A new heuristic for broadcasting in clusters of clusters

Hazem Fkaier, Christophe Cérin, Luiz Angelo Steffenel, Mohamed Jemni

A new heuristic for broadcasting in cluster of clusters

Hazem Fkaier[1,2], Christophe Cérin[2], Luiz Angelo Steffenel[3], Mohamed Jemni[1]

[1]Unité de recherche UTIC / ESSTT
5, Av. Taha Hussein, B.P. 56, Bab Mnara, Tunis, Tunisia
hazem.fkaier@esstt.rnu.tn mohamed.jemni@fst.rnu.tn
[2]LIPN, UMR 7030, CNRS, Université Paris-Nord
99, avenue J.B Clément, 93430 Villetaneuse, France
christophe.cerin@lipn.univ-paris13.fr
[3] Université de Reims Champagne-Ardenne
CReSTIC - Équipe SysCom
Département de Mathématiques et Informatique - Bâtiment 3
"Moulin de la Housse" - BP 1039
51687 REIMS CEDEX 2 – France
Luiz-Angelo.Steffenel@univ-reims.fr

Abstract. This paper deals with the problem of broadcasting for cluster of clusters. The construction of partial minimum spanning trees being NP-complete, several heuristic algorithms have been already formulated. Many of these heuristics (like the heuristic of Kruskal) use the shortest path to connect the components of the tree. They are not relevant in case of forwarding or overlapping communication during a step of the algorithm. In this paper, we study a new heuristic for the minimum broadcasting tree and we evaluate it through simulations with different communication parameters and also through real experimentation over the Grid'5000 testbed.
Keywords: scheduling in grids, resource management, collective primitives for communication in grids.

1 Introduction

It is well known that communication affects significantly the performance of applications deployed on large-scale architectures. Large-scale platforms are characterized by a collection of a great number of computing resources that are geographically distributed over sites from institutions and connected with a wide heterogeneous dedicated network. Since data sizes in Grid applications may be large as well as the number of nodes, the collective communication inherent to the applications is a critical bottleneck.

In this paper we study the problem of broadcasting, i.e., sending a message from one node to all the others in such environments. Collective communications are central elements in real life distributed applications, and most improvements to the broadcast problem (also known as *one-to-many* communication pattern) are also valid for other collective communication patterns.

Contrarily to heterogeneous clusters (see Section 3.1), optimal broadcast trees cannot be computed in advance and must therefore be determined accordingly to each network characteristics, a problem that is known to be NP-complete. Several approached have been proposed to approximate the best way to broadcast a message in a cluster and in a cluster of cluster [1, 2, 10, 3], mostly by decomposing the network in a two-layered structure: inter-cluster (heterogeneous communications) and intra-cluster (homogeneous communications). These works focus on different aspects of the network heterogeneity and therefore behave accordingly to specific situations. In our work we tried to encompass most of these situations, proposing a new heuristic that combines the advantages of the previous heuristics and adaptive techniques to reach the best performance in every situation.

We shall emphasize the fact that optimal broadcast tree is fundamentally different from the minimal spanning tree (MST) problem. In the optimal broadcast problem the issue is to minimize the time to reach the last node that minimizes the longest path in the tree. In the MST, the issue is to minimize the whole 'weight' of the tree. The two constructions may lead to very different trees.

The paper is organized as follows. In section 2, we define the problem of building a broadcast tree and the difference with building a spanning tree. In section 3, we recall related works dealing with the problem of achieving efficient broadcast operation. In section 4, we propose a new heuristics to fulfill this task. In section 5, we present the results of our simulation and experiments. Section 6 concludes the paper.

2 Broadcast tree versus spanning tree problem

From a theoretical point of view, the problem of broadcasting may find its roots on the construction of partial minimum spanning trees that is a problem in graph theory. A graph depicts the underlying infrastructure: nodes in the graph represent the machines and edges represent the cost of sending messages.

A graph often contains redundancies that there can be multiple paths between two vertices. This redundancy may be desirable, for example to offer alternative routes in the case of breakdown of an edge (road, connection, phone line) in a network. However, we often require the cheapest path that connects the vertices of a given graph. This must be an un-rooted tree, because there is only one path between any two vertices in a tree; if there is a cycle then at least one edge can be removed. The total cost or weight of a tree is the sum of the weights of the edges in the tree. We assume that the weight of every edge is greater than zero. Given a connected, undirected graph $G =< V, E >$, the minimum spanning tree problem is to find a tree $T =< V, E' >$ such that $E' \subseteq E$ and the cost of T is minimal.

Figure 1 is an example of a graph and its minimum spanning tree. Using the well-known *1-port* communication model, the broadcast of a message from vertices 0 to all the other nodes will send messages from vertex 0 to vertex 4, then from vertex 4 to vertex 3, then vertex 4 can redistribute to vertex 1 and

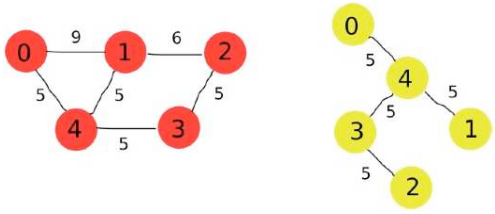

Fig. 1. A graph with 5 vertices (left) and its minimum spanning tree (right)

finally vertex 3 sends the message to vertex 2. Computing a solution of this problem can be done with Prim's or Kruskal's algorithms ***** REFERENCES ***** at a cost time of $\mathcal{O}(|V|^2)$ and $\mathcal{O}(|V|\log|V|))$ respectively.

***** Introduire un exemple où le MST n'est pas le MBT

To better describe the minimum broadcast time [1], we need to follow the formulation stated as follows:

INSTANCE: Graph $G = (V, E)$ and a source node $v_0 \in V$.
SOLUTION: A broadcasting scheme. At time 0 only v_0 contains the message that is to be broadcasted to every vertex. At each time step any vertex that has received the message is allowed to communicate the message to at most one of its neighbors.
MEASURE: The broadcast time, i.e., the time when all vertices have received the message.

This has been termed the minimum broadcast time problem under the telephone model and is known to be NP-complete. However, the minimum broadcast time in a graph has a solution that is reachable within $O(\log^2|V|/\log\log|V|)$ [12]. Instead of implementing the result, we go along heuristics. We have no idea, to our knowledge, if the theoretical result has been implemented for large-scale grid systems.

2.1 Description of the Environment

Heterogeneity Model: We assume a generic platform composed by heterogeneous clusters. The platform studied enjoys heterogeneity along three orthogonal axes: ***(i)*** the processors that populate the clusters may differ in computational powers, even within the same cluster; ***(ii)*** the clusters are organized hierarchically and are interconnected via a hierarchy of networks of possibly differing latencies and bandwidths. At the level of physical clusters, the interconnection networks are assumed to be heterogeneous; ***(iii)*** the clusters at each level of the hierarchy may differ in sizes.

Communication Model: We assume that the network is fully connected. The links between pairs of processes are bidirectional, and each process can

[1] See: http://www.nada.kth.se/~viggo/wwwcompendium/node127.html#5671

transmit data on at most one link and receive data on at most one link at any given time. This model is well known in the literature as *1-port full-duplex*.

Transmission Model: The literature contains several parallel communication models that differ on the computational and network assumptions, such as latency, heterogeneity, network contention, etc. In this work we adopted the *parameterized LogP* model (*pLogP*) [8]. Our choice on the *pLogP* model comes from the fact that we can experience different transmission rates according to the message size, as a consequence of transport protocols and hardware policies. Hence, all along this paper we shall use $\boldsymbol{L}$ as the communication latency between two nodes, $\boldsymbol{P}$ as the number of nodes and $\boldsymbol{g(m)}$ for the gap of a message of size m. The gap of a message m represents the time required to transmit a message through the network (excluding the latency), which is inversely proportional to the bandwidth of the link. In the case of inter-cluster communications, $L_{i,j}$ and $g_{i,j}(m)$ designate the specific latency and gap between two nodes i and j.

3 Related work: heuristics for broadcasting

One of the earliest papers, to our knowledge, dealing with communication in a wide sense and in a grid is certainly the paper of Ian foster [5]. In this paper authors investigated the need for a communication library in the frame of parallel programming on a distributed multi-site architecture where heterogeneity of network is an intrinsic property. Authors proposed a version of MPICH dedicated to grids and they called it MPICH-G. This version is built upon MPICH and Globus.

Other studies with implementations have been elaborated. Most of them consider the MPI communication library or one of its variant such as MPICH-G2, PACX-MPI, GridMPI. To deal specifically with broadcasting, we need to refer to the algorithm of Van de Gejin [2] that consists in a recursive scatter (a special broadcast) phase that puts a fragment of the message to distribute on each node, then a phase of recursive all-gather (a concatenation of messages that are stored on each node) occurs to each message fragment.

We can also cite the works achieved in " The Grid Technology Research Center, AIST " by M. Matsuda & al. In [10, 11], Masuda and al. studied collective operations on multi-site architectures with MPI library. The paper [9] of Matsuka & al. considers especially the broadcast operation in the case where nodes have 2 lane NIC's. The main contribution of the paper is the way of splitting the message to broadcast: it is broken into two pieces then they are broadcasted independently following two binary trees; then, nodes of the two trees exchange the two parts of the message.

Let us review now more in details some of know algorithms for both homogeneous and heterogeneous environments.

3.1 Basic approaches for broadcasting in homogeneous environments

We review hereby, some well-known algorithms for broadcasting a message in one cluster. In the remainder, we assume that we can reach any node in an equal time. Then, there is no need to choose a specific node since network is homogeneous and all nodes are symmetric. We also need definitions related to the network. Let L be the latency and g the gap as introduced in the LogP [4] and pLogP models [8].

Linear broadcast: It is the simplest basic algorithm. It consists in a one level (flat) tree where the root is the node detaining initially the message and all the other nodes are leaves. Then the root sends sequentially the message to the leaves. The broadcast time is proportional to number of nodes.

Binomial tree broadcast: The optimal approach for homogeneous clusters (without considering message partitioning) considers that all nodes having received the message at a given time may participate in the following broadcast steps. This algorithm proceeds basically as follows: in the first step, P_0 sends to P_1. In the next step, both P_0 and P_1 send to two other nodes (P_2 and P_3). Then all nodes possessing the message send it other processors and so on. The cost of broadcasting according to such tree is $(l + M/b)logp$, since the number of nodes containing the message doubles at each step.

Following another approach, Van de Geijn [2] proposes an algorithm that segments messages and acts in two steps: a recursive scatter in a binomial tree fashion, followed by collecting of segments using recursive doubling until the whole message becomes available on all nodes. The complexity of this algorithm is proportional to the size of the message ($O(M)$) since the message is split into two pieces, following two binary trees. We do not detail this algorithm because we do not use it in the remainder of the paper.

3.2 Advanced approaches for heterogeneous clusters

We assume now that the network is heterogeneous in one cluster. To calculate the broadcast tree we must determine at each step which nodes will participate in the communication. Therefore, a set A represents the set of nodes already having the message at a given time, while the set B represents the set of nodes that have not yet received the message.

Early Completion Edge First - ECEF: In the ECEF heuristic (Bhat & al. [3]) a couple of nodes P_i in set A and P_j in set B is chosen in such a way that P_j becomes ready to send the message as early as possible. This time is computed by:

$$RT_i + g_{ij}(m) + L_{ij}$$

where RT_i is the ready time of P_i, $g_{ij}(m)$ is the latency gap between P_i and P_j and L_{ij} is communication cost between P_i and P_j. Note that this heuristic aims at increasing the number of nodes in set A as fast as possible.

Early Completion Edge First with look-ahead - ECEF-LA: It is clear that ECEF heuristic allows to increase number of nodes in set A which is yet a good fact. But it also important to choose well the next destination to be itself a good sender in remainder steps.

As an enhancement of the latter heuristic, Bhat & al. [3] propose to estimate the efficiency of each node throughout a function that takes into consideration the speed of forwarding the message to another node of set B. For instance the following function can be considered:

$$F_j = \min_{P_k \in B}(g_{jk}(m) + L_{jk})$$

for $P_k \in$ set B. Then we select in set B the node P_j that minimizes

$$RT_i + g_{ij}(m) + L_{ij} + F_j.$$

From a complexity point of view, ECEF has a running time of $\mathcal{O}(N^2 \log N)$, whereas, due to look-ahead function, ECEF-LA has a running time of $\mathcal{O}(N^4)$.

3.3 Grid aware heuristics

We suppose, now, that we have a cluster of clusters environment. We suppose also that we have a coordinator (proxy) on each cluster. All communications between clusters are performed through these coordinators. Subsequently, global communications are ordered in two levels: inter-cluster and intra-cluster communications. Hence, if we have a message to broadcast in a grid architecture, then it is broadcasted between coordinators, and then each coordinator broadcasts the message locally. In the works elaborated by Steffenel [1], the local communication load is represented by one virtual node that is connected to a specific coordinator. Thus, the local communication load is depicted by

$$L_{kk'} + g_{kk'} = \begin{cases} T_k & \text{if } k^{'} \text{ is associated to node } k \\ \infty & \text{if } k^{'} \text{ is not associated to node } k \end{cases}$$

where P_k is a coordinator and $P_{k'}$ is a virtual node simulating a cluster.

Under this framework, Steffenel proposed three heuristics to broadcast a message in a grid environment. Let us review the heuristics.

ECEF-LAt: The first heuristic proposed in this context is the one that increases the number of nodes in set A in least time. Then, we choose at each time the coordinator that takes the least time to join set A as in ECEF-LA heuristic. It also adopts the look-ahead option. The efficiency function F_j is set to:

$$F_j = \min_{P_k \in B}(g_{jk}(m) + L_{jk} + T_k).$$

ECEF-LAT: The previous heuristic encourages the coordinator with the lowest load at each step, which may implies delays on the most loaded coordinators and subsequently it may increases the broadcast completion time. The opposed heuristic is to choose at each time the coordinator that have the greatest load i.e. the one that maximizes F_j. Hence, F_j is set to:

$$F_j = \max_{P_k \in B}(g_{jk}(m) + L_{jk} + T_k).$$

BottomUp: It is clear that the last heuristic can not be optimal because we choose at each step the least powerful coordinator; so the number of nodes in set A increases very slowly. The last proposed heuristic in [1] combines ECEF-LAt and ECEF-LAT. We need to begin by contacting the most loaded coordinator. We also need to contact it through the 'shortest path'. Then BottomUp heuristic uses a min-max approach to find the 'shortest path' to contact the most loaded coordinator. Hence it selects the coordinator verifying:

$$\max_{P_j \in B}(\min_{P_i \in A}(g_{ij}(m) + L_{ij} + T_j))$$

Another interesting work to mention here is the one achieved by Ching-Hsien Hsu [6]. Hsu proposed to consider the problem according to two patterns: graph pattern and tree pattern. In the graph pattern case, the author proposed the Neareast Neighbor First (*G-NNF*) and the Maximum Degree Neighbor First (*G-MDNF*) algorithms. for the tree pattern case, he proposed *T-NNF*, *T-MDNF*, the Maximum Height Subtree First (*T-MHSF*), the Maximum Subtree First (*T-MSF*) and finally the Maximum Weight Subtree First (*T-MWSF*) algorithms. The definitions of all these algorithms should be very intuitive and we will not detail them here.

To achieve simulations, Hsu elaborated a random graph generator that takes as parameter the number of nodes and the degree of heterogeneity (h) of the desired graph. Through his simulations he observed the makespan, the Amount of Best Schedules (*ABS*) for each heuristic and the speedup of each heuristic with respect to Single Source Shortest Path (*S3P*: the source sends sequentially the message to each node). Among its conclusions, we can cite:

- Network heterogeneity plays an important role as the factor to select suitable scheduling policies in heterogeneous computing environments;
- Graph-based scheduling approaches are better used in homogeneous-like systems.
- Tree-based scheduling approaches are better used in heterogeneous systems.
- G-NNF outperforms other graph-based approaches and has the best schedule in most test cases in low degree heterogeneous environments.
- *T-MWSF* outperforms other tree-based approaches and has the best schedule in most test cases in high degree heterogeneous environments.

4 A new approach for broadcasting in clusters and cluster of clusters

According to previous heuristics, to reduce global broadcast time, three factors impact the performance. First, we need to increase the size of set A with clusters, in the quickest possible way. Having more potential senders give us more chance to perform next communication in a better way, since we have more choices to consider.

The second factor is to give an advantage to communication-efficient clusters when choosing a receiver. As we explained before, it is important to communicate with the cluster that can forward the message, in the next steps, within a short time. This means that we want to augment set A with good senders.

The third factor is to begin by contacting the most loaded clusters, so that we insure the maximum of overlap between intra and inter-cluster broadcast. This strategy is the key of success of BottomUp and ECEF-LAT heuristics since, according to measured parameters, in most cases local broadcast needs more time than inter-cluster communication.

At this point, we would like to mention that the problem of building the optimal broadcast tree is not a multi-criteria problem, since we have only one objective function that is the minimization of the global broadcast time. However, we believe that it is possible to transform each factor to an independent criterion that we have to 'optimize'. Then we can try to solve the whole problem as a multi-criteria problem. We will let this approach for future work.

4.1 The MostCrit heuristic

The previous heuristics try to optimize one of these 'criteria' or to combine two or all of them at each iteration. And the better we consider these factors, the better the result is.

Each heuristic has a function to minimize. This function contains parameters linked to one or two of aforementioned factors. Hence all factors are merged in only one formula to be minimized at each iteration.

Merging all factors in one may give us a compromised solution. But compromise is not always a good solution. To explain this idea let us imagine the situation where we have, at a given iteration, a 'very' loaded cluster. Then we should contact it in priority otherwise it will delay the ending time.

If we combine all factors and look for a compromise, then previous heuristics may lead us to choose another cluster, not the most loaded and subsequently we do not achieve the best performance.

The same reasoning can be applied if we have a very good forwarder cluster or a very fast-to-communicate cluster at a given iteration. The conclusion of this example is to say that considering a single factor at a time can also be very efficient and even more efficient then combining much factors in one.

Following this idea we developed a new heuristic that considers each factor in a separated way. We proceed as follows:

We consider our two sets A and B. At each iteration we choose one sender from set A and one receiver from set B. Then at each iteration we shall decide which factor we need to satisfy. Either (1) to choose the fastest-to-communicate cluster from set B, or (2) best forwarder cluster from set B or (3) the most loaded cluster form set B.

Condition (1) implies to minimize $RT_i + g_{ij}(m) + L_{ij}$ which is to say to apply one iteration of ECEF heuristic.

Condition (2) implies to minimize $RT_i + g_{ij}(m) + L_{ij} + F_j$ which is to say applying one iteration of ECEF-LA heuristic.

And condition (3) implies to choose the most loaded cluster in set B and then to find the best sender in set A which is to say to apply one iteration of BottomUp heuristic, i.e., we apply $\max_{P_j \in B}(\min_{P_i \in A}(g_{ij}(m) + L_{ij} + T_j))$.

The question now is "How to choose the factor to satisfy?"

To answer this question, let us examine what would happen if we do not satisfy a given factor $fact_i$ i.e. we do not choose the best cluster according to this factor:

a- Either the chosen cluster (the optimal cluster according to another factor) behaves well with the factor $fact_i$ then $fact_i$ is not strongly violated. Then we estimate that the chosen cluster and the optimal one according to $fact_i$ behaves in relatively similar way according to $fact_i$.

b- Or the chosen cluster behaves badly with the factor $fact_i$ and then it violates it strongly. Then we estimate that the chosen cluster and the optimal one according to $fact_i$ are relatively different for $fact_i$.

At the end, it is important to choose the cluster that satisfies one factor and behaves well with the other ones, or at least does not violate them strongly.

We propose to compute the set of values associated to each factor as follows:
For factor (1), we compute set $E_1 = \min_{P_i \in A}(RT_i + g_{ij}(m) + L_{ij})/P_j \in B$
For factor (2), we compute set $E_2 = \min_{P_i \in A}(RT_i + g_{ij}(m) + L_{ij} + F_j)/P_j \in B$
For factor (3), we compute set $E_3 = \min_{P_i \in A}(RT_i + g_{ij}(m) + L_{ij} + T_j)/P_j \in B$

Having dispersed value in a given set means that clusters are very different according to the associated factor, then factor may be strongly violated if we do not satisfy it. Whereas, if a set contains close values, then it means that clusters behave in a quite similar way. Subsequently, choosing one cluster or another one will not be decisive.

Finally we choose to satisfy the factor which has the associated set with the most dispersed values i.e. we compute the mean deviation of each set values and we choose to satisfy the factor having the greatest mean deviation.

4.2 Simulation

In our simulations, we rely on works done by Steffenel [1] for the different parameters measured on a real grid environment. He measured values of different communications parameters (L, g, T) over French **Grid'5000**[2] infrastructure. He

[2] For details, refer to `https://www.grid5000.fr`

found out a lowest value and highest value for each parameter. In his simulation, he set randomly the values of L (in μs), g (in ms), T (in ms) in the corresponding interval and then he applied the different heuristics. In our simulation we do the same. The values that we introduced now are the mean of 100 iterations.

parameter	min value	max value
L	1	15
g	100	600
T	200	3000

Table 1. Grid'5000 settings

In the first simulation, we set L, g and T in intervals measured over Grid'5000 see Table 1. As seen in Figure 2, all heuristics give almost the same completion time. Then we cannot evaluate the efficiency or compare them. The second remark we shall note is that our new heuristic (noted MostCrit) gives exactly the same values as BottomUp, which is the best heuristics at present time. We can conclude that both heuristics behave exactly in the same way and it can be obtained only if our new heuristic chooses to apply bottomUp at each iteration. By observing parameters values we can expect this fact since the interval of T_j is much larger then intervals of L_{ij} and g_{ij}. This means that values of T_j will be sparser than values of L_{ij} and g_{ij} and consequently values in E_3 will be sparser than those in E_1 and those in E_2. And finally factor (3) (choosing the most loaded cluster) will be retained.

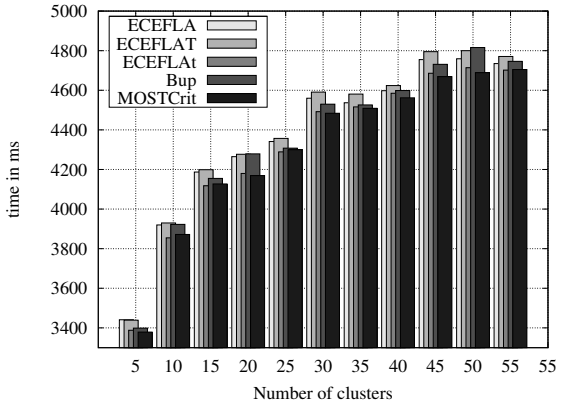

Fig. 2. Broadcasting time vs clusters number with Grid'5000 settings

To evaluate the efficiency of our new heuristics, we propose to achieve simulations with other settings. We changed the ratio of L and g (parameters linked to inter-cluster broadcast) and T (parameter associated to the local broadcast).

In the second simulation, top part of Figure 3, we multiplied L and g by 5 and divided T by 5, see left part of table 2. In the third simulation, bottom part of Figure 3, we multiplied L and g by 10 and divided T by 10, see right part of Table 2.

	Top part of fig. 3		Bottom part of fig. 3	
param	min	max	min	max
L	5	75	10	150
g	500	3000	1000	6000
T	40	600	20	300

Table 2. Grid'5000 settings

Simulation represented in Figure 2 show that bottomUp keeps giving good performances as well as our new heuristic 'MostCrit' even though they do not give exactly the same values. Other heuristics behave worse.

The last conclusion of our simulations is that BottomUp and MostCrit heuristics give evenly good results independently of the ratio of inter-cluster communication performances over intra-cluster communication performances. This point has never been observed before to our knowledge.

5 Experiments

We selected 3 sites/cities in France (Nancy, Rennes and Sophia-Antipolis) and 126 nodes on the Grid'5000 testbed. Grid'5000 is a Grid testbed composed of processors distributed over some clusters in 9 sites in France. RENATER, the French Educational and Research Network provides the inter-cluster connection through a 10Gbits/s "dark fiber" backbone. The different heuristics introduced in this paper were implemented in MPI, using the `MPI_send()` operation as the basic building block.

Figure 4 (left part) introduces the experimental results. We notice that the different curves are quite similar and confirm what we have found with simulations. We notice that the time to broadcast 32MB is about 5s... which is important and are probably due to the 'MPI stack' which slowdown the performance! We notice also some perturbation around 10kB and they should be due to TCP sliding windows. Perturbation around 300KB should be due to a change in MPI policy to send data, as already reported by Steffenel.

Figure 4 (right part) introduces another experimental result, on two clusters located in Rennes and Nancy again but with a new one in Sophia-Antipolis (*Sol* cluster instead of *Azur* cluster) and with new nodes. We also went further in the message sizes. We observe the same phenomena than those observed on Figure

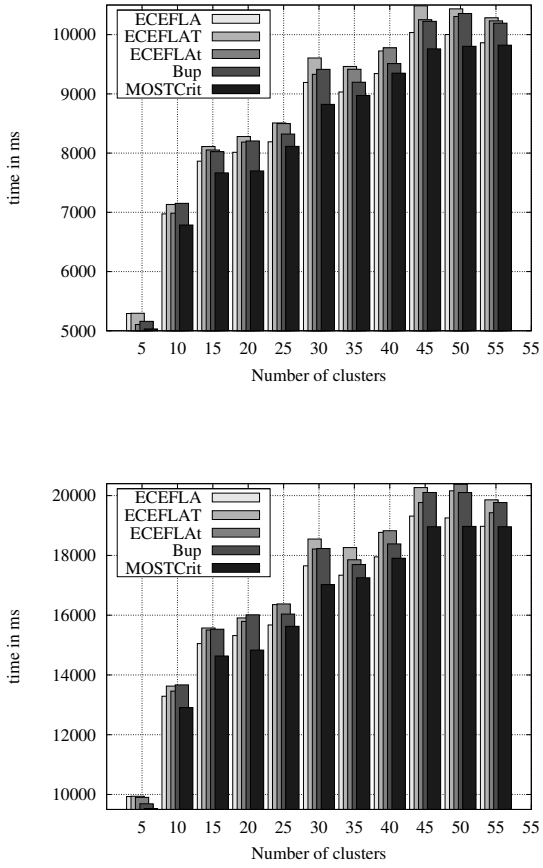

Fig. 3. Broadcasting time vs clusters number with other setting

4. We conclude that what we observe is reproducible and inherent to the tools and algorithms we used.

Figure 5 focus on the larger message sizes and the scales are no more logarithmic. We observe that in this case our MostCrit heuristic behaves well compared to BottomUp for instance and justify our work to optimize the broadcast operation.

6 Conclusions

In this paper, we investigated heuristics to achieve broadcast in a cluster of clusters. We developed a new heuristic inspired from other works in order to combine

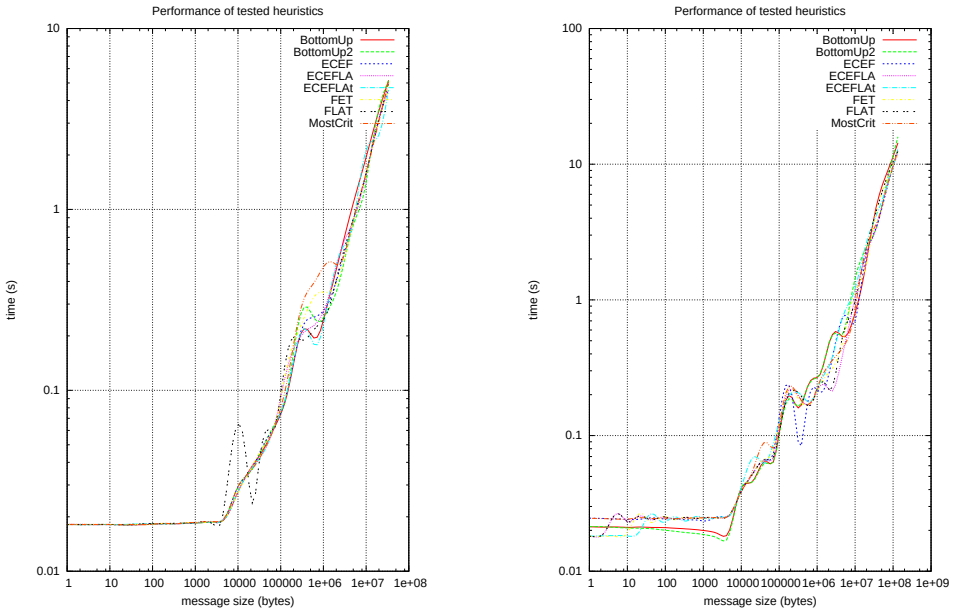

Fig. 4. First experiment on Grid'5000 (left) and second experiment on Grid'5000 (right)

the 'best' of each solution. Our idea is not to combine different elementary factors as it is done with previous published heuristics, but by applying only one factor at once. We proved the efficiency of our approach by simulation and by real experimentations.

We made simulation according to varying communication parameters and hence we cover a wide range of platforms. We also made effective experiments to prove the efficiency of our new heuristic in practice. Experiments have been carried out over Grid'5000 testbed through 126 nodes spread over 3 sites.

In future work, as stated in the paper, we plan to investigate others techniques such as dynamic programming. We plan also to 'translate' the problem to a multi-criteria problem and to solve it with tools from this discipline.

Another important work in experiments could be to increase the input size of the problem and to observe if curves will saturate. Sorting for instance may require exchanging partitions representing more than 95% of the input size; less than 5% of the initial data stay in place. The dual 'problem' is then to concentrate $n-1$ sampling intervals on one site and to realize n such communication steps in parallel.

Concerning the experiments, since the network link is shared among the Grid'5000 sites (but the nodes are dedicated) we also need to model congestion

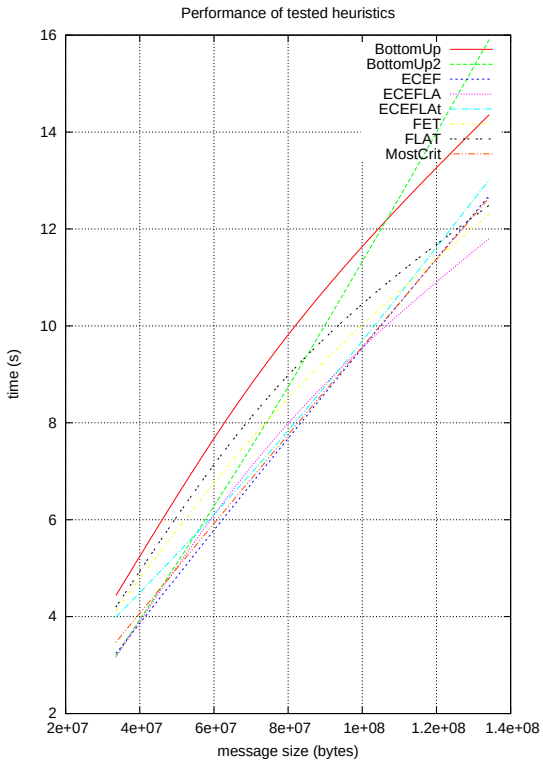

Fig. 5. A focus on messages size from 32MB to 128MB

phenomena or shared bandwidth. From a technical point of view, we could modify on the fly the matrix containing L, g and T parameters, but the problem is to have a realistic model for traffic in clusters of clusters. We are not sure that such model exists yet and it remains a challenging problem. We are currently guessing that our framework will behave better than today with disparate network bandwidth because in this case we can select a strategy among several strategies. What we observe today is due to a full utilization of the available bandwidth among sites.

Finally, instead of injecting perturbations in the network, we could use the Wrekavoc tool[3] able to slowdown the delivering of messages by modifying the TCP stack. Again in this case, we need a realistic model for inter sites communication.

[3] See: http://wrekavoc.gforge.inria.fr/

Acknowledgement We thank deeply the Regional Council of Ile-de-France for its support through the SETCI mobility program (http://www.iledefrance.fr).

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https://www.grid5000.fr).

References

1. L. A. Barchet-Steffenel and Grégory Mounie. Scheduling heuristics for efficient broadcast operations on grid environments. In *IPDPS*, 2006.
2. Mike Barnett, David G. Payne, Robert A. van de Geijn, and Jerrell Watts. Broadcasting on meshes with wormhole routing. *J. Parallel Distrib. Comput.*, 35(2):111–122, 1996.
3. Prashanth B. Bhat, C. S. Raghavendra, and Viktor K. Prasanna. Efficient collective communication in distributed heterogeneous systems. *J. Parallel Distrib. Comput.*, 63(3):251–263, 2003.
4. David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP - a practical model of parallel computing. *Communication of the ACM*, 39(11):78–85, 1996.
5. I. Foster and N. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In *Proceedings of SC'98*. ACM Press, 1998.
6. Ching-Hsien Hsu and Bing-Ru Tsai. Scheduling for atomic broadcast operation in heterogeneous networks with one port model. *The Journal of Supercomputing*, 50(3):269–288, 2009.
7. Natawut Nupairoj Ju-Young L. Park, Hyeong-Ah Choi and L.M. Ni. Construction of optimal multicast trees based on the parameterized communication model. *icpp*, 01:0180, 1996.
8. Thilo Kielmann, Henri Bal, Sergey Gorlatch, Kees Verstoep, and Rutger Hofman. Network performance-aware collective communication for clustered wide area systems. *Parallel Computing*, 27(11):1431–1456, 2001.
9. Tatsuhiro C. Toshio E. Satochi M. High-performance mpi. In *IPDPS*, 2004.
10. Y. Kodama R. Takano M. Matsuda, T. Kudoh and Y. Ishikawa. Efficient mpi collective operations for clusters in long-and-fast networks. In *Cluster2006*, 2006.
11. Motohiko Matsuda, Tomohiro Kudoh, H. Tazuka, and Yutaka Ishikawa. The design and implementation of an asynchronous communication mechanism for the MPI communication model. In *CLUSTER*, pages 13–22. IEEE Computer Society, 2004.
12. R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time. *Symposium on Foundations of Computer Science*, 0:202–213, 1994.
13. Luiz Angelo Steffenel. Modeling network contention effects on alltoall operations. In *Proceedings of the IEEE Conference on Cluster Computing (CLUSTER 2006)*, Barcelona, Spain, 9 2006. IEEE Computer Society.
14. Luiz Angelo Steffenel and Gregory Mounié. A framework for adaptive collective communications for heteroge neous hierarchical computing systems. *Journal of Computer & System Sciences, special issue on Perf ormance Analysis and Evaluation of Parallel, Cluster, and Grid Computer Systems*, 74(6):1082–1093, 2008.