Predicting the communication pattern evolution for scalability analysis

Javier Panadero*, Alvaro Wong, Dolores Rexachs, and Emilio Luque

Computer Architecture and Operating System Department, Universitat Autonoma of Barcelona, Barcelona, SPAIN.
{javier.panadero, alvaro.wong}@caos.uab.es,
{dolores.rexachs, emilio.luque}@uab.es

Abstract. The performance of the message-passing applications on a parallel system can vary and cause inefficiencies as the applications grow. With the aim of providing scalability behavior information of these applications on a specific system, we propose a methodology that allows to analyze and predict the application behavior in a bounded time and using a limited number of resources. The proposed methodology is based on the fact that most scientific applications have been developed using specific communicational and computational patterns, which have certain behavior rules. As the number of application processes increases, these patterns change their behavior following specific rules, being functionally constants. Our methodology is focused on characterizing these patterns to find its general behavior rules, in order to build a logical application trace to predict its performance. The methodology uses the PAS2P tool to obtain the application behavior information, that allow us to analyze quickly a set of relevant phases covering approximately 95% of the total application. In this paper, we present the entire methodology while the experimental validation, that has been validated for the NAS benchmarks, is focused on characterizing the communication pattern for each phase and to model its general behavior rules to predict the pattern as the number of processes increases.

Keywords: Prediction Scalability, Communication Pattern, MPI applications

1 Introduction

During the last years, due to constant hardware evolution, high performance computers have increased the number of cores significantly. Users of these systems want to get the maximum benefit of the number of cores and scale their applications, either by reducing the execution time or increasing the workload.

* This research has been supported by the MICINN Spain under contract TIN2007-64974, the MINECO (MICINN) Spain under contract TIN2011-24384, the European ITEA2 project H4H, No 09011 and the Avanza Competitividad I+D+I program under contract TSI-020400-2010-120.

To achieve an efficient use of high performance systems, it would be important to consider the analysis of the application behavior, before executing an application in a large system, since the ideal number of processes and resources required to run the application may vary from one system to another, due to hardware differences. The lack of this information may cause an inefficient use, causing problems at different levels, such as not achieving the expected speed up, or increasing the economic and energy cost. To avoid these problems and make an efficient system use, users and system administrators use predictive performance models selecting the most appropriate resources to run the application.

In this paper, we propose a methodology that will allow us to analyze and predict the scalability behavior for message-passing applications on a given system, in a bounded time and using a reduced set of resources. The objective of the methodology is to predict the application performance when increasing the number of processes, characterizing and analyzing the behavior of the communication and computation patterns.

The methodology is based on the fact that most scientific applications have been developed using specific patterns, which have functional similarity according to the number of processes, following behavior rules. To characterize these rules, we used the PAS2P methodology [1]. PAS2P identifies the application phases, which contain a specific communication pattern and allows us to reduce the complexity of the application analysis by creating the application signature, which contains the relevant communication and computation patterns of the application (phases), and their repetition rates (weights).

The application phases can be observed and analyzed during the execution time dynamically, this is without having the source codes, to relate them by functional similarity increasing the application processes, with the objective to model their general behavior rules, to build an application logical trace that will be independent of the machine. Once we have the logical trace, the last step is to convert this trace in machine dependent, through the machine parameters to get the physical trace This new trace will be used to predict the performance of the application.

This paper is organized as follows: Section II presents the related work, Section III presents an overview of the PAS2P methodology, Section IV presents the proposed methodology, Section V presents the experimental validation, which is focused in the communication pattern modeling, and finally Section VI presents the conclusions and future work.

2 Related Work

There are other works, related to the study of communication patterns of MPI applications. I. Lee et al [3] proposes to analyze the communication patterns of NAS-MPI benchmarks to understand the communication behavior in scientific workloads and to predict the larger scale program behavior. This work is focused on measuring the communication timing, the sources and destinations and mes-

sage sizes. This work differs from our proposal in that we model the general rules of the communication pattern and communication volume.

R. Preissl et al [4] presents an algorithm for extracting communication patterns. The algorithm finds locally repeated sequences on each node using a suffix tree algorithm and matches these local repetitions with other sequences on other nodes to generate a global pattern. This approach differs of our work in that we use a functional similarity algorithm, instead of the suffix tree algorithm.

M. Chao et al [5] proposes a methodology to determine the communication pattern similarities between two programs using two metrics to form a coordinate on a 2-dimensional Cartesian Space. Our work differs of this, because we search the similarity as the number of processes increases.

3 Overview about PAS2P methodology

The PAS2P methodology [1] is composed by two steps. The first step is done on a base machine and consists on analyzing the application, building the application model to extract its phases and weights that will use to construct the signature, which is an executable that contains the application phases. The second step consists on executing the signature on a target system, to measure the execution time of each phase. Once these times have been measured, equation 1 is used to predict the application execution time in the target system, where PET is the Predicted Execution Time, n is the number of phases, TEPhasei is the Phase i Execution Time and Wi is the weight of the phase i.

$$PET = \sum_{i=1}^{n} (TEPhasei)(Wi) \tag{1}$$

4 Proposed Methodology

Fig. 1 shows the proposed methodology, which is based on the fact that most message-passing applications have been developed using specific communication and computation patterns, which have functional similarity when the number of processes increases, following specific rules of behavior. These patterns compose the application phases, that can be observed and traced to relate them when the number of processes increases, in order to find and model their general behavior rules. Once the patterns have been modeled, it is possible to generate the logical trace, which will provide the communication and computation times to predict the application performance.

In this section, we will describe each of the methodology stages, focusing on the characterization and modeling of the communication pattern, which is the objective of this paper.

4.1 Characterization

The characterization step comprises two sub-stages: Machine Characterization and Application Characterization. The Machine Characterization is done once, regardless of the application which scalability we will predict.

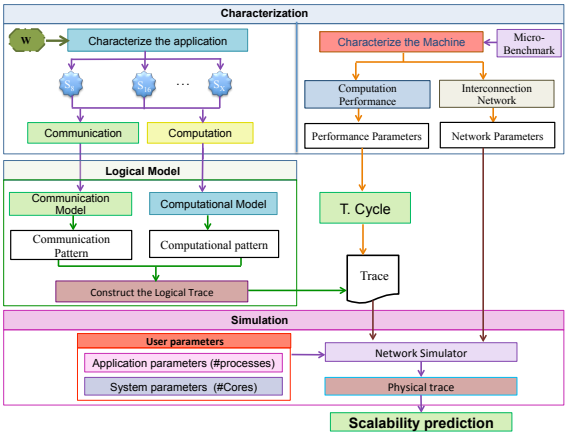


Fig. 1: Proposed Methodology

Machine Characterization. Consists on characterizing the machine performance in computation and communication. First of all, the performance is analyzed to obtain the real cycle time, when the processor is executing floating-point instructions and there are not misses on any level of the memory hierarchy. We characterize instead of using the theoretical time provided by the manufacturer, because there may be differences between them. In order to obtain the real cycle, a micro-benchmark was developed. On the other hand, the interconnection network is characterized using benchmarks.

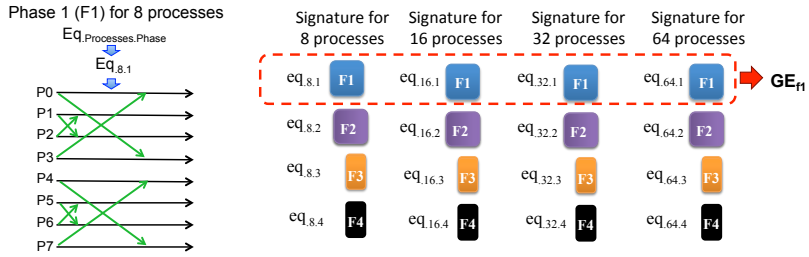
Application characterization. Consists on analyzing the application behavior (communication and computation) to obtain information and build its logical trace. We carry out a set of signature executions for a small and different number of processes, that will be analyzed to extract information of each phase of the signature. Each phase identifies a repetitive computation and communication behavior. The application signatures are obtained with PAS2P tool [2], which is a tool that applies the PAS2P methodology in an automatic and transparent way. It was decided to work with the signature rather than the whole application, since the signature contains only the relevant application phases. By analyzing this small set of relevant phases, we cover about 95% of the whole application. For this work, we integrated PAS2P with PAPI hardware performance tool [6] to obtain low-level performance information, such as the number of instructions and the number of cycles of each phase.

4.2 Logical trace generation

Once the relevant phases have been characterized, the communication and computation patterns are modeled for each phase to obtain the general behavior rules, which will be used to generate the application logical trace. This trace will be machine independent, according to how the application has been developed.

The parameters of the general behavioral rules will define the trace for a specific number of processes. Once the logical trace has been generated, the predicted computation and communication times are provided to generate the physical application trace, which will be dependent on the machine and will allow us to predict the application performance.

Communication pattern modeling. The communication pattern comprises the general behavior equations, which predict the destination from the source, and the data volume for each phase. The phases will be related by functional similarity between the signatures with different number of processes, in order to model their behaviors. It should be mentioned that a phase can have 1 to N communications, depending on the application. The predicted data volume of each communication will be obtained by mathematical regression models, while for obtaining the general communication rules (source - destination), we propose an algorithm, based on obtaining the communication equations that calculate the destination from the source ($\text{eq}._{processes.phase}$) for each phase of the signatures (local equations). From these equations, using functional similarity, the general equation behavior is modeled. We show an example of this algorithm, which considers that each phase (F_i) has only one communication, and therefore one local equation, as shown in fig. 2(a) for the phase 1 for 8 processes. Once the local equations have been obtained for each phase of each signature, these are analyzed by functional similarity to model the general behavior equation (GE_{Fi}) as shown in fig. 2(b) for phase 1.



(a) F1 for 8 processes (b) Comparison of phases by functional similarity

Fig. 2: Obtaining general equations by functional similarity

The algorithm to obtain the behavior rules is divided in two stages, in the first stage the local equations are generated for each phase, and in the second stage the general equations are obtained, as shown in fig. 3. Noteworthy that for both steps, the process identifier (process number) is converted to binary in order to work at bit level.

The first algorithm stage is composed of a first sub-stage of analysis and a second sub-stage of modeling. During the analysis phase, the dependencies between processes (Dependent, No-Dependent), the pattern type: Static (Mesh,

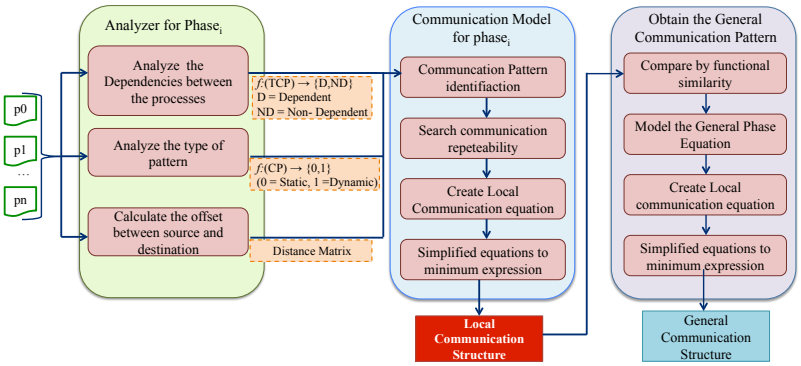


Fig. 3: Proposed algorithm to model the communication pattern

Ring, ...) or Dynamic (Exchange or Permutation), and the distance matrix between processes are obtained for each phase of the signatures.

All this information is provided to the modeling phase to generate the local equations. In this second sub-stage, the communication pattern of each phase is identified. Moreover, the repeatability of the communication is sought to generate a more structured equation model. Once this information has been identified, the local equation for each communication is generated and applied in a compression method to simplify the next step of modeling. The output of this module is a communication structure that identifies each local equation. The algorithm uses two different structures because the way to predict the communication pattern is different depending on the pattern type. If the pattern is dynamic, the obtention of the destination process is based on the exchange of a certain number of source bits, which are called bits involved. For this type of algorithm, the first structure (EC1) is used. In case of a static pattern, to obtain the destination process we search the repeatability of the communications, for example, in a 4 x 4 mesh, the first three processes in the first row have a displacement of 1, while the fourth process connects with the first and has a displacement of 3. This behavior is repeated for the remaining rows of the mesh. For this type of patterns, the second structure (EC2) is used.

The structure EC1 has the number of phase (#Phase), the communication number of the phase (#Comm), the algorithm type (Exchange, Permutation) and a vector with the bits involved in the pattern as parameters, while the structure EC2 has the number of phase, the communication number of the phase, and a list of communication and number of repetitions as parameters.

1. EC1 = {#Phase, #Comm, Algorithm Type , List of bits involved }

2. EC2 = {#Phase, #Comm, list[communication list[#repetition]] }

Fig. 4 shows a brief example of the procedure. We have a phase with 8 processes and three communications, where each communication has its own communication pattern. If we focus on the first communication, that shows its communication pattern, we generate the matrix distance between the source and

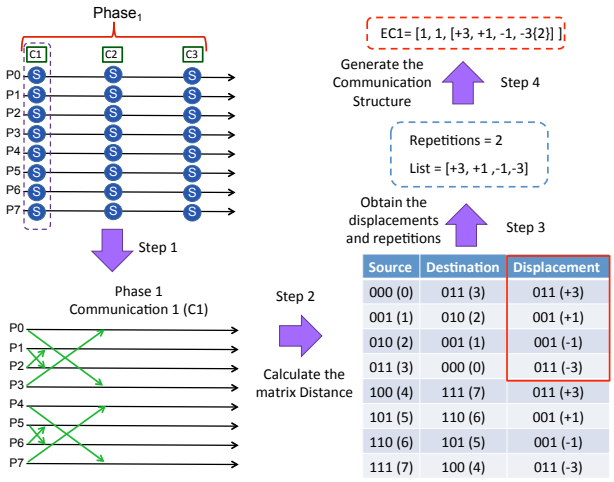


Fig. 4: Example of generating the Communication Structure

destination. The algorithm is static, then we search for repetitions, in this case, we have two, therefore the sequence {+3,+1,-1,-3} is repeated two times, once for processes from 0 to 3 and then for processes from 4 to 7. Once we have the sequences and repeatability, we create the local equation and generate the structure of communication EC1.

Once the communication structure has been obtained, it is submitted to the second algorithm stage with the purpose of obtaining the general equations, which model the general behavior of the communication pattern. To model the general equations of each phase, the local equations are analyzed by functional similarity. When the similarity has been identified, the general equations are modeled. The general equation has as variables the number of processes to predict, the displacements between the source and destination for the number of processes to predict, and the hops between the processes executed and the number of processes to predict. Finally, the last step that compresses the general equation in order to simplify the expression used to predict the communication pattern for a greater number of processes.

Computation pattern modeling. As shown in fig. 5, the algorithm is based on searching the repeatability of MPI primitives for each phase to identify the computation patterns. Then, they are compared using functional similarity between the different signature executions to predict the computation patterns for a larger number of processes. We search repeatability because these primitives are enclosed in repeated loops throughout the phase. The aim of the proposed algorithm, is to discover the minimum set of primitives and predict their repetition number to generate the logical trace, as it may vary depending on the number of processes. Once the information has been characterized, the computation time between the MPI primitives is modeled, based on separating the computation

time in: execution time, the time it takes the processor to execute instructions, the stall time and the time that the processor waits for memory accesses. We decided to separate computation times as they may have different trends when the number of processes increases. In order to separate these times, the model uses information from the hardware counter, obtained in the characterization stage. Then, the computation time will be predicted using statistical regression models .

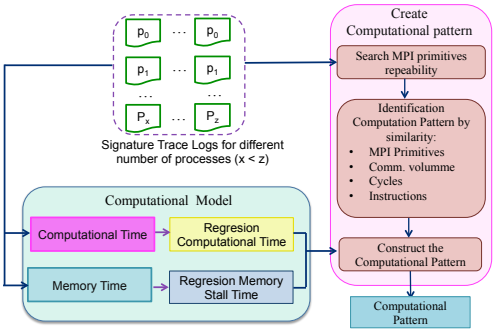


Fig. 5: Procedure to model the computation pattern

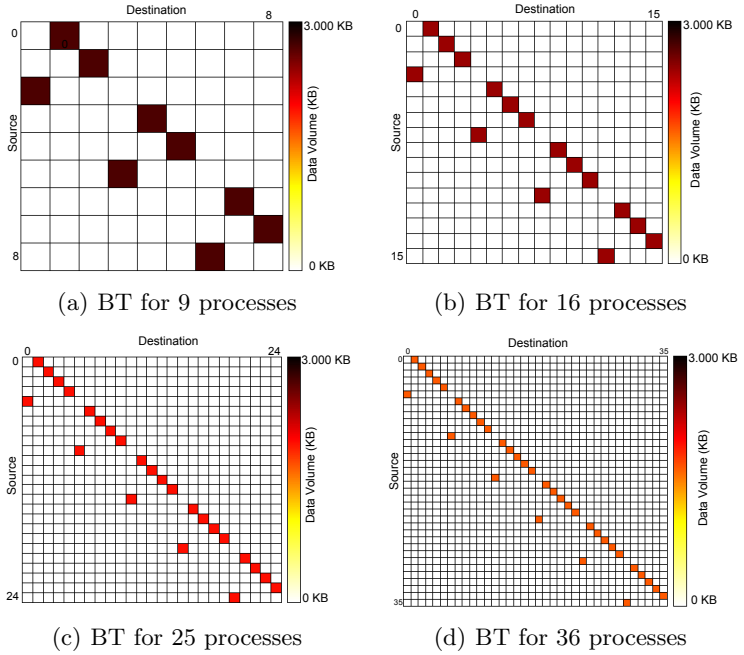
4.3 Simulation

Finally, in the last stage of the methodology we carried out a simulation of the trace in order to obtain the communication times of the messages from the trace on the physical machine. After the simulation, we obtained the runtime of each predicted phase that comprises the trace. Each phase time will be multiplied by the predicted weight of the phase to obtain the run time of the application for the number of processes we want to predict, as shown in Equation 1.

5 Experimental Validation

This section shows the experimental validation of the communication pattern modeling using the BT from the NPB class D. As experimental environment, we used a cluster of 16 nodes with 16 processors Intel Xeon quad-core.

To carry out the experimental validation, we executed a set of BT signatures and different number of processes (9, 16, 25 and 36). We predicted for 49 processes. The signatures were characterized to obtain the local equations for all phases and validated the proposed algorithm for all the communications in each phase, but due to a lack of space we only show the first communication in the first phase, as is shown in fig. 6. From this characterization, table 1 shows the local equations and the communication volumes. The pattern type is static (Toroidal), hence, the EC1 output structure is used. From the information of this structure, the general equation and the communication volume for each communication phase were modeled and validated for 49 processes.



(a) BT for 9 processes (b) BT for 16 processes

(c) BT for 25 processes (d) BT for 36 processes

Fig. 6: Characterization of the communication pattern and volume data

Table 1: Sumary of commnications for the different signatures

Signature (#Processes)	Phase	Com.	Local Equation	Communication Volume (KB)
9	1	1	[+1 {+2,+3}], [-2{+1,+3}]	2892
16	1	1	[+1 {+3,+4}], [-3{+1,+4}]	2496
25	1	1	[+1 {+4,+5}], [-4{+1,+5}]	2132
36	1	1	[+1 {+5,+6}], [-5{+1,+6}]	1849

Fig. 7 shows the general equation, corresponding to a static algorithm (Toroidal), which has a displacement +1 (first term of the equation), except for the processes located at the end, which connect with the initial nodes (second term of the equation). The variable *Disp* has a value of 1, since the distance between the source and destination is 1, while the variable K indicates the number of hops from the last characterized signature until the number of processes we want to predict. BT signature has been executed for 36 processes, and we want predict for 49 processes, then K has increased by 1 unit (1 hop). This is, by the application constraint, the next incremental step of 36 processes is 49, because BT only accepts processes of a square number as valid. On the other hand, in fig. 7 we show the communication volume equation, which is a potential regression and has a R-squared value of 0,97. If we apply this equation for 49 processes, we obtain a communication volume of 1703.59 KB. If we compare this value with the real execution, where we obtained a communication volume of 1628 KB, the prediction error is about 4.6%.

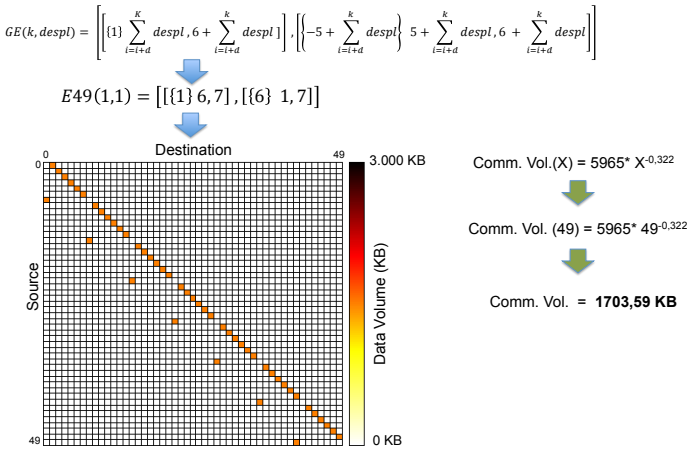


Fig. 7: Prediction of communication pattern and data volume for BT 49

6 Conclusions and future work

This paper proposes a methodology to analyze and predict the scalability behavior in message-passing applications in a given system, using a limited number of resources and bounded time. The methodology was presented and the modeling communication was experimentally validated. Currently, we are working on finishing the computational model validation and generating the logical trace, in order to insert it in a network simulator and obtain the physical trace, which will predict the application performance.

References

1. Wong, A., Rexachs, D., Luque, E.: Extraction of parallel application signatures for performance prediction. HPCC 10th (2010) 223–230
2. Panadero, J., Wong, A., Rexachs, D., Luque, E.: A tool for selecting the right target machine for parallel scientific applications. ICCS **18**(0) (2013) 1824 – 1833
3. Lee, I.: Characterizing communication patterns of nas-mpi benchmark programs. In: Southeastcon, 2009. SOUTHEASTCON '09. IEEE. (2009) 158–163
4. Preissl, R., Kockerbauer, T., Schulz, M., Kranzlmuller, D., Supinski, B., Quinlan, D.: Detecting patterns in mpi communication traces. In: Parallel Processing, 2008. ICPP '08. 37th International Conference on. (2008) 230–237
5. Ma, C., Teo, Y.M., March, V., Xiong, N., Pop, I., He, Y.X., See, S.: An approach for matching communication patterns in parallel applications. In: Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. (2009) 1–12
6. Dongarra, J., Malony, A.D., Moore, S., Mucci, P., Shende, S.: Performance instrumentation and measurement for terascale systems. In: European Center for Parallelism of Barcelona. (2003) 53–62