Geometrical interpretation for data partitioning and sorting on a grid architecture

Dominique Bernardi(1), Christophe Cérin (2), Hazem Fkaier (3), Mohamed Jemni (3) and Michel Koskas (4)
(1) Université Pierre et Marie Curie-Paris6
Théorie des nombres - Institut de Mathématiques de Jussieu, Paris, F-75005 France
(2) Université de Paris Nord - LIPN
UMR CNRS 7030, 99 avenue Jean-Batiste Clément, 93430 Villetaneuse - France
(3) Ecole Supérieure des Sciences et Techniques de Tunis, Unité de recherche UTIC
5, avenue Taha Hussein, B.P. 56, Bab Menara, Tunis – Tunisie
(4) Université de Picardie Jules Verne,
LAMFA, UMR CNRS 6140, 5 rue du moulin neuf, Amiens, F-80000, France
bernardi@math.jussieu.fr, christophe.cerin@lipn.univ-paris13.fr, hazem.fkaier@esstt.rnu.tn,
mohamed.jemni@fst.rnu.tn., michel.koskas@laria.u-picardie.fr

I-Introduction

Does a grid architecture can achieve performance similar to the 2005 Minute Sort[1] record: 116GB (100 bytes records with 10 bytes for the key) in one minute on 80 Itanium2, 2,520 SAN disks? To answer, we have to consider first the type of grid and second the type of algorithm that we have to deploy. In our work we develop new methods in order to control the execution time and the load balancing of each cluster node participating in a parallel sorting application. We deal with heterogeneity of CPU and network speeds [1,2]. In fact, partitioning is the key and sorting is just one application for the concept. In the paper, the partitioning step is based on geometrical interpretations to find a data partition schema offering good times both for computation and communication in an heterogeneous context.

II- Our motivating example

Consider first a single cluster and a homogeneous one. Initially, n data are distributed across the p processors proportionally to their speeds. In this case, each processor receives n/p data. This assumption describes the initial condition of the problem. An efficient parallel sorting schema on clusters (and even with heterogeneous processors) can be implemented in the following way:

- Each processor picks up representative values in the unsorted list. It sends the representative values to node 0;
- Node 0 sorts what it receives from the processors and it keeps p-1 pivots; it distributes the pivots to all the processors
- Each processor partitions its input according to the pivots and it sends p-1 portions to the others
- Each processor sorts the data it has.

One important problem in all environments (heterogeneous or homogeneous) is to select the "good number" and the "right values" of pivots, such that, the amount of data that processors receive are almost similar to the amount of initial data they have.

Initially data are distributed according to processors performances. If one processor is very fast among a two processors cluster, then the fastest processor has more data than the slow one. After redistribution, we assume that we will be also very close to this situation. Then the amount of data exchanged between the two processors will be very low. Otherwise, if the two processors have close speeds or let's say equal speeds, then initially they detain almost the same amount of data. In the communication phase each processor will send to other almost the half of what it has initially. Then during this phase, almost the half of the whole data will cross the link between the two processors.

We have developed a family of algorithms and MPI codes able to control the execution in the case of heterogeneity of the CPUs. After configuring for an homogeneous cluster (in our code, homogeneity is obtained by setting a vector with the same value representing the processor speeds), tuning and running on one cluster of the Grid'5000[2] project and on 200 processors, we obtained a performance of 150s to sort 116GB. More than half of this time is devoted to communication, about 20% to write on disks (Minute Sort requires that the final result is stored on disks), about 20% to do the in-core sorts and the remainder for selecting pivots.

The main tuning steps concern (a) the possibility to sort on keys only and to avoid the moves of whole the record; then to write on disks one record after one record and after retrieving the records (b) the possibility to multithread communication (thread safe MPI implementation is required). All these techniques do not improve the execution time in a significant way.

The reason is that the communication step involves near 1Tb of data and we have a 1GBb/s Ethernet, Opteron processors at 2Ghz and IDE disks. We are guessing that in order to achieve the record, we need 200 Opteron processors, a 10Gb/s low latency network and ultra wide SCSI disks. We note also

[1] See http://research.microsoft.com/barc/SortBenchmark/

[2] See http://www.grid5000.org

that the interconnection of another long distance cluster with the same configuration will not improve the result since the network speed between the cities is still a bottleneck.

Waiting for a new infrastructure, we are currently working at the algorithmic level. Let us introduce a heuristic to show how to split data in the case of a grid with two clusters and heterogeneous links.

III. The case of a 2 clusters system

We consider the case of a grid of two heterogeneous clusters, P_1 and P_2 having different speeds, respectively k_1 and k_2, linked by a network link allowing a full duplex transfer mode. Note that we consider that the "power" of a site is represented by, say the sum of the speeds of the processors.

In the first phase, we shall select among data a global pivot to partition data on the two sites. Then we assume that this pivot will partition the two sets of data with the same proportionality since data are supposed homogeneous and randomly distributed between the two sites. The situation is depicted on Figure 1.

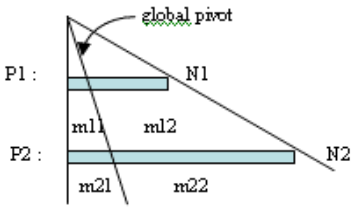

Fig1. Partitioning representation

It is obvious that m_{11}, m_{12}, m_{21} and m_{22} have correlated sizes. We consider the following matrix M that specifies what a site has at the beginning and what it sends to the other one:

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ a.m_{11} & a.m_{12} \end{bmatrix}$$

where $a = N_2/N_1$. We can also obtain the following relation: $m_{22} = N_2 - a.m_{11}$. m_{11} varies between 0 and N_1, while m_{22} varies between 0 and N_2. Figure 2 shows this relation. Then our estimated solution is a point of the segment of Figure 2.

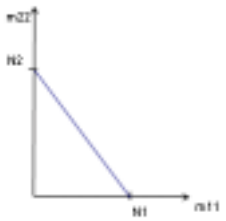

Fig 2. Distributions space

After communication, P_1 will have $m_{11}+m_{21}=N'_1$ to sort while P_2 will have $m_{12}+m_{22}=N'_2$ to sort. Our purpose is to find the values of N'_1 and N'_2 to minimize both processing and communications.
We consider, first, processing time. It is obvious that the parallel processing time is the maximum of the sequential processing time of all processors. Hence the optimal case is obtained when no processor is over-loaded, which is to say the two sites end processing at the same time.

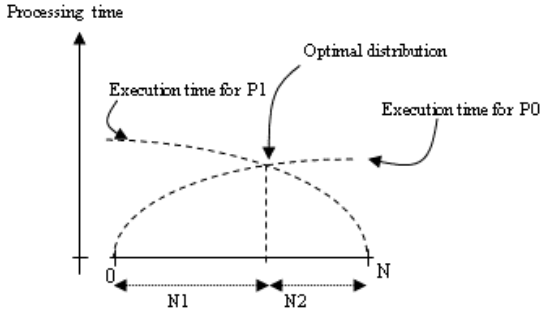

Fig 3. Processing time

Let's find now the distribution that minimizes communication time. We can model the communication time T_c between a processor p_i and a processor p_j with the equation $T_c = l_{ij} + d_{ij}.m$.
As we explained before, the more the two processors are balanced, the bigger the amount of exchanged data is. Subsequently, if the processors are "very" heterogeneous, the exchanged data will be very small. Then the communication graph will be a parabola-like (see Figure 4).

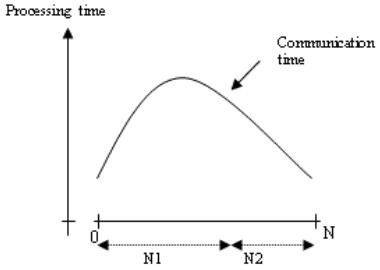

Fig 4. Communication time

If we suppose that the link is symmetric, then the communication time parabola will be symmetric too. We shall, now, merge the two solutions to find optimal distribution to have the best execution time. We shall consider in the same graph the sum of processing time and communication time (see Figure 5).

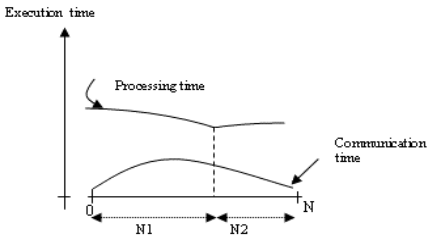

Fig 5. Execution time.

We do believe that the knowledge of the different cost functions makes the prediction of the optimal distribution easier.

IV. Master slaves approaches

The previous example suggests that we could have an organisation with two masters (one per site) in charge of distributing work to processors inside a site. The master slave paradigm has been extensively studied in the past for scheduling jobs and more recently for heterogeneous platforms [3,4].

The more relevant reference in our context is [5] because for the sorting application we deal with non linear cost function after the partitioning step. However, it appears that a polynomial needs to be solved to derive the solution and the order may be high. Deep investigation is still necessary to explain that running the construction of [5], locally on each master is benefit for the execution time and may approach the optimal solution, if it is known.

Conclusion

In this work, we investigated load balancing induced by data partitioning in a grid environment. The problem is to find the best partitioning making a compromise between processing time and communication time. Our approach is promising and we intend to run practical tests very soon to validate the heuristic depicted in section III.

Bibliography

[1] C. Cérin, M. Koskas, M. Jemni & H. Fkaier, Improving Parallel Execution Time of Sorting on Heterogeneous Clusters, 16th Symposium on Computer Architecture and High Performance Computing, Brazil, October 27-29, 2004.

[2] C. Cérin, H. Fkaier & M. Jemni, A Synthesis of Parallel Out-of-Core Sorting Programs on Heterogeneous Clusters, Third IEEE/ACM International Symposium on Cluster Computing and the Grid, Tokyo, 12-15 may 2003.

[3] Olivier Beaumont, Loris Marchal and Yves Robert, "Scheduling divisible loads with return messages on heterogeneous master-worker platforms", Report LIP RR-2005-21, May 2005. http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2005/RR2005-21.pdf

[4] Drozdowski, M. and Wolniewicz, P., "Out of Core Divisible Load Processing," IEEE Transactions on Parallel and Distributed Systems, vol.14, no. 10, Oct. 2003, pp. 1048-xx

[5] Jui-Tsun Hung and Thomas Robertazzi, “Distributed Scheduling of Nonlinear Computational Loads”, Conference on Information Sciences and Systems, The Johns Hopkins University, March 12–14, 2003