effects are accurate to 3% error, and (3) simulation performance of applications in high-level languages is accurate to less than 10% error. Furthermore, we have described an automated system that utilizes such models to a provide an effective development tool for parallel algorithm designers.

We have shown that given a specific virtual processor geometry configuration, for example, the cost of communicating n bytes from a processor to one of its neighbors can be modeled as a linear relationship consisting of a fixed startup cost and an amortized per-word communication latency. By analyzing the various memory hierarchies between virtual, physical and chip processor levels we have identified fundamental machine characteristics (t_E, t_M, t_I) that describe the communication costs between these various levels. By determining relationships for these coefficients, we have extended current models of grid communication [2] to include the effects of arbitrary message length, and communication/computation optimizations, as well as arbitrary virtual processor geometries.

By studying the underlying CM-2 floating point architecture and the relationships between the inner components (the Sprint chip, and Weitek 3132 FPU), we have developed a performance model accurate roughly to +/- 3%. We have shown that this behavior is similar to pipelined floating point vector processors, except that the pipelining effects arises from the virtual processor ratio. We have described the various transformations and stages in the transfer of data from processor memory through the Sprint chip and to the Weitek unit.

Given the various ways in which compilers and software tools translate high level code in the Paris constituents, an integrated and automated performance modeling system is necessary. We have described a prototype system, CMPS, that analyzes and predicts the communication and computation behavior of actual CM-Fortran and C* codes. This system is designed to work transparently with the CM compilers.

Finally, we have illustrated the usefulness of these performance models by studying a concrete application example: predicting the performance of a standard two-dimensional atmospheric simulation. This involved the solution of an explicit system of partial differential equations discretized on a rectangular staggered grid with periodic boundary conditions and leapfrog time differencing. Working through this concrete example, we demonstrated how our performance models can be used to predict PDE application performance to within ten percent error.

References

[1] Hoffman, G. R., Swartzrauber, P.N., Sweet, R. A., "Aspect s of using multiprocessors for meteorological modeling," in *Multiprocessing in Meteorological Models*, ed. D. Snelling, pp. 126-195, Springer-Verlag, Berlin, 1988.

[2] Levit, C., "Grid Communication on the Connection Machine: Analysis, Performance, and Improvements", *Proceedings of the Conference on Scientific Applications on the Connection Machine*, Moffett Field, California, September 12-14, 1988.

[3] Pozo, R., MacDonald, A. E., "Performance Characteristics of Scientific Computation on the Connection Machine", CS Tech Report, University of Colorado, Boulder, 1989.

[4] Pozo, R., "Connection Machine Status Report," Report to the Scientific Computing Division of National Center for Atmospheric Research and the Center for Applied Parallel Processing. September, 1989.

[5] Sadourny, R., "The dynamics of finite difference models o f the shallow water equations," *Journal of Atmospheric Science*, vol. 32, pp. 680-689, 1975.

[6] Sato, R. , Swarztrauber, P., "Benchmarking the Connection mach ine 2", *Proceedings of the Supercomputing Conference,* Orlando, Florida, November, 14-18, 1988.

[7] Thinking Machines Corporation, "Communications Architecture in the Connection Machine System", *Technical Report Series*, HA87-3.

Performance of SWM on the CM-2				
Grid Size	Observed (secs)	Predicted (secs)	Error	Gflops (scaled to 64K)
64 x 64	5.1	4.7	8.6%	0.9
64 x 128	8.0	8.0	0.5%	1.1
128 x 128	13.5	12.7	6.1%	1.2
128 x 256	23.2	22.7	2.0%	1.5
256 x 256	41.5	39.6	4.5%	1.6
256 x 512	77.1	75.0	2.7%	1.8
512 x 512	144.0	135.1	6.2%	2.0
512 x 1024	??	263.1	–	2.1
1024x1024	??	504.1	–	2.1

Figure 5: Results of CMPS on Shallow Water equations. The entries of "??" denote that the problem size would not fit on an 8K processor CM-2 with 8KB memory configuration.

At further steps, a time filter is applied to handle the weak instabilities of the leapfrog scheme for $f = u, v$, and P

$$F^{(0)} = f^{(n)} + \alpha(f^{(n+1)} - 2f^{(n)} + f^{(n-1)})$$

where α is a filtering parameter. The filtered values of the variables at the previous time-step are used in computing new values at the next time step.

The results for the complete application are shown in figure 5. This table shows various problem sizes for the application and the respective differences between the observed and predicted performance. In all cases the overall application performance generated by the CMPS model is below 10% relative error. We notice that the CMPS model always underestimates the running time of the application. This is due to the fact that we are only modeling the floating point aspect of the application. (We do not, for example, model CM-2 integer functions.) This capability could always be added, but the results from the CMPS are sufficient to illustrate that the basic concept works. Likewise the relative errors are slight higher than they were for individual Paris functions for the same reason. We also notice that the overall CMPS error is slightly less for the square Nx2N problem sizes. This is due to the fact that the communication parameters determined by the original analytical model (t_E, t_M, and t_I) more closely matches the physical configuration of the machine. (An 8K processor machine consists of two 64x64 subcubes.)

The last two entries in figure 5 illustrate an important use of these performance models. The Connection Machine we used for these tests did not have sufficient memory to hold the larger versions of the application. We notice that, as CMPS predicts, larger problem sizes generally result in more effective use of the computing hardware. The last column in figure 5 illustrates this. We were interested in seeing whether going a larger memory machine would improve performance. With the CMPS system, we could answer this question instantly before making the decision to purchase the memory upgrade.

Conclusion

In characterizing the performance of the Connection Machine CM-2, we developed fundamental models that accurately describe its behavior. Highlighting the major achievements from previous sections, we have developed (1) communication models of startup costs and transmission latency are accurate to 5% error for nearly all geometries, (2) computation model that account for virtual processor ratios and FPU pipelining

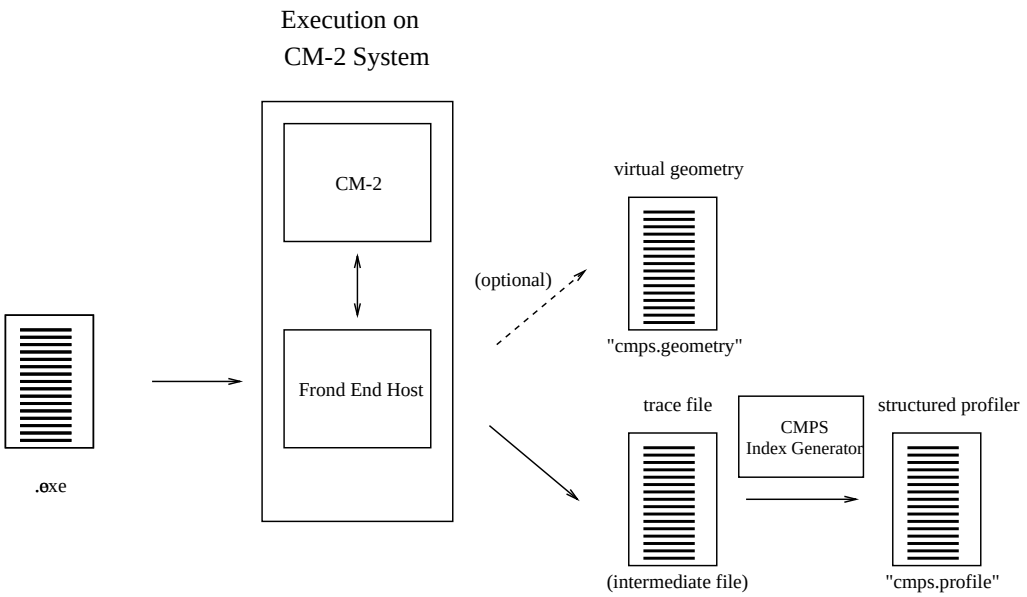

Figure 4: Run phase of the CMPS system.

necessary for the analytical model to reproduce the performance information.

An Application Example

As an example of the application performance analysis the CMPS system provides, we describe the implementation of a two-dimensional atmospheric/oceanographic simulation commonly used in the atmospheric sciences community to benchmark computers, [1], [6]. This mathematical system, the Shallow Water Equations, provides a primitive but useful model of the dynamics of the atmosphere or of certain ocean systems. The model is simple, yet captures computational features typical of more complex codes. These equations take the form

$$\begin{aligned} \frac{\partial u}{\partial t} - ZU + \frac{\partial H}{\partial x} &= 0 \\ \frac{\partial v}{\partial t} - ZU + \frac{\partial H}{\partial y} &= 0 \\ \frac{\partial P}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} &= 0. \end{aligned}$$

where u and v are the x and y velocities, respectively; P is the pressure; H, related to the height field is defined as $H = P + (u^2 + v^2)/2$, the vorticity, ζ, is given by $\zeta = \frac{\partial u}{\partial x} - \frac{\partial u}{\partial y}$, mass fluxes $U \equiv Pu$, $V \equiv Pv$, and the potential velocity is defined as $Z \equiv \zeta/P$.

The above equations are then discretized on a rectangular staggered grid with periodic boundary conditions. The variables P and H have integer subscripts, Z has half-integer subscripts, U has integer and half-integer subscripts, and V has half-integer and integer subscripts, respectively.

Initial conditions are chosen such that $\nabla \cdot \vec{v} = 0$ at all times. Time differencing is implemented using the Leap-frog method. The values at t_1 are computed using one step of the Euler's method

$$f^{(1)} = f^{(0)} + \frac{\partial f}{\partial t}^{(0)} \Delta t$$

`exp, sqrt`), and triad vector operations (`ax + y`). The model predicts these operations to within 2% error over all sample CM-2 geometries.

The basic model for floating point computation on the CM-2 is based on the movement of data from the processor memory onto the vector units. This transfer consists of five steps which are pipelined by the underlying hardware; if vp ratio is n, then this process is pipelined to require only $3n + 2$ stages rather than $5n$. Since this model closely resembles traditional pipelined vector processors, it is reasonable to expect that the floating point latency will consist of a basic startup cost and an amortized costs in one "pipelined" mode. Thus, we expect the model to have a linear relationship between vp ratio (which plays the role of vector length in traditional pipelined processors) and execution time.

Results show that the relative error of the model is never greater than 2%, and for vp ratios of two or greater the relative error is essentially neglible. The triad vector operations combine multiplication and an addition in a single function (the Weitek 3132 vector unit has an add and multiply pipe that can work in unison) thus effectively doubling the GFLOP performance rate in many cases.

CMPS

Once we have derived analytical performance models for floating point computation and grid communication for the CM-2, the next natural step is to see how such models can be used on basic numerical algorithms and higher level scientific applications.

There is a fundamental problem, however, in applying these analytical models to real problems: the algorithms and applications one is interested in modeling are typically coded in CM-Fortran or C* for the CM-2. Even with simple numerical expressions, we see that there are several ways in which compilers can parse and optimize the code generated for such constructs. This presents a practical problem to implementing our analytical performance model to high level code examples: we simply do not know how the compiler will interpret numerical expressions. Ideally, we would like to have a performance modeling system that mimics not only the underlying parallel hardware but also the software environment available to utilize such hardware.

In response to these issues we have developed a prototype software environment, the Connection Machine Performance Simulator (CMPS in the sequel), which essentially solves this problem. This software prototype is aimed at demonstrating functionality — it is primary a vehicle to illustrate that the basic ideas work, rather than being a commercial tool. The CMPS system works in integrated environment with the high-level CM-Fortran and C* compilers. It works with *unmodified* CM-Fortran and C* codes. Its features include a transparent integration with the C* and CM-Fortran compilers, no requirement of any source code modification, the utilization of CM analytical performance model as database to yield accurate analysis, the capability of working with arbitrary vp-geometries, and the capability of be used as an analytical tool for finding optimal data mappings.

.

The CMPS system consists of four phases: the compile, run, link, and analysis phases. The CMPS systems intercepts the assembly output of the compiler and alters the symbol table information. The CMPS version of the function is a wrapper which calls the real CM-2 instructions but also saves state information (current vp geometry, direction of communications, etc.) necessary to reconstruct its performance characteristics from the analytical CM model. Thus the CMPS system keeps a careful record of each CPM-2 function called and the global geometry conditions. Once the CMPS system completes its modification, compilation continues normally to produce a final object module. This object module is differs from the original only in that CM-2 instructions calls have been replaced by their CMPS counterparts.

At this point, we link the modified program object modules with the CMPS library, `libcmps.a`, to create a final executable. Once this executable image has completed running on the CM-2, a "structured" profile of the application is recorded in a text file, as shown in Fig. ??. This text file has all the call information

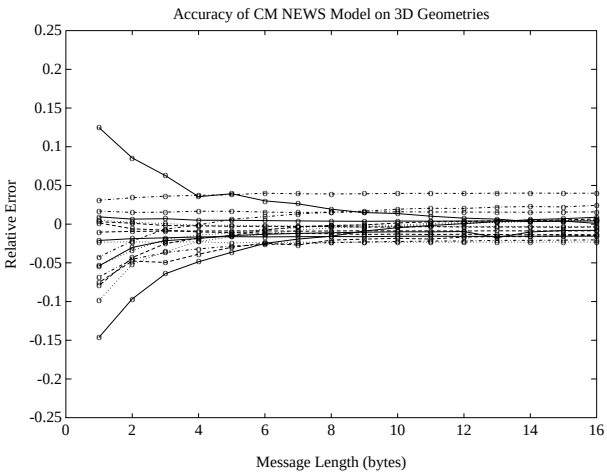

Figure 2: Relative error of CM NEWS model on CM measurements of various 1D and 3D geometries.

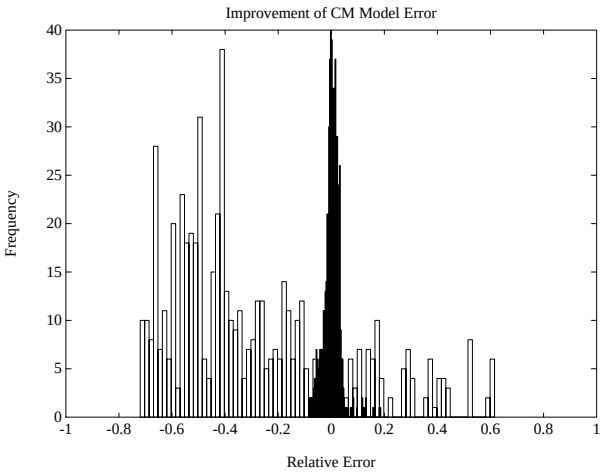

Figure 3: Improvements to previous communication models; relative error for arbitrary geometry layouts has been reduced from +/- 60% to roughly +/- 5 %.

One can then map a grid of 2^m points onto a d-dimensional CM-2 as

$$\underbrace{2^m}_{\text{total}} = \underbrace{2^{d-4}}_{\text{off-chip}} * \underbrace{2^4}_{\text{on-chip}} * \underbrace{2^{(m-d)}}_{\text{virtual}}.$$

Thus given a problem with an N-dimensional of size $n_1 \times n_2 \times \ldots n_N$ or $\prod_{i=1}^{N} n_i$, we can specify a mapping of off-chip components, $\vec{f} = \{f_1, f_2, \ldots, f_N\}$, on-chip components, $\vec{c} = \{c_1, c_2, \ldots, c_N\}$, and virtual components, $\vec{v} = \{v_1, \ldots, v_N\}$, where (1) for each dimension, the product of geometry lengths must match the problem dimension, $n_i = f_i * c_i * v_i, \;\; i = \{1, \ldots, N\}$, (2) the total on-chip dimensions must equal the number of processors on a single CM chip $c_1 \cdot c_2 \cdot \ldots, c_N = 16$ and (3) the total off-chip dimensions must match the size of the CM machine $f_i \cdot f_2 \cdot \ldots, f_N = = 2^{d-4}$.

Modeling Grid Communication

The NEWS communication phase on the CM-2 can be broken down into three parts, each corresponding to the off-chip, on-chip, and virtual processor phases. The first level, *virtual communication*, denotes the the communication between two virtual processors on the same physical processor. This is really a memory transfer within each physical processor, and requires no explicit communication. The second level, *on-chip communication*, denotes the communication between two physical processors on the same processor chip. This occurs over dedicated hardware on each chip and does not require use of the off-chip hypercube network. The third level, **off-chip communication** denotes the communication between two processors on different processor chips. This requires the off-chip hypercube network.

Given a CM geometry specification consisting of $G = \{\vec{f}, \vec{c}, \vec{v}\}$ specifying off-chip, on-chip, and virtual processor mapping, we can then determine the number of each type of communications performed along a given dimension by studying the boundary communication as follows. The number of memory-to-memory communications along axis j of a grid decomposition is given by $C_I = \prod_{j \neq i}^{N} v_i$, the number of on-chip communications is given by $C_M = (v_i - 1) \prod_{j \neq i}^{N} v_i$, and the number of off-chip communications is given by

$$C_E = \begin{cases} \prod_{j \neq i}^{N} o_i v_i & \text{if } f_i > 1 \\ 0 & \text{otherwise} \end{cases}$$

The communication bandwidth between these three levels can be determined by direct measurements to yield $t_M(n) = (2.4{+}5.6n)$ μsecs, $t_I(n) = (20.5{+}10.4n)$ μsecs, $t_E(n) = (0.0{+}5.5n)$ μsecs, where $t_M(n), t_I(n), t_E(n)$ denote the virtual, on-chip, and off-chip levels, repsectively.

Figure 2 provides an error distribution for the accuracy of the NEWS communication model. Here we measure the relative error of observed versus predicted values for 345 points over 23 different geometries. Most predicted values fall within 5 % relative error of the actual measurement. When this model is applied to the 2D data (which was not included in the least-squares fit) all predicted values fall within 6 % relative error of the measurements.

The model has been verified on 33 1D, 2D, and 3D geometries with message lengths ranging from 1 to 16 bytes. This comprises over 500 geometry/message-length combinations. Each of data point (circle) in these figures represents the average of five measurements (all experimental measurements agreed to within 1% relative error of each other), totaling over 2,500 CM measurements. Figure 3 illustrates the relative improvement of this model to earlier NEWS communication models [2].

Modeling Floating Point Operations

We present a performance model for the basic floating point kernel operations which constitute the computational core of all numerical applications: basic operations (+,*,-,/), transcendental library functions (`sin`,

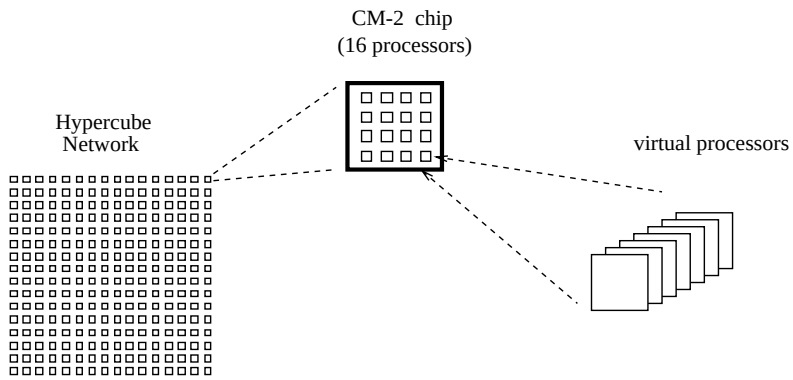

Figure 1: Hierarchical view of the Connection Machine 2

The basic architectural unit of the CM-2 is the CM bit-serial processor. Sixteen bit-serial processors are fabricated on a single physical chip; a full CM-2 contains 4096 such chips. Every pair of CM-2 chips (32 bit processors) shares a Weitek 3132 floating point unit (FPU). These Weitek chips provide the CM-2 with its tremendous number-crunching strength. When properly pipelined, each Weitek 3132 can produce one (single-precision) floating point multiply and one addition result per clock cycle. A CM-2 with an 6.7 MHz clock can run each Weitek 3132 at 13.3 MFLOPs, thus delivering a machine *peak* speed of 26.7 GFLOPs.

Communication between bit-processors takes place on the same physical chip. The 4096 physical CM processor *chips* in are connected in a hypercube topology: each chip forms the node of a 12 dimensional hypercube. General communication between CM chips occurs through "routers"; every group of 16 bit-serial processors share a router which is responsible for forwarding outgoing requests from the chip onto the network, and servicing incoming message requests from other chips residing in the Connection Machine. For applications that utilize regular communication patterns along N-dimensional grids, faster communication primitives exist that do not entail the overhead of using the chip router. This type of regular communication is termed NEWS communication.

An elegant feature of the Connection Machine is its availability to utilize virtual processors. The ratio of physical processors present on a specific CM to the number of virtual processors requested by the application is termed the **vp ratio.** Each physical processor can *simulate* more than one *virtual* processors by dividing its memory space into equal parts and sequentially serving the virtual processors.

Mapping N-dimensional Grids

As a mechanism for optimizing processor communications, the mapping of N-dimensional grids onto the pseudo-hypercube architecture of the CM-2 are specified by the use of virtual processor geometries. This mechanism specifies how the dimensions of the grid are mapped onto the CM-2's architecture components. Each CM-2 processor chip consists of 16 processors which can be arranged in an N-dimensional array of dimensions $\{c_1, c_2, \ldots, c_N\}$ where $c_1 \cdot c_2 \cdot \ldots, c_N = 16$. Furthermore, each physical processor may simulate several virtual processors v_i in each dimension. Thus we have the possibility of assigning each processor a $\{v_1 \times v_2 \times \ldots \times v_N\}$ subarray of virtual processors. The product $V = \prod_{i=1}^{N} v_i$ is termed the *vp-ratio* of the geometry and since it is always an exact power of two, one can denote the integer $v \equiv \log_2 V$.

Viewing the CM-2 as a d-Dimensional hypercube with 2^d processors, we see that 2^4 or 4 dimensions of the hypercube are embedded in the processor chip, and the remaining $d-4$ dimensions constitute the physical hypercube network connecting these chips. Furthermore, each processor can simulate 2^v virtual processors.

Performance Simulation and Modeling of Numerical Computations on the Connection Machine CM-2

Roldan Pozo

University of Tennessee, Knoxville, TN

Abstract

We present performance models of numerical computations on the Connection Machine CM-2. Our research identifies machine characteristics that are useful in predicting performance of numerical computations such as matrix/vector operations, relaxation techniques, and PDE applications on this parallel architecture. By conducting baseline experiments to establish parameters for floating point operations and grid-based communication, we can formulate practical performance models that predict grid algorithm behavior with over 95% accuracy. We also describe the design of a software package, CMPS, that analyzes and predicts the communication and computation behavior of actual CM-Fortran and C* codes. Utilizing a concrete example of a 2D fluid code describing atmospheric and ocean systems, we demonstrate how such models can be used to predict application performance to within ten percent error over a wide range of problem sizes and decompositions.

Introduction

Many scientific applications take place on two-dimensional and three-dimensional structured grids. Finite difference solvers for partial differential equations, for example, normally require processors to communicate with their local neighbors on the grid. Relaxation processes for elliptic PDEs also exhibit this type of communication pattern when discretized over regular domains. Quantum Chromodynamics (QCD) and mesoscale weather forecasting are two common examples of scientific applications that utilize rectangular lattice domains.

We present performance models for such numerical grid computations on the Connection Machine CM-2. Our research identifies important machine characteristics that are useful in predicting performance of numerical computations such as matrix/vector operations, relaxation techniques, and PDE applications on this architecture. By conducting baseline experiments aimed at establishing parameters for floating point operations and grid-based communication, we can formulate practical performance models that predict grid algorithm behavior with very good accuracy. We also describe the design of a software package, CMPS, that analyzes and predicts the communication and computation behavior of actual CM-Fortran and C* codes. Utilizing a concrete example of a 2D fluid code describing atmospheric and ocean systems, we demonstrate how such models can be used to predict application performance to within ten percent error.

CM-2 Overview

The Connection Machine CM-2 is a massively parallel single instruction multiple data stream (SIMD) distributed memory hypercube computer. A fully configured machine consists of 64K (65,536) bit-serial processors utilizing 2,048 Weitek 3132 floating point units (FPU) for an aggregate peak performance of over 25 Gflops.