Load-balancing iterative computations on heterogeneous clusters with shared communication links

Arnaud Legrand, Hélène Renard, Yves Robert, and Frédéric Vivien

LIP, UMR CNRS-INRIA-UCBL 5668, École normale supérieure de Lyon, France
{Arnaud.Legrand,Helene.Renard,Yves.Robert,Frederic.Vivien}@ens-lyon.fr

Abstract. We focus on mapping iterative algorithms onto heterogeneous clusters. The application data is partitioned over the processors, which are arranged along a virtual ring. At each iteration, independent calculations are carried out in parallel, and some communications take place between consecutive processors in the ring. The question is to determine how to slice the application data into chunks, and assign these chunks to the processors, so that the total execution time is minimized. A major difficulty is to embed a processor ring into a network that typically is not fully connected, so that some communication links have to be shared by several processor pairs. We establish a complexity result assessing the difficulty of this problem, and we design a practical heuristic that provides efficient mapping, routing, and data distribution schemes.

1 Introduction

We investigate the mapping of iterative algorithms onto heterogeneous clusters. Such algorithms typically operate on a large collection of application data, which is partitioned over the processors. At each iteration, some independent calculations are carried out in parallel, and then some communications take place. This scheme encompasses a broad spectrum of scientific computations, from mesh based solvers to signal processing, and image processing algorithms. An abstract view of the problem is the following: the iterative algorithm repeatedly operates on a rectangular matrix of data samples. This matrix is split into vertical slices that are allocated to the computing resources. At each step of the algorithm, the slices are updated locally, and then boundary information is exchanged between consecutive slices. This geometrical constraint advocates that processors be organized as a virtual ring. Then each processor only communicates twice, once with its predecessor in the ring, and once with its successor. There is no reason to restrict to a uni-dimensional partitioning of the data, and to map it onto a uni-dimensional ring of processors. But uni-dimensional partitionings are very natural for most applications, and we show that finding the optimal one is already very difficult.

The target architecture is a fully heterogeneous cluster, composed of different-speed processors that communicate through links of different bandwidths. On the architecture side, the problem is twofold: (i) select the processors that participate in the solution and decide for their ordering (which defines the ring); (ii) assign communication routes between each pair of consecutive processors in the ring. One major difficulty of this ring embedding process is that some of the communication routes will (most probably) have to share some physical communication links: indeed, the communication networks of heterogeneous clusters typically are far from being fully connected. If two or more routes share the same physical link, we have to decide which fraction of the link bandwidth is assigned to each route. Once the ring and the routing have been decided, there remains to determine the best partitioning of the application data. Clearly, the quality of the final solution depends on many application and architecture parameters.

Section 2, is devoted to the precise and formal specification of our optimization problem, denoted as SharedRing. We show that the associated decision problem is NP-complete. Then, section 3 deals with the design of polynomial-time heuristics to solve the SharedRing problem. We report some experimental data in Section 4. Finally, we state some concluding remarks in Section 5. Due to the lack of space, we refer the reader to [6] for a survey of related papers.

2 Framework

2.1 Modeling the platform graph

Computing costs. The target computing platform is modeled as a directed graph $G = (P, E)$. Each node P_i in the graph, $1 \leq i \leq |P| = p$, models a computing resource, and is weighted by its relative cycle-time w_i: P_i requires w_i time-steps to process a unit-size task. Of course the absolute value of the time-unit is application-dependent, what matters is the relative speed of one processor versus the other.

Communication costs. Graph edges represent communication links and are labeled with available bandwidths. If there is an oriented link $e \in E$ from P_i to P_j, b_e denotes the link bandwidth. It takes L/b_e time-units to transfer one message of size L from P_i to P_j using link e. When several messages share the link, each of them receives a fraction of the available bandwidth. The fractions of the bandwidth allocated to the messages can be freely determined by the user, except that the sum of all these fractions cannot exceed the total link bandwidth. The eXplicit Control Protocol XCP [5] does enable to implement a bandwidth allocation strategy that complies with our hypotheses.

Routing. We assume we can freely decide how to route messages between processors. Assume we route a message of size L from P_i to P_j, along a path composed of k edges $e_1, e_2, \ldots, e_k$. Along each edge e_m, the message is allocated a fraction f_m of the bandwidth b_{e_m}. The communication speed along the path is bounded by the link allocating the smallest bandwidth fraction: we need L/b time-units to route the message, where $b = \min_{1 \leq m \leq k} f_m$. If several messages simultaneously circulate on the network and happen to share links, the total bandwidth capacity of each link cannot be exceeded.

Application parameters: computations. W is the total size of the work to be performed at each step of the algorithm. Processor P_i performs a share $\alpha_i.W$, where $\alpha_i \geq 0$ and $\sum_{i=1}^{p} \alpha_i = 1$. We allow $\alpha_j = 0$, meaning that processor P_j do not participate: adding more processors induces more communications which can slow down the whole process, despite the increased cumulated speed.

Application parameters: communications in the ring. We arrange the participating processors along a ring. After updating its data slice, each active processor sends a message of fixed length H to its successor. To illustrate the relationship between W and H, we can view the original data matrix as a rectangle composed of W columns of height H, so that one single column is exchanged between consecutive processors in the ring.

Let $\text{succ}(i)$ and $\text{pred}(i)$ denote the successor and the predecessor of P_i in the virtual ring. There is a communication path $\mathcal{S}_i$ from P_i to $P_{\text{succ}(i)}$ in the network: let $s_{i,m}$ be the fraction of the bandwidth b_{e_m} of the physical link e_m that is allocated to the path $\mathcal{S}_i$. If a link e_r is not used in the path, then $s_{i,r} = 0$. Let $c_{i,\text{succ}(i)} = \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$: P_i requires $H.c_{i,\text{succ}(i)}$ time-units to send its message of size H to its successor $P_{\text{succ}(i)}$. Similarly, we define the path $\mathcal{P}_i$ from P_i to $P_{\text{pred}(i)}$, the bandwidth fraction $p_{i,m}$ of e_m allocated to $\mathcal{P}_i$, and $c_{i,\text{pred}(i)} = \frac{1}{\min_{e_m \in \mathcal{P}_i} p_{i,m}}$.

Objective function. The total cost of one step in the iterative algorithm is the maximum, over all participating processors (whose set is denoted $\mathcal{P}$), of the time spent computing and communicating:

$$T_{\text{step}} = \max_{P_i \in \mathcal{P}} \left(\alpha_i.W.w_i + H.(c_{i,\text{pred}(i)} + c_{i,\text{succ}(i)}) \right).$$

In summary, the goal is to determine the best way to select q processors out of the p available, to assign them computational workloads, to arrange them along a ring, and to share the network bandwidth so that T_{step} is minimized.

2.2 The SharedRing optimization problem

Definition 1 (SharedRing(G,W,H)). *Given p processors P_i of cycle-times w_i and $|E|$ communication links e_m of bandwidth b_{e_m}, given the total workload W and the communication volume H at each step, minimize*

$$T_{step} = \min_{1 \leq q \leq p} \min_{\substack{\sigma \in \Theta_{q,p} \\ \sum_{i=1}^{q} \alpha_{\sigma(i)} = 1}} \max_{1 \leq i \leq q} \left(\alpha_{\sigma(i)}.W.w_{\sigma(i)} + H.(c_{\sigma(i),\sigma(i-1 \bmod q)} + c_{\sigma(i),\sigma(i+1 \bmod q)}) \right) \quad (1)$$

In Equation (1), $\Theta_{q,p}$ denotes the set of one-to-one functions $\sigma : [1..q] \to [1..p]$ which index the q selected processors that form the ring, for all candidate values of q between 1 and p. For each candidate ring represented by such a σ function, there are constraints hidden by the introduction of the quantities $c_{\sigma(i),\sigma(i-1 \bmod q)}$ and $c_{\sigma(i),\sigma(i+1 \bmod q)}$, which we gather now. There are $2q$ communicating paths: the path $\mathcal{S}_i$ from $P_{\sigma(i)}$ to its successor $P_{\text{succ}(\sigma(i))} = P_{\sigma(i+1 \bmod q)}$ and the path $\mathcal{P}_i$

from $P_{\sigma(i)}$ to its predecessor $P_{\text{pred}(\sigma(i))} = P_{\sigma(i-1 \bmod q)}$, for $1 \leq i \leq q$. For each link e_m in the interconnection network, let $s_{\sigma(i),m}$ (resp. $p_{\sigma(i),m}$) be the fraction of the bandwidth b_{e_m} that is allocated to the path $\mathcal{S}_{\sigma(i)}$ (resp. $\mathcal{P}_{\sigma(i)}$). We have the equations:

$$\begin{cases} 1 \leq i \leq q, \quad 1 \leq m \leq E, \quad s_{\sigma(i),m} \geq 0, \quad p_{\sigma(i),m} \geq 0, \quad \sum_{i=1}^{q} (s_{\sigma(i),m} + p_{\sigma(i),m}) \leq b_{e_m} \\ 1 \leq i \leq q, \quad c_{\sigma(i),\text{succ}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{S}_{\sigma(i)}} s_{\sigma(i),m}}, \quad c_{\sigma(i),\text{pred}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{P}_{\sigma(i)}} p_{\sigma(i),m}} \end{cases}$$

Since each communicating path $\mathcal{S}_{\sigma(i)}$ or $\mathcal{P}_{\sigma(i)}$ will typically involve a few edges, most of the quantities $s_{\sigma(i),m}$ and $p_{\sigma(i),m}$ will be zero. In fact, we have written $e_m \in \mathcal{S}_{\sigma(i)}$ if the edge e_m is actually used in the path $\mathcal{S}_{\sigma(i)}$, i.e. if $s_{i,m}$ is not zero (and similarly, $e_m \in \mathcal{P}_{\sigma(i)}$ if $p_{i,m}$ is not zero). Note that, when q and σ are known, the whole system of (in)equations is quadratic in the unknowns α_i, $s_{i,j}$, and $p_{i,j}$ (we explicit this system on an example in [6]).

From Equation (1), we see that the optimal solution involves all processors as soon as the ratio $\frac{W}{H}$ is large enough: then the impact of the communications becomes small in front of the cost of the computations, and the computations should be distributed to all resources. Even in that case, we have to decide how to arrange the processors along a ring, to construct the communicating paths, to assign bandwidths ratios and to allocate data chunks. Extracting the "best" ring seems to be a difficult combinatorial problem.

2.3 Complexity

The following result states the intrinsic difficulty of the SharedRing problem (see [6] for the proof):

Theorem 1. *The decision problem associated to the* SharedRing *optimization problem is NP-complete.*

3 Heuristics

We describe, in three steps, a polynomial-time heuristic to solve SharedRing: (i) the greedy algorithm used to construct a solution ring; (ii) the strategy used to assign bandwidth fractions during the construction; and (iii) a final refinement.

3.1 Ring construction

We consider a solution ring involving q processors, numbered from P_1 to P_q. Ideally, all these processors should require the same amount of time to compute and communicate: otherwise, we would slightly decrease the computing load of the last processor and assign extra work to another one (we are implicitly using the "divisible load" framework [6]). Hence (see Figure 1) we have for all i (indices being taken modulo q):

$$T_{\text{step}} = \alpha_i.W.w_i + H.(c_{i,i-1} + c_{i,i+1}). \quad (2)$$

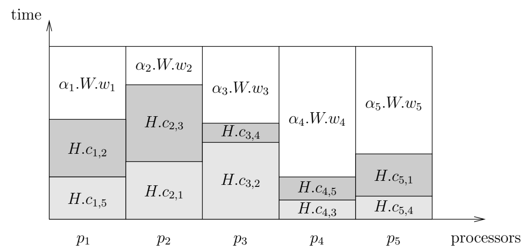

Fig. 1. Summary of computation and communication times with $q = 5$ processors.

Since $\sum_{i=1}^{q} \alpha_i = 1$, $\sum_{i=1}^{q} \frac{T_{\text{step}} - H.(c_{i,i-1}+c_{i,i+1})}{W.w_i} = 1$. With $w_{\text{cumul}} = \frac{1}{\sum_{i=1}^{q} \frac{1}{w_i}}$:

$$T_{\text{step}} = W.w_{\text{cumul}} \left(1 + \frac{H}{W} \sum_{i=1}^{q} \frac{c_{i,i-1} + c_{i,i+1}}{w_i} \right) \tag{3}$$

We use Equation (3) as a basis for a greedy algorithm which grows a solution ring iteratively, starting with the best pair of processors. Then, it iteratively includes a new node in the current solution ring. Assume we already have a ring of r processors. We search where to insert each remaining processor P_k in the current ring: for each pair of successive processors (P_i, P_j) in the ring, we compute the cost of inserting P_k between P_i and P_j. We retain the processor and pair that minimize the insertion cost. To compute the cost of inserting P_k between P_i and P_j, we resort to another heuristic to construct communicating paths and allocate bandwidth fractions (see Section 3.2) in order to compute the new costs $c_{k,j}$ (path from P_k to its successor P_j), $c_{j,k}$, $c_{k,i}$, and $c_{k,i}$. Once we have these costs, we can compute the new value of T_{step} as follows:

– We update w_{cumul} by adding the new processor P_k into the formula.
– In $\sum_{s=1}^{r} \frac{c_{\sigma(s),\sigma(s-1)} + c_{\sigma(s),\sigma(s+1)}}{w_{\sigma(s)}}$, we suppress the terms corresponding to the paths between P_i to P_j and we insert the new terms $\frac{c_{k,j}+c_{k,i}}{w_k}$, $\frac{c_{j,k}}{w_j}$ and $\frac{c_{i,k}}{w_i}$.

This step of the heuristic has a complexity proportional to $(p - r).r$ times the cost to compute four communicating paths. Finally, we grow the ring until we have p processors. We return the minimal value obtained for T_{step}. The total complexity is $\sum_{r=1}^{p} (p - r)rC = O(p^3)C$, where C is the cost of computing four paths in the network. Note that it is important to try all values of r, because T_{step} may not vary monotonically with r (for instance, see Figure 5).

3.2 Bandwidth allocation

We now assume we have a r-processor ring, a pair (P_i, P_j) of successive processors in the ring, and a new processor P_k to be inserted between P_i and P_j. Together with the ring, we have built $2r$ communicating paths to which a fraction of the initial bandwidth has been allocated. To build the new four paths involving P_k, we use the graph $G = (P, E, b)$ where $b(e_m)$ is what has been left by the $2r$ paths of the bandwidth of edge e_m. First we re-inject the bandwidths fractions used by the communication paths between P_i and P_j. Then to determine the four paths, from P_k to P_i and P_j and vice-versa:

– We independently compute four paths of maximal bandwidth, using a standard shortest path algorithm in G.
– If some paths happen to share some links, we use an analytical method to compute the bandwidth fractions minimizing Equation 3 to be allocated.

Then we can compute the new value of T_{step} as explained above, and derive the values of the α_i. Computing four paths in the network costs $C = O(p + |E|)$.

3.3 Refinements

Schematically, the heuristic greedily grows a ring by peeling off the bandwidths to insert new processors. To diminish the cost of the heuristic, we never recalculate the bandwidth fractions that have been previously assigned. When the heuristic ends, we have a q-processor ring, q workloads, $2q$ communicating paths, bandwidth fractions and communication costs for these paths, and a feasible value of T_{step}. As the heuristic could appear over-simplistic, we have implemented two variants aimed at refining its solution. The idea is to keep everything but the bandwidth fractions and workloads. Once we have selected the processor and the pair minimizing the insertion cost in the current ring, we perform the insertion and recompute all the bandwidth fractions and workloads. We can re-evaluate bandwidth fractions using a global approach (see [6] for details):

Method 1: Max-min fairness. We compute first the bandwidths fractions using the traditional bandwidth-sharing algorithm [1] maximizing the minimum bandwidth allocated to a path, then the α_i so as to equate all execution times (computations followed by communications), thereby minimizing T_{step}.

Method 2: Quadratic resolution. Once we have a ring and all the communicating paths, the program to minimize T_{step} is quadratic in the unknowns α_i, $s_{i,j}$ and $p_{i,j}$. We use the KINSOL library [7] to numerically solve it.

4 Experimental results

4.1 Platform description

We experimented with two platforms generated with the Tiers network generator [3]. Due to lack of space, and as the results are equivalent, we only report on the first platform. All results can be found in [6]. The Tiers generator produces graphs having three levels of hierarchy (LAN, MAN, and WAN). The platforms are generated by selecting about 30% of the LAN nodes (the boxed nodes in Figure 2) which are the computing nodes: the other nodes are simple routers. The processing powers of the computing nodes are randomly chosen in a list corresponding to the processing powers (evaluated using a LINPACK benchmark [2])

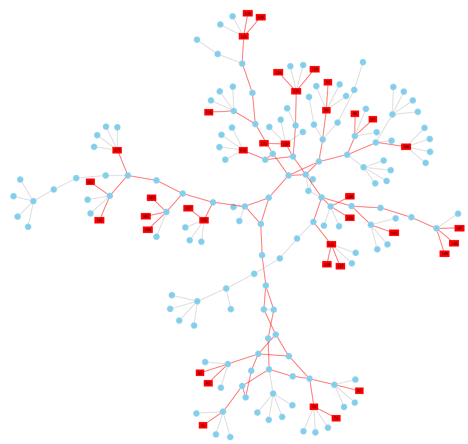

Fig. 2. Boxed nodes are computing nodes: there are 37 of them, connected through 47 routers, and 91 communication links.

of a wide variety of machines. The link capacities are assigned, using the classification of the Tiers generator (LAN, MAN, and WAN), with values measured by `pathchar` [4] between machines scattered in France, USA, and Japan.

4.2 Results

Figure 3 plots the number of processors used in the solution ring. As expected, this number decreases as the ratio H/W increases: additional computational power does not pay off the communication overhead. Figure 5 presents the normalized execution time as a function of the size of the solution ring for various communication-to-computation ratios: the optimal size is reached with fewer processors as the ratio increases. Finally, we try to assess the usefulness of the two variants introduced to refine the heuristic (Figure 4). Surprisingly enough, the impact of both variants is not significant: the best gain is 3%. Thus the plain version of the heuristic turns out to be both low-cost and efficient.

5 Conclusion

The major limitation to programming heterogeneous platforms arises from the additional difficulty of balancing the load. Data and computations are not evenly distributed to processors. Minimizing communication overhead becomes a challenging task. In this paper, the major emphasis was towards a realistic modeling of concurrent communications in cluster networks. One major result is the NP-completeness of the SharedRing problem. Rather than the proof, the result itself is interesting, because it provides yet another evidence of the intrinsic difficulty of designing heterogeneous algorithms. But this negative result should not be over-emphasized. Indeed, another important contribution of this paper is the

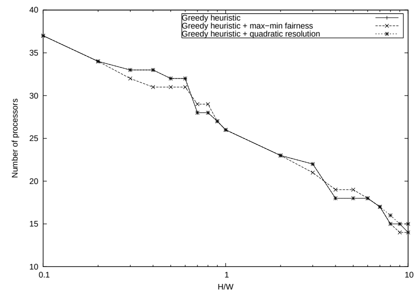

Fig. 3. Size of the optimal ring as a function of the ratio H/W.

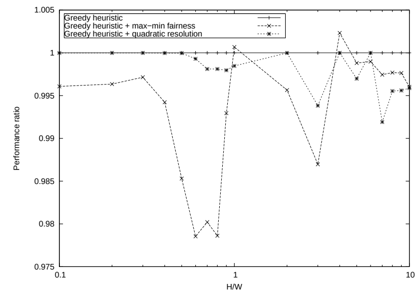

Fig. 4. Impact of the refinements on the quality of the solution.

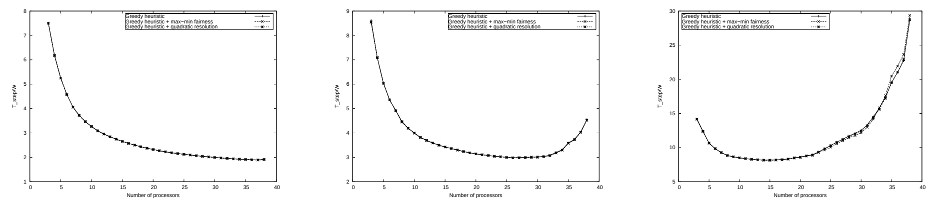

Fig. 5. Value of T_{step}/W as a function of the size of the solution ring, with a communication-to-computation ratio H/W equal from left to right to: 0.1, 1, and 10.

design of an efficient heuristic, that provides a pragmatic guidance to the designer of iterative scientific computations. Implementing such computations on commodity clusters made up of several heterogeneous resources is a promising alternative to using costly supercomputers.

References

1. D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, 1987.
2. R. P. Brent. The LINPACK Benchmark on the AP1000: Preliminary Report. In *CAP Workshop 91.* Australian National University, 1991. Website `http://www.netlib.org/linpack/`.
3. Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.
4. Allen B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.
5. D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM 2002*, pages 89–102. ACM Press, 2002.
6. A. Legrand, H. Renard, Y. Robert, and F. Vivien. Load-balancing iterative computations in heterogeneous clusters with shared communication links. Research Report RR-2003-23, LIP, ENS Lyon, France, April 2003.
7. A.G. Taylor and A.C. Hindmarsh. User documentation for KINSOL. Technical Report UCRL-ID-131185, Lawrence Livermore National Laboratory, July 1998.