Modeling, Predicting and Optimizing Redistribution between Clusters on Low Latency Networks

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

Frédéric Wagner
frederic.wagner@loria.fr
LIA
Avignon, France

Abstract

In this paper we study the problem of scheduling messages between two parallel machines connected by a low latency network during a data redistribution. We compare two approaches. In the first approach no scheduling is performed. Since all the messages cannot be transmitted at the same time, the transport layer has to manage the congestion. In the second approach we use two higher-level scheduling algorithms proposed in our previous work [10] called GGP and OGGP. The contribution of this paper is the following: We show that the redistribution time with scheduling is always better than the brute-force approach (up to 30%). As this speedup depends on the input redistribution pattern, we propose a modelization of the behavior of both approaches and show that we are able to accurately predict the redistribution time with or without scheduling and thus able to choose for each pattern whether or not to schedule the communications.

Key-words: Code-coupling, cluster computing, message scheduling, modelization.

1 Introduction

Code coupling applications (such as multi-physics simulations) are composed of several parallel codes that run concurrently on different clusters or parallel machines. In multi-physics simulations each code has in charge one part of the simulation. For instance simulating the airflow on an airplane wing requires to simulate fluid mechanics phenomena (the air) with solid mechanic one (the wing). In order for the simulation to be realistic each simulation code need to exchange data. This exchange of data is often called *data redistribution*.

The data redistribution problem has been extensively tackled in the literature [1, 2]. However in these works the redistribution takes place within the same parallel machine. In the context of code-coupling, several machines are involved and therefore the redistribution must take place between two different machines through a network. In many cases this network is a bottleneck because the aggregate bandwidth of network cards of each nodes of each parallel machines exceeds the bandwidth of the network that interconnect these two machines (think for instance of two clusters with 100 nodes each and 100 MBit cards interconnected by a 1 Gbit network).

In our previous work [5, 10] we have proposed several application-level messages scheduling algorithms (called GGP and OGGP) for the redistribution problem between clusters when the network is a bottleneck.

In this paper we study the problem in the context of a low latency network (LLN), such as LAN, where the transport layer gives very good performances. We provide a modelization of the behavior of TCP for the data redistribution problem as well as of (O)GGP. We compare two ways of predicting the data redistribution time and the performances of (O)GGP and a brute-force approach where no scheduling is performed.

2 Data Redistribution

2.1 Modeling

Let us consider a redistribution taking place between two clusters connected by a low latency network. We suppose that this network can handle at most k communications at the same time between the two clusters. For instance if both clusters have 100 Mbit network cards and are interconnected by a 1 Gbit network there can be at most $k = 10$ communications at the same time without contention. When k is smaller than the number of nodes in each cluster, it is required to schedule the communications in order to avoid contention.

Each redistribution phase is considered separately. A

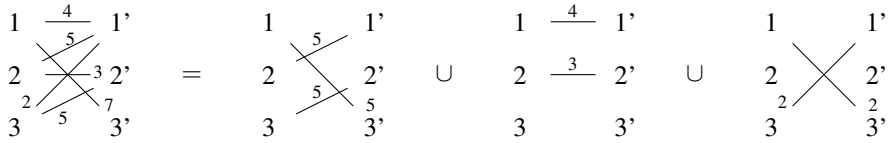

Figure 1. *Valid schedule (step 1 as a duration of 5, step 2 a duration of 4 and step 3 a duration of 2). If $\beta = 1$ the total cost of the schedule is 3+11=14. Note that, thanks to the preemption the communication of duration 7 is decomposed of a communication of duration 5 and a communication of duration 2*

redistribution phase is modeled by a redistribution matrix $M = m_{i,j}$ where $m_{i,j}$ is the time for sending the data from node i of the first cluster to node j of the second one. This time is computed by dividing the amount of data to be sent by the speed of the communication when no contention occurs. This matrix is given and computed by an other module of the application.

The redistribution is performed by steps. In order to optimize the solution we allow preemption. This means that at the end of a step, communication can be stopped and resumed in an other step.

We denote by β the time to setup a communication step (it takes into account the time to open the sockets, the latency of the network, the preemption, etc.).

Finally we do not allow contention on the network card. This means that we work in the 1-port model. During one step, a node can receive (resp. send) at most one communication from (resp. to) an other node. We provide here a graph formulation for the redistribution problem. It is called KPBS for K-Preemptive Bipartite Scheduling. A matrix can be seen as a bipartite graph. Let $M = m_{i,j}$ a redistribution matrix with n_1 lines and n_2 columns (this means that the first cluster has n_1 nodes while the second cluster has n_2 nodes). We build a bipartite graph with n_1 vertices on one side and n_2 vertices on the other. We draw an edge from node i of the first side to node j of the second side if element $m_{i,j}$ of M is not zero. We label this edge by $m_{i,j}$.

A step S is modeled by a bipartite graph too. In order for this step to be valid it has to model at most k communications and no node can perform more than one communication (1-port model). This means that the bipartite graph has to be a matching with at most k edges. The duration of a step is the startup time plus the duration of its longest communication ($m = \max(s_{i,j})$) where $s_{i,j}$ is the duration of the communication between node i and node j for this step.

A valid schedule of the redistribution is a set of s valid steps S_i such that $M = \cup_{i=1}^{s} S_i$. The cost of the schedule is the sum of the cost of each step : $\sum_{i=1}^{s}(\beta + m_i)$, where m_i is the length of step i (see Figure 1).

2.2 Related works

The KPBS problem 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, 12]. 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 ($k = \min(n_1, n_2)$). The problem has been partially studied in the context of Satellite-Switched Time-Division Multiple Access systems (SS/TDMA) [3, 8, 9]. The problem with $\beta = 0$ is studied in [3]. Finding the minimum number of steps is given in [8].The problem when no preemption is allowed is studied in [9]. Packet switching in communication systems for optical network problem also called wavelength-division multiplexed (WDM) broadcast falls in KPBS [4, 7, 11, 13, 8]. In [7, 8], minimizing the number of steps is studied. In [4] and in [11], a special case the KPBS problem is studied when $k = n_2$.

The problem of finding an optimal valid schedule has been shown NP-complete in [5].

In [10] we have proposed two approximations scheduling algorithms (the found schedule is never twice as long as the optimal one) for KPBS. These algorithms are called GGP (Generic Graph Peeling) and OGGP (Optimized GGP). It is out of the scope of this paper to describe them in details as they have already been presented in other publications, and due to space constraints (see [5, 10] for more information).

2.3 Other Notations

We note $G = (V_1, V_2, E, w)$ the bipartite graph, with V_1 and V_2 the sets of senders and receivers, E the set of edges, and w the edges weight function, $P(G) = \sum_{e \in E} w(e)$, $W(G) = \max_{s \in V_1 \cup V_2}(w(s))$ with $w(s)$ the sum of the weigths of all edges adjacent to s and η_o: the estimated redistribution time for when no scheduling is performed.

3 Predicting redistribution time

We present here how we predict the redistribution time. Two approaches are studied. One, called the *brute-force* approach, is used when no scheduling is performed at the application level and when the transport layer manage the contention by itself. In this case all the messages are sent at the same time. In the other approach, called the *scheduling* approach we use (O)GGP to schedule the messages and the contention is managed at the application level.

3.1 Brute-force approach

As the time needed by (O)GGP to compute and issue a schedule of all communications is not negligible, it is useful to develop a quick algorithm to estimate the time needed by the brute force approach. This could enable a quick decision algorithm as whether or not to use a scheduling approach.

We assume that the application is running on a dedicated platform and there is therefore no random perturbation on the network. In our modelization, we do not take into consideration the cost of network congestion.

We recall that η_o is the estimated redistribution time of the brute-force approach.

If more than k communications occur at the same time the network is assumed to be fairly shared.

Lower bound-based prediction

Several possibilities present themselves for computing η_o. The easiest method is to take the maximal time between $\frac{P(G)}{k}$ and $W(G)$. In other words η_o depends either from the bottleneck generated by the bandwidth of the network, or from the bottleneck generated on a local link by the sender or the receiver who has the most data to send. While this gives a lower bound of η_o, quick to compute, we can show that such an estimation of η_o is not always tight. For example consider the graph of Figure 2.

With $k = 2$ we can see that $\frac{P(G)}{k} = \frac{4}{2} = 2$ and $W(G) = 2$. So we know it needs at least 2 seconds to issue the redistribution.

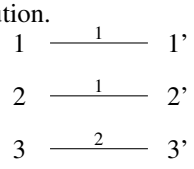

Figure 2. *Simple redistribution pattern, $k = 2$*

However the brute force redistribution takes place in two steps: At first, every node is sending data. As we assume that the network is fairly shared, the speed of the 3 communications is slowed-down by a factor of $\frac{3}{k}$.

Therefore It takes $1 \times \frac{3}{2} = 1.5$ seconds to send the data between nodes 1 and $1'$ and 2 and $2'$ because $k = 2$. After that time there is only one communication remaining (between node 3 and $3'$). It takes 1 second to send the remaining data. Therefore, the total time taken by the brute force redistribution is here of 2.5 seconds, that is 25% higher than the estimation.

Prediction algorithm

Computing η_o is done using a discrete event simulation of the redistribution. It is based on the same principle that leads to compute the brute-force redistribution time of the graph fig. 2 above: as the fraction of bandwidth available for an edge only depends on the redistribution pattern, we compute the fraction of bandwidth available for each communication. We then compute the time needed to finish each communication. We take the first communication to finish, remove it from the graph and add its time to η_o. At this point the redistribution pattern changed, so the amount of bandwidth available for everyone changes and we need to start again the simulation.

3.2 Scheduling approach

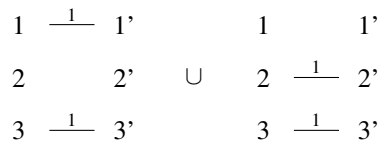

Figure 3. *Schedule of the pattern of Fig. 2 in two steps*

We can see on Fig. 3 that the scheduled redistribution of the graph Fig. 2 takes 2 second to execute, reaching the optimal time (without taking into account the time needed to establish the communications). This means that the brute-force approach is not optimal and gives inaccurate prediction of schedule redistribution.

Thankfully, estimating the time needed for a schedule redistribution is straightforward. We compute the cost of the schedule using the formula given in Section 2.1: $\sum_{i=1}^{s}(\beta + m_i)$

4 LAN Experiments

In order to prove both the accuracy of the evaluation of η_o and the existence of redistribution patterns giving

a suboptimal performance we conducted experiments on a real local area network.

The platform running the tests is composed of two clusters of 10 1.5Ghz PC running Linux and interconnected by two switches. All links, switches and Ethernet adapters have a speed of 100Mbits, but we limited the available incoming and outgoing bandwidth of each network card using the *rshaper* Linux kernel module. We conducted experiments for $k = 5$. All tests were conducted without interferences from other network users.

All communications routines have been implemented by hand using standard POSIX sockets and threads to get maximal performance and a complete control over communications.

We implemented a brute force redistribution algorithm. All communications are issued simultaneously, and the TCP is handling the network congestion. We also implemented a scheduled redistribution algorithm consisting of several redistribution steps synchronized using barriers. The scheduled has been obtained using our GGP algorithm [10].

The first tests are executed on some random redistribution patterns. The input data always has the same number of communications, however of random weights, and between random senders and receivers. The weights are generated uniformly between 10MB and xMB. All further plots are obtained as x increases from 10MB to 80MB. The first estimation for the brute force time is done computing the lower bound of the communication time and the second one using the discrete simulation. We executed the redistribution and computed the accuracy of the two estimations as the size of the data increases. We also compared the time estimated for the scheduled redistribution with the real time obtained. We can see in Figure 4 that all methods are giving a reasonable estimation of the real redistribution time, with η_o being slightly more precise for the brute force redistribution.

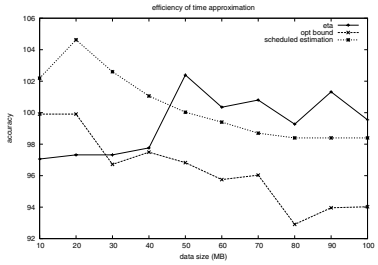

Figure 4. *Accuracy of redistribution prediction time on random patterns*

We then tested the redistribution on redistribution pattern where the lower bound is not tight (as in Fig 2). The results displayed in Figure 5 show that the experiments confirm the theory. The lower bound is far under the real time observed, whereas the η_o estimation is close from it. Furthermore, the time computed for the scheduled reditribution is close from the real time observed.

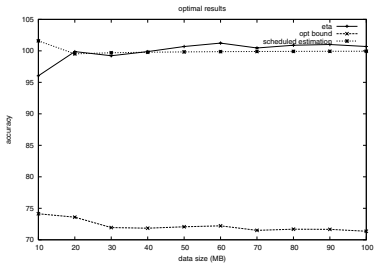

Figure 5. *Accuracy of redistribution prediction time on special patterns*

Brute-force vs. (O)GGP

We compared the efficiency of (O)GGP over a brute force redistribution: The results shown in Figure 6 confirm again the theory. The scheduled redistribution is on average 30% faster than the brute force redistribution. We can also see that as the redistribution pattern is kept the same with only increasing weights, the times scale linearly with the data size.

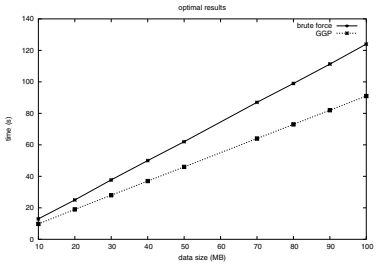

Figure 6. *GGP vs. Brute-force (socket implementation)*

While all the previous experiments have been done using POSIX socket, we have also implemented both algorithms using MPI. The use of MPI is justified by its portability and ease of programming. For this case, we have performed a all-to-all redistribution and compare the redistribution time in Fig. 7. In this case we see that (O)GGP outperforms the brute-force appraoch by a factor of 20%.

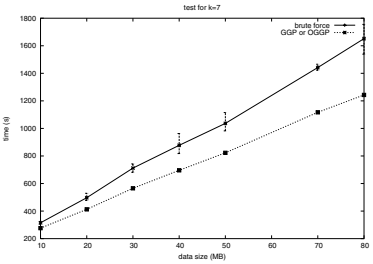

Figure 7. Brute-Force vs. GGP or OGGP (MPI implementation)

5 Conclusion

Data redistribution is a critical phase for many distributed-cluster computing applications such as code coupling or computational steering ones. Scheduling the messages that have to be transmitted between the nodes can be performed by the transport layer of the network (TCP) or by the application (using the knowledge of the redistribution pattern).

In this paper we have studied the redistribution phase in the context of a low latency network (LLN), such as LAN. In this case, the reactivity of TCP is nearly perfect and the transport layer should give very good performance. This is the best-case scenario for the use of a brute force redistribution.

We have experimentally compared the performance of (O)GGP against the brute-force approach that consist in letting the transport layer manage the congestion alone on LLN. Suprisingly, results show that even if the reactivity of transport-layer is nearly perfect, (O)GGP always outperforms the brute-force approach.

In order to estimate whether or not a scheduled redistribution is necessary, we have provided a modelization of the behavior of TCP for the data redistribution problem as well as of scheduling algorithms proposed in early works and called GGP and OGGP. We have compared two ways of predicting the data redistribution time. One is based on an inferior bound easily derived from the redistribution pattern, the other is based on the analysis of the pattern and our modelization of the platform. Whereas the first way is shown to be accurate only for certain redistribution pattern, the second approach gives very accurate predictions for any redistribution pattern. Whatever the scheduling is performed at the application level or at the transport level, we are able to accurately predict the redistribution time.

In future works we want to incorporate this modeling into simulators in order to help the design of scheduling parallel tasks. Indeed, in this case communication between parallel tasks are often redistribution.

References

[1] F. Afrati, T. Aslanidis, E. Bampis, and I. Milis. Scheduling in switching networks with set-up delays. In *AlgoTel 2002*, Mèze, France, May 2002.

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

[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 Transactions on Communications*, 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] Johanne Cohen, Emmanuel Jeannot, and Nicolas 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] A. Ganz and Y. Gao. A Time-Wavelength Assignment Algorithm for WDM Star Network. In *IEEE INFOCOM'92*, pages 2144–2150, 1992.

[8] 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 Transactions on Communications*, COM-30(11):2475–2481, November 1982.

[9] 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.

[10] E. Jeannot and F. Wagner. Two Fast and Efficient Message Scheduling Algorithms for Data Redistribution through a Backbone. In *18th International Parallel and Distributed Processing Symposium*, page 3, April 2004.

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

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

[13] 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.