Final Report for
Cooperative Agreement
NASA Ames Research Center
NASA NCC 2-5284
Developing Information Power Grid Based Algorithms and Software

8/1/98 to 11/1/98
$50,000

Principal Investigator
Jack Dongarra
University of Tennessee, Knoxville
Computer Science Department

Technical Officer for the Cooperative Agreement
Subhash Saini
Numerical Aerospace Simulation Systems Branch, T27A-1

Grant Officer
Venoncia Braxton

This exploratory study initiated our effort to understand performance modeling on parallel systems. The basic goal of performance modeling is to understand and predict the performance of a computer program or set of programs on a computer system. Performance modeling has numerous applications, including evaluation of algorithms, optimization of code implementations, parallel library development, comparison of system architectures, parallel system design, and procurement of new systems.

Our work lays the basis for the construction of parallel libraries that allow for the reconstruction of application codes on several distinct architectures so as to assure performance portability. Following our strategy, once the requirements of applications are well understood, one can then construct a library in a layered fashion. The top level of this library will consist of architecture-independent geometric, numerical, and symbolic algorithms that are needed by the sample of applications. These routines should be written in a language that is portable across the targeted architectures.

An important basis for accurate performance prediction is the accurate characterization of parallel architectures such as the Intel Paragon, Cray T3E, SGI/Cray Origin 2000, and the IBM SP/2. Not only the architecture, but also the message passing, data parallel, and shared memory programming interfaces to these systems need to be characterized. After identifying an appropriate set of architectural and system parameters for the range of parallel computers under consideration, this research collected data on the key architectural features and software systems of these machines and stored this data in a publicly available architecture characterization database. The data collection used vendor specifications as well as the execution of parameter-specific low-level benchmarks specifically designed to evaluate particular features. The benchmarking employed maximizes the use of already available benchmark execution results to speed

data collection time and utilize low-level, vendor-supplied, math function optimization for a particular architecture to reproduce optimal application coding techniques. Candidate quantities considered included the following:

- single processor performance
- single processor memory bandwidth
- single processor cache effects
- multiple processor shared memory latency and bandwidth
- cost of communication between processors
- latency of communication operations
- communication bandwidth
- I/O capabilities

In our study we measured some of these quantities using our current set of benchmark codes. This was done to evaluate the appropriateness of different benchmarks and to gain the insight necessary to further develop more advanced benchmark programs. The benchmark codes used included:

- MPBench to evaluate the communication between processors
- CacheBench to characterize the single processor cache effects
- BLASBench to evaluate the single processor floating point performance

These programs were ported and executed on the major NAS IPG system, the SGI Origin 2000. To ensure fair and optimal measurements of the architectural and system parameters, we made specific changes to the programs in question. These three benchmark codes were then packaged together and made available as LLCbench at http://icl.cs.utk.edu/projects/llcbench/

A major goal of performance modeling, beyond the scope of the measurement of basic architectural and system parameters, is the understanding of the performance behavior of full complex application codes. But as described above, the programs and codes used in different areas of science differ in a large number of features. The performance of full application codes cannot be characterized in a general way, independent of the application and code used. The understanding of the performance characteristics is tightly bound to the specific computer program code used. Therefore the careful selection of an interesting program for analysis is the crucial first step in any more detailed and elaborate investigation of full application code performance.

Working jointly with NAS personal the computational fluid dynamics (CFD) code we identified OVERFLOW as a code of special interest to NAS due to its wide spread use in that community. Several different versions of this code exist. Due to the importance of this code, NAS has developed versions of OVERFLOW capable of parallel execution on a variety of systems. These parallel versions are of particular interest to the NAS research community.

Contact with developers and users of OVERFLOW at NAS was established. Through informal discussions the specific performance modeling needs of OVERFLOW users and developers were determined. The parallel execution of OVERFLOW requires several data layout choices, which greatly influence the achievable performance values.

The optimal choice of data layout depends in turn on several architectural and system parameters. Currently the choice of data layout is left to the user. At present smaller test runs and prior experience of users with different input data sets are typically used to make informed guesses about good data layouts. Obviously this method is limited in scope, especially for large scale parallel systems for which little experience exists. A more profound understanding how to choose the parallel data layout according to different system parameters was therefore determined to be the most desirable and beneficial result for NAS users. The necessary detailed performance modeling study to answer this question was however out of the scope of this explorative study and is the focus of a currently ongoing effort.

In preparation of this future performance modeling of OVERLOW we obtained access to the source code of the different versions of OVERFLOW at NAS and its accompanying documentation. Using test data set first performance experiments were conducted to familiarize ourselves with OVERLOW. A profile of the program execution was obtained using the NAS Origin hardware performance counters in preparation the necessary instrumentation of the code for a more detailed performance modeling study.