Proceedings of the 17th IASTED International Conference
Parallel and Distributed Computing and Systems
November 14-16, 2005, Phoenix, AZ, USA

MESSAGES SCHEDULING FOR DATA REDISTRIBUTION BETWEEN HETEROGENEOUS CLUSTERS

Emmanuel Jeannot
INRIA – LORIA
Nancy, France
email: emmanuel.jeannot@loria.fr

Frédéric Wagner
LIA, Université d'Avignon
Avignon, France
email: frederic.wagner@lia.univ-avignon.fr

ABSTRACT
In this paper, we tackle the problem of redistributing data between clusters connected by a backbone. On distributed environments, communications often take more time [12] and thus lead to worse results than on local clusters. There is therefore a strong need to optimize the time needed by communications.Indeed, when an application composed of several codes running on distant clusters is executing, data are required to be redistributed between the clusters. We propose a general solution to the problem when the platform is fully heterogeneous platforms (each node of each cluster can communicate at different speed) or when some nodes have several network interface cards. We provide an algorithm for scheduling the messages that gives a solution at most twice as long as the optimal one. Simulation results show that it is giving almost optimal schedules on large redistribution patterns, and very good results in the general case.

KEY WORDS
Distributed cluster computing ; Message scheduling ; Data redistribution ; Approximation algorithm

1 Introduction

In this paper, we study the problem of data redistribution for heterogeneous cluster computing. This problem arise in the context of code-coupling [14, 16] or computational steering [9]. In such an application, two parallel machines (clusters) have to exchange data and redistribute [1, 6] this data from each node of the sending cluster to each node of the receiving one [15]. In general, the backbone that connects the two clusters is a bottleneck: it cannot handle all the communications at the same time because the aggregate bandwidth of each cluster exceeds its own bandwidth. Think for instance of two clusters of 100 nodes with 100 Mbit/s network cards interconnected by a backbone of 1 Gbit/s: in this case the aggregate bandwidth of each cluster is 100×100 Mbit/s = 10 Gbits/s. In order not to exceed the backbone bandwidth, no more than a given number of k messages have to be sent at the same time. In the previous example $k = 10$ because 10 communications at 100 Mbit/s leads to a 1 Gbit/s aggregated bandwidth.

All previous works of the litterature [6, 13, 18], propose several algorithms for solving this problem when a node operates in the 1-port model: at a given moment, each node can send or receive at most one message. In this paper, we extend previous results to the case where a node can send or receive an arbitrary number of messages (δ-port mode). The δ-port assumption is more general and is driven by these very important remarks: (1) the one-port model implies that all nodes of each cluster have the same network interface cards (see [13] for the details). However a distributed environment can be composed of heterogeneous resources: the δ-port model is one way of dealing with clusters composed of nodes with different network cards, (2) a cluster node can have several network cards and therefore can send more than one message at a given moment. Nowadays, this situation is very common as many clusters are composed of bi/quadri processors each having its own network interface card. This is not captured with the 1-port model, (3) very fast network cards are sometimes not able to communicate at full speed with one TCP socket. Several sockets have to be opened for communicating at full speed.

The contribution of this paper is the following. We propose a fast (polynomial) algorithm for solving the messages scheduling problem for data redistribution within the δ-port model called DGGP. We provide new lower bounds of the problem. We prove that this algorithm is an approximation algorithm (for any instance of the problem the given solution is at most twice as long as the optimal one). Experimental results show that DGGP provides results close to the optimal and, in most of the cases, outperforms our previous algorithm that do not consider the delta-port constraint.

2 The Redistribution Problem

2.1 Background

The redistribution problem has been widely studied in the case of homogeneous parallel programming. Indeed, data placement/distribution on each parallel computing element determine which algorithm can be applied and which performance can be expected. When different parallel algorithms have to be applied on the same data it is sometimes efficient to change the data placement and therefore redistribute this data. Redistribution routines have been implemented in many programming environments such as Scalapack (P#TRMR2D BLACS function) [2] and HPF

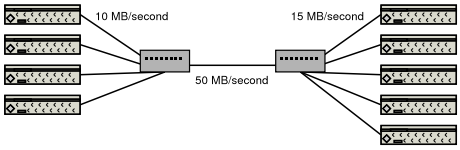

Figure 1. Exemple topology

(`redistribute` directive) [7]. Several steps are required to redistribute data:

1. **Data identification**. This step consists in determining which data is to redistribute, what is it size, what is its initial location what is its final location.

2. **Messages generation**. It consists, for each processor pair, to determine the data to be exchange. The output of this step is often call a redistribution matrix where element (i,j) of this matrix gives the amount of data to be exchanged between processor i and processor j as well as some other useful informations (data address, type, etc.).

3. **Messages scheduling**. This step computes in which order the message will take place. This step has to take into account the network topology in order to avoid contention. It also needs to balance communications in order to optimize the redistribution time.

4. **Communication**. The actual exchange of data between processors pair takes place during this step according to the scheduling computed during the previous step.

In this paper we focus on the message scheduling step. We assume that the communication matrix(see Fig. 2) is given.

We consider here the following case: the topology is composed of two distant clusters: the senders cluster and the receivers cluster as shown in Figure 1. Each node in the cluster is connected to the backbone via a switch. The backbone may be a bottleneck where only a limited number of communications can take place at the same time. Finally we allow preemption, that is any communication can be executed in several steps i.e. splitted into smaller communications.We associate a penalty to the preemption.

Data redistribution between distant cluster arise, for instance, in the context of distributed cluster computing such as code coupling [14, 16] when one part of the computation is done on one cluster and the other part is done on an other cluster. It arise also in the case of co-allocated parallel program, or in the case of execution of parallel tasks with dependencies.

2.2 Modeling of the Problem

The redistribution problem we study here is modeled by a weighted bi-partite graph: $G = (V_1, V_2, E, w, \delta)$ an integer k, a rational β. V_1 is the set of vertices on the left part of G. V_2 is the set of vertices on the right part of G. Each vertex of V_1 (resp . V_2) represents one node of the sending (resp. receiving) cluster. E is the set of edges. There is an edge between two vertices if a communication has to be issued between the two corresponding nodes during the redistribution. w is the weight function of each edge and represents the time needed to transfer the data between the two nodes. δ is a function that for each vertex of $V_1 \cup V_2$ gives the number of communications that can be issued at the same time from a given node. Note that the value of this function can be different for each vertex. k, the maximum number of communications that can be issued at the same time. It comes from the fact that the backbone interconnecting the two clusters can be a bottleneck. In order not to exceed its capacity k has to be computed such that the aggregated bandwidth of any k simultaneous communications is never greater than the backbone bandwidth. Since no node will be able to send more than k messages at the same time, we impose that $\forall n \in V_1 \cup V_2, \delta(n) \leq k$. β is the time to set up a communication step (a synchronized set of communications that all start at the same time).

2.3 Running Example

Throughout this paper we will use the following example of the redistribution between two heterogeneous clusters. Let us take the communication matrix of Figure 2. For instance, we see that node 1 of the first cluster has 700 Mbit to send to node A of the second cluster. Let us suppose that node 1 and B have a 300 Mbit/s network card, node 2 has 200 Mbit/s network card and node A a 100 Mbit/s NIC. Let us assume that the backbone has a 200 Mbit/s bandwidth[1] .

nodes	*A*	*B*
1	700	300
2	0	100

Figure 2. Exemple communication matrix (in Mbit)

In order to compute G, δ and k we need to introduce the notion of *base speed*. The base speed is the greatest common divisor of the backbone and all different network card speeds. In this case the base speed is 100 Mbit/s. We compute k, δ and w from the base speed that will be the communication speed during the redistribution.

Since all communications will take place at 100 Mbit/s and the backbone speed is 200 Mbit/s we deduce that $k = 2$. Concerning node 1 the network card is 300 Mbit/s and the base speed is 100 Mbit/s therefore $\delta(1) = 3$. For the same reason $\delta(B) = 3$, $\delta(2) = 2$ and $\delta(A) = 1$. We can see that $\delta(1)$ and $\delta(B)$ are greater than $k = 2$. We therefore reduce $\delta(1)$ and $\delta(B)$ to 2.

Since the base speed is 100 Mbit/s the weight of the edge connecting 1 to A in G is $\frac{700}{100} = 7$. The full graph G obtained is shown on Figure 3.

β depends on the system and the network. It is mainly the time to open a connection to a remote node plus the network latency.

[1] These numbers are a little bit unrealistic but they show that we can deal with a highly heterogeneous environment.

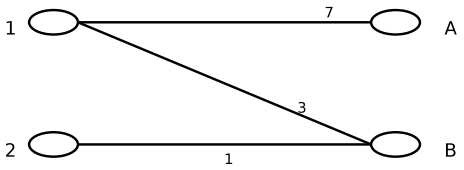

Figure 3. Graph G modeling the exemple

Note that we would have obtained the same modeling if node 1 and B had 3 network cards at 100 Mbit/s, node 2 two NIC at 100 Mbit/s and node A only one 100 Mbit/s card.

2.4 Modeling of the Solution

Given a bipartite graph $G = (V_1, V_2, E, w, \delta)$ an integer k and a rational β. We have to decompose G into s communications steps. A communication step i is given by a bipartite graph $G_i = (V_{1_i}, V_{2_i}, E_i, w_i, \delta)$ that describes which nodes are involved in this step, what communications are performed and what is the duration of each communication. The union of all the s steps must give the original graph: $G = \cup_{i=1}^{s} G_i$. In order to be valid a step has to respect the two following constraints:

1. The number of communications of the step must not exceed k. $\forall i : |E_i| \leq k$.
2. A node n cannot perform more than $\delta(n)$ communications. If we denote $d_i(n)$ the degree of n at step i, we have: $\forall i, n : d_i(n) \leq \delta(n)$.

The cost of each step is given by the cost to setup the step (i.e. β) plus the duration of the longest duration (i.e. $W_i = \max_{e \in E_i} w_i(e_i)$). Therefore, the cost of the solution is: $s \times \beta + \sum_{i=1}^{s} W_i$. Note that in this model we allow preemption: a communication can be cut into pieces and performed in several steps.

2.5 Example

Let us go back to our running example. For sake of simplicity we assume that $\beta = 1$. The example can be decomposed in a set of 3 steps as given Fig. 4. Thanks to preemption the edge between node 1 and A is decomposed in 3 edges. Steps cost are respectively 3, 1 and 3 and $3 \times \beta$ are needed for initializing each steps. Therefore, the total time needed for the redistribution is $3+1+3+3\times 1 = 10$.

2.6 Related Works

Up to the best of our knowledge all previous works deal with the 1-port constraint ($\forall n \in V_1 \cup V_2, \delta(n) = 1$) which is less general than the δ-port constraint. Therefore, it generalizes several well known-problems of the literature. The problem of redistributing data inside a cluster has been studied in the field of parallel computing for years [6, 18]. In this case the modelization and the formulation is the same except that the number of messages that can be send during one step is only bounded by the number of nodes ($k = \min(|V_1|, |V_2|)$). The problem has been partially studied in the context of Satellite-Switched Time-Division Multiple Access systems (SS/TDMA) [3, 10, 11]. The problem with $\beta = 0$ is studied in [3]. Finding the minimum number of steps is given in [10].The problem when no preemption is allowed is studied in [11]. Packet switching in communication systems for optical network problem also called wavelength-division multiplexed (WDM) broadcast falls in this problem [4, 8, 17, 19, 10]. In [8, 10], minimizing the number of steps is studied. In [4] and in [17], a special case the problem is studied when $k = |V_2|$. The problem of finding an optimal valid schedule has been shown NP-complete in [5]. In [13] we have proposed an approximation algorithm called Optimized Generic Graph Peeling algorithm (OGGP) for the problem with the 1-port constraint.

3 Algorithm

OGGP [13] provides a solution for the 1-port constraint. It is very important to note that, in this case, all communication step is a matching. In an informal description, OGGP is subdivided into two parts:

- An initialization step: enhancing the input graph to be able to match the constraints: (1) normalize all weights by β to limit the use of preemption, (2) add some new nodes and edges to enable the main step to respect the constraint given by k.
- The heart of the algorithm: peeling the graph into the final set of matchings.

We extend OGGP into Delta Generic Graph Peeling Algorithm (DGGP) in order to take into account the δ-port assumption. We start by normalizing all weights by β as in OGGP. We then transform the graph $G = (V_1, V_2, E, w, \delta)$ into the graph $G' = (V_1', V_2', E', w')$ using the algorithm given Fig. 5.

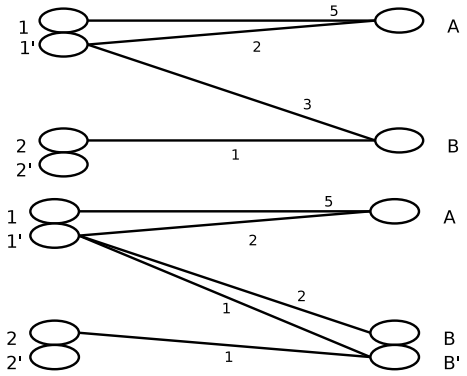

Figure 6. Decomposition of G into G' (top : after step 2; bottom after step 3).

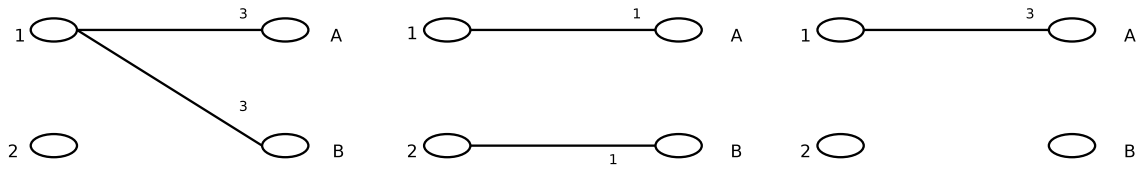

Figure 4. Decomposition in communication steps

1. $V_2' = V_2$
2. $\forall s \in V_1$:
   (a) compute $w(s) = \sum_{(s,s_2)\in E} w((s, s_2))$
   (b) build $(l_i)_{0\leq i<\delta(s)}$ the list representing the share of all communications of s given to each new node. We try to give an equal share of d to each new node. $\forall i \in \{1, \ldots, w(s) \bmod \delta(s)\}, l_i = \left\lceil \frac{w(s)}{\delta(s)} \right\rceil ; \forall i \in \{(w(s) \bmod \delta(s)) + 1, \ldots, \delta(s)\}, l_i = \left\lfloor \frac{w(s)}{\delta(s)} \right\rfloor$.
   (c) add to V_1' the nodes obtained by splitting s: $s_{i0\leq i<\delta(s)}$
   (d) now we split the edges: $\forall i | 0 \leq i < \delta(s)$
      i. while $l_i \neq 0$
         A. take an edge $a = (s, a_2) \in E$, such that a is of maximal weight
         B. if $w(a) \leq l_i$
            - $l_i = l_i - w(a)$, remove a from E, add (s_i, a_2) of weight $w(a)$ to E'
            - else $w(a) = w(a) - l_i$, add (s_i, a_2) to E' with $w'((s_i, a_2)) = l_i$, $l_i = 0$
3. issue the same operations to split edges from V_2

Figure 5. Formal algorithm splitting the nodes and edges

This algorithm first split each node n of V_1 into $\delta(n)$ nodes. Then, it dispatches equally all communications of node n to the new virtual nodes. Finally, the same procedure is applied for the nodes of V_2. The idea behind this transformation is that a matching (a communication step for OGGP) of G' corresponds to a set of edges of G (when merging back the nodes) having at most $\delta(n)$ edges adjacent to n. It is therefore a valid communication step for DGGP. Fig. 6 shows the result of the decomposition on our running example of Sec. 2.3.

We give the graph G' to OGGP and then merge back its nodes to obtain the final schedule.

Complexity. To split the edges, we iterate on each node s. For each of them we have $\delta(s) \leq k$ new nodes. This is done by iterating at most once on each edge. Hence splitting the edges is done at worst in $O(n \times k \times m)$. We create at most $\delta(s)$ edges when splitting s. This means that $|E'| \leq |E| + k \times (|V_1| + |V_2|)$ hence $m' \leq m + kn$. Finally it is clear that $n' \leq kn$. OGGP being in $O((m + n)^{\frac{5}{2}})$ we can see that this new algorithm is in $O((m+kn+kn)^{\frac{5}{2}}) = O((m + kn)^{\frac{5}{2}})$.

Approximability DGGP is an approximation algorithm. It means that whatever is the input, the solution is never too long compare to the optimal solution (See [20] for a formal proof).

4 Simulation Results

We study the behavior of DGGP under different situations, by simulations. All simulations consist in executing DGGP or OGGP on some random input pattern, computing the lower bound on the communication time and computing the optimum ratio defined as the time achieved by the algorithm divided by the lower bound [20]. Thus a value of 1 means we achieved optimal result. The value of δ for each node is randomly generated between 1 and 5.

We present several graphs detailing the obtained results. Each point in a graph is obtained as the average of computations on 100 random graphs. The number of parameters of the algorithm being pretty high, for all graphs mostly all parameters are fixed in order to obtain a 2D graph. Only the most significant results are presented here.

At first, we evaluated the performance of DGGP. We choosed here to execute DGGP on random graphs with 14 nodes in each set, with $k = 10$ and $k = 3$. We then plotted Figure 7 which shows that the results are getting better as the number of edges increases, with the variation on graphs of same size decreasing in the same way. Worst result is

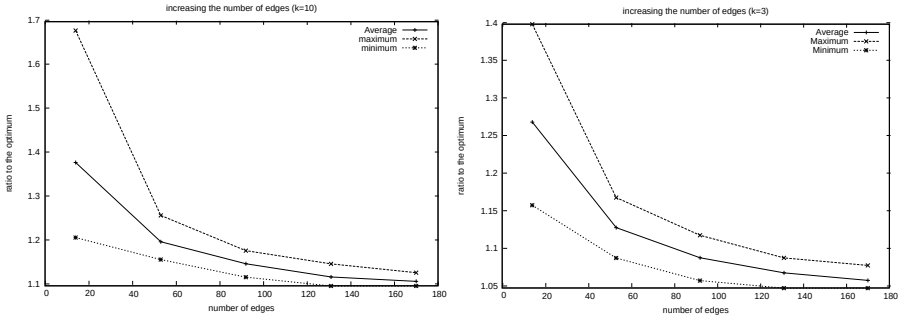

Figure 7. *DGGP Evaluation*

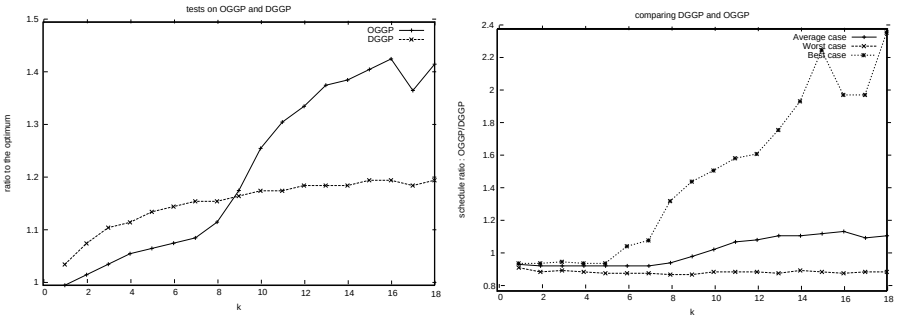

Figure 8. *DGGP- Comparisons*

of approximately 1.7 while for large graphs we are close from the optimum. This is explained because it is easier for the algorithm to compute matchings with edges of the same size if the graph has a high number of edges. As the variation in time is kept the same and the lower bound increases, the variation of the *ratio* decreases. The behavior is the same for graphs of another size.

We then compared DGGP with our previous OGGP algorithm. Figure 8 shows on the left side the optimum ratios for OGGP and DGGP as k increases. The input pattern is a random graph of 18 nodes in each set, with a random number of edges. We can see that OGGP starts as being slightly better than DGGP, but as k increases, results obtained for OGGP worsen with peaks at more than 20 % degradation. DGGP is slightly worse than OGGP for small k because we are dividing more edges than with OGGP, which we cannot really schedule in parallel as k is small.

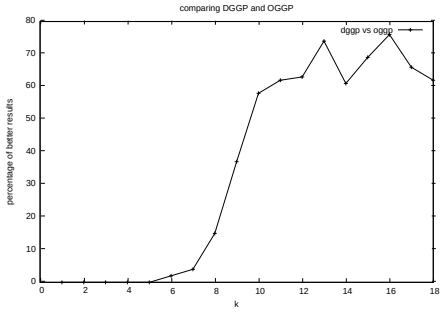

Figure 9. DGGP Improvements

The graph on the right side of Figure 8 plots the schedule ratio: the schedule time found by OGGP divided by the schedule time found by DGGP. We plot 3 graphs, the average case, the worst case and the best case. When this ratio is greater than 1 this means that DGGP outperforms OGGP. We see that on the average DGGP is slightly better than OGGP when k is large. On some cases it outperforms OGGP up to a factor of 240 % and this factor grows with δ while OGGP never outperforms DGGP to a factor greater than 10%.

Finally, we computed on Fig. 9 the number of times DGGP performs better than OGGP for each value of k. This confirms the results displayed in previous figures. DGGP is performing better than OGGP in 65% of cases for high value of k.

5 Conclusion

Data transmission is an important feature for cluster computing applications. Performance depends on how communications between resources are performed. This is especially true on a distributed environement where network performances are often poorer than on local clusters.

Here, we study the message scheduling problem for redistribution: a cluster has to send data to an other cluster through a backbone. When this backbone is a bottleneck, it is required to schedule the messages in order not to exceed its capacity.

As none of the previous works we allow the use of the δ-port model. This model enables us to take into account the heterogeneity of the nodes within a given cluster, the fact that there is more than one network card on some nodes or the necessity to perform multi-socket data transfer.

We proposed DGGP, an approximation algorithm, Simulation results show that it can achieve nearly optimal results on complex input patterns while improving results of OGGP in general.

Acknowledgment

This work is supported by the INRIA ARC RedGRID.

References

[1] P.B. Bhat, V.K. Prasanna, and C.S. Raghavendra. Block Cyclic Redistribution over Heterogeneous Networks. In *11th Int. Conf. on Parallel and Distrinuted Computing Systems (PDCS 1998)*, 1998.

[2] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.

[3] G. Bongiovanni, D. Coppersmith, and C. K. Wong. An Optimum Time Slot Assignment Algorithm for an SS/TDMA System with Variable Number of Transponders. *IEEE Trans. on Com.*, 29(5):721–726, 1981.

[4] H. Choi, H.-A. Choi, and M. Azizoglu. Efficient Scheduling of Transmissions in Optical Broadcast Networks. *IEEE/ACM Transaction on Networking*, 4(6):913–920, December 1996.

[5] J. Cohen, E. Jeannot, and N. Padoy. Messages Scheduling for Data Redistribution between Clusters. In *Algorithms, models and tools for parallel computing on heterogeneous networks (HeteroPar'03) workshop of SIAM PPAM 2003, LNCS 3019*, pages 896–906, Czestochowa, Poland, September 2003.

[6] F. Desprez, J. Dongarra, A. Petitet, C. Randriamaro, and Y. Robert. Scheduling Block-Cyclic Array Redistribution. *IEEE Transaction on Parallel and Distributed Systems*, 9(2):192–205, 1998.

[7] High Performance Fortran Forum. *High Performance Fortran Language Specification, Version 2.0.α.3*. Department of Computer Science, Rice University, August 1996.

[8] A. Ganz and Y. Gao. A Time-Wavelength Assignment Algorithm for WDM Star Network. In *IEEE INFOCOM'92*, pages 2144–2150, 1992.

[9] G.A. Geist, J.A. Kohl, and P.M. Papadopoulos. CUMULVS: Providing Fault-Tolerance, Visualization and Steering of Parallel Applications. *International Journal of High Performance Computing Applications*, 11(3):224–236, August 1997.

[10] I.S. Gopal, G. Bongiovanni, M.A. Bonuccelli, D.T. Tang, and C.K. Wong. An Optimal Switching Algorithm for Multibean Satellite Systems with Variable Bandwidth Beams. *IEEE Trans. on Com.*, COM-30(11):2475–2481, November 1982.

[11] I.S. Gopal and C.K. Wong. Minimizing the number of switching in an ss/tdma system. *IEEE Trans. on Communications*, 33:497–501, 1885.

[12] Marc Grunberg, Stéphane Genaud, and Catherine Mongenet. Seismic Ray-Tracing and Earth Mesh Modeling on Various Parallel Architectures. *the journal of supercomputing*, 29(1):27–44, july 2004.

[13] E. Jeannot and F Wagner. Two Fast and Efficient Message Scheduling Algorithms for Data Redistribution through a Backbone. In *Proceedings of the IEEE 18th International Parallel and Distributed Computing Symposium (IPDPS 2004)*, Santa-Fe, New-Mexico, USA, April 2004.

[14] K. Keahey, P. Fasel, and S. Mniszewski. PAWS: Collective Interactions and Data Transfers. In *10th IEEE Intl. Symp. on High Perf. Dist. Comp. (HPDC-10'01)*, pages 47–54, August 2001.

[15] Oak Ridge National Labs. Mxn. `http://www.csm.ornl.gov/cca/mxn`.

[16] C. Pérez, T. Priol, and A. Ribes. A parallel corba component model for numerical code coupling. In Manish Parashar, editor, *Proc. of the 3rd International Workshop on Grid Computing*, number 2536 in LNCS, pages 88–99, Baltimore, Maryland, USA, November 2002. Springer-Verlag.

[17] G.R. Pieris and Sasaki G.H. Scheduling Transmission in WDM Broadcast-and-Select Networks. *IEEE/ACM Transaction on Networking*, 2(2), April 1994.

[18] L. Prylli and B Tourancheau. Efficient Block-Cyclic Data Redistribution. In *Europar'96, LNCS 1123*, pages 155–164, 1996.

[19] N. Rouskas and V. Sivaraman. On the Design of Optimal TDM Schedules for Broadcast WDM Networks with Arbitrary Transceiver Tuning Latencies. In *IEEE INFOCOM'96*, pages 1217–1224, 1996.

[20] F. Wagner. *Redistribution de données à travers un réseau à haut débit*. PhD thesis, Université Henri Poincaré, Nancy-1, 2005.