Modeling Parallel Applications for Scalability Analysis: An approach to predict the communication pattern

Javier Panadero[1], Alvaro Wong[1], Dolores Rexachs[1] and Emilio Luque[1]
[1]Department of Computer Architecture and Operating System (CAOS),
University Autonoma of Barcelona, Spain
javier.panadero@caos.uab.es, alvaro.wong@caos.uab.es, dolores.rexachs@uab.es, emilio.luque@uab.es

Abstract— *The performance of message-passing applications varies depending on the parallel system, causing potential inefficiencies when its number of processes increases. By this reason, it is critical to predict the application behavior before executing it, in order to use the system efficiently. We propose a methodology that allows us to predict the application scalability behavior in a specific system, providing information to select the most appropriate resources to run the application. The methodology strives to use a bounded analysis time, and a reduced set of resources. This paper presents the general methodology, focusing on validating the step of the methodology concerning to the generation of scalability communication model. We can predict the evolution of the communication pattern using a reduced set of resources. Analyzing from 16 to 256 processes, we can predict the communication pattern for 4,096 processes.*

Keywords: Modeling MPI applications, Application Scalability, Communication pattern

1. Introduction

With the advent of multicore and the constant hardware evolution, High Performance Computing (HPC) clusters have increased the number of cores significantly [1]. The users of these systems want to get the maximum benefit from this large number of cores, scaling their applications.

To achieve an efficient use of these HPC systems using a large number of cores, a point to consider before executing an application is to know its performance in the system. It is known that using more resources does not always imply a higher performance. The lack of this information may produce an inefficient use, resulting in not achieving the expected speedup.

Parallel applications are composed of a set of phases, which are segments of code delimited by communications events, that are repeated throughout the application [2]. These phases were written in the application code using specific communicational and computational patterns, which follow behavior rules. When the application increases the number of processes, the number of phases remains constant, but its patterns change their behavior following their behavior rules, being functionally constant. To obtain these phases, we use the PAS2P tool [3], which identifies the application phases and allows us to create the application signature. As is shown in fig. 1, the signature only contains the relevant application phases and their repetition rates (weights). Therefore, it allows us to cover approximately 97% of the total application code, in about 1% of the application execution time.

We propose a methodology to analyze and predict the strong scalability [4] behavior for message-passing applications on a given system, by running a set of small-scale signatures. It strives to use a bounded analysis time, and a reduced set of resources. Moreover, the methodology could also be useful for scheduling and code optimization.

The methodology focuses on characterizing and analyzing the communication and computational patterns of each phase, in a transparent way (without analyzing and modifying the source code), from a set of executions for a small-scale signatures, in order to model general behavior rules, to build the Scalable Logical Trace (STL), which is machine independent, depending on the way in which the application was developed. The STL will be generated for a N number of processes, and it will be used in the future, to predict the communication and computational time, in order to obtain the application execution time.

We present an overview of the methodology, focusing on explain in detail the scalability communication model, to predict the evolution of the communication pattern (spatial and volume parameters) of each phase as the application scales. In a previous paper [5], the key ideas of the method were presented. In this study, the whole procedure and its algorithms are explained in detail. In order to validate the method, we executed from 16 to 256 processes and we predict the communication pattern (Spatial and volume parameters) for 4,096 processes, which is validated with the real communication pattern obtained with PAS2P.

This paper is organized as follows: Section II presents the related work, Section III presents the proposed methodology, Section IV presents the scalability communication model, Section V presents experimental validation and finally Section VI, the conclusions and future work.

2. Related Work

Similar works to our approach have been presented in the literature. Wu et al. [6] generate the application communi-

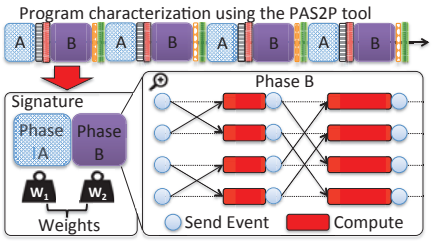


Fig. 1: Relevant phases that represent the signature.

cation trace for a large number of nodes, by extrapolating from a set of smaller traces. Their methodology is focused on SPMD (Single Program, Multiple Data) applications with stencil/mesh topology. Our proposal differs from this work, because it covers a wide range of MPI application, not only SMPD applications. Zhai et al [7] present the tool FACT, which collect MPI communication traces and extract application communication patterns through program slicing. This tool uses a set of code analysis techniques to generate a program slice that only contains the variables and code sections related to MPI events, and then executes the program slice to acquire communication traces. This tool allows to predict the communication pattern for a specific number of processes, our methodology differs because we execute a set of small-scale signatures to predict evolution of the communication pattern as the application scales.

There are other works based on analytical regression and machine learning methods, from executions for small-cores. Barnes et al [8] propose studying the scalability using regression models, isolating computation and communication to predict the application performance. Ipek et al [9] present a different approach based on multilayer neural networks. From a training set of the application executions, the application model is created automatically. This approach is interesting for its ease of use and its obliviousness to details of application internals. These works are based on the input parameter space to obtain the regression models and extrapolate its behavior. Our methodology focuses on obtaining the general behavior rules for each relevant application phase, to extrapolate its behavior to predict the application performance.

3. Proposed methodology

The main goal of the methodology is to analyze and predict the strong scalability behavior for parallel applications on a given system, using as input a limited set of small-scale signatures, as is shown in fig. 2.

The methodology is made up of three steps: Application characterization, Generation of the logical application model and Performance prediction.

As we mentioned before, the parallel applications are typically composed of patterns of computation and communication that are repeated throughout the application. These patterns are grouped in phases, which compose the application signature. The number of phases remains constant when the number of processes increases, but their patterns change their behavior following behavior rules. The objective of the characterization step is to obtain information about the communication and the computation patterns of each phase, to model the general behavior rules, which will be used to predict their behavior, as the number of the application processes increases. The methodology is restricted to applications where the communication pattern follows deterministic behavior rules.

To obtain the predicted application execution time, we carry out a set of signature executions for small-scale, which will be analyzed to obtain information from each phase. When the signature is executed in the system, it generates a trace file per process, which contains information of each phase. The trace provides information about the phase id, the type of MPI primitive, the source and destination of the communication, the communication volume in bytes, the computational time in nanoseconds and finally, the number of instructions for the computational time.

Once the phases have been characterized, their communication pattern, the computational pattern and the weight of each phase are analyzed and modeled to generate the general behavior rules, in order to construct the Scalable Logical Trace (STL) for a N number of processes. The input parameters of the general behavior rules will project the STL for a specific number of processes. The STL is composed of the intrinsic parameters for each phase needed to model the scalability of the parallel application, which are: the phase ID, communication pattern (spatial and volume parameters), number of instructions of the computational time and phase weight. The STL is generated per process instead of a global trace, with the objective being to model each process independently.

The STL has to be complemented with the computational time, in order to generate the physical trace, which is dependent of the machine, because contains the information

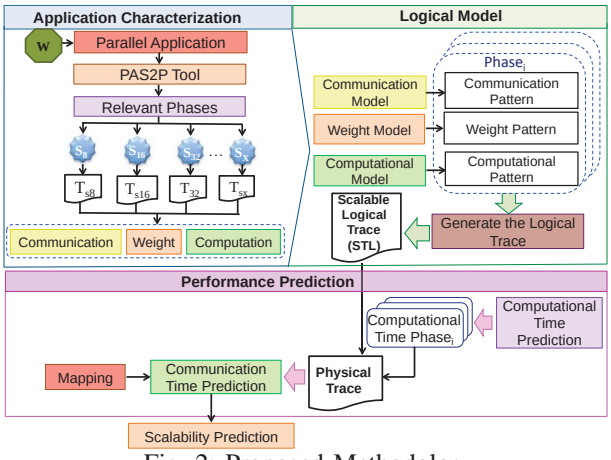


Fig. 2: Proposed Methodology

of the STL more the computational times for each phase for a N number of processes. To predict the computational time, we use a regression-based model by phase. The physical trace will be used to predict the communication time and obtaining the application performance.

To predict the communication time, the physical trace will be executed by segments of processes in a reduced number of resources, in an iterative way, until all the processes have been executed. Once the communication time has been predicted, the predicted execution time of each phase will be obtained. Then, we apply eq. 1 to obtain the application performance, where PET is the Predicted application Execution Time, m is the number of phases, *PhaseETi* is the Phase i Execution Time and *Wi* is the predicted weight of the phase i. As the objetive of this paper is to focus on explaining the computation method, we do not explain this step in detail.

$$PET = \sum_{i=1}^{m} (PhaseETi)(Wi) \quad (1)$$

In the next section, we explain in detail the scalability communication model, which predicts the evolution of the communication pattern for a large number of processes.

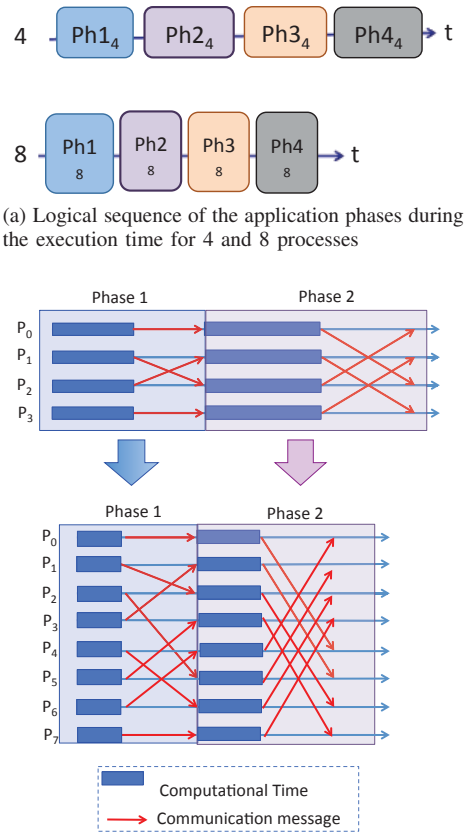
4. Scalability Communication Model

The scalability communication model comprises the general behavior equations and the data volume equations for each communication of each phase. The general behavior equations calculate the message destination from the source, while the data volume equations calculate the size of the message.

When we analyze the behavior of the phases, we know that as the application scales, the communications (number of messages and destinations), the communication volume, the computational time and the number of instructions of computation of a phase can change, but the work to be carried out will still be the same, distributed among more processes.

To model the communication behavior of each phase, it is necessary to recognize and relate the phases of the small-scale signatures. In order to relate the phases for a different number of processes, we use functional similarity. Two phases will have functional similarity, when the computation work to be carried out for both phases is the same, because is the same code segment, distributed between different number of processes, changing only the structure of the communication pattern.

To relate the phases, we use a method which is based on how the sequence of phases occurs, since it does not depend on the number of processes, only the way in which the application was developed. Fig 3 shown an example. As we can see in the fig. 3.a, the number of phases remains constant as the application increases from 4 processes to 8 processes. If we focus on the fig. 3.b, where the phases 1



(a) Logical sequence of the application phases during the execution time for 4 and 8 processes

(b) Behavior of the phases for 4 and 8 processes

Fig. 3: The functional similarity relates the phases for different number of processes

and 2 are showed in detail, we can see that the behavior is different from 4 processes to 8 processes, because the phases have different communication pattern and different computational time between them, but the work carried out by the phases is the same, distributed between different number of processes, because they are in the same logical position in the application.

Once the phases have been related, the predicted data volume of each communication will be obtained by mathematical regression models, while for obtaining the general communication rules (Source-Destination), an algorithm has been proposed. This algorithm is based on obtaining the communication equations (eq.$_{processes.phase.comm}$) for each phase (Local Equations) of the set of small-scale signatures executed, which identifies the communication pattern for each phase. From these Local Equations, the General Equations are modeled, which are used to predict the communication pattern for a N number of processes. Fig. 4 shows an example, where it has been considered that each phase has only one communication, and therefore one local equation. Once the local equations have been obtained for each phase of each signature, they are analyzed to model the General Equations (GE_{Ph_i}), as is shown in the figure for phase 1. The algorithm converts the source and destination of the

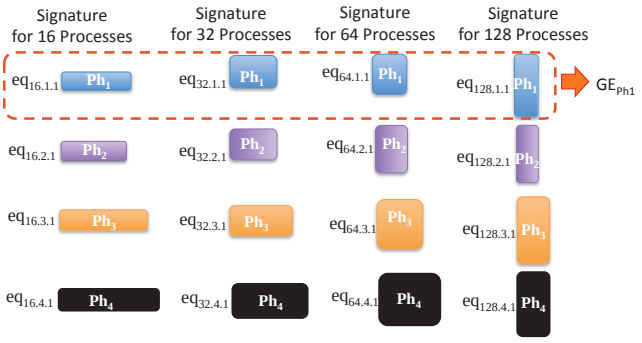


Fig. 4: Obtaining the General Equations from the Local Equations

communications from decimal to binary to work at bit level.

4.1 Generating the Local Equations

This stage is composed of two steps: a first step of analysis, in which the information obtained in the characterization stage is analyzed to obtain information about the communication pattern of each phase, and a second step of modeling, where the Local Equations for each phase are generated. The Local Equations are a representation of the communication pattern of a phase for a specific number of processes. During the analysis step, the dependencies between processes, the pattern type: Static (Mesh, Ring, etc.) or Dynamic (Exchange, Permutation, etc.), and the distance matrix between processes are obtained for each phase. All this information is provided to the second step to generate the Local Equations. In this second step, the Local Equation of each communication of the phase is obtained using an algorithm of identification. This algorithm is based on the fact that the application is well developed, and it executes a deterministic communication pattern for all the processes, without non-predictive conditional sentences as the number of processes increases. The algorithm compares the source-destination of each send primitive for all the processes of the phase, to identify the specific rule to obtain the destination from the source for that number of processes. Moreover, the repeatability of a set of communications is sought to generate easier equations and simplify the analysis and modeling for the General Equations. Finally, the Local Equations for each communication are generated.

The algorithm uses two different structures to generate the Local Equations, because the way to predict the communication pattern is different, depending on the pattern type. If the pattern is dynamic, the way to obtain the destination processes is based on the exchange of certain numbers of source bits, which are called *bits involved*. For this type of pattern, the EC1 structure is used. In case of static patterns, to obtain the destination processes, the distance between the processes and the repeteability of the communications are identified. For this type of pattern the EC2 structure is used. The EC1 structure has as parameters the phase number (#Phase), the number of communication in the phase (#Comm), the algorithm type (Exchange, Permutation) and the list of *bits involved*, (*EC1(p) = {#Phase, #Comm, Type , List of bits involved }*). The EC2 structure has the number of phase, the number of communication in the phase and the list of communication distances and its number of repetitions (*EC2(p) = {#Phase, #Comm, list[communication distances{#repetition}] }*).

Fig. 5 shows a brief example of the procedure. We have a phase with 8 processes (p=8) and three communications. These three communications compose the communication pattern of the phase, which is static because it is a 4x2 mesh, identified in the analysis phase. Then, the EC2 structure will be used. If we focus on the first communication of the pattern (Step 1), we generate the matrix distance between the source and the destination (Step 2). Then, we search for repetitions, in this case, the sequence {+1,+1,+1,-3} is repeated two times (Step 3), once for processes from 0 to 3 and another for processes from 4 to 7. Once we have the sequences and repeteability, we create the Local Equation with the structure of communication EC2 (Step 4).

Depending of the communication pattern, it can be possible that the processes of the phase do not have the same number of communications. With the aim of obtaining the correct Local Equations, all the processes of the phase must have the same number of communications for all the processes, because the algorithm could not relate the communications properly.

To solve this problem, the algorithm selects the process with the maximum number of communications, and it completes the phase structure for the other processes with null communications, until all the processes have the same number. To complete the phase structure with null communications, the algorithm is based on the fact that the application is written in a deterministic way and the communication events follow a logical order with a specific behavior.

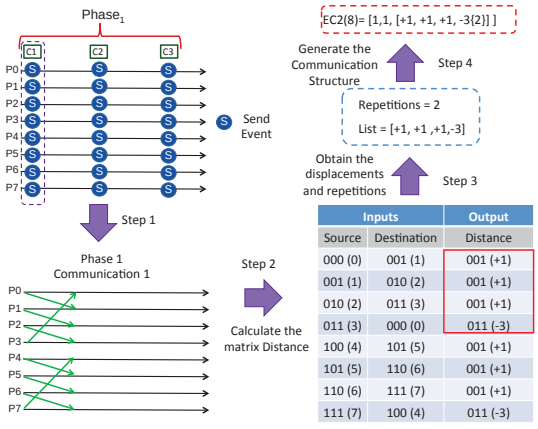


Fig. 5: Example of generating the Local Equations

4.2 Modeling the Global Equations

From the Local Equations, the General Equations of each phase are modeled, which will be used to predict the communication pattern for a N number of processes. To generate the General Equations of each phase, the Local Equations are analyzed in order to model the evolution of the communication pattern. The method consists of comparing the Local Equations to model by a function, as the parameters change their values as the number of processes increases, as is shown in fig. 6. The General Equations have the number of processes to predict as input. The structure of these equations is the same as the one for the Local Equations, the difference is that the parameters have been modeled as a function. Finally, this structure will be simplified to manage easier equations to use.

In some applications, when the number of processes increases, the communication pattern expands communicating with new processes, and new communications appear. To predict these communications, the algorithm models the behavior of how these new communications will appear (number of communications and their destination) for N processes. Fig. 7 shows this procedure. The signature traces of processes 2, 4, 8 and 16 were obtained by the signature executions, and we want to predict the communication pattern for 64 processes. As we can observe, when the number of processes is increased per two, a new communication appear. To predict the communication pattern for 64 processes, first of all, the algorithm models a function to predict the number of communications of the phase as the application scales. The function has as input parameter the number of processes to predict the number of communications. When we the number of communications of the phase has been obtained, we apply the General Equations to predict the destination.

Once we have modeled the General Equations and the communication volume equations, which are obtained by regression models, we have evolution of the communication pattern (spatial and volume) for each phase of the application. These equations will be used to generate the STL.

5. Experimental validation

In this section, the scalability communication model has been validated. Of all the experiments that we have made, we present the BT and CG from the NPB NAS [10] suite, using as input class D, and the applications: Sweep3D [11] and N-Body. We have selected this set of applications because of their distinctive behavior. As an experimental environment,

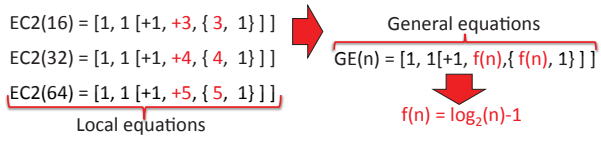


Fig. 6: Modeling the General Equation from Local Equations

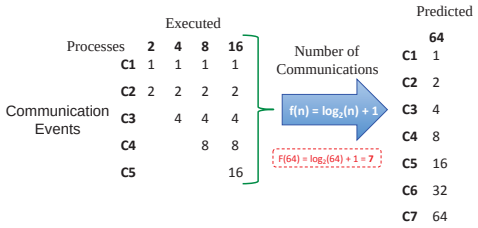


Fig. 7: Generating new communications

a cluster of 8 nodes with 64 processors AMD Opteron 6262 (512 cores) was used.

To carry out the experimental validation, we follow this workflow:

a) We executed five signatures for a small-scale and we obtain their physical traces. Four signatures are used to generate the model and the last one to validate our model before predicting. We executed the signatures with a 1:1 mapping (one process per core).
b) We generate the Local Equations for each phase of the four signatures executed.
c) We model for each phase the General Equations from the Local Equations, and the regression equations of the communication volume.
d) To validate our model, we use the fifth signature to compare whether the models that we predicted are correct. If the predicted values are correct, we use these models to predict for a greater number of processes. Otherwise, we use the fifth signature to improve our model and we create another signature with a greater number of processes to validate the model. We consider that the generated model is correct if we are able to predict the communication pattern without error, and the communication volume with an error less than 10%.
e) Finally, we use these models to generate the Scalability Logical Trace (STL) for a N number of processes.

To validate the generated STL, we compared them with the traces obtained through PAS2P tool. We are only interested in their logical information, so for executing the signatures, we allocate with an x:1 mapping (more than one process by core). By lack of space, we focus on showing the experimentation of process 0.

For BT, we executed the small-scale signatures for 16, 36, 64, 81 and 100 processes. We obtained 6 phases. We used the signatures from 16 to 81 processes to generate the model and the fifth signature to validate the results. Using the generated model, we predicted the communication pattern (spatial and volume) for 1024 processes.

Once we executed the signatures, we generated the communication model for each phase of the application. The predicted values of the communication pattern (Dest.) were obtained by the General Equations, which are shown at the left of Table 1. Due to the type of pattern (static), we used the EC2 type structure to model the General Equations

Table 1: Generated Communication Model and predicted communication pattern from these equations

Communication Model			Real Phases			Predicted Phases			Prediction Error
Phase ID	Global Comm. Equations (Dest.)	Comm. Volume Equations (Bytes)	MPI Prim.	Src-Dest.	Comm Volume	MPI Prim.	Src-Dest.	Comm. Volume	Comm. Volume
BT for process 0 with n = 1024 processes									
1	1)$f(n)=1$	$y(n)=4E+07n^{(-0.997)}$	1)Isend	0-1	40,560	Isend	0-1	39,883	1.69%
	2)$f(n)=\sqrt{(n)}-1$	$y(n)=4E+07n^{(-0.997)}$	2)Irecv	0 -31	40,560	Irecv	0-31	39,883	1.69%
	3)$f(n)=\sqrt{(n)}-1$	$y(n)=4E+07n^{(-0.997)}$	3)Wait	0-31	40,560	Wait	0-31	39,883	1.69%
2	1)$f(n)=n-\sqrt{(n)}+1$	$y(n)=7E+06n^{(-0.997)}$	1)Isend	0-993	6,760	Isend	0-993	6,979	3.13%
	2)$f(n)=17+2(\sqrt{(n)}-9)$	$y(n)=7E+06n^{(-0.997)}$	2)Irecv	0-63	6,760	Irecv	0-63	6,979	3.13%
	3)$f(n)=17+2(\sqrt{(n)}-9)$	$y(n)=7E+06n^{(-0.997)}$	3)Wait	0-63	6,760	Wait	0-63	6,979	3.13%
Sweep3D for process 0 with n = 4096 processes									
1	1) if $log_2(n) mod 2 = 0 \rightarrow, f(n)=\sqrt{(n)}$	$y(n)=6E+07n^{(-1)}$	1)Send	0-64	15120	Send	0-64	14648	3.2%
	if $log_2(n) mod 2 != 0 \rightarrow, f(n)=\sqrt{(2*n)}$								
	2)$f(n)=1$	$y(n)=6E+07n^{(-1)}$	2)Recv	0-1	15120	Recv	0-1	14648	3.2%
2	1)$f(n)=1$	$y(n)=6E+07n^{(-1)}$	1)Send	0-1	15120	Send	0-1	14648	3.2%
	2) if $log_2(n) mod 2 = 0 \rightarrow, f(n)=\sqrt{(n)}$	$y(n)=6E+07n^{(-1)}$	2)Recv	0-64	15120	Recv	0 -64	14648	3.2%
	if $log_2(n) mod 2 != 0 \rightarrow, f(n)=\sqrt{(2*n)}$								
CG for process 0 with n = 4096 processes									
1	1..18)[$\#Comm., c(n)=log_2(n)/2$,	$y(n)=8E+08n^{(-1)}$	1)Isend	0-1	187504	Isend	0-1	195312	4%
		$y(n)=8E+08n^{(-1)}$	2)Irecv	0-1	187504	Irecv	0-1	195312	4%
	$f(y)_{y=1..\#Comm.,c(n)}=2^{(y-1)}$],	$y(n)=8E+08n^{(-1)}$	3)Wait	0-1	187504	Wait	0-1	195312	4%
		...	...	...	...	...	...	...	...
		$y(n)=8E+08n^{(-1)}$	16)Isend	0-32	187504	Isend	0-32	195312	4%
		$y(n)=8E+08n^{(-1)}$	17)Irecv	0-32	187504	Irecv	0-32	195312	4%
		$y(n)=8E+08n^{(-1)}$	18)Wait	0-32	187504	Wait	0-32	195312	4%
N-Body for process 0 with n = 4096 processes									
1	1)$f(n)=1$	$y(n)=1E+07n^{(-1)}$	1)ISend	0-1	2342	ISend	0-1	2441	4.0%
	2)$f(n)=n-1$	$y(n)=1E+07n^{(-1)}$	2)IRecv	0-4095	2342	IRecv	0-4095	2441	4.0%
	2)$f(n)=n-1$	$y(n)=1E+07n^{(-1)}$	3)WaitAll	0-4095	2342	WaitAll	0-4095	2441	4.0%
	4)$f(n)=0$	$y(n)=1E+07n^{(-1)}$	4)WaitAll	0-0	2342	WaitAll	0-0	2441	4.0%

to predict the communication pattern. This structure has been simplified and specified for process 0 for readability. Moreover, in this table, we show the communication volume regression equations. All these equations have the number of processes to predict as their input parameter.

At the right of Table 1, we show the real and predicted communication patterns of phase 1 and 2 for the process 0. The predicted communication pattern was predicted by means of providing the parameter 1024 (number of processes) to the General Equations.

The predicted communication pattern corresponds with the real one for both phases. The communication volume was predicted with an error of about 2% for the first phase and 3% for the second phase. For the other four phases of the application, the communication pattern was predicted without error for all of them. In the top of Table 2, we present a summary of these phases with the predicted error of communication volume. We show the maximum error of all the communications of the phase. As we can see, the communication volume was predicted with a maximum error of about 4% (phase 5).

For CG, we executed the small-scale signatures for 16, 32, 64, 128 and 256 processes. This application has 3 phases. We used the signatures from 16 to 128 processes to generate the model and the fifth signature to validate the results. We predicted the communication pattern for 4096 processes. At the right of Table 1, we show the real and predicted trace of process 0 for phase 1. Due to the similarity between the phases, we only show phase 1. At the left of Table 1, we show the generated equations of communication for this phase. Due to the type of pattern (dynamic), we used the EC1 structure to model the General Equations. For this application, the communication pattern expands, communicating with more processes as the application scales. For this reason, we also have to model an equation to predict the number of communications of the phase. The general equation calculates the number of communications of the phase and their destinations. First of all, the number of communications is predicted using the equation $\#Comm.c(n) = log_2(n)/2$, this equation calculates the number of times that the sequence Isend, Irecv and Wait repeats in the phase. Applying this equation for 4096 processes, the sequence is repeated 6 times. Then, we apply the equation $f(y)_{y=1..\#Comm.c(n)} = 2^{(y-1)}$ to calculate the destination of the sequence, obtaining a destination sequence of 1, 2, 4, 8, 16 and 32. The communication volume was predicted with an error of about 4% for all the communications of the phase.

For the other two phases of the application, the communication pattern was predicted without error for all of

them. In Table 2, we present a summary of these phases. The communication volume was predicted with a maximum error of about 6% (phase 2).

For Sweep3D, we executed the signatures for 16, 32, 64, 128 and 256 processes. This application has 4 phases. The fifth signature was used to validate the model. We predicted the communication pattern for 4096 processes. At the right of Table 1, we show the real and predicted trace of process 0 for phases 1 and 2. The generated equations are shown in the left of Table 1. Due to the type of pattern (static), we used EC2 type structure. The pattern obtained by the General Equations corresponds to a pipeline wavefront. As we can see, the predicted pattern corresponds with the real one for both phases. The communication volume was predicted with an error of about 3.2% for both phases.

For the other two phases of the application, the communication pattern was predicted without error for all of them. At the bottom of table 2, we present a summary of these two phases. As in phases 1 and 2, the communication volume was predicted with an error of about 3.2%. This is because the application sends in its messages the same volume of bytes in all its phases.

For N-Body, we executed the signatures for 16, 32, 64, 128 and 256 processes. This application has 1 phase. We used the signatures from 16 to 128 processes to generate our model and the fifth signature to validate the results. We predicted the logical trace for 4096 processes. At the left of Table 1 we show the generated equations for this phase for process 0. Due to the type of pattern (static), we used the EC2 structure to model the general equations.

The communication pattern corresponds to a pipeline. The process 'x' communicates with the process 'x+1' until the last process, which communicates with the first process. At the right of Table 1, we show the real and predicted trace of process 0 for the phase 1. The phase has two WaitAll primitives. The first WaitAll primitive waits until the Irecv releases its request, while the second WaitAll waits until the Isend releases its request. This second WaitAll is written in the application in order to synchronize the application. The communication volume was predicted with an error of about 4%.

Table 2: Summary of prediction error for the rest of phases

Phase Number	Comm. volume equation (Bytes)	Comm. volume (Error %)
Summary of phases of BT (from 3 to 6)		
Phase 3	$y(n) = 4E + 07n^{(-0.996)}$	2.7%
Phase 4	$y(n) = 4E + 07n^{(-0.997)}$	3.1%
Phase 5	$y(n) = 5E + 07n^{(-0.997)}$	3.6%
Phase 6	$y(n) = 4E + 07n^{(-0.996)}$	2.7%
Summary of phases of CG (from 2 to 3)		
Phase 2	$y(n) = 8E + 08n^{(-0.995)}$	5.8%
Phase 3	$y(n) = 8$	0%
Summary of phases of Sweep3D (from 3 to 4)		
Phase 3	$y(n) = 6E + 07n^{(-1)}$	3.2%
Phase 4	$y(n) = 6E + 07n^{(-1)}$	3.2%

6. Conclusions and future work

We propose a methodology to analyze and predict strong scalability behavior for MPI applications on a given system. It strives to use a bounded analysis time, and a reduced set of resources. The methodology has been explained focusing on presenting the scalability communication model, to predict the evolution of the communication pattern of each phase as the application scales.

As future work, we are analyzing the internal behavior of the collective MPI primitives. Internally, the collective MPI primitives make point-to-point communications to exchange information between all the processes. We are analyzing this kind of primitives, to expand the model to consider the evolution of the internal communications of the Collective MPI primitives.

Acknowledgment

This research has been supported by the MINECO (MICINN) Spain under contract TIN2011-24384

References

[1] N. Attig, P. Gibbon, and T. Lippert, "Trends in supercomputing: The european path to exascale," *Computer Physics Communications*, vol. 182, no. 9, pp. 2041 – 2046, 2011.

[2] A. Wong, D. Rexachs, and E. Luque, "Parallel application signature for performance analysis and prediction," in *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2014 (Accepted).

[3] J. Panadero, A. Wong, D. Rexachs, and E. Luque, "A tool for selecting the right target machine for parallel scientific applications," in *ICCS*, 2013, pp. 1824–1833.

[4] R. Nishtala, P. Hargrove, D. Bonachea, and K. Yelick, "Scaling communication-intensive applications on bluegene/p using one-sided communication and overlap," in *IPDPS 2009*, 2009, pp. 1–12.

[5] J. Panadero, A. Wong, D. Rexachs, and E. Luque, "Analysis of scalability: A parallel model approachâĂİ," in *CLUSTER*, 2014, pp. 294–295.

[6] X. Wu and F. Mueller, "Scalaextrap: Trace-based communication extrapolation for spmd programs," *ACM Trans. Program. Lang. Syst. Article 5*, vol. 34, no. 1, 2012.

[7] J. Zhai, T. Sheng, J. He, W. Chen, and W. Zheng, "Fact: fast communication trace collection for parallel applications through program slicing," in *SC*, 2009.

[8] B. J. Barnes, J. Garren, D. K. Lowenthal, J. Reeves, B. R. de Supinski, M. Schulz, and B. Rountree, "Using focused regression for accurate time-constrained scaling of scientific applications," in *IPDPS*, 2010, pp. 1–12.

[9] E. Ipek, B. R. de Supinski, M. Schulz, and S. A. McKee, "An approach to performance prediction for parallel applications," in *Euro-Par*, 2005, pp. 196–205.

[10] D. Bailey, E. Barszcz, J. Barton, and D. Browning, "The NAS Parallel Benchmarks," *International Journal of Supercomputer Applications*, vol. 5, no. 3, pp. 66–73, Jan 1991.

[11] A. Hoisie, O. Lubeck, and H. Wasserman, "Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional," *Journal of High Performance Computing Applications*, vol. 14, pp. 330–346, Jan 2000.