Resource Allocation for Clusters

Elise de Doncker, Laurentiu Cucos
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008
{elise,lcucos}@cs.wmich.edu

Yuqiang Guan
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
yguan@cs.utexas.edu

Keywords: Quadratic assignment, min-cut max-flow, Lagrangian dual, general Lagrange multiplier method, branch-and-bound.

Abstract

The mapping of processes to a network with hierarchical structure is considered. Variations of the resource allocation model result in 0-1 quadratic optimization problems for the minimization of the total data transfer time as a pseudo-boolean quadratic function. Computational and other low resource restrictions can be dealt with via penalty functions or additional constraints. We show that the problem has structure allowing special purpose solution methods for certain constraint sets.

1. INTRODUCTION

We consider the mapping of processes to a network with hierarchical structure, where the processors are grouped in clusters which may have different intra-cluster bandwidths. We combine the communication requirements of the processes with the bandwidth characteristics of the target network by minimizing the total transfer time. Variations of the resource allocation model result in 0-1 combinatorial optimization problems for the minimization of the total data transfer time as a pseudo-boolean quadratic function.

Section 2 introduces the resource allocation problem at hand. The task allocation models will be developed further in Section 3. Section 4 will discuss the corresponding 0-1 optimization problems and solution methods. Conclusions are given in Section 5. Target application areas include process allocation for multi-cluster computing or meta-computing.

2. RESOURCE ALLOCATION

As an architectural model, consider a number of processor clusters (generally with different bandwidths), all tied into the same network. Assume there are p processes to be assigned, and κ clusters. Let $\beta_{\ell_1\ell_2}$ be the required communication traffic (# bytes) between processes ℓ_1 and ℓ_2. Let τ^{ij} be the transfer time per byte between clusters i and j. Then the communication time between processes ℓ_1 and ℓ_2 is $\beta_{\ell_1\ell_2}\tau^{ij}$ if processes ℓ_1 and ℓ_2 reside in clusters i and j, respectively. Define indicator variable $x_\ell^j = 1$ if process ℓ belongs to cluster j and $x_\ell^j = 0$ otherwise, for $\ell = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, \kappa$, and let $\bar{x}_\ell^j = (\text{not})x_\ell^j$. The total transfer time to be minimized is $t(x) = \sum_i \sum_j \sum_{\ell_1} \sum_{\ell_2} \beta_{\ell_1\ell_2}\tau^{ij}x_{\ell_1}^i x_{\ell_2}^j$.

We denote the intra-cluster transfer time per byte within cluster j by $\tau^j = \tau^{jj}$. For the inter-cluster transfer time per byte we use $\tau_g = \tau^{ij}, i \neq j$ (which may be considered as an estimated average transfer time/ byte).

By requiring that each process is mapped to one cluster, the problem is to obtain an assignment x which solves

$$\begin{aligned} &\min_x t(x) \\ &\textstyle\sum_j x_\ell^j = 1, \quad 1 \le \ell \le p \\ &x_\ell^j \in \{0,1\}, \quad 1 \le j \le \kappa, \quad 1 \le \ell \le p \end{aligned} \tag{1}$$

The objective function which can be written as

$$\tau_g \sum_j \sum_{\ell_1} \sum_{\ell_2} \beta_{\ell_1\ell_2} x_{\ell_1}^j \bar{x}_{\ell_2}^j + \sum_j \sum_{\ell_1} \sum_{\ell_2} \beta_{\ell_1\ell_2}\tau^j x_{\ell_1}^j x_{\ell_2}^j,$$

can furthermore be put in the form of the pseudo-boolean quadratic function

$$t(x) = x^\tau Q x + y^\tau x, \tag{2}$$

where $x = (x_1^1, \ldots, x_p^1, \ldots, x_1^\kappa, \ldots, x_p^\kappa)^\tau$ and the matrix Q is upper triangular, and block diagonal with κ upper triangular blocks each of dimensions $p \times p$, i.e.

$$Q = \begin{pmatrix} Q_{11} & & & \\ & Q_{22} & & \mathbf{0} \\ \mathbf{0} & & \ddots & \\ & & & Q_{\kappa\kappa} \end{pmatrix}.$$

The jth upper triangular block Q_{jj} has zeros on the diagonal and the element $-2\beta_{\ell_1\ell_2}(\tau_g - \tau^j)$ at location (ℓ_1, ℓ_2) in the block. Q can also be given in symmetric form.

As a small example, for $\kappa = 2$, $p = 3$,

$$Q_{jj} = \begin{pmatrix} 0 & -2\beta_{12}(\tau_g - \tau^j) & -2\beta_{13}(\tau_g - \tau^j) \\ & 0 & -2\beta_{23}(\tau_g - \tau^j) \\ \mathbf{0} & & \\ & & 0 \end{pmatrix},$$

for $j = 1, 2$.

Denoting $x^j = (x_1^j, \ldots, x_p^j)^\tau$, the objective function is

$$t(x) = \sum_{j=1}^{\kappa} {x^j}^\tau Q_{jj} x^j + \tau_g \sum_{j=1}^{\kappa} \sum_{\ell_1=1}^{p} (\sum_{\ell_2=1}^{p} \beta_{\ell_1\ell_2}) x_{\ell_1}^j.$$

Note that the objective function as the total (sum, or average) transfer time reflects the total usage of the network,i.e., the extent to which network links/ channels are tied up by the application, loose from the actual point in time any transfer occurs, thereby allowing transfer overlaps (where transfers occur simultaneously on different network links). $t(x)$ does not depend on the underlying topology, and the processes and message passing may be synchronous or asynchronous. However, $t(x)$ does generally not represent the total communication time.

For example, consider a two-cluster system with $\tau^1 = 1$, $\tau^2 = 2$, $\tau_g = 100$ (using suitable units), and four processes to be assigned, with $\beta_{\ell_1\ell_2} = 0$ except $\beta_{12} = \beta_{21} = 100$, $\beta_{34} = \beta_{43} = 50$, and $\beta_{13} = \beta_{31} = 1$, characterizing an application with two "groups" of processes where each group requires a large amount of intra-group transfer, and little communication is needed in between the groups. Assuming further that no more than two processes can be assigned per cluster, then with respect to communication requirements it makes sense to assign the first two processes to the first cluster, and the other processes to the second.

3. TASK ALLOCATION MODELS

The objective function (2) can be augmented with a sum (or average) computational time function, for which a simple model is given by $t_c(x) = \sum_j \sum_\ell c_\ell^j x_\ell^j$, where c_ℓ^j is the computational requirement of process ℓ when run on cluster j (e.g., as a number of operations c_ℓ multiplied with the time/ operation). Alternatively, "penalty" weights can be incorporated to avoid overloading low capacity clusters (de Doncker and Kapenga, 1989), by incrementing the objective function with functions of the form $p_c(x) = \sum_j C_j \sum_\ell c_\ell x_\ell^j$ and $p_m(x) = \sum_j M_j \sum_\ell m_\ell x_\ell^j$, where c_ℓ and m_ℓ, $\ell = 1, 2, \ldots, p$ represent the computational and memory requirements, respectively, of process ℓ, and C_j and M_j determine computational and memory penalties, respectively, for cluster j. These modifications affect $t(x)$ linearly.

The problem of minimizing an unconstrained pseudo-boolean quadratic function of the form (2), although *NP* complete in general (Barahona et al., 1989), has an efficient algorithmic solution when the elements of Q are all nonpositive, which is satisfied by $t(x)$ assuming $\tau_g \geq \tau^j$; cf. the following property from (Picard and Ratliff, 1975). The problem

$$\min_x \{\sum_{j=1}^{n} y_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j\} \tag{3}$$

subject to

$$x_j \in \{0, 1\}, \quad 1 \leq j \leq n$$

where y_j and q_{ij} are real valued constants and $q_{ij} \leq 0$ for $1 \leq i \leq n$ and $1 \leq j \leq n$ can be solved efficiently by solving for a minimum cut on a related graph.

The constructions in (Picard and Ratliff, 1973; Picard and Ratliff, 1975) deliver a network with a set of non-negative capacities, so that the capacity through the cut equals the objective function, thereby reducing the 0-1 minimization problem to determining a cut with minimum capacity. The algorithm of Ford and Fulkerson or its improvements (Papadimitriou and Steiglitz, 1982) can then be used to obtain a minimum cut.

Taking the constraint of (1) into account leads to a problem of the form considered in (Billionnet et al., 1992). In the latter, a branch-and-bound method is used where, at each branch-and-bound node, bounds (for the primal problem) are obtained via the Lagrangian dual, which is mapped to a max-flow problem with an efficient solution.

The task allocation models given so far in this section will not, in general, guarantee a particular restriction on the number of processes assigned to any cluster, which would be necessary if, for example, the number of processes assigned to cluster j cannot exceed its number of processors (allowing for assigning at most one process per processor). With this type of constraint the quadratic program becomes

$$\begin{array}{l} \min_x t(x) \\ \sum_j x_\ell^j = 1, \quad 1 \leq \ell \leq p \\ \sum_\ell x_\ell^j \leq \nu_j, \quad 1 \leq j \leq \kappa \\ x_\ell^j \in \{0,1\}, \quad 1 \leq j \leq \kappa, \quad 1 \leq \ell \leq p \end{array} \tag{4}$$

assuming $p \leq \sum_j \nu_j$.

4. METHODS

We did some test runs with a "general" optimization code (for mixed integer programs) available from the NAG library (NAG, 1999). The algorithm uses a branch-and-bound technique which branches by splitting a variable range, and obtains bounds for the objective function by solving the corresponding continuous problem at each node. The problem specification is of the form $\min_x\{y^\tau x + \frac{1}{2}x^\tau Hx\}$, where each variable $x_i,\ 1 \leq i \leq n$, is specified to be in a given range and may be restricted to be integer, subject to a linear constraint set. The only restriction on H is to be symmetric. In general the method is only expected to deliver a global optimum when H is positive definite; yet for some applications a result corresponding to a local optimum may yield useful information.

For the small example at the end of Section 2, the optimal result was obtained using five branch-and-bound nodes, with the linear terms in the objective function replaced in view of $x_i = x_i^2$. For a test run of problem (4) in the original form with $p = 100, \kappa = 5, \nu_j = 20$, random $\beta_{ij} \in [0,1]$, $\tau_g \approx 45$ (chosen at random in [10,50]), random $\tau^j \in [1,10]$, an integer solution was found using 38 branch-and-bound nodes. However, this method is generally too time consuming for the problem at hand.

Alternatively, special purpose methods can be investigated, in order to exploit the structure of the objective function. *Generalized Lagrange multiplier* type methods solve a sequence of Lagrangian dual (relaxation) problems, searching the multiplier space to maximize the minimum Lagrange function. Indeed, it is well known that a solution minimizing the Lagrange function provides a lower bound on the original objective function. We thus seek to maximize the lower bound.

Using the definition of (Greenberg, 2001), the Lagrangian dual corresponding to $\min_{x\in X} f(x)$ under constraints $h(x) = 0$ and $g(x) \leq 0$ (primal) is

$$\max\{L^*(\lambda,\pi) : \lambda \geq 0\}$$

where

$$L^*(\lambda,\pi) = \inf_{x\in X}\{L(x,\lambda,\pi)\}$$

with

$$L(x,\lambda,\pi) = f(x) + \lambda g(x) + \pi h(x).$$

If L^* is reached at x^*, then

$$L^*(\lambda,\pi) - \lambda b - \pi c \leq \inf_{x\in X}\{f(x)|g(x) \leq b, h(x) = c\}.$$

Furthermore, $x^* \in \operatorname{argmin}\{L(x,\lambda,\pi) : x \in X\}$ implies $x^* \in \operatorname{argmin}\{f(x) : g(x) \leq b \text{ and } h(x) = h(x^*)\}$ for all b satisfying $(g(x^*) \leq b)$ and $(\lambda g(x^*) = \lambda b)$.

A generalized Lagrange multiplier method then incorporates a strategy of the form (Greenberg, 2001)

1. For (λ,π) with $\lambda \geq 0$, find $x \in \operatorname{argmin}\{L(x,\lambda,\pi) : x \in X\}$.
2. Terminate if 0 is in the convex hull of $S(x,\lambda,\pi) = \{(b,c) : g(x) \leq b, h(x) = c, \lambda g(x) = \lambda b\}$ (then x solves the original program). If no min generates 0 as a member of S, the right hand side (0) is in a duality gap.
3. Given (x,λ,π), determine (λ',π') to increase $L^*(\lambda,\pi)$.

(Everett, 1963) proposed a strategy of this form. (Greenberg, 1973) put the Lagrangian model in the framework of a generalized penalty function/ surrogate model. Considering the Lagrange multiplier method as a mapping from the space of the multipliers to the space of the constraint vectors, the duality gap arises from the fact that this mapping is not necessarily onto. This may leave inaccessible regions or *gaps* corresponding to (right hand side) constraint vectors which can never be generated (Everett, 1963). In (Billionnet et al., 1992), an approximate Lagrangian dual problem is solved to provide a lower bound; the duality gap is tested by solving a posiform equation at each branch-and-bound node.

5. CONCLUSIONS

The allocation of processes to a multi-cluster network was cast in the form of a 0-1 combinatorial optimization problem. It was shown that the sum data transfer time is minimized as a pseudo-boolean quadratic function $y^\tau x + x^\tau Qx$, where Q can generally be written in symmetric or upper-triangular form with negative coefficients.

Strategies are considered for solving the constrained problem, involving the solution of subproblems which can be solved efficiently (such as branch-and-bound combined with generalized Lagrange multiplier type methods). Using penalty functions to impose restrictions on low capacity clusters yields a problem to which the method of (Billionnet et al., 1992) can be applied. Modifications are needed for models which impose restrictions through additional constraint sets.

Future work is needed to run practical comparisons of the different models, to compare their efficiency for given problem sets. This task will be simplified by using branch-and-bound code which provides a general framework for incorporating different optimization problems.

A motivating application is that of the hierarchical implementation of PARINT (de Doncker et al., 1999), for the parallel/distributed computation of large sets of multivariate integrals. The processes are organized in groups, each of which solves a sequence of integration problems adaptively, i.e., using considerable communication between its worker processes (for example as a result of load balancing). The groups communicate with a global controller process (which may or may not belong to one of the groups) for exchanging problem data and results. However, there is no communication between the groups for the integral calculations. The groups are assigned statically at the beginning of the application.

Network computing is becoming important as more resources are available, and their use becomes scalable, fast and secure (Tavangarian, 2000).

References

Barahona, F., Jünger, M., and Reinelt, G. (1989). Experiments in quadratic 0-1 programming. *Mathematical Programming*, 44:127–137.

Billionnet, A., Costa, M. C., and Sutter, A. (1992). An efficient algorithm for a task allocation problem. *J. ACM*, 39(2):502–518.

de Doncker, E., Gupta, A., and Zanny, R. (1999). Large scale parallel numerical integration. *Journal of Computational and Applied Mathematics*, 112:29–44.

de Doncker, E. and Kapenga, J. (1989). Mapping asynchronous processes onto broadband networks. In *Proc. 2nd Int. Conf. on Graph Theory, Combin., Algorithms and Applications*, pages 151–161.

Everett, H. I. (1963). Generalized Lagrange multiplier method for solving problems of optimal allocation of resources. *Operations Research*, 11:399–417.

Greenberg, H. J. (1973). The generalized penalty function/ surrogate model. *Operations Research*, 21:162–178.

Greenberg, H. J. (2001). Math. prog. glossary. http://carbon.cudenver.edu/greenbe/˜glossary.

NAG (1999). Numerical Algorithms Group, C Library Mark 5.

Papadimitriou, C. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Picard, J. and Ratliff, H. (1973). A graph theoretic equivalence for integer programs. *Operations Research*, 21:261–269.

Picard, J. and Ratliff, H. (1975). Minimum cuts and related problems. *Networks*, 5:357–370.

Tavangarian, D. (2000). Local-area and wide-area high performance computing. In *Proceedings of High Performance Computing*, pages 193–200.