A Semi-Empirical Approach to Scalability Study *

Xiaodong Zhang Zhichen Xu
High Performance Computing and Software Laboratory
The University of Texas at San Antonio
San Antonio, Texas 78249
zhang, zxu@ringer.cs.utsa.edu

1 Three Limitations in Existing Studies

A major limitation in existing scalability studies is the lack of effective ways to precisely predict the scalabilities. Experimental and simulation methods are highly time consuming. Pure analytical methods usually focus on asymptotic behaviors of a parallel system, and are only applicable to simple problem/architecture structures.

Secondly, how to evaluate the scalability of applications and systems as independently as possible is a complex issue. In [1], algorithm scalability is defined as the maximum achievable speedup on an architecture with an idealized communication structure. This speedup measures the inherent parallelism and overhead patterns of a program for a given size. To isolate program effects, the architecture scalability for a program is defined as the ratio of the speedup of the program on a real machine with the asymptotic speedup of the program on an EREW PRAM. An analytical model of a program structure may not precisely and completely describe overhead patterns of the program. The architectural scalability is still, to a great extent, dependent on the application program executed. In [5], latency is divided into memory reference latency, processor idle time, and parallel primitive execution overhead time, which comprehensively covers the major sources of system/architecture overheads. However, the same type of latency in this classification may also come from both the program and the machine. It is still difficult to identify and distinguish the performance bottlenecks from the hardware and the algorithm. In [2], parallel computing overheads are classified into the algorithmic overhead and the interaction overhead, which effectively isolates the overhead from the application program. However, it does not provide direct information about how performance changes when the program and the machine are scaled. In addition, the simulation-based approach is highly computation intensive.

*This work is supported in part by the National Science Foundation under grants CCR-9102854 and CCR-9400719, and by the Air Force Office of Scientific Research under grant AFOSR-94NM151 Part of the experiments were conducted on the CM-5 machines in Los Alamos National Laboratory and in the National Center for Supercomputing Applications at the University of Illinois, and on the KSR-1 machines at Cornell University and at the University of Washington.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association of Computing Machinery.To copy otherwise, or to republish, requires a fee and/or specific permission.
SIGMETRICS '95, Ottawa, Ontario, Canada
© 1995 ACM 0-89791-695-6/95/0005 ..$3.50

Finally, although the concept problem size is essential in the definitions of scalability, most scalability studies consider either the input data size of a problem, or the number of floating point operations in the computation as the problem size. For applications with more than one parameter, the observed scalabilities can be significantly different when the application parameters are scaled differently. The existing definitions of problem size simplify program scaling behavior.

To address these limits, we make two extensions to the latency metric in [4], and propose a semi-empirical approach to scalability evaluation. The objectives of this study are to significantly reduce simulation and measurement cost, to isolate algorithm scalability from algorithm-machine combination scalability, and to further investigate effects of architectures, algorithms and programming models on parallel computing scalability.

2 Extensions to the Latency Metric

Scalability measures the ability of a parallel system to improve performance when the sizes of the program and the machine are scaled. The latency metric defines algorithm-machine combination scalability as an average increase of the latency (L) needed to keep its computation efficiency equal to a constant (E) when the size of a parallel program increases from W to W', and the size of the parallel architecture increases from N processors to N' processors. It is expressed as

$$scale(E,(N,N')) = \frac{L(W,N)}{L(W',N')}. \quad (1)$$

Using (1), we obtain an upper triangular table, where each element represents the scalability measurement of (1) between a meaningful pair of processors.

To isolate the scalability of the algorithm from an algorithm-machine combination, we define the algorithmic scalability of a program based on the definition of algorithmic efficiency, E_a, and latency, L_a, of the program.

$$E_a(\Pi,\chi,N) = \frac{T_p^{\mathcal{M}}(\Pi,\chi,N)}{T_p^{\mathcal{P}}(\Pi,\chi,N)}, \quad (2)$$

$$\mathcal{L}_a(\Pi,\chi,N) = T_p^{\mathcal{P}}(\Pi,\chi,N) - T_p^{\mathcal{M}}(\Pi,\chi,N). \quad (3)$$

where Π is the parallel program, χ are the application parameters, N is the number of processors employed, $T_p^{\mathcal{M}}(\Pi,\chi,N)$ is the parallel execution time of Π on a MIRACLE machine that always exhibits linear speedup regardless

of the characteristic of the program executed, $T_p^{\mathcal{P}}(\Pi, \chi, N)$ is the parallel execution time of Π on an idealized PRAM machine that has an ideal communication structure, and $T_p(\Pi, \chi, N)$ is the parallel execution time of Π on the real machine. In (2), E_a measures the percentage of execution time that is dedicated to meanful computation on an ideal machine — the efficiency of the algorithm.

The algorithmic scalability of a parallel application is defined as the amount of the increase in algorithmic latency to keep a desired algorithmic efficiency, $E_a \in [0, 1]$, that is

$$Scale_a(E_a, (N, N')) = \frac{\mathcal{L}_a(\Pi, \chi, N)}{\mathcal{L}_a(\Pi, \chi', N')}, \qquad (4)$$

where N' is a larger number of processors than N, χ' is the scaled application parameter to keep $E_a = E_a(\Pi, \chi', N') = E_a(\Pi, \chi, N)$. The larger the increase of the algorithmic latency to keep a desired algorithmic efficiency, the smaller the algorithmic scalability will be. This definition of algorithmic scalability is, to a great extent, independent of the parallel machine employed.

The second extension to the latency metric is the replacement of problem size W for the application parameters, χ. This enables us to explicitly keep track the change of application parameters in studying scalability.

3 Outline of the Semi-Empirical Approach

The semi-empirical approach is based on a two-level hierarchical performance model. The kernels of the high-level model are a graphical representation of a program, called the *thread graph*, and a graphical algorithm. The task-level behavior of a program is modeled by a thread graph, in which the computation demands of the program are captured and expressed as functions of application and architectural parameters, and all explicit communications are isolated and classified. The graphical algorithm estimates the execution time of the entire program with information provided by the lower level model. In the lower model, the elapsed time of individual segments in the thread graph are captured based on a small sample run of the program. Implicit and nondeterministic system effects are obtained through experimental measurements.

There are some advantages to the semi-empirical approach in comparison with pure analytical, pure experimental, or simulation approaches. First, performance predictions are based on the small-scale execution of the program, and thus the time is significantly reduced from experiments and simulations. Second, the representation of a program as a thread graph enables the program's overhead pattern to be studied according to its synchronization and communication structures. Third, since experimental measurement is used to estimate the complex system effects, this performance model is less dependent on real architectures. Finally, because important and implicit system effects are obtained through experimental measurements, the semi-empirical performance model should be more precise than pure analytical models or pure simulations.

4 Preliminary Performance Results

To validate the semi-empirical scalability evaluation method, and to further investigate effects of architectures, algorithms and programming models on parallel computing scalability, we implemented three applications on two different machines. The three applications are *All Pairs Shortest Path* (APSP), *Gauss Elimination* (GE) and a large *Electromagnetic* (EM) *Simulation* application. The two parallel architectures are the KSR-1 and CM-5.

Our preliminary performance results indicate that the time to estimate the scalability is significantly reduced compared with pure experimental and simulation methods. For instance, it took 395.5 seconds to obtain the combination scalability of a shared memory implementation of the GE program on the KSR-1, and 53652.7 seconds for the combination scalability of the APSP program on the KSR-1 using the pure experimental method in the best case. In contrast, it only took 1.08 and 0.76 seconds respectively using the semi-empirical method (the time to collect the sample data is excluded) — approximately 366 times faster for the GE program and 70595 times faster for the APSP program. Besides the time reduction, if the program abstraction is sufficiently detailed, the predicted results can be very precise. According to our experiments, the average error is less than 10% in most cases. The average is only about 7%.

Using different program implementations on the two different machines, we are able to compare the effects of the shared-memory and data-parallel programming models on a program's algorithmic scalability, and the effects of the architectures on a program's combination scalability. Different programming models can significantly affect the algorithmic scalability of a program because the parallelisms inherent in the program are exploited differently. Even though the differences among algorithmic scalabilities can be moderate for different implementations of a program, the combination scalabilities can be significantly different. This indicates that different communication and synchronization patterns and structures, and different architectural support, affect performance differently.

Acknowledgement: We would like to thank Yong Yan for many technical discussions about this topic. We appreciate Neal Wagner for his careful reading of the original technical report.

References

[1] D. Nussbaum, A. Agarwal, "Scalability of parallel machines", *Communication of the ACM*, March 1991, Vol. 34, No. 3, pp. 57-61.

[2] A. Sivasubramaniam, A. Singla, U. Ramachandran, H. Venkateswaran, "An approach to scalability study of shared memory parallel systems", *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1994, pp. 171-179.

[3] X. Zhang, Z. Xu, L. Sun, "Performance prediction on implicit communication systems", in *Proceedings of the Sixth IEEE Symposium of Parallel and Distributed Processing*, IEEE Computer Society Press, Oct. 1994, pp. 560-568

[4] X. Zhang, Y. Yan, K. He, "Latency metric: an experimental method for measuring and evaluating parallel program and architecture scalability", *Journal of Parallel and Distributed Computing*, Vol. 22, No. 3, 1994, pp. 392-410.

[5] X. Zhang, Y. Yan and K. He, "Evaluation and measurement of multiprocessor latency patterns", in *Proceedings of the 8th International Parallel Processing Symposium*, IEEE Computer Society Press, April, 1994, pp. 845-852.