Quickstep: A Performance Monitoring and Debugging System for Alewife[*]

Sramana Mitra[†]
Alewife Research Group, MIT LCS
smitra@lcs.mit.edu

Introduction

Even though the peak performance rating of multiprocessor systems has improved substantially over the past several years, the initial implementation of parallel applications almost never harnesses the full processing power. Performance bottlenecks abound and it is difficult for the programmer to keep track of all aspects of performance optimization. Consequently, there is the need for tools to assist in performance debugging.

The Alewife machine is a large-scale multiprocessor with distributed shared memory built at the MIT Laboratory for Computer Science [1]. The Alewife Controller and Memory Management Unit (CMMU) has features to support performance monitoring in hardware. QuickStep utilizes these features to provide a performance monitoring and debugging platform.

Background

Several efforts have been directed at identifying performance bottlenecks in parallel programs [3].

Static Analysis involves predicting the performance of loops, counts of local and global memory references, estimates of MFLOPS, etc. based on simple models of instruction latencies and memory hierarchies. The techniques are in general inaccurate.

Simulation is a slow but precise method. It is reasonably accurate, but it is not realistic to simulate the behavior of an entire large program. Therefore, simulation is hardly an effective tool for performance debugging.

Emulation is a method of hardware system debugging that is becoming increasingly popular. Field-programmable gate arrays have made possible an implementation technology that is ideal for full system prototyping, yet does not require the construction of actual silicon chips [5].

Software Instrumentation is fast and flexible. Manually done, it involves instrumenting a program with write statements to print out special purpose information. More sophisticated tools involve automatic instrumentation. Software instrumentation, however, introduces inaccuracies due to their intrusive nature.

Hardware Instrumentation involves using dedicated counters and registers to monitor events. Monitoring of events occurs in hardware and hence is virtually unintrusive. The biggest advantages of hardware instrumentation are its accuracy and speed.

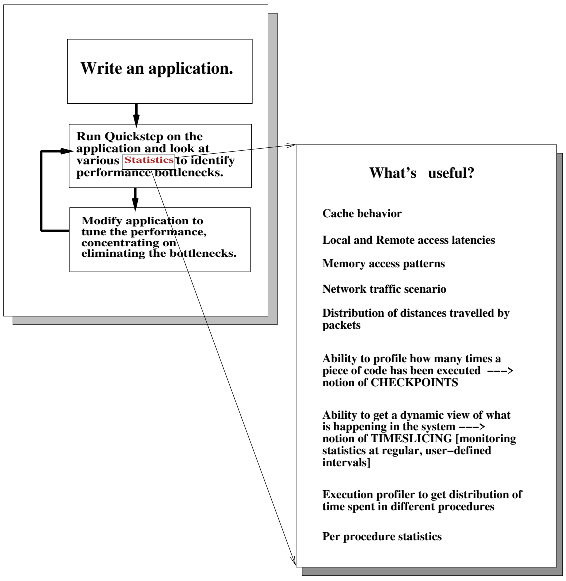

Figure 1: Flow chart of tuning the performance of an application using QuickStep.

The drawback of hardware instrumentation is that it is not widely available and it may not be as flexible as simulation. In our case, availability is not an issue since Alewife hardware was designed to support instrumentation counters. However, it is only possible to provide a finite amount of instrumentation support in hardware, so it is not as flexible as software. In Alewife, for example, we have 4 statistics counters that monitor a subset of all events. Therefore, only a finite set of events can be monitored during a single run. However, since runs can happen fast, multiple runs allow monitoring of larger sets of statistics. Furthermore, the event monitoring hardware was carefully architected so that most key events could be captured.

QuickStep takes a hybrid of hardware and software approaches and provides a friendly interface for viewing the data collected by the kernel.

Goals of the work

The main goal of this work has been to develop the QuickStep system with adequate features to provide a vehicle of

[*]This work was supervised by Anant Agarwal.

[†]The Alewife project is funded in part by ARPA contract # N00014-94-1-0985, in part by NSF Experimental Systems grant #MIP-9012773, and in part by NSF Presidential Young Investigator Award.

further research on the Alewife machine. QuickStep provides a platform for monitoring cache and memory statistics, network statistics, various latencies and message frequencies, etc. for applications that are run on Alewife. It thus enables users to analyze the performance characteristics, bottlenecks and enhancement potentials and to accordingly fine-tune applications. Figure 1 shows a flow chart of tuning the performance of an application using QuickStep. It also shows what kind of statistics are useful for performance debugging. In principle, QuickStep is capable of providing all those features, although some of the profiling features have not been implemented yet.

The Alewife CMMU provides basic hardware support for monitoring events. 11% of the total chip area of the CMMU is dedicated to performance monitoring hardware. However, it is not possible to utilize this feature without a well-developed software interface which can handle the bit manipulations and produce comprehensible information. QuickStep provides this interface, as well as a graphical display environment for the information gathered.

Features

The high-level interface for the QuickStep system is an integrated part of the general Alewife interface developed by Patrick Chan. It consists of menu items for:

- Connecting to the Alewife machine or to the NWO simulator (NWO is a simulator that has been developed as a part of the Alewife design process by David Chaiken)
- For selecting the statistics to monitor and display the graphical output of QuickStep
- For running *Parastat*— a graphical monitor of the status of the individual nodes of the Alewife machine (also developed by David Chaiken)

The code for the interface is written in Tcl, an X-windows programming environment. The user requests the statistics that he or she wants to monitor by clicking on the relevant menu items.

Statistics can either be recorded at the end of the run, or at regular intervals during the run. QuickStep provides both these options, allowing the user to get some amount of profiling information. Often, however, the user simply wants an overall summary statistic for the run. In such cases, the timesliced mode is turned off and the counter values are recorded at the end of the run. It is possible to post-process data gathered in time-sliced mode to obtain the overall statistics. However, as discussed in the previous section, timesliced statistics perturb program behavior, while non-timesliced statistics do not. Furthermore, non-timesliced mode is naturally faster than the timesliced mode. As always, there is a trade-off between accuracy of statistics, perturbation of program and speed. Hence, we provide both timesliced and non-timesliced modes of collecting statistics.

Future Work

The first major addition that is planned for the next version of QuickStep is a profiling feature. Currently, statistics gathering cannot be turned on or off in the middle of a run. However, this is a feature that would be of enormous usefulness. For instance, users of the QuickStep system have commented that it would be useful if a certain set of statistics could be computed on a per procedure basis. The statistics could be of various types: the amount of time spent in the procedure, the cache behavior and the network statistics for the procedure, etc.

This feature can be incorporated easily, by encoding the turning on and off of statistics counters into a function. Ideally, the user should be able to specify the name of the procedure and the statistics to be monitored. The compiler/linker would then incorporate the function in that procedure automatically, the process being transparent to the user.

Furthermore, there are several classes of statistics that the CMMU supports which have not been implemented in this version. These include *synchronous trap statistics*, *hit-miss statistics*, *remote transaction statistics*, *memory controller statistics* and *transaction buffer statistics*.

From a presentation point of view, we are currently using the Proteus [2] *Stats* program as the display environment. Most of the data we are presenting would be easier to read in a 3-dimensional graphical display environment, which stats does not support. There is room for improvement in the way the statistics are represented through graphical displays.

User Documentation

The Alewife Memo #41 contains elaborate documentation for using QuickStep.

References

[1] Anant Agarwal *et. al.* The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprcoessor. Proceedings of Workshop on Scalable Shared Memory Multiprocessors, 1991.

[2] Eric A. Brewer and Chrysanthos N. Dellarocas. *Proteus User Documentation.*

[3] Aaron J. Goldberg. Multiprocessor Peformance Debugging and Memory Bottlenecks. *Stanford Technical Report No. CSL-TR-92-542*, August 1992.

[4] S. Graham, P. Kessler and M. McKusick. Gprof: A call graph execution profiler. In *Proceedings of the ACM/SIGPLAN Symposium of Compiler Construction*, pp 120-126, 1982.

[5] Stephen Walters. Computer-Aided Prototyping for ASIC-Based Systems. IEEE Design and Test of Computers, 1991.