HAL
archives-ouvertes.fr

Message Scheduling for Data Redistribution through High Performance Networks

Emmanuel Jeannot, Frédéric Wagner

▶ To cite this version:

Emmanuel Jeannot, Frédéric Wagner. Message Scheduling for Data Redistribution through High Performance Networks. DistRibUtIon de Données à grande Echelle - DRUIDE'2004, May 2004, Le Croisic/France, pp.10, 2004. <inria-00099881>

HAL Id: inria-00099881
https://hal.inria.fr/inria-00099881

Submitted on 26 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Message Scheduling for Data Redistribution through High Performance Networks

Emmanuel Jeannot
LORIA, Université H. Poincaré
Nancy, France
Emmanuel.Jeannot@loria.fr

Frédéric Wagner
LORIA, INRIA-Lorraine
Nancy, France
Frederic.Wagner@loria.fr

30th March 2004

Abstract

With the emergence of large scale distributed computing, new problems bound to data transfers are appearing. We present the problem of data redistribution between two clusters connected by a high performance network. This problem consists in finding the best way to transfer data from the first cluster to the second one in the shortest possible time. In order to avoid slowing down the network, and the transfer, it is necessary to schedule the messages. This problem (named as KPBS) is known to be NP-complete and we present here a 2-approximation algorithm we developped.

1 Problem

1.1 Presentation

We consider data redistribution between two distant clusters connected by a backbone as shown in Figure 1. Each cluster is connected to the backbone by a switch. For each cluster we consider the bandwidth between a node and its switch to be equal for each node. Each network card is assumed to be full-duplex and 1-port.

Given the different bandwidths we can compute k, the maximum number of simultaneous connections generating no interferences. In the example of Figure 1, $k = 4$ since all nodes in the left cluster can communicate at full bandwidth without exceeding the backbone's bandwidth. If we note b_1, b_2 the bandwidths

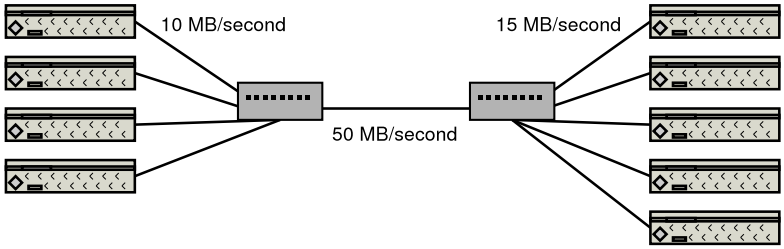

Figure 1: two distant clusters

connecting cluster's nodes to switchs, bb the backbone bandwidth, v_1, v_2 the number of nodes in each cluster, we have $k = \min(v_1, v_2, \left\lfloor \frac{bb}{\min(b_1,b_2,bb)} \right\rfloor)$. As we want to limit the network saturation, we need to ensure we do not issue more than k communications at any time, and also that any network card does not issue or receive more than one communication at any time.

The representation chosen to modelize our data redistribution problem is a weighted bipartite graph $G = (V_1, V_2, E, f)$, with V_1 representing the first cluster's nodes, V_2 the second cluster's nodes and $E \subset V_1 \times V_2$ (the graph's edges) the communications to perform. The f function ($f : E \to \mathbb{Q}$) represents the cost in terms of time associated to each communication.

For example consider the communications of Figure 2 (on the previously introduced network) in Megabytes :

cluster nodes	$y1$	$y2$	$y3$	$y4$
$x1$	30	20	0	0
$x2$	20	0	20	0
$x3$	0	0	40	20
$x4$	0	15	10	10

Figure 2: Communication matrix

These communications can be represented by the bipartite graph of Figure 3. To obtain this graph we first need to convert the megabytes to communicate into a time unit. Therefore we divide each entry in the communication matrix by the worst bandwidth in the network ($\min(b_1, b_2, bb)$). In our example we have at least 10 Megabytes/second, so the 40 MB communication from node $x3$ to node $y3$ will take us $40/10 = 4$ units of time.

1.2 Goal

The problem we want to solve is the following : given G, a bipartite graph, k, the maximal number of simultaneous communications and β, the setup delay,

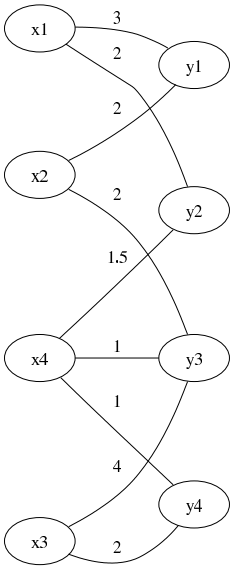

Figure 3: Communications of Figure 2 seen as a graph

slice G into a sequence of communication steps, with of course the time needed to perform all steps being optimal. This problem is known as KPBS and has been proven in [1] to be NP-complete. For example consider the graph of Figure 3 with $k = 4$ and $\beta = 3$. A feasible schedule is given in Figure 4.

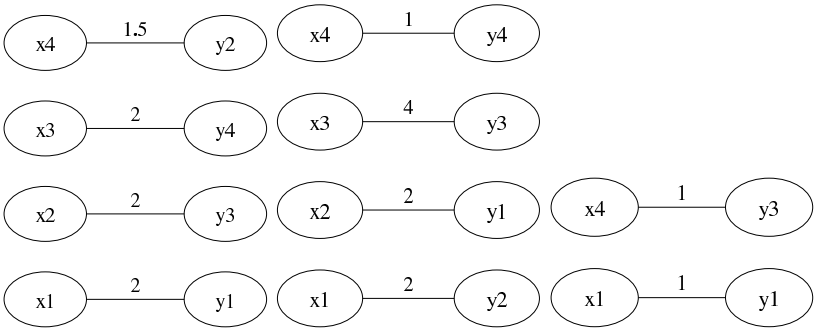

Figure 4: Example schedule

This schedule is feasible because at each step the number of communications is no more than 4, and no node issues two communications simultaneously. On our results this means that we obtain a set of matchings whose cardinality is at most k. We allow preemption, so, it is possible to cut one edge accross different communication steps. For example, here, the 3 data units long communication

from x_1 to y_1 is issued in two times. Such a schedule has an easily computable cost : each step takes as time β (initialization) + the time of it's longest communication.

Here the schedule time is : $\beta + 2 + \beta + 4 + \beta + 1 = 3 \times 3 + 7 = 16$ units of time.

1.3 Notations

We call $G = (V_1, V_2, E, f)$ our bipartite graph, with V_1 and V_2 the two sets of nodes, $E \subset V_1 \times V_2$ the edges, and $f : E \to \mathbb{Q}$ the weight function.

We note: $m(G) = |E(G)|$ the number of edges, $\Delta(G)$ the maximal degree of all nodes, $w(s)$ the sum of the weights of all edges incident to node s, $W(G)$ the max of all $w(s)$, and $P(G)$ the sum of the weights of all edges.

Given a graph G, it is possible to compute $\eta(G)$ a lower bound to the optimal communication time for G. With η_e the minimal number of communication steps and η_d the minimal time required to transfer datas, we have $\eta = \eta_d + \beta\eta_e$.

It is easy to see that η_e is the maximum between $\Delta(G)$ and $\left\lceil \frac{m(G)}{k} \right\rceil$ as shown in Figure 5. We cannot have less than $\Delta(G)$ steps since this would mean having in one communication step one node issuing two communications, and less than $\left\lceil \frac{m(G)}{k} \right\rceil$ since it would mean issuing more than k communications in one step.

Similarly we have $\eta_d = \max\left(W(G), \left\lceil \frac{P(G)}{k} \right\rceil\right)$ It should be noted that η is not the optimal time but a lower bound to it. In particular, there are cases where η cannot be reached.

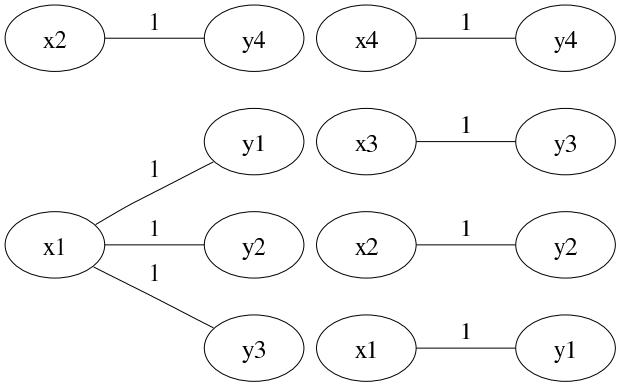

Figure 5: $k = 2$, we have respectively $\eta_e = 3$ and $\eta_e = 2$

2 Algorithm

In [1], several heuristics, and one approximation algorithm have been proposed for KPBS. However, we have proven than these heuristics may in some cases behave badly. The approximation algorithm proposed being of high complexity, we introduced in [3] two new polynomial-time, 2-approximations.

We are going to present GGP, one of our 2 algorithms. We give here the complete formal description of GGP, together with a short informal description. However this algorithm is complex, so proofs of approximation and complexity will not be described in this paper. One should refer to [2] for more details regarding this algorithm.

2.1 GGP Algorithm (formal description)

1. $R = \emptyset$
2. Build $H = (V_1, V_2, E, f_H)$ such that $\forall e \in E, f_H(e) = \left\lceil \frac{f_G(e)}{\beta} \right\rceil$
3. If $\frac{P(H)}{k} < W(H)$ build the graph $I = (V_{1_I}, V_{2_I}, E_I, f_I)$ such that :
   - $E \subset E_I, V_1 \subset V_{1_I}, V_2 \subset V_{2_I}$
   - $\forall e \in E, f_I(e) = f_H(e)$
   - $\frac{P(I)}{k} = W(H)$
   - $\exists e \in E_I | e \notin E, f_I(e) \leq \max\left(W(H), \left\lceil \frac{P(H)}{k} \right\rceil\right) = \frac{P(I)}{k}$
   - $\forall d \in E_I | d \notin E, d \neq e$ we have $f_I(d) = \max\left(W(H), \left\lceil \frac{P(H)}{k} \right\rceil\right) = \frac{P(I)}{k}$
   - $\forall v \in V_{1_I} | v \notin V_1$ the degree of v is 1, the edge e incident to v is incident to $v_2 \in V_{2_I} | v_2 \notin V_2$

   Else if $\frac{P(H)}{k} \notin \mathbb{N}$
   - $E \subset E_I, V_1 \subset V_{1_I}, V_2 \subset V_{2_I}$
   - $\forall e \in E, f_I(e) = f_H(e)$
   - $\frac{P(I)}{k} = \left\lceil \frac{P(H)}{k} \right\rceil$
   - $\exists e \in E_I | e \notin E, f_I(e) \leq \max\left(W(H), \left\lceil \frac{P(H)}{k} \right\rceil\right) = \frac{P(I)}{k}$
   - $\forall d \in E_I | d \notin E, d \neq e$ we have $f_I(d) = \max\left(W(H), \left\lceil \frac{P(H)}{k} \right\rceil\right) = \frac{P(I)}{k}$
   - $\forall v \in V_{1_I} | v \notin V_1$ the degree of v is 1, the edge e incident to v is incident to $v_2 \in V_{2_J} | v_2 \notin V_2$

   Else $I = H$
4. Build the graph $J = (V_{1_J}, V_{2_J}, E_J, f_J)$ such that :

- $E_I \subset E_J, V_{1_I} \subset V_{1_J}, V_{2_I} \subset V_{2_J}$
- $\forall e \in E_I, f_J(e) = f_I(e)$
- J is $\frac{P(I)}{k}$-weight-regular
- $\forall (v_1, v_2) \in E_J | (v_1, v_2) \notin E_I$, either $v_1 \notin V_{1_I}$ or $v_2 \notin V_{2_I}$

5. While $E_J \neq \emptyset$ do :
   (a) Choose a perfect matching M in J
   (b) Add M to S a set of matchings
   (c) Compute s the smallest weight of the edges in M
   (d) For each edge in M reduce it's weight by s
   (e) Remove from E_J all edges of weight 0
6. Remove all edges e in S such that $e \notin E$
7. While $S \neq \emptyset$
   (a) Choose a matching M in S
   (b) Modify f_M such that $\forall e \in M, f_M(e) = \min(f_M(e)\beta, f_G(e))$ and f_G such that $\forall e \in M, f_G(e) = f_G(e) - f_M(e)$
   (c) Add M to R

2.2 GGP Algorithm (informal description)

The main part of GGP is the step 5. At each iteration, we compute a matching which will correspond to a matching i.e. to a communication step in our results set. We simply take a perfect matching, and use preemption to assure all edges in this matching have the same weight (i.e. we cut all edges to the length of the smallest one). This is because once a node has finished its communication, it needs to wait until the longest communication in the step is finished to proceed further. By ensuring every communication in the step has the same size, no one is waiting, and no bandwidth is lost.

However taking at each step a perfect matching is troublesome, as we in fact don't want more than k edges in each matching. To get around that problem, we first add some "virtual" edges in steps 1 to 4 to G, to ensure that in our perfect matching of size s computed at step 5 there will be at least $s-k$ "virtual" edges, that is at most $s - (s - k) = k$ edges belonging to G.

In these steps we also ensure the J graph builded at step 4 is weight-regular. This property is important because it guarantees that it is possible at step 5 to find a perfect matching. As at each iteration the remaining graph stays regular, it is always possible to find a perfect matching, and the algorithm is working.

The final steps simply consists in removing all virtual edges from the intermediate results set to get the final set of matchings.

2.3 Example

To illustrate this algorithm, we take the graph of Figure 2.3, with $k = 3$ and $\beta = 1$.

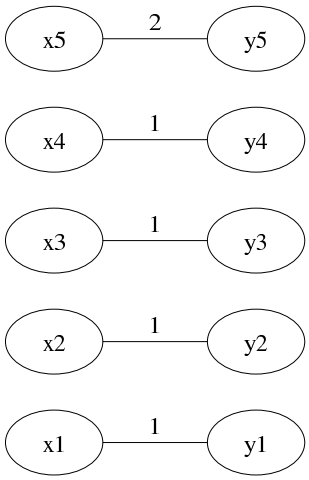

Figure 6: Input graph G

At steps 1 one 2 we don't do anything. At step 3 we compute $\frac{P(H)}{k} = \frac{1+1+1+1+2}{3} = 2$ and $W(H) = \max(2, 1) = 2$ so we also don't do anything. At step 4, we build the graph J as shown in Figure 7, by adding 4 virtual edges and nodes. It should be noted that J is a 2 weight-regular graph.

Now we can start the main loop. We choose a perfect matching. You should note that it isn't possible to choose a matching with more than 3 real edges. We choose here the matching shown in Figure 7.

In this matching, the smallest weight is 1, so we reduce all weights of its edges by 1. All edges whose weights reach 0 are removed from J, this means only the edge from x_5 to y_5 with a weight of 2 (now 1) is keeped. We now only have the graph of Figure 8 left.

In the second iteration, the perfect matching is given by all remaining edges. As they are of the same weight, we now have completed the algorithm.

We now remove all virtual edges to get the final result, R, displayed in Figure 9. The cost is here : $1 + 1 + 2\beta = 4$ units of time.

2.4 Evaluation

The evaluation of GGP has been done in several steps.

- we have proven that GGP is a 2-approximation algorithm.
- GGP has been tested on random bipartite graphs. The tests have shown

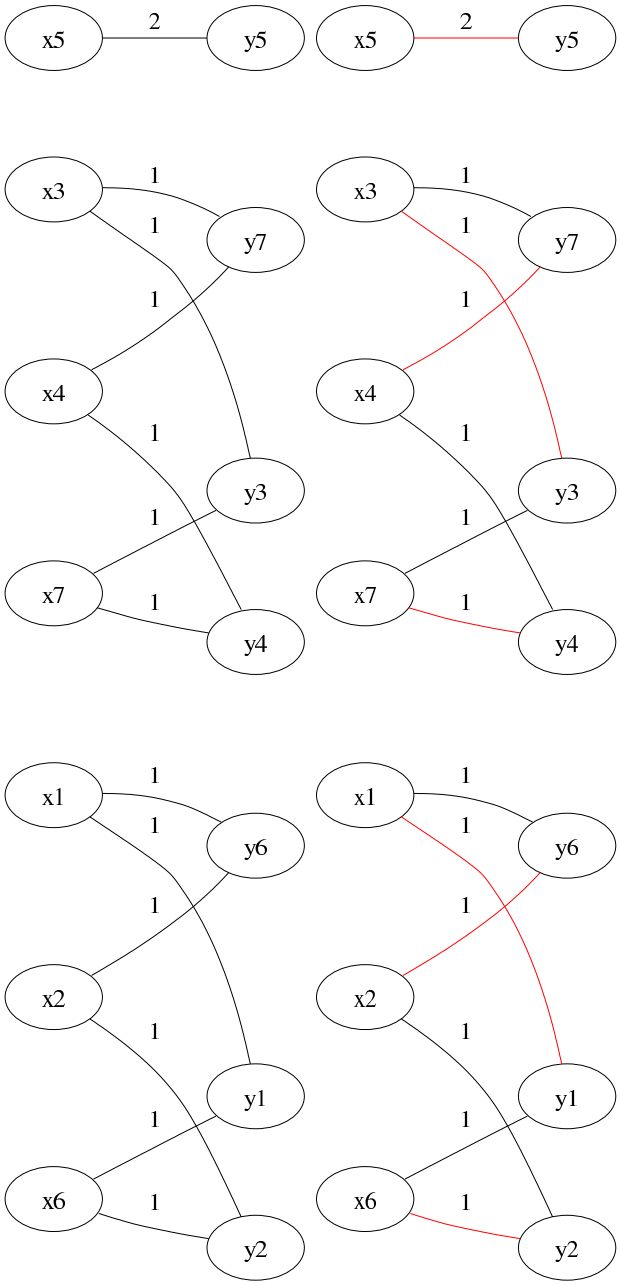

Figure 7: Graph J, first perfect matching chosen

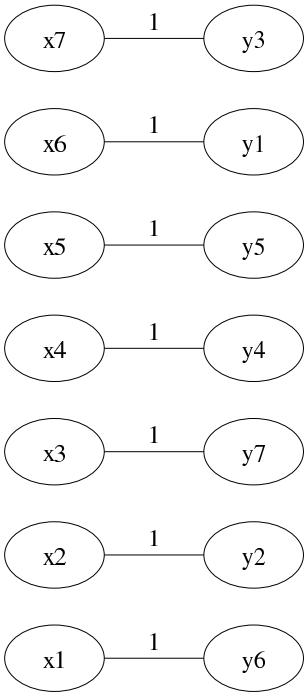

Figure 8: Graph J, first perfect matching removed

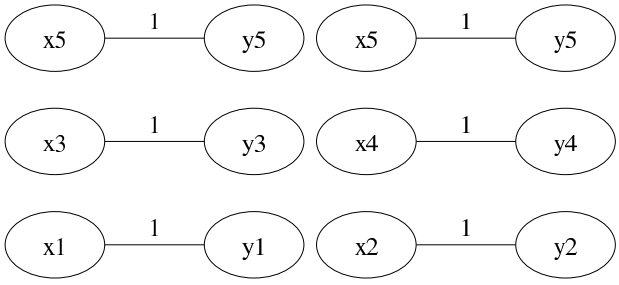

Figure 9: Final scheduling

some good results with weights close in range from β, and some quasi-optimal results with weights large before β.

- we conducted several redistributions on a LAN, using both MPI and low-level communication primitives. Results are various and depends both on the configuration, and on the input set.
- tests are underway within the ARC Redgrid.

3 Conclusion

In this paper we have studied and proposed a message scheduling algorithm called GGP for the redistribution problem. The novelty of our approach is that we set the maximum number of messages during one step. This is especially useful when the redistribution is performed between two clusters interconnected by a backbone and when this backbone is a bottleneck. However the algorithms we have proposed can also be used when redistribution happen on the same parallel machines or in the context of SS/TDMA systems or WDM network.

In our future work , we want to extend the model to handle more complex redistributions.

First we would like to consider achieving a local pre-redistribution in case a high-speed local network is available. This would allow to aggregate small communications together, or on the opposite to dispatch communications to all nodes in the cluster.

Second, we would like to study the problem when the throughput of the backbone varies dynamically or when the redistribution pattern is not fully known in advance. We think that our multi-step approach could be useful for these dynamic cases.

The final goal of this work is to produce (together with the people involved in the INRIA ARC redGRID) a fully working redistribution library.

Bibliography

References

[1] J. Cohen, E. Jeannot, and N. Padoy. Parallel data redistribution over a backbone. Technical Report RR-4725, INRIA-Lorraine, February 2003.

[2] E. Jeannot and F. Wagner. Message scheduling for data redistribution through high performance networks. Technical report, INRIA, 2004.

[3] E. Jeannot and F. Wagner. Two fast and efficient message scheduling algorithms for data redistribution through a backbone. In *IPDPS 2004*, April 2004.