Methodology to predict scalability of parallel applications

Claudia Rosas[1], Judit Giménez[1, 2] and Jesús Labarta[1, 2]
Barcelona Supercomputing Center[1] and Technical University of Catalonia[2]
crosas@bsc.es

Abstract- *In the road to exascale computing, the inference of expected performance of parallel applications results in a complex task. Performance analysts need to identify the behavior of the applications and to extrapolate it to nonexistent machines. In this work, we present a methodology based on collecting the essential knowledge about fundamental factors of parallel codes, and to analyze in detail the behavior of the application at low core counts on current platforms. The result is a guide to generate the model that best predicts performance at very large scale. Obtained results from executions at low core counts showed expected parallel efficiencies with a low relative error.*

Keywords: parallel efficiency, analysis and prediction, exascale computing.

LIST OF PUBLICATIONS

- Rosas, C.; Giménez, J. and Labarta, J. "Scalability prediction from fundamental performance factors." *Supercomputing Frontiers and Innovations. An Intl. Journal*. vol. 1. Num 2. pp. 4-19, Oct 2014.

I. INTRODUCTION

Inferring the scaling capacity of parallel codes is getting harder than ever [1]; applications need to be modeled for restricted or nonexistent systems.

Our goal is to use *efficiency principles* of parallelism in the code to infer its potential performance. We consider as fundamental factors, components that represent essential features of the program and their evolution may be related to primary models of parallelism such as Amdahl's Law.

Our method relies on the detailed preprocessing of available traces to determine appropriate sections and clean noise. We follow this approach because: (i) having few points to fit, the blind use of functions with many parameters would lead to undetermined systems with many possible solutions; and (ii) low core count runs provide enough information on the fundamental behavior of the code.

The main contributions of this work are:

- The capability to identify interesting regions within the execution of parallel applications;
- The collection of fundamental factors, such as: Load Balance, Serialization and Transfer, to infer their evolution in the application;
- A general model to extrapolate parallel efficiency based on Amdahl's Law.

II. METHODOLOGY

A. Identify Structure

Our method begins with a trace, obtained from executing an instrumented parallel code on a target machine. We visualize the trace to generate clean cuts of the representative phases from the temporal structure of the execution. A main phase suggests regions of computation and/or communications that may show different behaviors or that are independent between them.

B. Phase Performance Analysis

We detail the performance of the application based on observations from collected measurements. Data measured from the trace of each phase is put together into a multiplicative analytical model, as shown in (1).

$$\eta_{||} = LB * Ser * Trf. \quad (1)$$

The model decomposes the parallel efficiency metric ($\eta_{||}$) as a product of factors with normalized values between 0 (very bad) and 1 (perfect) [2]. The factors correspond to fundamental components of parallel coded and are load balance (*LB*), serialization (*Ser*) and transfer (*Trf*)[1].

C. Scalability Prediction

By independently extrapolating the individual components of this model we can observe how relevant they are to the overall performance of the parallel code. The basic default model is the Amdahl's law formulation. This is a first approach to describe the effect of non-parallel regions, where inefficiencies are caused by an activity that cannot be executed concurrently. Other possible patterns of concurrency correspond to pipelined computations or suggest a constant value when there are no changes in efficiency when scaling.

III. EXPERIMENTAL EVALUATION

We use three applications form the CORAL suite: HACC [3], Nekbone [4] and AMG2013 [5]; and the CFD application AVBP [6]. CORAL applications were executed in MareNostrum III, and AVBP was executed in Juropa. The machines operated in normal production using fully populated nodes. Executions are summarized in Tab. I.

TABLE I
APPLICATIONS USED FOR THE EXPERIMENTS

Strong Scaling	Ranks	Weak Scaling	Ranks
HACC	16, 32, 64, 128, 256, 512, 1024, 2048, 4096	AMG2013	32, 64, 96, 128, 192, 256, 384
Nekbone	2, 4, 8, 16, 32, 64, 128, 256, 512	AVBP	16, 32, 64, 96, 128, 192, 256, 520, 768, 1024, 1040, 1280, 1536

A. Identify Structure

From a manual process, we obtain clean cuts of interesting phases within the execution; for example, one iteration of HACC shows a large computationally intensive phase of around 250 sec (left side of Fig.1), with low communications

[1]For more details, please refer to [3].

and some imbalances at the end. It also has a small communication phase (4 sec, at the right of Fig. 1).

B. *Phase Performance Analysis*

With the automatic framework, we extract the main performance metrics for each phase. Because of the space, we show here only the results for HACC. In this application, the evolution of the three factors for the computational phase (left side in Fig. 2) describes a highly efficient region with almost none unbalance or contention, what suggest that it scales well. In the communication phase, it can be seen how Transfer and Serialization are dominant factors in the overall performance lost.

C. *Scalability Prediction*

- Model fitting: Amdahl's Law reflects the presence of contention when increasing parallelism. In consequence, we fit the fundamental factors of most of the applications following the Amdahl model by default. Inside phase 2 of HACC (Fig. 3 right), processors seem to perform computations in stages we adapted the fitting model to a pipeline behavior.
- Validation of results: Our method extrapolates expected efficiencies for a specific machine from traces obtained from it. The error of prediction for each factor is summarized in Tab. II, it shows how optimistic (+) or pessimistic is our prediction (-). We used executions of HACC from 16 to 256 ranks to predict the efficiency for the next values, and compared the real measurements with the expected efficiencies.
- Projection for large core counts: our framework extrapolates the expected total parallel efficiency for up to 10^6 cores. Phase 1 of HACC shows in Fig. 4 (left) a constant behavior for serialization and transfer, and a soft degradation of load balance. For phase 2, we expect that at 2k cores the parallel efficiency is below 0.4, thus suggesting that main problem will be communication contention.

TABLE II
PREDICTED EFFICIENCY AND RELATIVE ERROR FOR LARGER CORE COUNTS HACC (PHASE 2), EXTRAPOLATED FROM RUNS USING 16 TO 256 CORES

Ranks	Load Balance	Serialization	Transfer	Parallel Efficiency
512	0.952 (-0.82%)	0.860 (+1.81%)	0.517 (-6.78%)	0.424 (-5.61%)
1024	0.948 (-0.17%)	0.857 (+2.34%)	0.452 (-15.47%)	0.368 (-13.64%)
2048	0.943 (-0.68%)	0.855 (+2.45%)	0.391 (-9.39%)	0.315 (-7.80%)
4096	0.937 (+0.82%)	0.853 (+1.83%)	0.333 (-27.19%)	0.267 (-25.25%)

IV. CONCLUSIONS

In this paper, we presented a methodology to collect primary components of current parallel codes at low core counts and infer their expected behavior when scaled to larger core counts. We evaluated the method with 3 applications from the CORAL suite and a CFD application in two different machines. Scalability projections showed initial insights of expected parallel efficiency and obtained results seem promising to provide our methodology as a tool to infer the scalability of parallel codes.

ACKNOWLEDGMENT

This work has been done thanks to the Intel-BSC Exascale Lab and the DEEP Project.

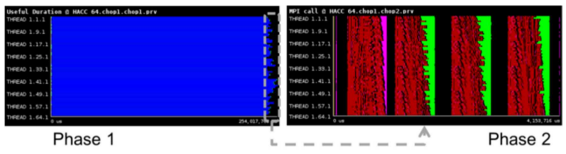

Fig. 1. Phases within one iteration of HACC code.

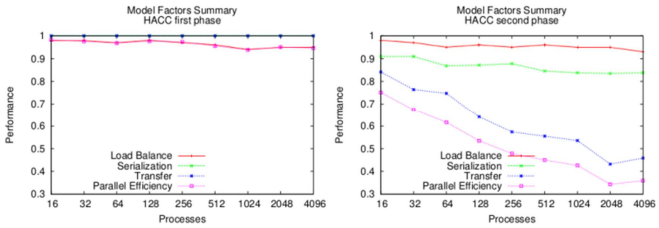

Fig. 2. Fundamental factors phase 1 and 2 for HACC

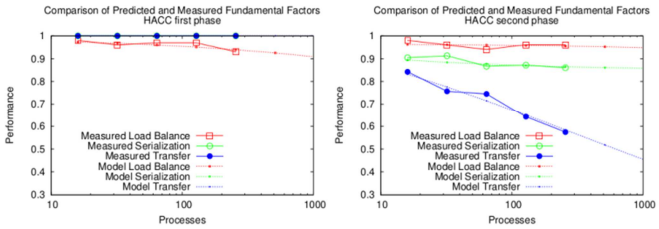

Fig. 3. Fitting phase 1 and 2 for HACC

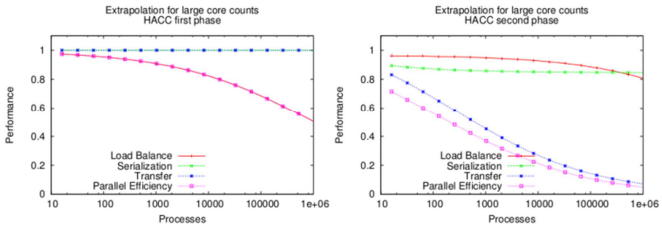

Fig. 4. Extrapolation phase 1 and 2 for HACC

REFERENCES

[1] A. Geist and R. Lucas, «Major Computer Science Challenges At Exascale,» *Int. J. of High Perform. Comput.*, vol. 23, nº 4, pp. 427-436, 2009.

[2] M. Casas, R. M. Badia and J. Labarta, «Automatic Analysis of Speedup of MPI Applications,» de *Proc. 22nd Intl. Conf. on Supercomputing*, 2008.

[3] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumara, T. Peterka, J. Insley, D. Daniel, P. Fasel, N. Frontiere and Z. Lukic, «The Universe at Extreme Scale: Multi-petaflop Sky Simulation on the BG/Q,» de *Proc. Intl. Conf. on High Performance Computing, Networking,*

Storage an Analysis., 2012.

[4] Center for Exascale Simulation of Advanced Reactors. Reactors, «Proxy-Apps for Thermal Hydraulics,» [En línea]. Available: https://cesar.mcs.anl.gov/content/software/thermal_hydraulics. [Last access: May 2014].

[5] V. E. Henson and U. M. Yang, «BoomerAMG: A parallel algebraic multigrid solver and pre-conditioner,» *Appl. Numer. Math.*, vol. 41, nº 1, pp. 155-177, 2002.

[6] N. Gourdain, L. Gicquel, M. Montagnac, O. Vermorel, M. Gazaix, G. Staffelbach, M. Garcia, J. Boussuge and T. Poinsot, «High performance parallel computing of flows in complex geometries: I. Methods,» *Comp. Sci. Disc*, vol. 2, nº 1, p. 015003, 2009.

[7] C. Rosas, J. Giménez and J. Labarta, «Scalability prediction from fundamental performance factors,» *Supercomputing Frontiers and Innovations. An Intl. J.*, vol. 1, nº 2, pp. 4-19, 2014.