On Optimization Criteria for Task Assignment in Cluster and Wide-Area Computing

Dmitriy Litvinov

V.N.Karazin Kharkov National University, Kharkov, Ukraine,
dimalit@yandex.ru

Abstract. Most of techniques proposed for task to processor mapping in distributed computing minimize communication cost that is calculated with model based on the quadratic assignment problem (QAP). I.e. to each pair of tasks and each pair of processors is assigned a cost, and then total cost of particular mapping is computed as the sum of products of the corresponding costs. However, shortcoming of such approach is that it cannot adequately address both bandwidth and contention considerations. Therefore, we propose an alternative communication cost model. It is based on the multicommodity flow (MCF) approach, and can directly compute communication time considering both link bandwidths and contention in the network. We evaluate our model through simulation using NPB Multi-Zone benchmarks on several irregular network topologies. Evaluation results show that MCF-based cost function has higher correlation with actual application run time than popular QAP-based cost functions. Thus, the usage of proposed cost model can potentially improve quality of solutions obtained with the task assignment algorithms.

Keywords: task assignment, wide-area computing, Grid computing, multicommodity flow, mapping problem

1 Introduction

One of the most widely used paradigm of programming multicomputers is "single process multiple data" (SPMD) programming. It assumes that application consists of multiple interacting processes or tasks which are assigned to processors and then executed. If the application is communication-limited rather then computation-limited its runtime heavily depends on network performance. In the case of distributed computing platform with irregular network topology, such as cluster or Grid, this means that application run time also depends on exact way how tasks are assigned (mapped) to the processors.

There exist a number of algorithms that optimize such mapping. Because of NP-hard nature of the problem these algorithms are mainly based on heuristics. In most of researches on this topic application is represented by a task interaction graph (TIG), which we will denote as $G(V, E)$. Nodes of this graph correspond to tasks, and edges represent intertask communication. Both nodes and edges can be weighted with node weight representing amount of computation done by the task and edge weight representing amount of data transferred between different task pairs. To describe the platform, another graph $G' = (V', E')$ is used. Vertices V' of this graph are processors. The graph is complete. Its weighted edges reflect cost of transferring a unit amount of data between every pair of processors. Sometimes nodes of this graph are also assigned weight which represents cost of performing a unit of computation on that processor. The goal is to find mapping from V to V' which minimizes some objective function π:

$$\pi : V \to V' | c(\pi) \to min \ .$$

Consider cost function $c(\pi)$. In general case it reflects both communication and computation costs. In this paper we focus on communication, and therefore will ignore cost asccociated with computation which generally belongs to a separate problem (load balancing). Usually communication cost is written as

$$c(\pi) = \sum_{(u,v)\in E} w(u,v) d(\pi(u), \pi(v)) \ , \tag{1}$$

where π is a mapping, w is (u, v) edge weight, and d is "distance" (cost of communication) between two processors (vertices of G'). This formulation represents an instance of the *quadratic assignment problem* and therefore such models and similar ones we will call *QAP-based models.*

Now consider distance function d mentioned in (1). It determines the "expence" of communication between each pair of processors. Clearly, it should assign larger costs to communications through slow links. In such a case it should be roughly inversely proportional to the bandwidth that network can deliver between a pair of processors (e.g. maximum flow value). But at the same time it should minimize contention. To account this, d can be made proportional to "hop count" distance between processors. Rationale behind this is that minimizing hop count that a message must travel we also minimize the probability of contention. An example of compromise between these two goals is "RTT" metric, which makes link cost proportional to round trip time (RTT) of a test packet between nodes in question. Clearly, RTT is both inversely proportional to the bandwidth between nodes and directly proportional to the number of links that a message should travel.

However, in either case QAP approach can measure contention only *indirectly* through its *probability.* But with applications where network is a system performance bottleneck it is unwise to model contention in such a rough way. Thus, in this paper we describe communication cost model which can directly account contention. In our model we consider multicommodity flows generated by pairs of processors and route them optimally, such as total communication time is minimized. This time is determined by a "bottleneck" links that have largest traffic/bandwidth ratio. Afterwards, this time is used as a cost of particular task-to-processor mapping. Such formulation is closely related to multicommodity flow feasible problem [1] and similarly can be formulated and solved as a linear program (LP).

2 MCF-Based Cost Function

2.1 Basic Notations

Let $G = (V, E)$ be a (directed) task interaction graph. Vertices of this graph V correspond to tasks and edges E represent intertask communication. Every edge is assigned weight w_{ij} which is amount of data transferred through it.

Let $G' = (V', E')$ be (directed) platform graph. Here vertex set V' represents platform nodes and edge set E' represents links between them. A node can contain zero or more processors. Such a way we uniformely model both compute nodes and network nodes (a.k.a. routers). Every link has bandwidth b_{ij}.

Now suppose we are given some assignment of task to processors (and consequently, to nodes). As the amount of communication between each pair of tasks is known, from these we can derive the amount of communication between every pair of nodes.

Let W be a set of (ordered) pairs of nodes having nonzero communication between them. Amount of data sent between pair $w \in W$ we denote r_w. Let P_w be a set of all possible (simple) paths between two nodes belonging to pair w and let $P = \bigcup P_w$. A routing algorithm used in the network somehow distributes data r_w between these paths. Denote the amount of data sent over path $p \in P_w$ as x_p. Obviously

$$\forall w \in W : \sum_{p \in P_w} x_p = r_w \ . \tag{2}$$

Now consider an arbitrary edge of the platform graph $(i, j) \in E'$. Amount of data that should be sent through it is as follows:

$$v_{ij} = \sum_{p \in P | (i,j) \in p} x_p \ ,$$

i.e. it equals to the total amount of data sent along all paths containing this edge. Minimum amount of time needed to transfer all these data through edge (i, j) is

$$t_{ij}^{min} = \frac{v_{ij}}{b_{ij}} \ . \tag{3}$$

To finish all communication we need to finish the most long lasting one. Therefore, the lower bound on total communication time can be estimated as follows:

$$T^{comm} \geq \max_{(i,j) \in E'} t_{ij}^{min} \ . \tag{4}$$

It can be shown that this value always can be achieved — i.e. for all communicating node pairs $w \in W$ their communication r_w can be performed within amount of time mentioned in (4). So

$$T^{comm} = \max_{(i,j) \in E'} t_{ij}^{min} \ . \tag{5}$$

The value of T^{comm} obtained in such way can be then used as a cost of a mapping. In contrast to QAP-cost it can directly measure contention. We call it

"MCF-cost" because all the flows $x_p \in P_w$ between different source-destination pairs $w \in W$ constitute a multicommodity flow (MCF) and T^{comm} is determined by a "bottleneck" edge of this flow.

2.2 MCF-cost Estimation in General Case

Note that equation (5) for communication time T^{comm} was derived based on the knowledge of the routing algorithm used in the network. Obviously, the value of T^{comm} can be easily estimated if the routing algorithm is static such as in case of tree-structured networks. However, if dynamic routing is utilized we cannot *a priori* know values of x_p which are needed for estimation of T^{comm}. Now we will derive the values x_p for such case too. We will assume that a) network uses dynamic multipath routing and b) that this routing works almost perfectly. In the context of mapping problem we say that routing works perfectly if it can deliver all the data in the shortest possible time.

Let $\boldsymbol{x}$ be a "routing" vector consisting of traffic volumes for all paths in P:

$$\boldsymbol{x} = \{x_p\}, p \in P \ .$$

Given this vector, the values of t_{ij}^{min} (3) and consequently T^{comm} (5) can be determined in a streightforward way. So, our aim is to find such routing vector $\boldsymbol{x}$ that gives

$$\min T^{comm} = T^{min} \ .$$

Let D be a set of all possible routing vectors that satisfy to the equation (2). It can be shown that D is a *convex polyhendron.* As such, any vector $\boldsymbol{x} \in D$ can be represented as a convex combination of vertices of D:

$$\boldsymbol{x} = a_1\boldsymbol{x^1} + a_2\boldsymbol{x^2} + \ldots + a_n\boldsymbol{x^n} \tag{6}$$

$$0 \le a_k \le 1 \ , \sum_{k=1}^{n} a_k = 1 \ ,$$

where $\boldsymbol{x^1} \ldots \boldsymbol{x^n}$ are vertices of the polyhedron D.

Suppose, we are given a routing vector $\boldsymbol{x}$. Assign to every edge (i,j) of graph G' a flow value of

$$F_{ij} = \frac{v_{ij}}{T^{comm}} \ ,$$

where T^{comm} can be found from (5). It can be shown that these flows always are feasible in the network G' and flow on path $p \in P$ $f_p = \frac{x_p}{T^{comm}}$.

As we did it for vector $\boldsymbol{x}$, construct from the path flows f_p vector $\boldsymbol{f} = \{f_p\}$ and similarly to (6) write:

$$\boldsymbol{f} = c_1\boldsymbol{x^1} + c_2\boldsymbol{x^2} + \ldots + c_n\boldsymbol{x^n} \ ,$$

where $c_k = \frac{a_k}{T^{comm}}$ and consequently $\sum_{k=1}^{n} c_k = \frac{1}{T^{comm}}$.

Thus, searching for such $\{c_k\}_{k=1}^{n}$ that $B = \sum_{k=1}^{n} c_k$ is maximized we can minimize T^{comm}.

Finally, recalling that flows $\{f_p\}$ must be feasible in G', we can write problem of minimizing T^{comm} as a linear program with respect to $\{c_k\}$:

$$\begin{array}{l} \text{Maximize} \\ \quad B = \sum_{k=1}^{n} c_k \\ \text{Subject to} \\ \quad \sum_{k=1}^{n} \sum_{p \in P | (i,j) \in p} c_k x_p^k \le b_{ij}, (i,j) \in E' \\ \quad c_k \ge 0, k = \overline{1,n} \ , \end{array} \tag{7}$$

where x_p^k denotes the volume of traffic that goes through path p in a routing vector $\boldsymbol{x}^k$.

3 Preliminary Experiments

To compare proposed MCF-based cost model with the "classical" QAP-based ones we conducted a series of experiments. Using simple heuristic we generated a number of "good" task-to-processor mappings. For each of these mappings we computed QAP-cost, MCF-cost and evaluated by simulation actual application execution time. As our simulation environment didn't support adaptive or multipath routing, we were forced to use static routing only — which is nevertheless common in modern platforms. Experiments showed that MCF-based cost function had higher correlation with actual execution time then QAP-based one.

3.1 Experimental Setup

We conducted our experiments for three different platforms:

1. Compute cluster with hierarchial communication topology and bottleneck at higher levels of the hierarchy ("cluster" topology).
2. Wide-area computing system where nodes are connected via a tree-structured network with homogeneous links ("tree" topology).
3. Wide-area computing system having cycles in the topology graph ("grid" topology).

All three platforms consisted of 16 compute nodes, each having power of 16 GFlops. So, the only difference between them was in interconnect. Topologies of platforms 2) and 3) were derived from the actual topology of wide-area computing system [2].

Using MSG framework of SimGrid [3] we implemented the model of NPB Multi-Zone [4] benchmarks. These benchmarks use overset-grid approach to solve discretized versions of the unsteady, compressible Navier-Stokes equations. With this approach complex domain is covered by a set of partially overlapping meshes or *zones.* Then the equations are solved independently in each zone, and after each iteration the zones exchange boundary values with their immediate neighbors with which they overlap.

With some number of zones assigned to it, each benchmark process repeatedly executes the following steps.

1. Post asynchronous receive request for next step boundary values.
2. Compute current time step using current boundary values.
3. Post asynchronous send reqest with boundary values for neighbouring zones.
4. Finish asynchronous receive requests (1) (synchronize).
5. Go to step 1.

In this model asynchronous communication is used to tolerate network latency and utilize computation/communication overlap.

NPB-MZ benchmarks are particularly suitable to run in a distributed environment because they have relatively low communication-to-computation ratio. At the same time, their communication pattern makes them sensitive to network contention and consequently, particularly appopriate for studying the mapping problem. There are three benchmarks in this suite: LU-MZ, SP-MZ, and BT-MZ. They have very similar communication patterns but while SP-MZ is the most simple and straightforward benchmark, LU-MZ has poor scalability and BT-MZ has additional load-balancing issues.

To make emphasis on communication cost rather than computation cost or other issues we present here results for SP-MZ banchmark only and force the number of tasks equal to the number of compute nodes. So, in all experiments we had one-to-one mapping of tasks to nodes and computation was perfectly balanced. For different mappings only communication produced change in total run time of the application.

For our experiments we used such configuration of the benchmark (class D, 16 tasks) that send relatively large messages which are relatively insensitive to link latencies. Communication matrices for the banchmark were obtained through measurements during test runs on a real cluster and computational complexity of each task was evaluated analitically using formulas from the paper [4].

For generation of "good" random mappings on which we evaluated our model we used simple randomized local search heuristics described in [5]. It begins with a random mapping. Then it choses a random pair of tasks, and if the exchange of their places gives a decrease to target function, it is performed. The algorithm stops when it cannot decrease target function in certain predefined number of tries. Afterwards, the procedure can be repeated with another initial mapping. For our purpuse, we ran the mapping heuristic 1000 times with the stop criterion being 500 tries without improvement of the target function.

3.2 Experimental Results

Figure 1 shows scatter plots with MCF-cost on x-axis and QAP-cost on y-axis for different mappings. To compute QAP-cost for "tree" and "grid" topologies we used RTT-metric, and for "cluster" topology we used "bandwidth"-metric.

It is clear that in all three cases there exists correlation between these two functions. This correlation is particularly high for "cluster" topology: points on the plot are arranged almost in a straight line. However, for two other topologies for each value of QAP-cost there exists some spread of the MCF-cost values. This

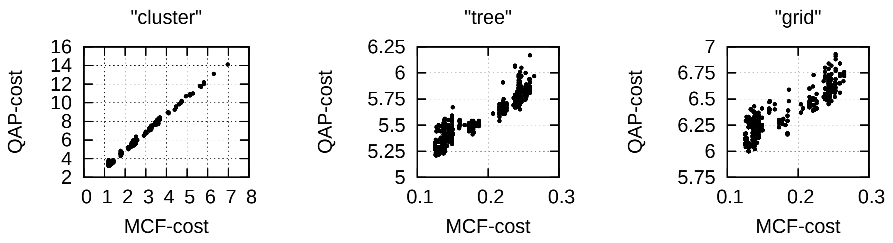

Fig. 1. Dependency of QAP-cost on MCF-cost for different topologies

means that even minimizing QAP-cost can often leave some space for improvement of the MCF-cost and, potentially, the run time of the application. Based on the above arguments, further experiments we conducted only for "tree" and "grid" topologies.

Fig. 2 depicts an example of how QAP-cost, MCF-cost and actual run time may change on successive iterations of the mapping heuristic (figure shows 10 runs (214 iterations) of the heuristic for "grid" topology).

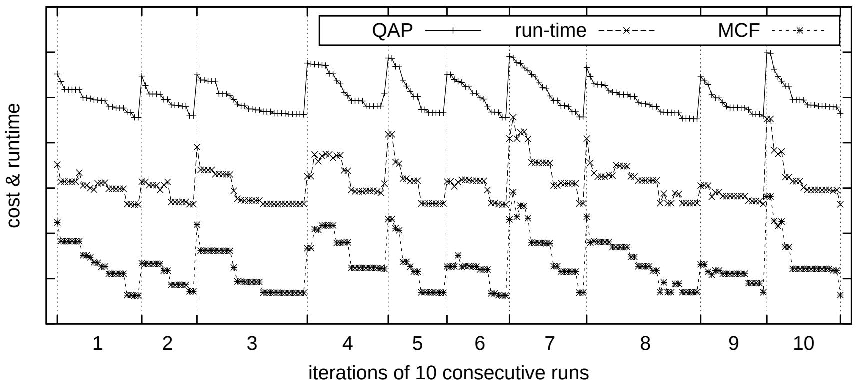

Fig. 2. Heuristic search iterations

While QAP-cost, being a target function of the optimization, constantly decreases, other two curves behave a bit differently. For example, at runs 4, 7, and 8 there are points where actual run time increases, and so does MCF-cost. Generally there are 44 points where QAP-cost goes "wrong" (in opposite direction from the run time). On the contrary, there exists only 15 points where MCF-cost makes such mistake.

To estimate the difference between QAP and MCF costs quantitively, we generated 1000 "final" mappings (i.e. where local search stops) and computed correlation coefficients between each cost function and actual run time of the application. Additionaly, we separately computed correlation for those mappings that had actual run time falling into best 10% of its full range. Scatter plots are shown on Fig. 3 and numerical resulsts are summarized in Table 1. Clearly, in terms of correlation, for both "tree" and "grid" topologies MCF-cost outperforms QAP-cost. The gain is especially notable for "top 10%" of the mappings which are, obviously, the most important.

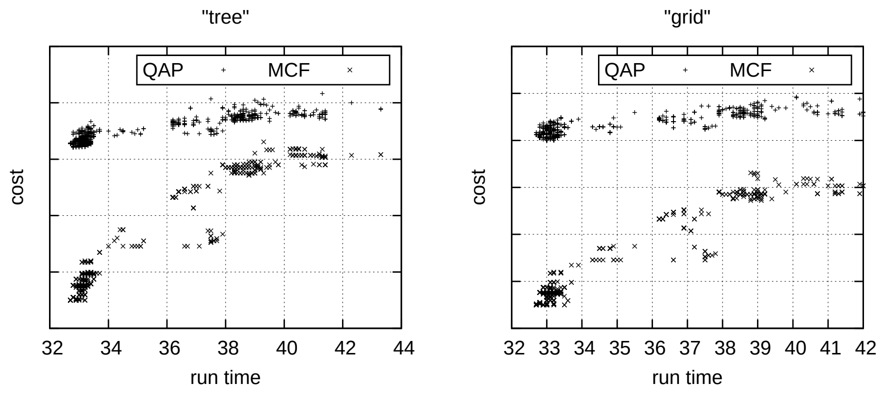

Fig. 3. Dependency between QAP-cost, MCF-cost, and actual run-time

Table 1. Summmary of correlation coefficients

	"tree" topology		"grid" topology	
	100%	top 10%	100%	top 10%
QAP-cost	0.91	0.38	0.86	0.36
MCF-cost	0.97	0.66	0.96	0.55

4 Conclusions and Future Work

Although QAP approach to communication cost estimation is the most widespread in the literature on the mapping problem, other approaches also can be found there. Particularly, in the paper [6] communication cost is estimated as the maximum *occupacy* over links of the platform graph. As the occupacy is defined as the ratio of (traffic) *load* to communication *capacity* of the link, this approach is exactly equivalent to formula (5). However, the paper [6]:

1. Doesn't prove the choise of used flow-based model of communication cost.

2. Considers only static routing and ignores possibility of adaptive and multipath routing.

In this paper we have shown experimentally the advantage of flow-based (MCF) models over traditional distance-based (QAP) models. Certainly, this advantage exists just as long as the messages that tasks send to each other are relatively long, so network latency can be neglected. This condition holds well for NPB-MZ benchmarks running on a medium-sized platform, which we used in our experiments. Also we have shown that for tree-structuted clusters having bottleneck at higher levels of the communication hierarchy MCF-based communication cost model is likely to be equivalent to traditional QAP-based ones and probably has no advantages over them.

Second, theoretical contribution of this paper is that it extends flow-based approach to evaluation of communication cost onto networks with adaptive and multipath routing — i.e. most of modern networks. Furthermore, expression for T^{comm} (7) derived here constitutes theoretical lower bound on the communication time and therefore can be used for evaluation of the efficiency of routing algorithms used in distributed computing platforms. This also suggests that the model presented here can be used for *improvement* of routing algorithms for use in this field.

Based on the conclusions presented just before, we anticipate the following directions of further research in the scope of this paper:

1. Experimental evaluation of proposed MCF-based model of communication cost on platforms that utilize modern adaptive multipath routing algorithms.
2. Development of a mapping technique that uses MCF-based cost model.
3. Extending routing algorithms to utilize the knowledge of structure of the application running on a computational platform with the goal of improvement of routing efficiency.

References

1. Hu, T.C.: Integer Programming and Network Flows. Addison-Wesley, Reading, MA (1970)
2. Saito, H.: Design and Implementation of Scalable High-performance Communication Libraries for Wide-area Computing Environments. Ph.D. thesis, University of Tokyo (2008)
3. Casanova, H., Legrand, A., Quinson, M.: SimGrid: A Generic Framework for Large-Scale Distributed Experiments. In: 10th International Conference on Computer Modeling and Simulation, pp. 126-131 (2008)
4. Van Der Wijngaart, R.F., Jin, H.: NAS Parallel Benchmarks, Multi-Zone Versions. NAS Technical Report NAS-03-010, NASA Ames Research Center, Moffett Field, CA (2003)
5. Orduna, J.M., Silla, F., Duato, J.: On the development of a communication-aware task mapping technique. Journal of Systems Architecture, Volume 50, Issue 4, pp. 207–220 (2004)
6. Taura, K., Chien, A.: A Heuristic Algorithm for Mapping Communicating Tasks on Heterogeneous Resources. In: 9th Heter. Computing Workshop, p. 102 (2000)