Performance Modeling for Dynamic Algorithm Selection

Michael O. McCracken[1], Allan Snavely[2], and Allen Malony[3] *

[1] Department of Computer Science, University of California, San Diego
mike@cs.ucsd.edu
[2] San Diego Supercomputer Center
allans@sdsc.edu
[3] Department of Computer and Information Science, University of Oregon
malony@cs.uoregon.edu

Abstract. Adaptive algorithms are an important technique to achieve portable high performance. They choose among solution methods and optimizations according to expected performance on a particular machine. Grid environments make the adaptation problem harder, because the optimal decision may change across runs and even during runtime. Therefore, the performance model used by an adaptive algorithm must be able to change decisions without high overhead. In this paper, we present work that is modifying previous research into rapid performance modeling to support adaptive grid applications through sampling and high granularity modeling. We also outline preliminary results that show the ability to predict differences in performance among algorithms in the same program.

1 Introduction

Grid environments [1] present novel performance challenges, adding variability to many characteristics of high performance code. Heterogeneous platforms and varying network performance mean that the best algorithm for an application may change between runs of an application, and even during execution. Adaptive algorithms, developed to support portable performance in libraries, present an excellent opportunity to deal with these challenges by switching algorithms based on runtime information. To choose the optimal algorithm, a performance prediction must be made based on this information and the performance characteristics of the candidate algorithms. Because it is important to keep the combined overhead of measurement, modeling, prediction, and adaptation low, current time-consuming modeling techniques are not suitable for grid environments. We propose using a combination of ongoing research into rapid performance modeling and new development of a general adaptive algorithm framework to support exploration of portable performance on Grids.

We will initially focus our research on data-intensive applications running on Virtual Private Grids (VPGs), which combine the heterogeneity and physically

* We would like to acknowledge the help of Wayne Pfeiffer.[2]

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 749–758, 2003.

distributed qualities of a public Grid, without the complication of distributed administration. Examples of important VPGs that this work is applicable to include GAMESS (General Atomic Molecular Electronic Structure Systems) Portal [2, 3], Encyclopedia of Life (EOL) [4], Biomedical Informatics Research Network (BIRN) [5], GriPhyn (Grid Physics Network) [6] projects, and Real-time observatories, applications, and data-management (ROADNet) [7].

2 Rapid Performance Modeling

Performance modeling is the science of constructing analytical and empirical models to predict and explain the performance of applications. Past techniques involve very detailed low-level predictions on a single architecture, as in [8–10], or developing custom models for single applications, for example [11]. Our goal is to develop general methods fast enough to update models at runtime, therefore such methods requiring expensive cycle-accurate simulation, slow evaluation, or human intervention are not suitable. Because it is sometimes possible to amortize the cost of more detailed analysis, we are investigating the tradeoff between analysis speed and prediction accuracy.

Current techniques for rapid performance modeling include convolution and Petri net methods. These approaches combine properties of abstraction and rapid computability. Only the most significant properties of an application are measured to facilitate building a model quickly. Our current approach uses convolution methods to model node-level performance, and Petri nets to model network performance. Current work using these methods has had success with HPC application modeling [12], so we propose to use a similar approach for VPGs.

2.1 Convolution Methods

It is now feasible to rapidly estimate the performance of an application across several candidate platforms, using techniques shown in [12–14]. We have developed a framework for automatic generation of performance models for whole HPC applications through convolution of application signatures with system profiles. Application signatures are summaries of the operations performed during execution. Ideally, signatures are machine independent. System profiles are measurements of the rates at which machines can carry out fundamental operations such as floating-point operations, memory accesses, message transfers, and disk accesses. Convolution methods are techniques for mapping signatures to profiles for predicting and understanding performance. The separation of signatures and profiles means that signatures need only be gathered once to support prediction on multiple machines. Convolutions are fast and allow exploration of varying accuracy by adding detail to application signatures.

2.2 Petri Net Methods

A Petri net is a tool for modeling concurrent systems. Details about Petri Nets are available in the references of [15], which describes Petri nets applied to Grid

performance prediction. In previous work with Petri nets using the DIMEMAS tool [16], we found that the performance of many data-intensive HPC applications is bottlenecked by their interaction with the memory hierarchy while their scalability is mostly a function of their interaction with the interconnect. Thus our application signatures focused on memory access patterns and communications patterns.

Our focus on rapid modeling of deep memory hierarchies and multi-tiered networks makes our previous modeling work most pertinent for modern HPC architectures. The fact that these same architectural factors are even more influential in Grid performance, along with the need to rapidly respond to changing network conditions and data sparsity, suggests that our methods will be useful to predict performance on Grids rapidly and cheaply.

3 Algorithmic Selection and Adaptation

Adaptive algorithms, also called poly-algorithms, encapsulate a number of algorithms for solution of the same numerical problem, along with a mechanism for selecting the best algorithm amongst the available alternatives [17–19]. The software mechanism responsible for making the determination of the best available choices at run-time is known as a switching function. The optimal choice of algorithm, can be determined at run time, typically using data such as measurements of hardware characteristics, system load, and input-data dependent properties of the application. Prototypes for adaptive dense matrix multiplication such as ATLAS [19], and FFTs [20, 21] show that adaptive algorithms can frequently do as well as or even better than hand-tuned vendor code.

Currently, adaptive algorithms and the switching functions employed by them run within standard HPC computing environments such as a tightly coupled cluster of SMP nodes [22] The next generation of adaptive algorithms must be able to run within distributed computational environments such as the TeraGrid [23], the NASA Information Power Grid (IPG) [24], or the Grid Physics Network (GriPhyN) [6], with heterogeneous compute resources and varied and fluctuating network bandwidths and latencies. Therefore, the ability to update algorithms choices automatically at runtime will become more important.

4 Applications of Performance Modeling in Adaptive Grid Software

Rapid performance modeling techniques can be used to enable real-time adaptive decisions in VPG environments. Such environments require rapid decisions about the best combination of software and hardware resources for the given application at a particular time. We are extending current rapid modeling techniques to grid applications, by identifying additional performance factors which must be measured to model grid applications. These include machine availability and heterogeneity, increased variance in network characteristics, and grid middleware service overhead.

We will continue to use convolution-based modeling to capture compute node-level characteristics of the VPGs such as expected CPU and memory subsystem performance. Petri net modeling will be used to model higher level characteristics such as varying network load, middleware service overheads, and machine availability. As our previous modeling work assumed a relatively fixed-state computing platform, our new research will explore how to make performance models capture changing conditions. Possible approaches we are evaluating include sampling of performance characteristics. We are currently extending the tracing technology to use the DynInst run-time code patching API [25] to allow sampling.

Present convolution methods gather signatures for an entire application. Using DynInst, we are extending the signatures to describe individual procedures and loops. This additional capability will allow us to use the convolution method to predict performance of candidate algorithms within an adaptive program.

5 Preliminary Results

Current investigations have already shown promising results in whole-application performance modeling with rapid evaluation [13]. To support modeling at a finer granularity to enable comparisons of candidate algorithms, we have extended the convolution method to the loop level. We are developing a tool similar to MetaSim [12] that gathers information at the loop level.

In order to validate that the convolution approach can work at the loop level, we show the results of predicting performance of a set of 8 synthetic loops which represent a range of memory usage and computational intensity. The loops were padded to avoid cache conflicts, and compiled using the IBM compiler with the following options: `xlf90 -O3 -qtune=pwr3 -bmaxdata:0x40000000`. Table 1 shows a summary of the loops studied. The experiments were performed on the SDSC Blue Horizon IBM SP, with 375 MHz Power3 processors.

Figure 1 shows the results of our convolution under development for use with loops. A simple linear fit using the L1 and L2 cache miss rates predicts the efficiency of a loop very well. The units are efficiency in terms of the limiting operation, which is either memory references or floating-point operations, depending on the loop. This set of loops will be expanded in ongoing work to include strided access, tiling optimizations, and then on to loops from application kernel benchmarks, to support building a robust convolution for modeling. These results were generated by hand and show that the dynamic instrumentation tool under development, which includes the fitted variables among those it measures, has the potential to support useful performance models.

6 Related Work

Past related work in the areas of performance modeling and adaptive algorithms are outlined in the following sections.

Table 1. Loops studied

Loop	Memory References per Iteration	FP operations per Iteration	Computational Intensity
fill			
$a(i) = s$	1	0	0
copy			
$a(i) = b(i)$	2	0	0
vadd			
$a(i) = b(i) + c(i)$	3	1	0.33
sum			
$s = s + a(i)$	1	1	1.0
daxpy			
$a(i) = a(i) + s * b(i)$	3	2	0.67
dot			
$s = s + a(i) * b(i)$	2	2	1.0
poly2	2	4	2.0
$a(i) = (c2 * b(i) + c1)$ $*b(i) + c0$			
poly5	2	10	5.0
$a(i) = ((((c5 * b(i) + c4)$ $*b(i) + c3) * b(i) + c2)$ $*b(i) + c1) * b(i) + c0$			

6.1 Performance Modeling

Gustafson and Todi first proposed the convolution method in [26]. Because cycle-accurate simulation is slow, there is a range of techniques that have been developed to improve its efficiency [10, 27–31]. These techniques are useful in domains such as architecture design but fall short of the rapid estimation capability needed for adaptable grid applications. To enable more rapid performance estimation, Saavedra [32–34] proposed to model applications as a collection of independent Abstract Fortran Machine tasks. For parallel system predictions, Mendes [35, 36] has proposed a cross platform approach. Traces are used to record the explicit communications among nodes and to build a directed graph based on the trace. Then sub-graph isomorphism is used to study trace stability and to transform the trace for different machine specifications. Simon [37] proposed to use a Concurrent Task Graph to model applications using the dependence relationship between nodes.

Scheduling using performance contracts is a current topic of grid research which requires models of expected performance. The GrADS Project [38] work on performance contracts measures the intrinsic application signature, monitors system processing and communication rates, and uses fuzzy logic to determine whether the application processing and communication speeds for each task are within the bounds defined by the user-specified performance contracts. However, algorithms for adjusting the application and/or system resource configuration

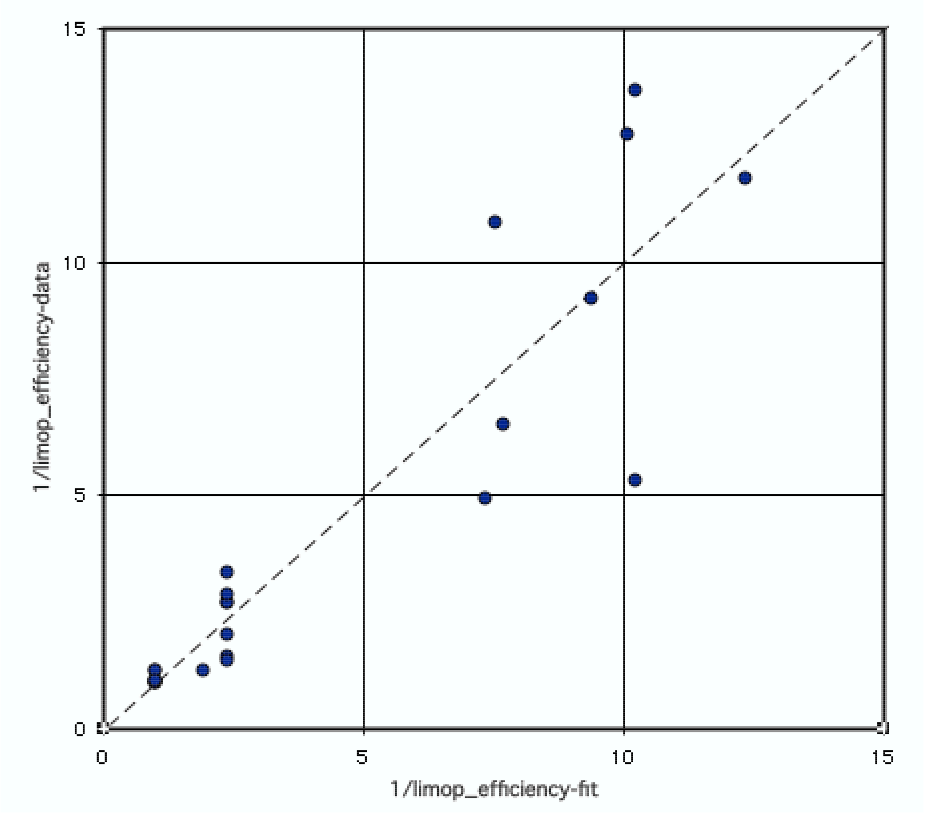

Fig. 1. Efficiency vs. predicted efficiency for loops in Table 1.

when the performance contract is violated are not yet addressed in those works. Furthermore, the performance models developed so far for GrADS are ad hoc and application-specific, for example the model developed for ScaLAPACK in [39]. Heymann et al. [40] measure task execution times and worker node efficiencies in each iteration of an executing Condor Master-Worker (MW) application. Results from previously measured synthetic MW applications provide a table of the estimated number of worker nodes to allocate, as a fraction of the number of parallel tasks, to achieve 80% efficiency with no more than a 10% increase in iteration execution time, based on the relative processing times of the largest 20% of the tasks. This algorithm assumes homogeneous processing nodes, that each task performs (approximately) the same work in successive iterations, homogeneous processing nodes, and that the application is synchronous (i.e., the results from one iteration are completed before a new iteration executes) while our target VPGs are heterogeneous and many of our applications are asynchronous. Hollingsworth and Keleher [41] propose an approach in the Harmony system, in which the application developer specifies the processing time and communication

time, or provides a model to predict these values at runtime, for each possible multidimensional configuration of the application. The Harmony system then dynamically allocates resources to each executing application to achieve a particular system objective (rather than application objective), such as maximizing throughput. By contrast our work is applications-centric.

The Delphi project [42] uses a language-directed whole-system approach to performance prediction and analysis. Compilers, instrumentation, custom libraries, and analysis tools all work together to support performance modeling. In contrast, our work does not require the same extensive system support. We focus on developing tools that are as lightweight as possible, specifically requiring minimal change in the application workflow. This focus will allow us to build performance models of existing applications and enhance decisions made by their adaptive libraries transparently.

6.2 Adaptive Algorithms

Some of the well-known numerical software packages with adaptive algorithms are the Linear Solver package (LINSOL) [43], ATLAS [19], HPL [44], Lapack for Clusters (LFC) [22], FFTW [20], and UHFFT [21]. The primary issue is how some of these packages use parameters for switching functions. In LINSOL, adaptive algorithms are chosen based on different numerical characteristics and their effect on convergence. All of these packages modify computation patterns in order to deliver high performance. Other aspects on which algorithms may adapt are data storage and communication patterns. All of these packages differ on criteria for switching functions based on functionality and goals specific to their application. In an alternative approach, Brewer made an extensive treatment of adaptive algorithms to achieve portable high performance sensitive to machine and instance issues [45]. The switching functions employed by Brewer are based upon statistically generated data from a database of empirical performance data. This approach, though highly accurate, is expensive, as shown by Sussman [46].

7 Conclusions and Future Work

Our targeted data-intensive Grid applications make substantial use of libraries, some of which already have adaptive features. Switching functions for Grid applications will need additional capabilities beyond those currently available on HPC platforms to deal with heterogeneity in compute resources and varying network conditions. Future work will investigate generalizing adaptive algorithms' switching functions to use output from general purpose performance models. These new switching functions will need access to a wider range of data, including past performance history, available machine characteristics, and configuration information about the Grid environment, including topology and network conditions. Custom development of performance models for each application is already time consuming, and this additional data will make it more so. Therefore, adaptive algorithms will benefit from separate development of automatic

performance models. Switching functions will then easily employ runtime performance estimates for making algorithm and machine selections, using the present work on rapid performance modeling.

References

1. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
2. Baldridge, K.K., Greenberg, J.P., Elbert, S.T., Mock, S., Papadopoulos, P.: QMView and GAMESS: Integration into the world wide computational grid. In: Proceedings of Supercomputing. (2002)
3. Schmidt, M., Baldridge, K.K., Boatz, J.A., Elbert, S., Gordon, M., Jenson, J.H., Koeski, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., Montgomery, J.A.: The general atomic and molecular electronic structure system. Journal of Computational Chemistry **14** (1993) 1347 – 1363
4. EOL: Encyclopedia of life (2003) http://eol.sdsc.edu/.
5. BIRN: Biomedical informatics research network (2003) http://www.nbirn.net/.
6. GriPhyN: Grid physics network (2003) http://www.griphyn.org/.
7. ROADNET: Real-time observatories, applications, and data management network (2003) http://roadnet.ucsd.edu/.
8. Gibson, J., Kunz, R., Ofelt, D., Horowitz, M., Hennessy, J., Heinrich, M.: FLASH vs. (simulated) FLASH: Closing the simulation loop. In: Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). (2000) 49–58
9. Chatterjee, S., Parker, E., Hanlon, P.J., Lebeck, A.R.: Exact analysis of the cache behavior of nested loops. In: Proceedings of the International Symposium on Programming Language Design and Implementation (PLDI), ACM, ACM (2001)
10. Weikle, D., Skadron, K., McKee, S., Wulf, W.: Caches as filters: A unifying model for memory hierarchy analysis. Technical Report 16, University of Virginia Department of Computer Science (2000)
11. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.: Predictive performance and scalability modeling of a large-scale application. In: Supercomputing, Denver (2001)
12. Snavely, A., Wolter, N., Carrington, L.: Modeling application performance by convolving machine signatures with application profiles. In: IEEE Workshop on Workload Characterization. (2002)
13. Snavely, A., Carrington, L., Wolter, N., Labarta, J., Badia, R., Purkayastha, A.: A framework for performance modeling and prediction. In: Supercomputing. (2002)
14. Carrington, L., Wolter, N., Snavely, A.: A framework for application performance prediction to enable scalability understanding. In: Scaling to New Heights Workshop, Pittsburgh (2002)
15. Anglano, C.: Predicting parallel application performance on non-dedicated cluster platforms. In: Proceedings of the 12th ACM International Conference on Supercomputing (ICS), Melbourne, Australia (1998)
16. Dimemas: Internet (2003) http://www.cepba.upc.es/dimemas.
17. Im, E., Yelick, K.: Optimizing sparse matrix-vector multiplication on smps. In: Ninth SIAM Conference on Parallel Processing for Scientific Computing, SIAM (1999)

18. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.0.29, Argonne National Laboratory (2000)
19. Whaley, R.C., Petitet, A., Dongarra, J.J.: Automated empirical optimizations of software and the ATLAS project. Parallel Computing **27** (2001) 3–35 http://math-atlas.sourceforge.net/.
20. Frigo, M., Johnson, S.: FFTW: An adaptive software architecture for the FFT. In: Proceedings of ICASSP. Volume 3. (1998) 1381 – 1384
21. Mirkovic, D., Johnsson, S.: Automatic performance tuning in the UHFFT library. In: International Conference on Computational Science. (2001)
22. Roche, K., Dongarra, J.J.: Deploying parallel numerical library routines to cluster computing in a self adapting fashion. http://icl.cs.utk.edu/news_pub/submissions/dyn_pnumlibs.pdf (2002)
23. Teragrid: The national science foundation's teragrid project (2003) http://teragrid.org/.
24. Johnston, W.E., Gannon, D., Nitzberg, B.: Grids as production computing environments: The engineering aspects of NASA's information power grid. In: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing. (1999)
25. Buck, B., Hollingsworth, J.K.: An API for runtime code patching. The International Journal of High Performance Computing Applications **14** (2000) 317–329
26. Gustafson, J.L., Todi, R.: Conventional benchmarks as a sample of the performance spectrum. The Journal of Supercomputing **13** (1999) 321–342
27. Sherwood, T., Perelman, E., Hamerly, G., Calder, B.: Automatically characterizing large scale program behavior. In: International Conference on Architectural Support for Programming Languages and Operating Systems. (2002)
28. Haskins, J., Skadron, J.: Minimal subset evaluation: Rapid warm-up for simulated hardware state. In: Proceedings of International Conference on Computer Design. (2001)
29. Lafage, T., Seznec, A.: Choosing representative slices of program execution for microarchitecture simulations: A preliminary application to the data stream. In: Proceedings of the Third IEEE Annual Workshop on Workload Characterization. (2000)
30. Conte, T., Hirsch, M., Menezes, K.: Reducing state loss for effective trace sampling of superscalar processors. In: Proceedings of IEEE International Conference on Computer Design, IEEE Computer Society (1996) 468–477
31. Nussbaum, S., Smith, J.E.: Modeling superscalar processors via statistical simulation. In: 10th International Conference on Parallel Architectures and Compilation Techniques (PACT'01), Barcelona (2001)
32. Saavedra, R.H., Smith, A.J.: Analysis of benchmark characteristics and benchmark performance prediction. ACM Transactions on Computer Systems **14** (1996) 344–384
33. Saavedra, R.H., Smith, A.J.: Measuring cache and TLB performance and their effect on benchmark runtimes. IEEE Transactions on Computers **44** (1995) 1223–1235
34. Saavedra, R.H., Smith, A.J.: Performance characterization of optimizing compilers. Software Engineering **21** (1995) 615–628
35. Mendes, C.L., Reed, D.A.: Integrated compilation and scalability analysis for parallel systems. In: IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT). (1998) 385–392

36. Mendes, C., Reed, D.: Performance stability and prediction. In: Proceedings of the IEEE/USP International Workshop on High Performance Computing, Sao Paulo (1994)
37. Simon, J., Wierum, J.M.: Accurate performance prediction for massively parallel systems and its applications. In: Euro-Par 96. Volume 1124 of LNCS., France, Lion, Springer (1996)
38. Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crummey, J., Reed, D., Torczon, L., Wolski., R.: The GrADS project: Software support for high-level grid application development. International Journal of High Performance Computing Applications **15** (2001) 327 – 344
39. Petitet, A., S.Blackford, J.Dongarra, B.Ellis, G.Fagg, K.Roche, S.Vadhiyar: Numerical libraries and the grid: The GrADS experiment with ScaLAPACK. International Journal of High Performance Computing Applications **15** (2001) 359 – 374
40. Heymann, E., Senar, M.A., Luque, E., Livny, M.: Evaluation of an adaptive scheduling strategy for master-worker applications on clusters of workstations. In: HiPC. (2000) 310–319
41. Hollingsworth, J.K., Keleher, P.J.: Prediction and adaptation in active harmony. Cluster Computing **2** (1999) 195–205
42. Delphi: (2003) http://www-pablo.cs.uiuc.edu/Project/Delphi/DelphiOverview.htm.
43. Weiss, R., Haefner, H., Schoenauer, W.: LINSOL (LINear SOLver) - description and user's guide for the parallelized version (1995) http://www.uni-karlsruhe.de/ linsol/html/documentation.html.
44. Dongarra, J., Luszczek, P., Petiet, A.: The LINPACK benchmark: Past, present and future (2001) http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf.
45. Brewer, E.: High-level optimization via automated statistical modeling. In: Proceedings of PPOPP. (1995)
46. Sussman, A.: Model-driven mapping onto distributed memory parallel computers. In: Proceedings of the 1992 ACM/IEEE conference on Supercomputing, IEEE Computer Society Press (1992) 818–829