Characterization of Communications between Processes in Message-Passing Applications *

J. M. Orduña, V. Arnau
Departamento de Informática
Universidad de Valencia
E-mail: `Juan.Orduna@uv.es`

José Duato
D.I.S.C.A.
Universidad Politécnica de Valencia
E-mail: `jduato@gap.upv.es`

Abstract

Many research activities have focused on the problem of task scheduling in heterogeneous systems from the computational point of view. However, an ideal scheduling strategy would also take into account the communication requirements of the applications and the communication bandwidth available in the network. One of the major problems to be solved in the development of this scheduling strategy is precisely the measurement of the communication requirements for each application.

In this paper, we propose a clustering-based method to characterize the communications between processes generated by message-passing applications. This technique provides a model consisting of several partitions of the processes generated by the application. Also, we propose a criterion to measure the quality of the obtained partitions. This approach can be used when a given application is repeatedly executed with different input data. Results show that the proposed method can provide a partition with the highest ratio between the intracluster and the intercluster required communication bandwidth. This partition can be used to map groups of processes to processors in the heterogeneous system.

1 Introduction

In order to fully exploit the computing power of heterogeneous systems, a lot of research has focused on solving the *NP-complete* problem of efficiently scheduling diverse groups of tasks to the machines that form the system [14, 10, 7, 11, 15]. Nevertheless, these proposals only focus on computing power requirements, and they do not consider communication cost, thus assuming that the communication subsystem provides enough bandwidth in any case. However, as the computing power of new processors increases, the interconnection network in these heterogeneous systems may become the system bottleneck. In this case, the scheduler should also consider the estimated communication cost between processes. Given an heterogeneous system (that may be formed by different groups of interconnected homogeneous systems) and given a certain set of different (parallel or sequential) applications from different users, an ideal scheduling strategy would map the processes to processors taking into account both the computing and the communication requirements of the applications running on the machine. The scheduler would choose either a computation-aware or a communication-aware task scheduling strategy depending on the kind of requirements that leads to the system performance bottleneck.

In order to develop a communication-aware task scheduling strategy for parallel applications on heterogenous systems, several problems must be solved. First, the communication requirements of the applications running on the machine must be measured or estimated. On the other hand, the available network resources must also be characterized. Additionally, some criterion is needed to measure the suitability of each allocation of network resources to each one of the parallel applications, according to their communication requirements. Based on this criterion, some mapping technique based exclusively on the communications requirements should be developed. Finally, this technique must be integrated with process scheduling, in order to be used when the communication requirements are the ones that lead to the system performance bottleneck.

In previous papers, we proposed a model of communication cost that provides a characterization of the network resources of any given irregular topology [1]. Also, we proposed a clustering method to provide a network partition adapted to the communication requirements of the applications running on the machine [2], a criterion to measure the suitability of each allocation of network resources to each of the parallel applications [2, 12], and a mapping technique based exclusively on the communication requirements[12] . Due to the complexity of estimating the communication

*Supported by the Spanish CICYT under Grant TIC97–0897–C04–01

requirements of the applications, in these proposals we left this task as future work, and some simplified assumptions were applied. In this paper, we propose the experimental evaluation of the communication requirements of message-passing parallel applications. Although this approach does not completely solve the problem (because it requires the execution of the application to measure the communication cost), it will be very useful when parallel applications are repeatedly executed with different input data. Also, we propose a clustering approach to identify the clusters of processes that require most of the communication bandwidth. This approach can provide a partition of the processes generated by the application into clusters that show a high ratio between the intracluster and the intercluster required bandwidth. Thus, this partition can be used to obtain an efficient mapping of processes to processors.

The rest of the paper is organized as follows: Section 2 describes the execution environment, the benchmarks used as message-passing parallel applications, and the data structures extracted as a result from the processing of the execution traces. Section 3 shows the proposed clustering approach to be performed on these data structures in order to characterize the communication requirements of the applications. Section 4 shows the evaluation results obtained with the proposed method. Section 5 presents some considerations about the obtained results. Finally, Section 6 presents some concluding remarks.

2 Execution Environment and Applications

In order to evaluate the communication requirements of real message-passing parallel applications, we have executed several parallel benchmarks in a simulated network of workstations using the MPI message-passing standard. Each parallel benchmark creates N processes that communicate between them by message passing, were N is an input parameter set by the user. We have used the MPICH portable implementation of the MPI Message-Passing Standard [8]. This implementation allows us to simulate several machines of a given architecture in a single machine [9]. Additionally, it can be set to provide execution traces of each MPI call. We have executed the CG, EP, IS, LU, MG and SP NAS Parallel Benchmarks 2.0 [3]. While SP benchmark requires N to be a square number, the rest of the benchmarks require N to be a power of 2. Each benchmark has been executed with a number of processes ranging from 8 (9 in the case of SP) to 64.

The execution traces of the benchmarks show that most of the communication between processes is performed through MPI point-to-point communication calls. Only some benchmark traces contain a few MPI collective communication calls, that in no case reach 0.5 % of the total MPI communication calls. Therefore, we have only considered point-to-point communications.

Starting from the execution traces, we have constructed a *table of communication* between processes. This table contains $N \times N$ elements, were N is the number of processes that each benchmark has generated. Each element T_{ij} represents the number of messages that process i has sent to process j. As an example, Table 1 shows the table of communication between processes obtained for benchmark CG configured for eight processes. In this table it can be seen that, for example, process 0 has sent 1266 messages to process 1 (element T_{01} shows the value 1266).

No.	0	1	2	3	4	5	6	7
0	416	1266	1266	0	2	0	0	0
1	1267	416	0	1266	0	2	0	0
2	1267	0	0	1266	416	0	2	0
3	0	1266	1267	0	0	416	0	2
4	3	0	416	0	0	1266	1266	0
5	0	2	0	416	1267	0	0	1266
6	0	0	2	0	1267	0	416	1266
7	0	0	0	2	0	1266	1267	416

Table 1. Table of communication between processes for CG benchmark configured for 8 processes

Although this table does not have the same properties as the table of distances defined in [1] (the table of distances between processes is not symmetrical and does not always contain a zero diagonal), a clustering method is applicable here in order to find the groupings of processes with the highest bandwidth requirements.

3 Characterization of the Communication Requirements

In order to characterize the communication requirements of the considered applications, we have applied a hierarchical agglomerative clustering method to the table of distances between processes, obtaining a *dendogram*. In particular, we have applied the furthest-neighbor algorithm [5, 6] to compute the optimal dendogram. This algorithm uses a similarity measure. In each step the algorithm merges two of the existing clusters into a new one, choosing the two clusters that result in the lowest similarity measure when the step is applied. The similarity measure usually used in this algorithm is the intracluster distance, and therefore it is called the furthest-neighbor algorithm. However, we have considered a different similarity measure f. Let cluster A be formed by x processes $a_1, a_2, \cdots, a_x$, and let cluster B be formed by y processes $b_1, b_2, \cdots, b_y$. Then, we have defined f as

$$f = \frac{\sum_{i=1}^{x}\sum_{j=1}^{y} T_{a_i b_j}}{x\,y} \quad (1)$$

where T_{rs} is the amount of messages exchanged between process r and process s in the table of communication between processes. In each step of the algorithm, the next two clusters to be merged into one larger cluster will be A and B if this pair of clusters provides the maximum value for f. The initial partition consists of N clusters of one process each, and therefore the table of communication between processes contains the value of f for each possible cluster of two processes. Thus, the first step of the algorithm simply consist of merging the two processes with the highest value in the table of communication between processes. In each step a new partition is formed, decreasing the number of clusters by one. The algorithm ends when all the processes are grouped into a single cluster.

However, the main problem with this method is the large amount of identical values in the table of communication between processes. For example, consider Table 1. In this table there are 6 pairs of nodes that exchange 2533 messages, resulting in 6 different possible optimal partitions with 7 clusters. Additionally, the range of valid solutions for the first step increases in the following steps, providing a solution tree that grows exponentially with respect to the number of processes, N. Thus, an additional criterion is required to select among solutions with the same f value. For solving this problem, we have performed a random sorting of the clusters with the highest value for f. Then, the furthest-neighbor clustering algorithm [5] is computed using the table of communication between processes. In each step of this algorithm, if there exist two or more clusters with the same highest f value, we select the cluster in such a way that it contains the first node in the sorted list. Table 2 shows the obtained dendogram for the table of communications between processes shown in Table 1.

Partition 0:	(0) (1) (2) (3) (4) (5) (6) (7)
Partition 1:	(0,2) (4) (6) (1) (5) (3) (7)
Partition 2:	(0,2) (4,6) (1) (5) (3) (7)
Partition 3:	(0,2) (4,6) (1,3) (5) (7)
Partition 4:	(0,2) (4,6) (1,3) (5,7)
Partition 5:	(0,2,1,3) (4,6) (5,7)
Partition 6:	(0,2,1,3) (4,6,5,7)
Partition 7:	(0,1,2,3,4,5,6,7)

Table 2. Dendogram provided by the furthest-neighbor algorithm

The obtained dendogram hierarchically shows which are the groupings of processes that exchange more messages for each partition. That is, it shows the groupings of processes with the highest communication requirements. However, the dendogram by itself does not provide any information about which partition is the most suitable one for the traffic generated by the application. For example, Table 2 does not show if a partition formed by four clusters of two nodes each (partition 4) represents the traffic generated by CG benchmark better than a partition formed by two clusters of four nodes each (partition 6). Therefore, a quality function is necessary in order to sort the partitions in the dendogram depending on the ratio between the intracluster and the intercluster required bandwidth. The partition that shows the highest value for this function will be the most suitable one for the traffic generated by the application.

3.1 Quality Function

We have defined two distinct and complementary global quality functions based on the table of communications between processes, the *similarity* and the *dissimilarity* functions. The first function measures the required intracluster communication bandwidth, and the second one measures the required intercluster communication bandwidth.

Let a partition P be formed by M clusters $A_1, A_2, \cdots, A_M$, and let a cluster A_i be formed by x_i processes $a_1, a_2, \cdots, a_{x_i}$ ($x_i < N$, where N is the number of processes generated by the application). Under these conditions, the cluster similarity function for A_i is defined as

$$F_{A_i} = \sum_{k=1}^{x_i}\sum_{j=1}^{x_i} T_{a_k a_j} \quad (2)$$

where T_{ij} is the amount of messages exchanged between process i and process j in the table of communication between processes. If a cluster A_i contains x_i processes, then F_{A_i} is defined as the sum of all the messages exchanged between the x_i processes that form cluster A_i.

The similarity global function for each partition P is defined as

$$F_G = \frac{\dfrac{\sum_{i=1}^{M} F_{A_i}}{\sum_{i=1}^{M} x_i^2}}{\dfrac{\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}}{N^2}} \quad (3)$$

where M is the number of clusters in the partition P, F_{A_i} represents the cluster similarity function for each cluster A_i and the term

$$\sum_{i=1}^{M} x_i^2 \tag{4}$$

is the total number of elements in the table of communication between processes that imply communication with processes in the same cluster. F_G is computed as the sum of all the F_{A_i} values divided by the total number of intracluster elements in the table of communication between processes in partition P, and normalized by the average amount of messages sent by each process. Thus, a value of F_G greater than 1 means that the partition shows greater intracluster communication bandwidth requirements than when grouping processes randomly, while values for F_G close to 0 mean that the obtained partition shows very small intracluster communication bandwidth requirements (a high portion of exchanged messages are destined to processes in other clusters), compared to the average amount of messages sent by each process.

For the dissimilarity global function we define the cluster dissimilarity function D_{A_i} for a cluster A_i as

$$D_{A_i} = \sum_{k=1}^{x_i} \sum_{j=1}^{N-x_i} T_{a_k j} \quad \forall j \notin A_i \tag{5}$$

That is, D_{A_i} is defined as the sum of all elements in the table of communication between processes that imply communication of processes in cluster A_i with processes in the rest of the clusters. The dissimilarity global function for partition P is defined as

$$D_G = \frac{\dfrac{\sum_{i=1}^{M} D_{A_i}}{\sum_{i=1}^{M} x_i\,(N - x_i)}}{\dfrac{\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}}{N^2}} \tag{6}$$

where M represents the number of clusters in the partition P and D_{A_i} represents the cluster dissimilarity function for each cluster A_i. D_G is computed as the sum of all the D_{A_i} values divided by the number elements in the table of communication between processes that imply intercluster communication of processes in A_i with processes in any other cluster in partition P, and then normalized by the average amount of messages sent by a process. Thus, a value of D_G close to 1 means that the partition has intercluster communication bandwidth requirements very close to the communication bandwidth required when considering each process as a cluster. Lower values for D_G mean that the partition shows smaller intercluster bandwidth requirements (the clusters are better defined than when considering each process as a cluster).

F_G and D_G provide a measurement of the required intracluster and intercluster communication bandwidth, respectively. Thus, the quotient of F_G divided by D_G provides the relationship between the intracluster and intercluster required bandwidth for a given partition P. We will denote this relationship as the *grouping coefficient* G_c. The grouping coefficient can be used to select the most suitable partition from the dendogram provided by the clustering algorithm: a higher grouping coefficient corresponds to better defined groupings of processes. Effectively, a higher grouping coefficient means that fewer messages are exchanged between the clusters of processes of that partition, and more messages are destined to processes in the same cluster.

As an example, Figure 1 shows the grouping coefficients obtained for each one of the partitions shown in Table 2. In this figure the grouping coefficient is shown on the Y-axis, while the number of clusters that form each partition is shown on the X-axis.

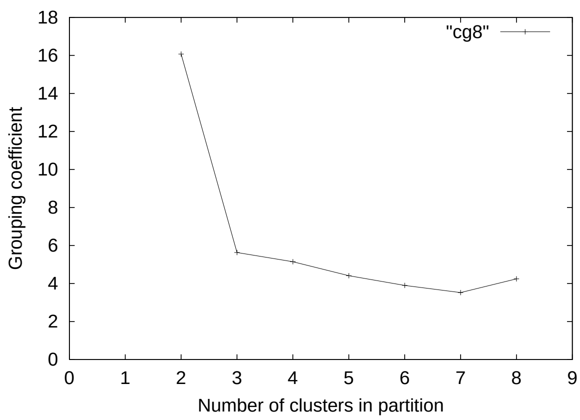

Figure 1. Grouping coefficients for the dendogram shown in Table 2

Figure 1 clearly shows that the partition formed by two clusters is the one that provides the highest grouping coefficient. The grouping coefficient significantly decreases for partitions consisting of more than two clusters, meaning that when there exist more than two clusters then the number of intercluster messages is relatively higher with respect to the number of intracluster messages. The partition formed by two clusters corresponds to Partition 6 in the dendogram shown in Table 2, and it is formed by two clusters of four processes each. Therefore, Partition 6 can be considered as the most appropriate model of the traffic generated by CG benchmark when configured for eight processes. Additionally, Figure 1 provides a classification of the partitions shown in Table 2 based on the grouping coefficient. Thus, the proposed clustering algorithm and the grouping coefficient can provide a method for the characterization of the communication requirements of the applications.

4 Evaluation Results

The machines used to simulate the execution of the NAS Parallel Benchmarks have been PC clones ranging from a 90 MHz Pentium with 32 Mbytes of RAM to a 450 MHz Pentium-III processor with 192 Mbytes of RAM. As stated above, we have configured the benchmarks for a number of processes ranging from 8 to 64.

Figure 2 shows the grouping coefficients obtained for CG benchmark configured for 16 processes. In this case, the clustering coefficient drastically increases from two to four clusters, and then rapidly decreases until the partition formed by seven clusters. There is a local maximum value of the grouping coefficient (G_c) for the partition formed by 8 clusters, and then G_c slowly decreases for any partition formed by a higher number of clusters. G_c slightly increases again only for a partition formed by 16 clusters of one process each. Therefore, in this dendogram the partition that shows the highest value for G_c is the one formed by four clusters of four processes each. Table 3 shows this partition.

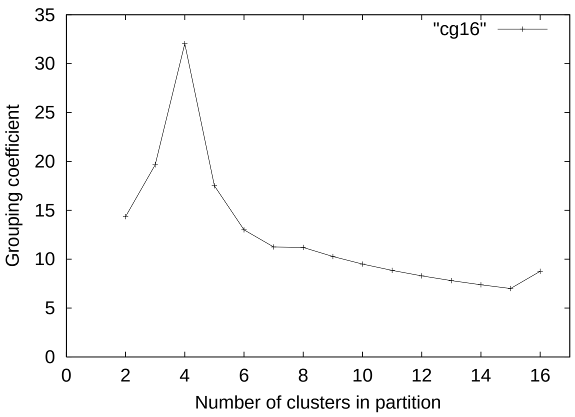

Figure 2. Grouping coefficients for Benchmark CG configured for 16 processes

(0, 1, 2, 3)	(4, 6, 5, 7)	(8, 9, 10, 11)	(12, 13, 14, 15)

Table 3. Partition with the highest grouping coefficient for CG benchmark with 16 processes

Figure 3 shows the values of G_c for the dendogram obtained from execution traces of CG benchmark configured for 32 processes. When comparing this figure with Figures 2 and 1, it can be seen that the shape of the plots are very similar. In all of them there is a peak (maximum value of G_c) that in Figure 1 is obtained for a partition formed by two clusters of four processes each, and in Figures 2 and 3 is obtained for a partition formed by four clusters. In the case of Figure 2 all the clusters are formed by four processes, and in the case of Figure 3 all the clusters are formed by eight processes. Additionally, G_c reaches a local maximum for a partition with 16 clusters in Figure 3. In this case all the clusters are formed by pairs of processes.

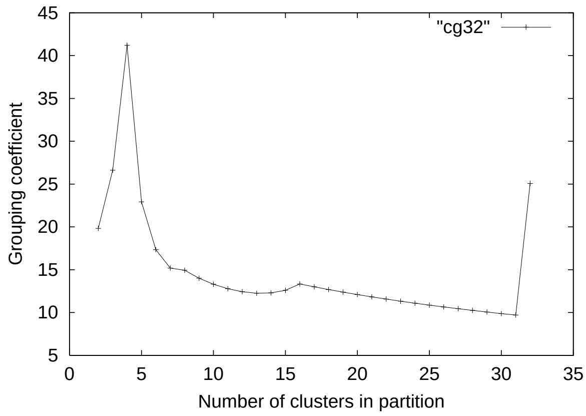

Figure 3. Grouping coefficients for Benchmark CG configured for 32 processes

The values of G_c for the dendograms obtained from execution traces of EP, LU, MG and SP benchmarks produce very similar plots. For this group of benchmarks, the values of G_c start with relatively low values for the partitions formed by a few clusters. They increase as the number of clusters increases, resulting in a central region of the plot with the highest values of G_c. Then, these values decrease, until reaching zero for the partition formed by N clusters, where N is the number of processes the benchmark is configured for. As an example, Figures 4 and 5 show the values of G_c for the dendogram obtained from execution traces of EP benchmark configured for 16 and 32 processes, respectively. For this benchmark the shape of the plot in both figures is also very similar, showing the same behavior of G_c for different number of processes. The value for G_c increases while the number of clusters in the partitions increases, reaching a local maximum, slightly decreasing and reaching the absolute maximum for the partition formed by 21 clusters in the case of $N = 32$ and for the partition formed by 11 clusters in the case of $N = 16$. From these points, the value for G_c decreases as the partitions contain

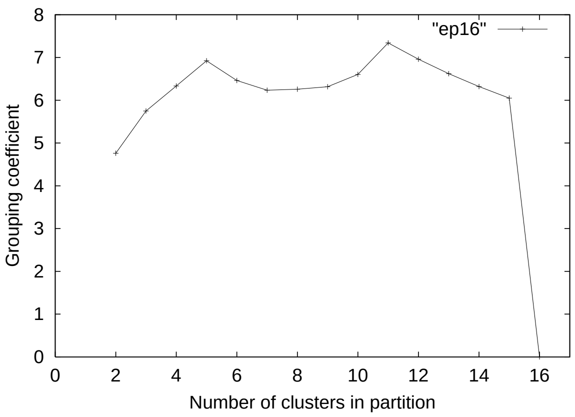

Figure 4. Grouping coefficients for Benchmark EP configured for 16 processes

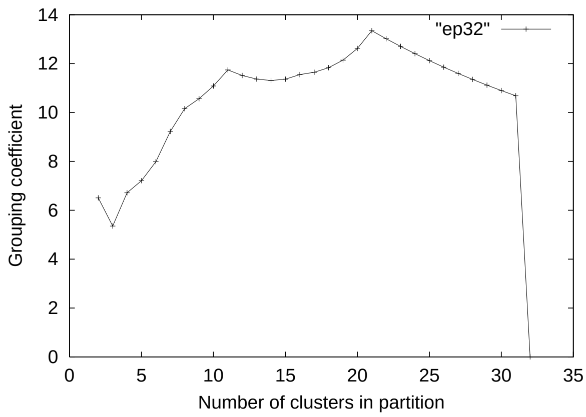

Figure 5. Grouping coefficients for Benchmark EP configured for 32 processes

more clusters. Since in these cases the table of communication between processes contains a zero diagonal, the value for G_c is zero when the partition is formed by N clusters of one process each. Although they are not shown here due to space limitations, the partitions that provide the highest value for G_c in both figures are very similar. In the case of Figure 4, the partition that provides the maximum value for G_c is formed by five clusters of two nodes each and six clusters of one node each. The partition that provides a local maximum value for G_c is formed by three clusters of four processes each and two clusters of two processes each. Similarly, the partition that provides the maximum value for G_c in Figure 5 is formed by eleven clusters of two nodes each and ten clusters of one node each. The partition that provides the local maximum value for G_c is formed by five clusters of four processes each and six clusters of two processes each.

It is worth mentioning that the absolute maximum values reached by G_c are similar for EP, LU, and MG benchmarks (around 8 when they are configured for 16 processes and around 15 when they are configured for 32 processes). For SP benchmark the absolute maximum value decreases to 3 when configured for 16 processes, and to 7.5 when configured for 32 processes.

Figures 6 and 7 show the values of G_c for the dendograms obtained from execution traces of IS benchmark configured for 16 and 32 processes, respectively. For this benchmark all the elements in the obtained table of communications between processes are very similar, indicating that this benchmark produces uniform traffic between processes. Therefore, the shape of the plots in both figures is almost a flat line. Only for the partition formed by N clusters (where N is the number of processes the benchmark is configured for) G_c increases, showing that for this benchmark the best characterization is not to group any process.

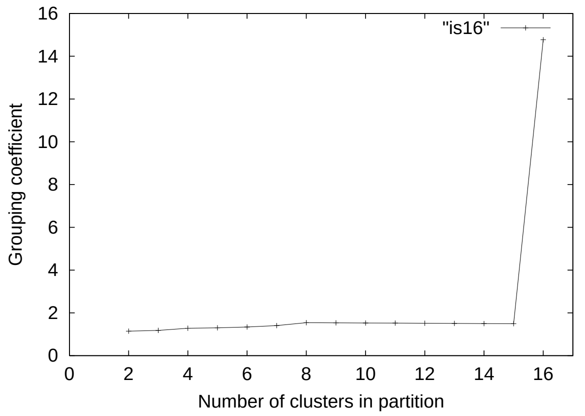

Figure 6. Grouping coefficients for Benchmark IS configured for 16 processes

5 Final Considerations

The characterization of communication requirements for message-passing parallel applications presented in this paper clearly shows that, in general, communication is not uniformly distributed among processes. Instead, some pairs of processes communicate much more frequently than others. This information can be very useful when mapping processes to processors in heterogeneous environments where the communication bandwidth available is not the same for each group of processors.

In particular, the proposed characterization method can be used when a given application is repeatedly executed

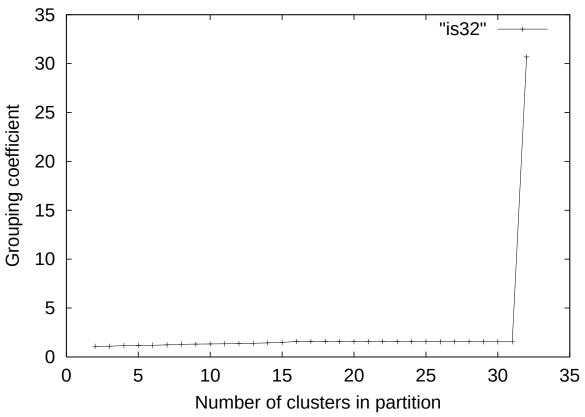

Figure 7. Grouping coefficients for Benchmark IS configured for 32 processes

with different input data, as in the case of weather forecasting applications, particle physics computing, and many other applications. In these cases, both task execution times and communication bandwidth requirements are similar to the ones for previous executions of the application. Therefore, the characterization of communication requirements for an application run can be used to obtain better mappings of processes to processors when the network becomes the system bottleneck.

6 Conclusions and Future Work

In this paper we have proposed a method of characterization of communications between processes in message-passing applications. In this method, a table communications between processes is extracted from each execution trace. Then, a clustering algorithm is applied on this table, obtaining a dendogram. Also, a quality function is proposed to sort the partitions in the dendogram depending on the ratio between the intracluster and the intercluster required bandwidth for each partition.

Evaluation results show that the proposed approach is able to find the most suitable partition to model the traffic generated by the applications as clusters of processes. In all of the considered applications the quality function provides a single maximum value that clearly identifies the best partition. Additionally, this function provides a way of sorting the partitions in the dendogram depending on the ratio between the intracluster and the intercluster required bandwidth.

The results also show that, in all of the considered applications, the partitions that provide the best ratio between the intracluster and the intercluster required bandwidth are well balanced in regard to the number of processes that form the clusters. Therefore, if all the processes have a similar computational load, then these results show that when the computational load is well balanced then the communication requirements are also well balanced.

As for future work, we plan to develop scheduling strategies that match the communication requirements of the applications to the communication bandwidth available in different parts of the network. Later, we plan to integrate these scheduling strategies with strategies that consider the computing requirements of the tasks.

References

[1] V. Arnau, J.M. Orduña, A. Ruiz, J. Duato, "On the Characterization of Interconnection Networks with Irregular Topology: A New Model of Communication Cost", in *XI IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'99)*, November 1999.

[2] V. Arnau, J.M. Orduña, S. Moreno, R. Valero, A. Ruiz, "A Clustering Approach for Improving Network Performance in Heterogeneous Systems", in *Euro-Par'2000 – Parallel Processing*, *Springer-Verlag, 2000. Lecture Notes in Computer Science.*

[3] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, M. Yarrow , "The NAS Parallel Benchmarks 2.0". Technical Report NAS-95-020. NAS Systems Division, NASA Ames Research Center, December 1995.

[4] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, *J. Wiley, 1993.*

[5] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, *J. Wiley, 1973.*

[6] B. Everitt, *Cluster Analysis*, *Wiley ,New York, 1974.*

[7] R.F. Freund, M. Gherrity, S. Ambrosious et al., "Scheduling in Multi-User, Heterogeneous Computing Environments with SmartNet", in *Proceedings of* 7^{th} *IEEE Heterogeneous Computing Workshop (HCW'98)*, March 1998, pp. 184-199.

[8] W. Gropp, E. Lusk, N. Doss and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard", in *Parallel Computing*, Vol. 22, No. 6, pp. 789-828, September 1996.

[9] W. Gropp and E. Lusk , "User's Guide for `mpich`, a Portable Implementation of MPI". Technical Report ANL-96/6. Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[10] M. Kafil and I. Ahmad, "Optimal Task Assignment in Heterogeneous Distributed Computing Systems", in *IEEE Concurrency*, Vol. 6, No. 3, 1998, pp. 42-51.

[11] M. Maheswaran, S. Ali, H. Siegel, D. Hensgen and R. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogenous Computing Systems", in *Proceedings of* 8^{th} *IEEE Heterogeneous Computing Workshop (HCW'99)*, April 1999, pp. 30-44.

[12] J.M. Orduña, V. Arnau, A. Ruiz, R. Valero, J. Duato, "On the Design of Communication-Aware Task Scheduling Strategies for Heterogeneous Systems", in *Proceedings of International Conference on Parallel Processing (ICPP-2000)*, Toronto (Canada), August 2000.

[13] A. L. Peressini, F. E. Sullivan, J. J. Uhl, Jr.*The Mathematics of Nonlinear Programming* , *Springer-Verlag, 1991*.

[14] S. C. S. Porto and C. C. Ribeiro, "A Tabu Search Approach to Task Scheduling on Heterogeneous Processors under Precedence Constraints", in *Int. Journal of High Speed Computing*, Vol. 7, No. 1, 1995, pp. 45-71.

[15] H. Singh and A. Youssef, "Matching and Scheduling Heterogenous Task Graphs using Genetic Algorithms", in *Proceedings of* 5^{th} *IEEE Heterogeneous Computing Workshop (HCW'96)*, April 1996.