International Journal of Grid and Distributed Computing
Vol. 2, No. 4, December, 2009

Meshlization of Irregular Grid Resource Topologies by Heuristic Square-Packing Methods

Uei-Ren Chen[1], Chin-Chi Wu[2], and Woei Lin[3]

[1]*Department of Electronic Engineering, Hsiuping Institute of Technology*
[2]*Department of Information Management, Nan-Kai University of Technology*
[3]*Department of Computer Science, National Chung Hsing University*
wlin@nchu.edu.tw

Abstract

Traditionally, the assignment problem between tasks and processors is worked out by scheduling methods. In this paper, we have an innovative idea to construct a problem-solving environment for computational grids. To bridge the gap between the parallel processing technology and the heterogeneous computing environment, the grid computing resources are reorganized architecturally to form a virtual regular topology. In this research, we put emphasis on finding a heuristic algorithm of transferring an irregular grid resource topology into a mesh virtually.

Keywords: *Computational grid, Heuristic algorithm, Grid resource topology, Mesh.*

1. Introduction

Grid computing [3] is a new generation of computing paradigm that coordinates the use of resources, such as high performance computers, networks, databases, and scientific instruments owned and managed by multiple organizations. In comparison with the previous task-resource mappings by the scheduling algorithms, we have other ideas of constructing a problem-solving environment on computational grids. Figure 1 depicts the major difference between the task scheduling and our methodology. We first perform the transformation called meshlization of the resource topology selected from the computational grid, and then we assign tasks to the virtual mesh by using traditional parallel processing techniques, so as to solve the matrix computing problems [6]. From the aspect of virtual mesh, the concept is similar to the mesh-partition or the graph-partition problems [9] which consider how to divide a mesh or a graph into parts. Recent works on partitioning for heterogeneous environments include PART [1], JOSTLE [10], SCOTCH [8], and PaGrid [4]. Huang et al. have compared their PaGrid with above and shown the partitions with better quality [4]. In the proposed algorithm, transforming computational nodes into square blocks and permuting them in a virtual mesh, we translate the mesh-partition problem into a variant of the off-line two dimensional bin-packing problem [2] using m square items and a fixed-size bin. Leung et al. have proved that the problem of packing a set of squares into a square is NP-complete [5]. The difference between the bin-packing problem and ours are that: i) the square items in this research can be changed in size to pack in a bin; ii) our objective is neither maximizing the items packed in bins nor minimizing the number of bins for packing items; iii) we find the permutation with better computational and communicational abilities.

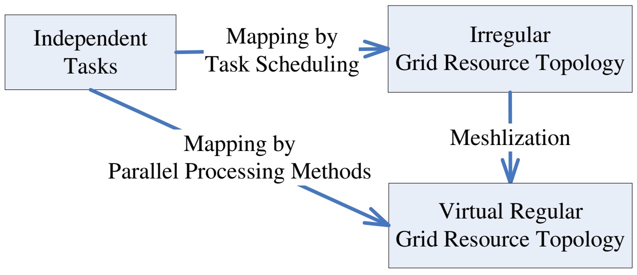
Independent Tasks
Mapping by Task Scheduling
Irregular Grid Resource Topology
Meshlization
Mapping by Parallel Processing Methods
Virtual Regular Grid Resource Topology

Figure 1. Comparison of Mappings

2. Proposed Algorithms

The computational grid includes a set of resources, network devices, and links used to connect them. Theoretically, the grid resource model can be represented as an undirected connected graph $G = (R, L)$, where R is the *resource set*, and L is the *link set*. The resource set consists of *computational nodes* and *routing nodes*. After selecting resources from the computational grid, an example of the grid resource topology is given in Figure 2 (a). The computational node may be a supercomputer, a cluster of workstations or computing devices practically and it is denoted as c with a parameter P_C called *computational ability*. The link can be defined with a parameter P_B which represents the *communication ability*.

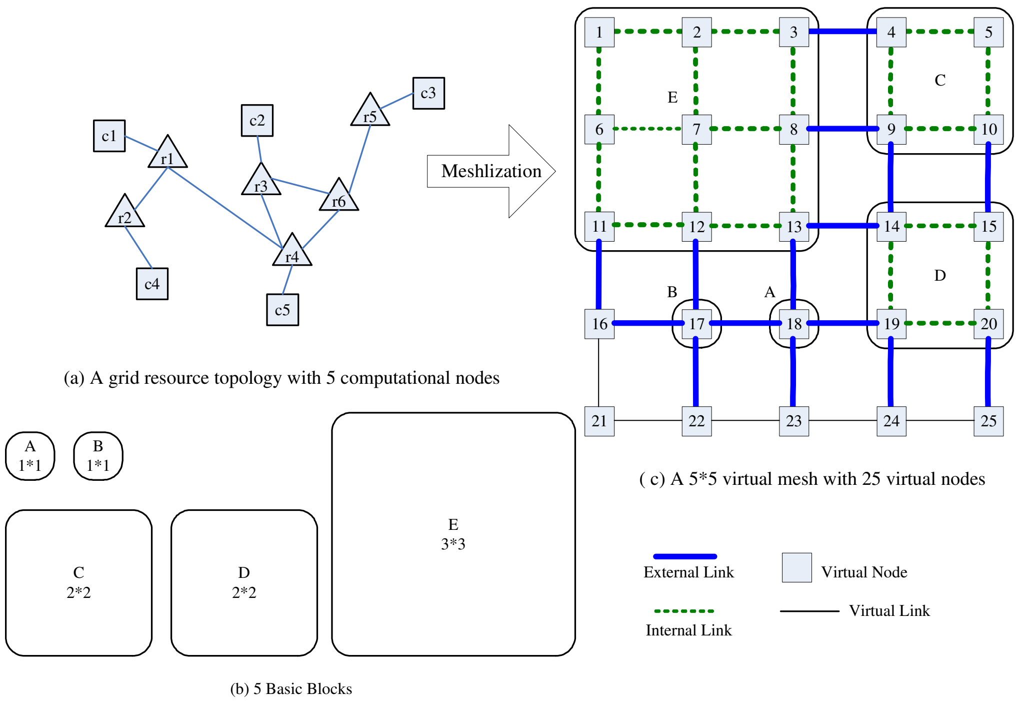

Figure 2. An Example of Meshlization

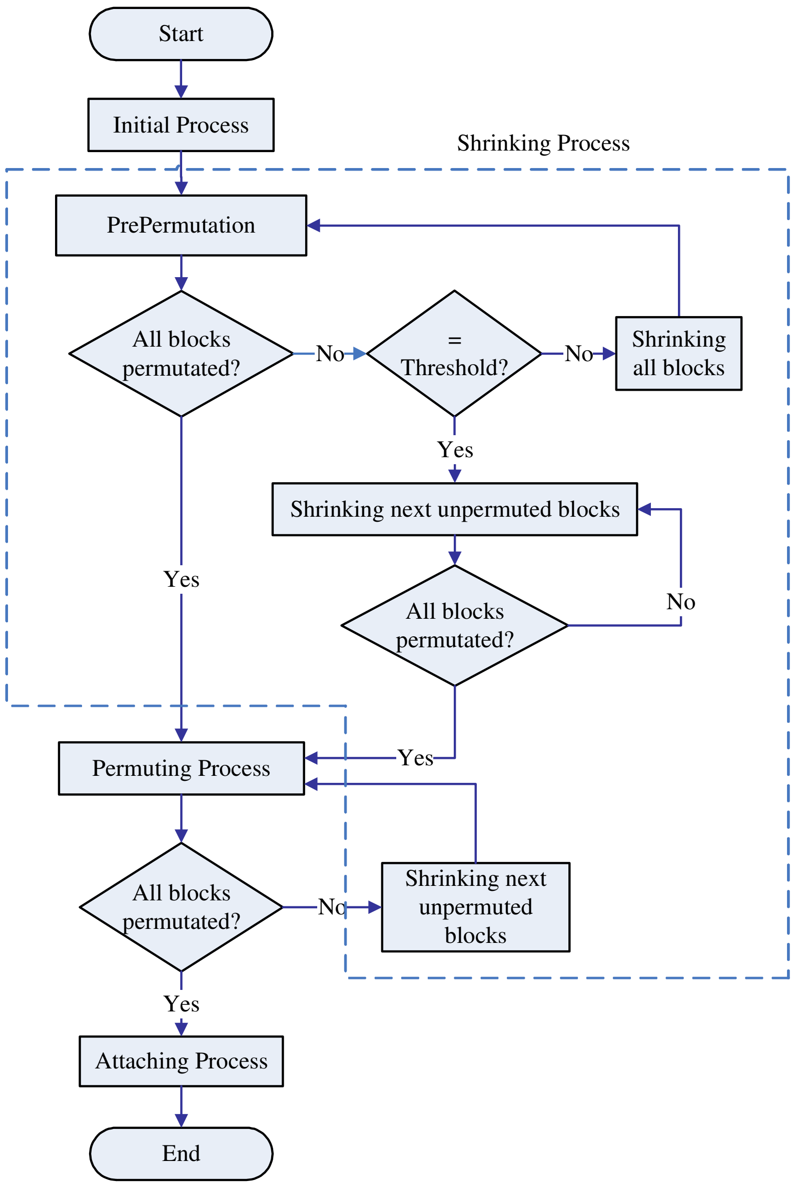

Figure 3. Flowchart of ISPA Algorithm

In our meshlization algorithm, the square basic block in Figure 2 (b) which represents a computational node is used to permute in a mesh. In Figure 2 (c), the edges can be turned into two kinds, that is, *internal links* and *external links* after placing the basic blocks in a virtual mesh. Internal links are those edges inside the same basic block or those edges that connect virtual nodes belong to the same computational node. External links are those edges that connect virtual nodes of basic blocks represent distinct computational nodes. Internal links have better communicational ability than external links because the messages are communicated in the same computational node. The reason of choosing the square block is that it owns the most number of internal links than other rectangulars under the same size. The input of this algorithm is a virtual mesh and a grid resource topology selected from the computational grid. The output is the mapping result from each computational node to the virtual nodes in the mesh. Figure 3 shows the flowchart of ISPA algorithms.

Initial Process. The ability of computational nodes selected from the grid resource topology is normalized in quantity. The minimal computational ability denoted PC_{min} will be normalized to one; other denoted $PC(c_i)$ for the computational node c_i will be set to the relative ratio nr_i of the minimal one.

$$nr_i = PC(c_i)/PC_{min} \tag{1}$$

The computational node is turned into a representative form of a basic block with size denoted a_i. The size of a basic block represents the number of virtual nodes in this block. We expand the size of basic block by Equation 2. The mesh size denoted N is the total number of virtual nodes in this mesh.

$$a_i = nr_i \Big/ \sum_i nr_i \times N \tag{2}$$

The side for each square basic block denoted s_i can be defined as follows.

$$s_i = \left\lfloor \sqrt{a_i} \right\rfloor \tag{3}$$

Shrinking Process. To check rapidly that all blocks can fit in the mesh, the prepermuation of basic blocks is performed by using the First-Fit-Decreasing algorithm [10]. Prepermutation will stop if it can successfully place all blocks in a mesh. If not all of basic blocks can be deposited in the virtual mesh, the shrinking process will be started up. A *shrinking threshold* is a ratio of the total size of permuted blocks to the mesh size. If the total size of permuted blocks is not larger enough to reach this threshold, the side of all square blocks will decrease by one, otherwise just shrinking the side of remaining blocks by one.

Permuting Process. To make the basic blocks join together, the permuting process will always place the succeeding blocks at the neighbor of previous ones. The first basic block is always placed at the top left corner of the mesh. A search tree can be produced by permuting a series of basic blocks in a mesh. If the search tree is fully expanded, the growth will be highly exponential. To reduce the complexity and find the best placement of basic blocks, a heuristic searching method called A^* search is used [11]. Three permuting policies are considered.

1. *Large Block First (LBF)*: The large basic block takes precedence over small ones and it will be selected to permute in the virtual mesh.

2. *Small Block First (SBF)*: The small basic block has higher priority than big one to be selected and permuted in the virtual mesh.

3. *Random*: The order of selecting basic blocks is made at random. The cost function $f(.)$ used by A^* search consists of two parts, $g(.)$ and $h(.)$.

$$f(.) = g(.) + h(.) \tag{4}$$

The function $g(.)$ can be defined as follow, where $x > 0$ is the total number of internal links, and $y > 0$ is the total number of external links while permuting from the first block to the current one.

$$g(.) = y\,/\,x \tag{5}$$

The *estimative function* $h(.)$ is defined as follows, where $x' > 0$ is the total number of internal links after setting the next block, and $y' > 0$ is the total number of external links.

$$h(.) = (y' + y) / (x' - x) \qquad (6)$$

By definition of the cost function $f(.)$, it has small value while the permutation of blocks in a mesh with small number of external links and large number of internal links. The object of A^* search is to find the solution with small cost.

Attaching Process. For the remaining nodes in the virtual mesh, we have the option of taking one from five policies mentioned below.

1. *Strong Node First (SNF)*: The remaining virtual nodes are attached to the block (represents a computational node) with the maximal computational ability shared by its virtual nodes. We assume that the ability of a computational node is shared by its virtual nodes evenly.

2. *Near Node First (NNF)*: The remaining virtual nodes are attached to the nearest basic block which has the minimal Euclidean distance between them.

3. *First Strong Then Near (FSTN)*: The block with powerful ability has higher attaching priority than the nearest one. The remaining virtual nodes are attached to the block with the maximal shared ability, but if the abilities of two nodes are equal, the near computational node will be attached.

4. *First Near Then Strong (FNTS)*: The nearest computational node will be attached, but if the Euclidean distances of two nodes are equal, the powerful node will be attached.

5. *Random*: Attach virtual nodes to the target randomly.

3. Experiment Results

In the simulation, the mesh size is 70×70. The grid resource topologies with different ratios of computational abilities are listed in Table 1. The communicational abilities in grid resource topologies are set to 100 for external links and 1000 for internal links. The shrinking threshold is set to 0.6.

Table 1. Computational Abilities of Grid Resource Topologies

Topologies	Computational Nodes				
	c_1	c_2	c_3	c_4	c_5
Topology1	500	1000	2000	2500	4000
Topology2	2000	2000	2000	2000	2000
Topology3	500	500	500	4250	4250
Topology4	525	525	2100	2100	4750

The *average computational abilities* $\overline{PC}$ is the optimal shared value for each virtual node and can be defined as follows.

$$\overline{PC} = \sum_{i=1}^{N} PC(c_i) \Big/ N \tag{7}$$

In Equation 8, *variance of computational abilities* evaluates the difference between the abilities of virtual nodes and the average computational ability. N is the total number of computational nodes and $PC(c_i)$ is the computational ability of the computational node c_i.

$$\sigma^2 = \sum_{i=1}^{N} \left[PC(c_i) - \overline{PC}\right] \Big/ (N-1) \tag{8}$$

Attaching Policies	Permuting Policies		
	LBF	SBF	Random
SNF	lb-s	sb-s	r-s
NNF	lb-n	sb-n	r-n
FSTN	lb-sn	sb-sn	r-sn
FNTS	lb-ns	sb-ns	r-ns
Random	lb-r	sb-r	r-r

Table 2. Abbreviations for Combinations of Permuting and Attaching Policies

The *average communicational ability* $\overline{PB}$ is the ratio of total communicational ability to total number of links. In Equation 9, $PB(l_{in})$ is the communicational ability of the internal link l_{in} and $PB(l_{ext})$ is that of the external link l_{ext}. Let N_{in} is the total number of internal links and N_{ext} is that for external links.

$$\overline{PB} = \left[\sum PB(l_{in}) + \sum PB(l_{ext})\right] \Big/ \left[N_{in} + N_{ext}\right] \tag{9}$$

Table 2 shows the abbreviations for combinations of permuting and attaching policies mentioned in Section 2. Table 3(a) shows the Strong Node First and Strong First Then Near policies can effectively reduce the variance of computational abilities in comparison with other attaching policies. Table 3(a) shows that the difference between the ratio of computational ability in the resource topology and its ratio of block size can influence the variance of computational abilities in a virtual mesh. Topology 2 and Topology 4 have the equivalent ratio of computational ability to their block sizes, but Topology 1 and Topology 3 are not. The results in Table 3(a) show that we can get the smaller variance of computational abilities while the ratio of computational ability is close to the ratio of block size such as Topology 2 and Topology 4 in Table 3(a). Moreover, the ISPA has the smaller variance than PaGrid, while the ratio of computational ability approaches the ratio of block size. For the attaching policies of the ISPA algorithm, the First Near then Strong or Near Node First policy can get less number of external links than others as shown in Table 3(b). The reason is that the First Near then Strong and Near Node First policies assign the neighbor nodes in the mesh to the same computational node so as to reduce the number of external links. In Table 3(b), the ISPA has less number of external links than PaGrid while using the Near Node First and First Near Then Strong attaching policies. As shown in Table 3(c), the average communicational ability of the First Near then Strong or Near Node First is greater than other attaching policies, because they have a larger number of internal links and less number of external links. Table 3(c) shows that the ISPA algorithm with the Near Node First and First Near Then Strong attaching policies can have greater communicational ability than PaGrid.

Table 3. Performance of the 70 × 70 Mesh for Test Topologies

(a) Variance of Computatonal Abilities

Topology #	lb-sn	lb-ns	lb-s	lb-n	lb-r	sb-sn	sb-ns	sb-s	sb-n	sb-r	r-sn	r-ns	r-s	r-n	r-r	PaGrid
1	0.07	0.56	0.07	1.04	0.40	0.07	0.81	0.07	0.81	0.38	0.07	0.59	0.07	0.22	0.40	0.0003
2	≈ 0	0.34	≈ 0	4.39	1.61	≈ 0	0.34	≈ 0	4.39	1.68	≈ 0	1.97	≈ 0	4.39	1.94	0.0002
3	0.30	1.14	0.30	1.25	1.64	0.30	0.62	0.30	0.56	1.62	0.30	1.08	0.30	0.65	1.60	0.003
4	≈ 0	0.15	≈ 0	0.88	0.10	≈ 0	0.43	≈ 0	0.61	0.09	≈ 0	0.37	≈ 0	0.61	0.10	0.0014

(b) Number of External Links

Topology #	lb-sn	lb-ns	lb-s	lb-n	lb-r	sb-sn	sb-ns	sb-s	sb-n	sb-r	r-sn	r-ns	r-s	r-n	r-r	PaGrid
1	425	220	683	192	1277	214	157	673	204	1285	235	177	458	204	1278	323
2	344	202	406	241	1695	344	202	406	241	1681	344	202	406	241	1711	416
3	223	203	261	246	1761	217	162	261	256	1779	242	196	258	245	1796	336
4	393	228	1885	225	2263	522	135	2035	163	2277	459	171	1893	163	2249	340

(c) Average Communicational Ability

Topology #	lb-sn	lb-ns	lb-s	lb-n	lb-r	sb-sn	sb-ns	sb-s	sb-n	sb-r	r-sn	r-ns	r-s	r-n	r-r	PaGrid
1	957	978	931	981	871	979	984	932	980	872	978	982	953	980	869	967
2	965	980	959	976	827	965	980	959	976	830	964	980	959	979	830	957
3	978	980	974	975	823	978	984	974	974	821	976	980	974	975	819	965
4	961	977	810	977	772	948	986	795	984	771	954	983	810	984	774	965

(d) Branch Amount Visited by A^*

Topology #	lb-sn	lb-ns	lb-s	lb-n	lb-r	sb-sn	sb-ns	sb-s	sb-n	sb-r	r-sn	r-ns	r-s	r-n	r-r
1	16375	16375	16375	16375	16375	13799	13799	13799	13799	13799	8492	18554	18554	17870	8492
2	19358	19358	19358	19358	19358	19358	19358	19358	19358	19358	27058	23710	27058	19358	19358
3	27515	27515	27515	27515	27515	17092	17092	17092	17092	17092	25709	25709	28270	31978	21817
4	25544	25544	25544	25544	25544	14641	14641	14641	14641	14641	28468	28509	23598	14641	21241

The branch amount in Table 3(d) shows that the A^* search can massively reduce the complexity of block permutation. For example, the branch amount of complete search tree is about 10^8 for permuting five blocks in a 70×70 mesh according to our estimation. For test topologies in Table 3(d), their branches actually travelled by A^* search are less than 32000. This result shows that there are above ninety percent of branches reduced by A^* search. Table 3(d) shows that the precedence of permuting basic block has a massive impact on the branch amount of search trees while the difference of the block size is enormous. For instance of Topology 3 in Table 3(d), Small Block First has less branch amount than other permuting policies while the ratio difference of the computational ability and the block size is massive.

4. Conclusions

In this paper, we propose a heuristic square-packing algorithm to transfer a grid resource topology into a virtual mesh. Our experiment results show that the ISPA algorithm can evenly distribute the computational ability of the grid resource to the virtual mesh. Because having more internal links, the virtual mesh gets larger average communicational ability than PaGrid.

Acknowledgement

We are grateful to the National Center for High-performance Computing (NCHC) in Taiwan, R.O.C. for computer time and facilities.

References

[1] J. Chen and V. E. Taylor, "Mesh partitioning for efficient use of distributed systems," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 1, 2002, pp. 67–79.

[2] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," Approximation Algorithms for NP-hard Problems, 1997, pp. 46–93.

[3] I. Foster and C. Kesselman, Eds., The Grid: Blueprint for a New Computing Infrastructure , 1st ed. San Francisco, CA, USA: Morgan-Kaufman, July 1998.

[4] S. Huang, E. Aubanel, and V. C. Bhavsar, "PaGrid: A mesh partitioner for computational grids," Journal of Grid Computing, vol. 4, no. 1, 2006, pp. 71–88.

[5] J. Y.-T. Leung, T. W. Tam, and C. S. Wong, "Packing squares into a square," Journal of Parallel and Distributed Computing, vol. 10, 1990, pp. 271 – 275.

[6] J. J. Modi, Parallel Algorithms and Matrix Computation. Clarendon Press, 1988.

[7] J. Pearl, Heuristics: intelligent search strategies for computer problem solving. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[8] F. Pellegrini and J. Roman, "Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs," Lecture Notes in Computer Science, vol. 1067, 2006, pp. 493–498.

[9] K. Schloegel, G. Karypis, and V. Kumar, Sourcebook of Parallel Computing. Morgan Kaufmann, 2003, pp. 491–541.

[10] C. Walshaw and M. Cross, "Multilevel mesh partitioning for heterogeneous communication networks," Future Generation Computer Systems, vol. 17, no. 5, 2001, pp. 601–623.

[11] A. C. C. Yao, "New algorithms for bin packing," Journal of the ACM, vol. 27, no. 2, 1980, pp. 207–227.