLC transfers block b to node s from node having smallest transfer cost to s.

$$x_{bv}+\textstyle\sum_{t\leq i}\sum_w y_{bwvt}=0 \tag{25}$$

$$x_{bs}+\textstyle\sum_{t\leq i}\sum_w y_{bwst}=0 \tag{26}$$

$$tcost_1=k_{wv} \text{ where } w=\min_{w'} k_{vw'} \text{ and } x_{bw}+\textstyle\sum_{t<i}\sum_{w'} y_{bw'wi}=1 \tag{27}$$

$$tcost_2=k_{ws} \text{ where } w=\min_{w'} k_{sw'} \text{ and } x_{bw}+\textstyle\sum_{t<i}\sum_{w'} y_{bw'wi}=1 \tag{28}$$

$$tcost_3=k_{wv} \text{ where } w=\min_{w'} k_{vw'} \text{ and } x_{bw}+\textstyle\sum_{t\leq i}\sum_{w'} y_{bw'wi}=1 \tag{29}$$

$$tcost_4=k_{ws} \text{ where } w=\min_{w'} k_{sw'} \text{ and } x_{bw}+\textstyle\sum_{t\leq i}\sum_{w'} y_{bw'wi}=1 \tag{30}$$

The last step of PPLC is to transfer all blocks which cause network to be not saturated (*TR*3). This process is performed at the end of last iteration. During this transfer f_i is set basing on (15) and transfers are performed according to (21). At this point, all blocks are possessed by all nodes (4).

5. Experimentation results

Experimental results of PPLC algorithm showed, that it is able to provide solution to the problem. Experimentation system was created using C++ under g++ compiler, and then compiled under Microsoft Visual Studio 2003. The system implements PPLC algorithm and is able to do value overriding, so values of coefficients may be set manually (this was used to experiments showed on Fig. 3). Network data generator

was also created – it produces network with parameters enclosed in given ranges.

For researches about PPLC algorithm itself, there were 20 networks generated – using following parameters given to network data generator: quantity of nodes: 50-90, quantity of blocks: 1-10 per node, transfer costs: 1-50. *TR*0 and *TR*3 affect on total transfer costs in networks having *CVGs* lowering total transport cost. For experiment where *TR*0 is not performed, not all blocks are transferred (transfers occurs only inside *CVG*s). Condition (4) is not satisfied then, so result where both *TR*0 and *TR*3 are not used cannot be accepted. For networks with *CVGs* when using only *TR*3, cost is highest – blocks between *CVGs* are transferred only during *TR*3 phase (what implies high costs). The relation between cost and *m* coefficient is presented on Fig. 5. To obtain this relation, experimental system was modified: value of *m* was not set using (11b), but it was set to fixed value. Experiments for given network were made for each value of *m* in range [1, 20]. Fig. 5 shows, that value of *m* should be low, to limit the cost in adequate way. Too low values cause cost to increase very quickly – more transfer is performed in *TR*3 on higher (>*m*) costs. Cost values have one minimum at $m=m_0$, for values higher than m_0, cost of transfer related to *m* is growing up. Lowering *m* values causes more blocks to be transferred in *TR*3, what is not efficient. Low values of *m* also result in presence of *CVGs* – and some transfer costs emerge in a phase of *TR*3. When values of *m* become higher, more blocks are transferred in main iterations – obeying f_i costs. Consequently, *TR*3 and *TR*0 perform less transfers cost. PPLC algorithm was also experimented in the scope of *m* designation. For 20 experimented networks, $m=m_0$ was designated in 16 cases.

PPLC algorithm results were compared with optimal solution generated by CPLEX solver. Time limit for CPLEX computation process was set to 3600 seconds, in many cases CPLEX did not find optimal solution during this time, and returned feasible solution. On Fig. 6 we can observe the relation of solution values for both PPLC and CPLEX, and Fig. 7 shows parameters of networks used for experiments. All CPLEX results on this graph are optimal, and PPLC result values are 1%-20% close to optimal solution. The same experiments were made for larger networks, and results are presented on Fig. 8 and Fig. 9. In this case, CPLEX delivered feasible solution, as it was not able to produce optimal solution in 3600 seconds time. For smaller networks, CPLEX feasible solutions are still lower than PPLC, but for larger networks PPLC produces better solutions.

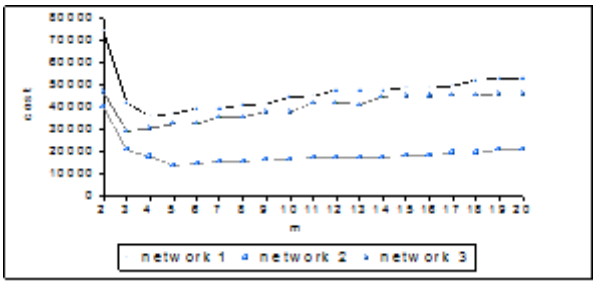

Fig. 5 Cost of transfer according to *m*

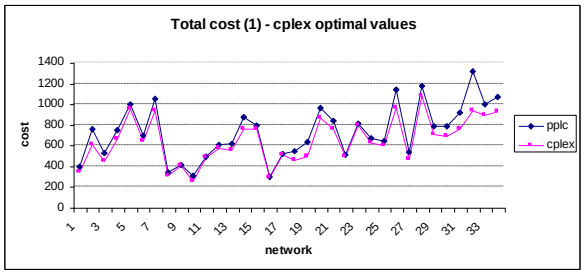

Fig. 6 Total cost of processing

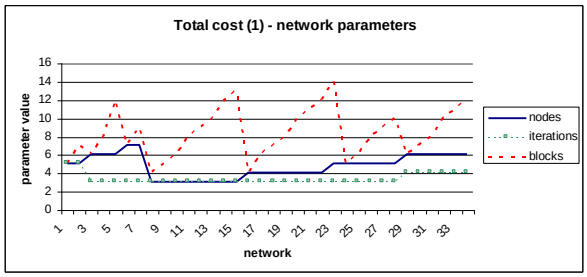

Fig. 7 Network parameters for Fig. 6

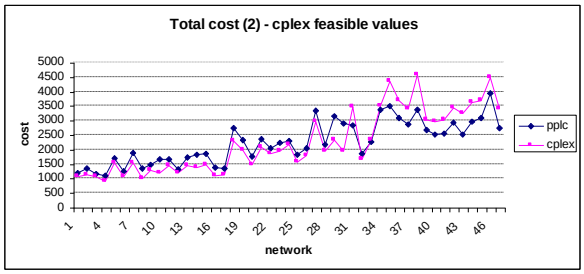

Fig. 8 Total cost of processing

Fig. 10 shows results for one network (8 nodes, 5 iterations), where number of blocks was increased. We observe that for 160 blocks processed, PPLC and CPLEX solutions are still close, but for larger number of blocks PPLC algorithm results in much better solutions. Experiments showed that CPLEX is able to deliver optimal solutions in 3600 seconds only for

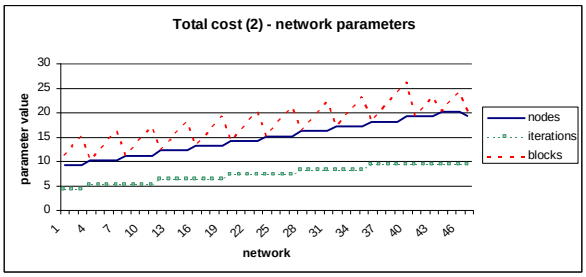

Fig. 9 Network parameters for Fig. 8

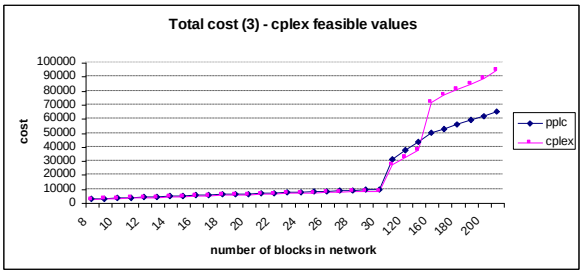

Fig. 10 Total cost of processing / blocks

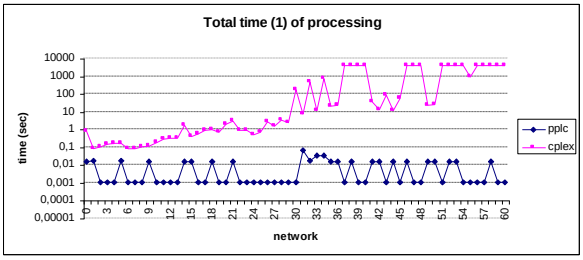

Fig. 11 Total time of processing

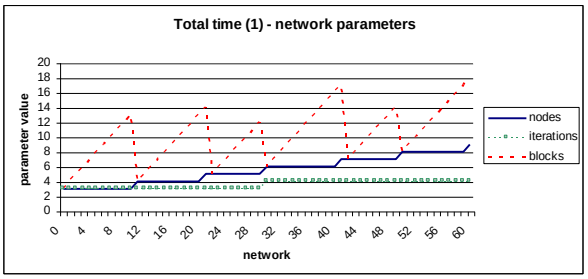

Fig. 12 Network parameters for Fig. 11

small networks. For bigger structures, time of experiment grows significantly, and quickly reaches 3600 seconds limit. PPLC delivers solutions in the time of single seconds, for all researched networks. Time relations are shown on Fig. 11 and Fig. 12.

6. Conclusion

Presented approach of P2P-based data distribution in grid systems provides many processing possibilities, but requires efficient algorithm to manage transfers and blocks' distribution. The joint of grid and peer-to-peer point of view has been proposed as one system, serving as grid or p2p at one time. Algorithm PPLC – proposed in this paper – achieves satisfying efficiency results of transfer costs and planning, suitable both for grid and p2p. Future work is to extend network model with real limitations, such as computation capacity and network transfers limitations.

7. References

[1] I. Foster, A. Iamnitchi, "On Death, Taxes and Convergence of Peer-to-Peer and Grid Computing", *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 978-3-540-40724-9, pp. 118-128

[2] D. Milojicic and others, "Peer to Peer computing", *HP Laboratories Palo Alto*, HPL-2002-57, 2002.

[3] K. Krauter, R. Buyya, M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", *Software - Practice and Experience*, 2002.

[4] C. Wang, B. Li, "Peer to peer overlay networks: A Survey", Department of Computer Science, 2003.

[5] G. Antoniu, M. Jan, D. Noblet, "A practical example of convergence os P2P and grid computing: an evaluation of JXTA's communication performance on grid networking infrastructures", *Rapport de recherche n°5718*, Systèmes communicants Projet Paris, September 2005.

[6] F. Travostino, J. Mambretti, G. Karmous-Edwards, "Grid Networks Enabling grids with advanced communication technology", *Wiley*, ISBN 0-470-01748-1, 2006.

[7] M. Pioro, D. Medhi, "Routing, Flow, and Capacity Design in Communication and Computer Networks", *Morgan Kaufmann Publishers*, ISBN 0-12-557189-5, 2004