Distributing a Chemical Process Optimization Application Over a Gigabit Network

Robert L. Clay
Department of Chemical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213 U.S.A.

Peter A. Steenkiste
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 U.S.A.

Abstract

We evaluate the impact of a gigabit network on the implementation of a distributed chemical process optimization application. The optimization problem is formulated as a stochastic Linear Assignment Problem and was solved using the Thinking Machines CM-2 (SIMD) and the Cray C-90 (vector) computers at PSC, and the Intel iWarp (MIMD) system at CMU, connected by the Gigabit Nectar testbed. We report our experience distributing the application across this heterogeneous set of systems and present measurements that show how the communication requirements of the application depend on the structure of the application. We use detailed traces to build an application performance model that can be used to estimate the elapsed time of the application for different computer system and network combinations. Our results show that the application benefits from the high-speed network, and that the need for high network throughput is increasing as computer systems get faster. We also observed that supporting high burst rates is critical, although structuring the application so that communication is overlapped with computation relaxes the bandwidth requirements.

Keywords: chemical process optimization, distributed computing, heterogeneous computing, gigabit networks, stochastic linear assignment problem, optimal resource allocation.

1. Introduction

High-performance networks have made it attractive to distribute compute-intensive applications across computer systems connected by local-area and wide-area networks. The obvious benefit is that the applications can combine the resources of several systems to reduce execution time. Heterogeneous computing is a special case of distributed computing. It has the added benefit that each application component can be mapped onto the most appropriate architecture, thus optimizing the efficiency of the computation. As a result, heterogeneous computing can result in super-linear speed up, i.e., using N systems, the application runs more than N times faster than on any of the individual systems (e.g. [22]).

An important question is how critical network performance is to the success of distributed computing. While many coarse-grain applications have been distributed successfully across relatively slow networks, high-speed networks are needed if we want to apply distributed computing to a wide class of applications, including grand challenge and national challenge applications. A first indication is that many applications are not only computationally intensive, but also data intensive, so large data sets have to be exchanged between the distributed tasks. A second observation is that as computer systems get faster, the networks will have to keep up.

In this paper we evaluate the impact of network bandwidth on the performance of a chemical process optimization application. This application is economically important for the chemical process industry, and it is representative of large class of optimization problems in other fields. We use a two-step evaluation strategy. We first distributed the application across three systems connected by the Gigabit Nectar testbed: the Intel iWarp system (MIMD) at CMU, and the Thinking Machines Corp. CM-2 (SIMD) and the Cray C-90 (vector) computers at PSC. This implementation shows the feasibility of distributed computing for this class of problems, and allows us to identify the problems in distributing large applications across heterogeneous systems.

The above evaluation, however, is limited to a single set of systems and a single network. In the second step of the evaluation, we use the detailed tracing information collected during execution to build a performance model of the application. The model captures the communication and computation requirements of the application, and the data dependencies between the different tasks. The model can be used to estimate the execution time for different computational systems and for different networks.

The remainder of this paper is organized as follows. In Section 2 we give an overview of the application and in Section 3 we discuss its mapping on the Nectar testbed. In Section 4 we describe the experimental set up and the computational results. We then present the application performance model (Section 5) and we use the model to study the impact of network and node performance on the application execution time (Section 6). We summarize in Section 7.

2. Application Overview

Modeling and optimization applications are a primary tool for decision making throughout the chemical process industry. Typical engineering examples are process design, materials allocation, optimal control, real-time optimization, process scheduling, production and capacity planning applications. Uncertainty is inherent in our decision models and processes (i.e., the model parameters are estimates of the real values). Although failure to account for the uncertainty of key parameters in decision problems can lead to non-optimal solutions [2], deterministic optimization methods are still predominant in chemical engineering today [10] because including the explicit treatment of uncertainty in models increases the computational requirements beyond the capabilities of today's computers.

The optimal resource allocation problem can be expressed as a Linear Assignment Problem (LAP), which can be stated as follows: given n resources (e.g., raw materials), n demands (e.g., processing units), and a set of costs c_{ij} of assigning resource i to demand j, find the set of (one-to-one) resource-demand assignments which minimizes the total cost, z. The primal LAP is defined as follows:

$$
\begin{array}{lll}
\text{minimize} & z = \sum_{i \in S} \sum_{j \in T} c_{ij} x_{ij} & \qquad\qquad \text{(LAP)} \\
\text{subject to} & \sum_{j \in T} x_{ij} = 1 & \forall i \in S \\
& \sum_{i \in S} x_{ij} = 1 & \forall j \in T \\
& x_{ij} \geq 0 & \forall i \in S, j \in T
\end{array}
$$

where

$$x_{ij} = 1 \Rightarrow \text{edge } (i,j) \in M,$$

$$x_{ij} = 0 \Rightarrow \text{edge } (i,j) \notin M.$$

The terms S and T denote the source and terminal node sets corresponding to resources and demands, respectively. The term M denotes a (bipartite graph) matching, which is an edge set, $\{(i_1,j_1), (i_2,j_2),...,(i_n,j_n)\}$, connecting source and terminal nodes in an exclusive manner.

Despite the combinatorially large solution space, a variety of polynomial time algorithms have been developed. They achieve their efficiency by exploiting the special structure of the Linear Assignment Problem. See [20, 21, 14, 17] for further details on both serial and parallel algorithms.

The LAP problem is stochastic when some or all the costs c_{ij} are defined probabilistically i.e., represented by a probability distribution instead of a single value. Stochastic LAP problems can be solved by converting them into deterministic problems using the certainty equivalent transformation [8]. However, this transformation increases the size of the LAP problem exponentially. Furthermore, this transformation alters the structure of the problem such that solution procedures specialized for deterministic LAP no longer apply. An algorithm appropriate for the stochastic LAP where costs are independently, normally distributed variables is presented in [5, 6].

3. The Distributed Stochastic LAP

Solutions to the stochastic LAP problem combine a diversity of tasks, each best suited to a different type of computer system (e.g., MPP or fast scalar processor). Consequently, this application is a good candidate for heterogeneous computing. In this section we describe how we mapped the stochastic LAP onto a set of heterogeneous systems connected by the Nectar gigabit testbed. Further discussion of the solution methods as well as the MPP computing implementation and results are reported in [5, 6, 7].

3.1 Application Overview and Mapping

Figure 1 shows the structure and mapping of the distributed chemical process optimization application. The "real world" model (i.e., the production cost data) is simulated on iWarp by sampling from a normal distribution with means ranging from (0,100000) with standard deviations randomly set for each cost element. The production data is sent to the C-90 where it is analyzed and used to compute the LAP cost matrix means (statistical model analysis). After this statistical reduction, the C-90 sends the cost matrix to the CM-2 for the LAP solver initialization. Once the entire cost matrix is received, the CM-2 computes the reduced cost matrix and sends the indices for the potentially optimal assignments to the C-90. The maximum cardinality matching is computed on the C-90 based on the initial zero-element indices of the reduced cost matrix (unweighted bipartite matching). The CM-2 completes the weighted matching portion of the LAP solution [6]. A complete implementation of the application would include a Bayesian inference step that generates a probability manifold from the production data, and would also use both means and variances in the LAP solver. These simplifications do not fundamentally alter the structure or mapping of the application.

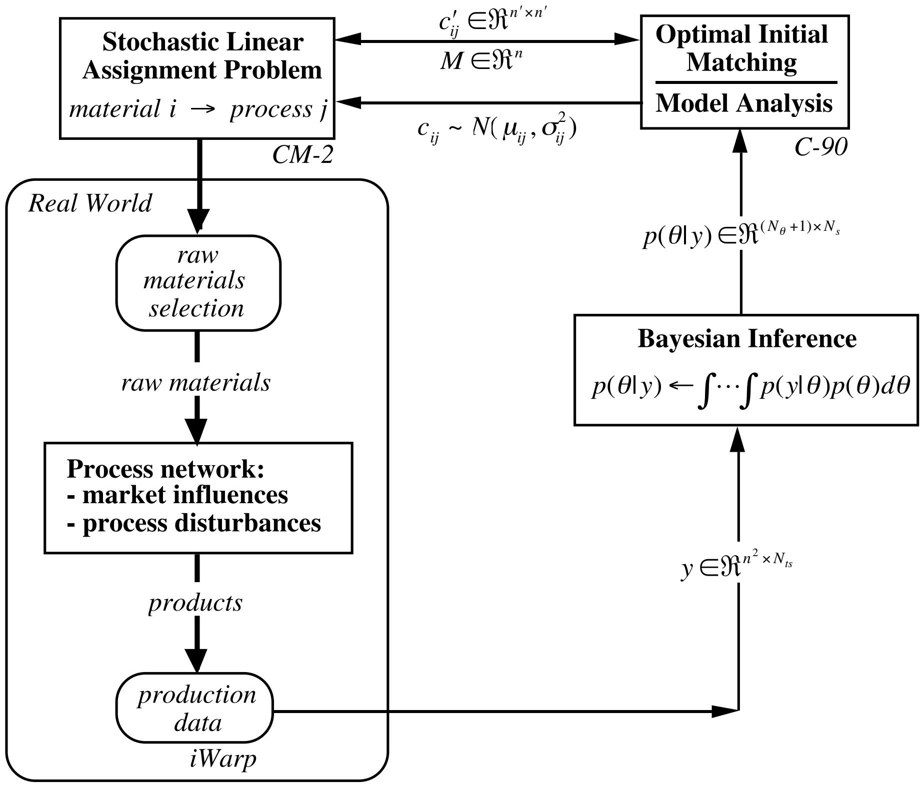


Figure 1 Mapping of the stochastic Linear Assignment Problem on the Nectar testbed.

The mapping of the application shown in Figure 1 is driven by the performance and suitability of each task on each system. The iWarp is an MIMD system and is well suited for the independent, complex function evaluations required for the simulation and integration problem. The C-90 is used as a high-performance vector machine, well suited for the statistical reduction computations and the serial initial matching procedure. The CM-2 is an SIMD machine, well suited for the row and column scanning and matrix arithmetic operations characteristic of the weighted matching portion of the LAP solver. Indeed, while each task could theoretically run on any of the systems, none of the tasks are well suited for the other machines. Since there is no single-machine version of the application, we cannot measure speedups, but we expect that the efficient processor utilization results in super-linear speedup.

3.2 Implementation

This application was developed by adding the iWarp sample generation phase to an existing, highly optimized LAP solver running on the C-90 and CM-2. This illustrates an important advantage of heterogeneous computing: large applications can be built from existing, separately-developed application components, allowing fast application development without porting effort. We expect that this opportunity will be one of the main motivations for heterogeneous computing over high-speed networks.

In our implementation, program control was carried out using PVM 3.2 [9], with the CM-2 acting as the master node. After reading in key parameters, the CM-2 (*Lap*) started the data generation job on the iWarp (*Gen*) and the analysis job on the C-90 (*Ana*). The simulated plant data generated on the iWarp was sent using the streams package [24, 11] over the HIPPI network, and read on the C-90 using the socket interface. Communication between the C-90 and the CM-2 was through the DHSC library [19]. DHSC supports distributed computing across the CM-2 and C-90, using the HIPPI network as the interconnect. Both the streams package and DHSC used raw HIPPI for the actual communication. The fact that a variety of tools had to be used added considerable complexity to distributing the application. Moreover, since the tools only support message passing and do not hide the details of the systems, individuals with expertise on each system had to help in program development. Programming tools that hide system details, such as parallelizing compilers [26], are clearly needed to make distributed computing more accessible to users.

Figure 2 shows how the different computational tasks in the application interact; the figure is not to scale. The horizontal bars represent computational tasks executed on the different systems and the arrows represent communication. The application can be delineated into two main phases: (*i*) data generation and analysis and (*ii*) LAP solution. The first phase consists of the data generation (iWarp), data analysis (C-90) and LAP initialization (CM-2). Since the time required for the LAP initialization on the CM-2 is insignificant, this phase ends when the last packet of the cost matrix data reaches the CM-2 from the C-90. The generation and analysis are pipelined to minimize the execution time; communication should be viewed as separate stages in the pipeline. As with any pipelined computation, the stages execute concurrently and the execution time is determined by the slowest stage in the pipeline. The LAP solution phase of the application is mainly performed on the CM-2, with a remote procedure call (RPC) to the C-90 to perform the (unweighted) initial matching. In this phase, all operations, including communication, are performed serially, and the execution time is the sum of the execution time of each component.

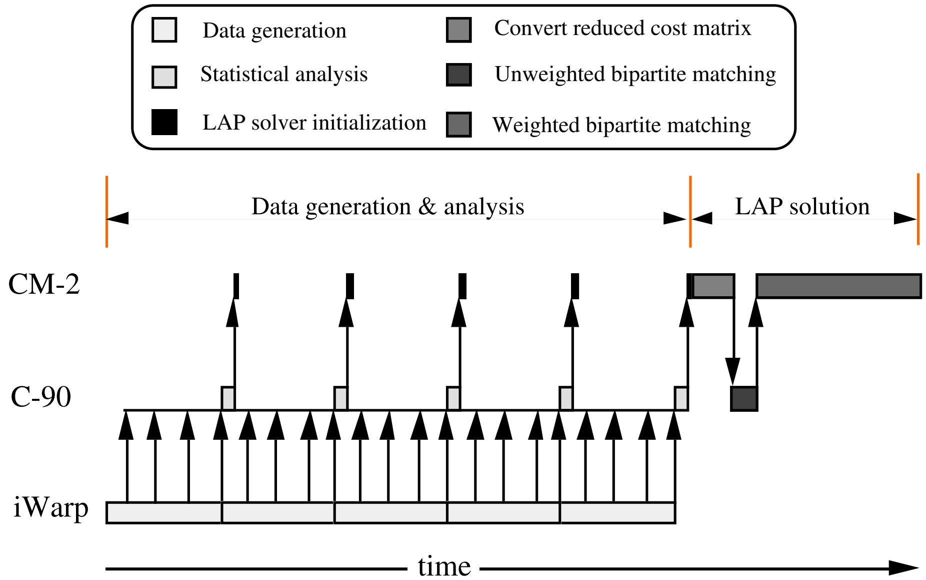


Figure 2 Application flowchart for the stochastic optimization problem.

3.3 Communication Requirements

The application communication requirements are summarized in Table 1. The table gives the source, destination, data format and data size for each of the four data streams in the application. The component items in Table 1 refer to the application executable images residing on the compute nodes. Programs *Gen*, *Ana* and *Lap* (initialization) comprise the data generation and analysis phase of the application. Programs *Lap* and *Fem* denote the LAP solver and initial matching in the LAP solution phase of the application.

Data	Source	Data Type	Destination	Size
Plant data	*Gen* on iWarp	32 bit float	*Ana* on C-90	$n^2 \cdot N_{ts}$
Stochastic model	*Ana* on C-90	32 bit int	*Lap* on CM-2	n^2
Cost data	*Lap* on CM-2	32 bit int	*Fem* on C-90	$\alpha \cdot n^2$
Initial matching	*Fem* on C-90	32 bit int	*Lap* on CM-2	n

Table 1 Summary of application communications requirements.

The communication requirements of the application depend on the size (n) of the LAP, the number of samples (N_{ts}) used to simulate the plant data, the number of grid points (N_s) used to map the probability manifold, and the number of stochastic parameters (N_θ). For the stochastic LAP application that was executed, $N_s = N_{ts}$ since the probability manifold mapping (i.e., the Bayesian inference problem) was omitted, and $N_\theta = n^2$ since all costs were assumed to be uncertain. Using both the mean and the variance in the LAP solution, instead of just the mean, would double the communication requirements for the Stochastic Model data stream. For our implementation, the reduction factor α is inversely proportional to n ($\alpha = 1.37 / n$); this is discussed in more detail in Section 5.2 As can be seen from the table, the data reduction during the analysis step (*Ana* on C-90) is such that the first phase of the application has significantly larger data requirements than the second.

The data format changes several times during the computation, resulting in presentation layer overhead. The generated samples are represented as 32 bit IEEE floating point numbers on the iWarp, and transmitted to the C-90 where they are then converted to 64 bit Cray floating point format. After the statistical analysis on the C-90, the sample means are converted to 32 bit integers and transmitted to the CM-2. Further data exchange between the CM-2 and the C-90 for the initial matching procedure use 32 bit integer representation. However, as a result of the different data representations on the C-90 and CM-2, these transfers still require an expensive transformation [16, 27, 18].

Because of the structure of the application (Figure 2), the four data streams have very different characteristics. The first stream is sent as a continuous stream of 64 Kbyte packets, since iWarp sends the data as it is generated. The second stream is sent as a sequence of bursts, where the number of bursts is the degree of pipelining (i.e. the number of blocks used for computation and data transfer on the C-90). For example, for a degree of pipelining of 16 and a problem size of 4k, the stream would consist of 16 bursts of 4 Mbyte each. Finally, the last two data streams are each sent as a single burst.

4. Experimental Results

The distributed optimization application described above was executed over the Nectar gigabit testbed on May 9, 1994. This section describes the execution environment, and presents and analyzes the measurements.

4.1 Nectar Testbed

The Nectar testbed is one of the five national gigabit testbeds [23] funded ARPA and NSF through CNRI. The Nectar testbed is a joint effort between the Nectar group at CMU, Bellcore, the Pittsburgh Supercomputer Center (PSC), and Bell Atlantic. The goal of the testbed is to build a gigabit Metropolitan Area Network (MAN) and to demonstrate its value to applications. The testbed consists of twenty-five DEC Alpha workstations, an iWarp parallel array [3] and a Paragon [12] on the CMU campus, and a Cray C-90, Cray T3D [1], CM-2 and Alpha cluster at the Pittsburgh Supercomputer Center (PSC). The Alpha workstations and iWarp use network interfaces that provide architectural support for copy avoidance to optimize throughput [24, 25, 15].

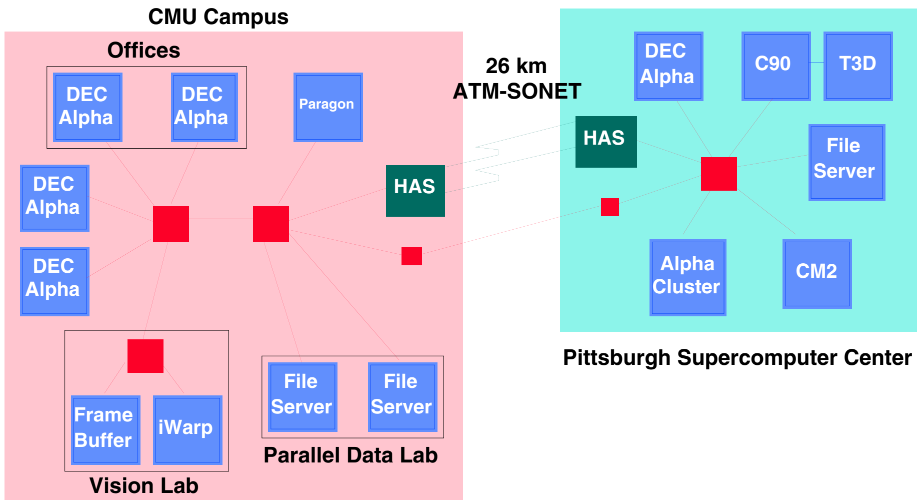


Figure 3 Nectar testbed high-performance distributed computing system.

The network (Figure 3) connecting the systems consists of two HIPPI-based LANs that are linked by a 2.4 Gbs ATM-SONET link [13, 4] representing a metropolitan area network (MAN). For the execution runs reported on in this paper, we used a HIPPI link that runs in parallel with the ATM-SONET link. The peak throughput of that link is 100 MB/sec. However, because of the latency of setting up the HIPPI connections across the 26 km link for each packet, the maximum achievable throughput is 73 MB/sec for packets of size 64 Kbyte.

4.2 Experimental Set Up

We solved LAP problem instances of sizes $n = 1\text{k}$, 2k, and 4k, with the number of samples N_{ts} fixed at 64. Another parameter in the experiment was the *degree of pipelining* in the generation and analysis phase; it is the block size used by the C-90 for the analysis of the model data. For each similarly sized run, the random number seeds used on iWarp for the generation of the simulated plant data were identical, leading to the generation of identical (pseudo-random) cost matrices. This was done to allow repeated identical runs to help isolate elements of the system performance.

For all experiments, the CM-2 was run in dedicated mode (i.e., all 32k processors *attached* with no other users on the front end), resulting in repeatable runs for the *Lap* component on the CM-2. The C-90 application components were run on a single node in interactive mode, so variations in the elapsed times occurred for repeated instances of the same problem. The iWarp array was run in dedicated mode, but variations in the iWarp elapsed times occurred for repeated instances of the same problem due to non-constant load on the front end.

4.3 Measurements and Analysis

The experimental data for a problem size of 4k are plotted in Figures 4 through 6; the problem sizes of 1k and 2k give similar results. Figure 4 shows the total execution time of the application, broken up into generation (*Gen*), LAP initial matching on the C-90 (*Fem*), and the reduction of the cost matrix plus LAP weighted matching solution on the CM-2 (*Lap*). Because the iWarp times dominate the pipelined execution in the generation and analysis phase, the time in *Gen* represents the total time of that phase, and the times on the C-90 and CM-2 are not shown. The times are shown as a function of the degree of pipelining.

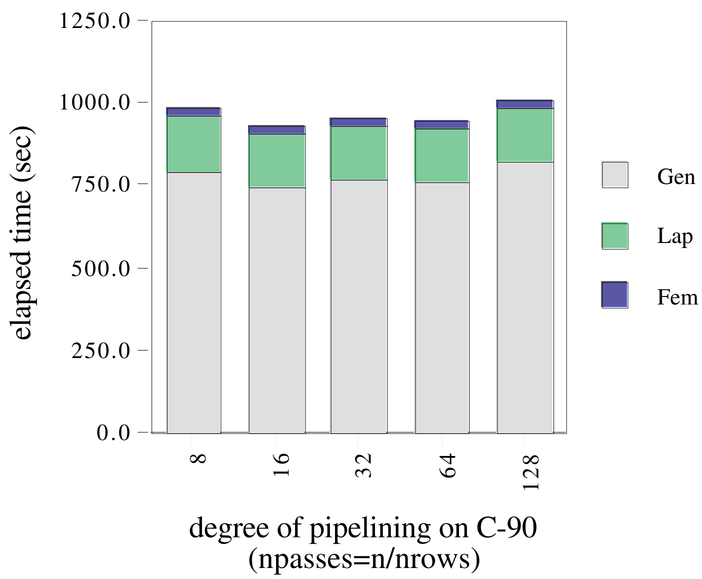


Figure 4 Elapsed time versus 'degree of pipelining' in generation and analysis phase (size 4k).

For the 4k problem the generation component accounts for about 80% of the total execution time, i.e., the iWarp is the bottleneck. We observe that the first phase takes slightly longer for the highest and lowest degree of pipelining (8 and 128), but the effect is small. This is no surprise since all block sizes (ranging from 500 Mbyte to 32 MByte for the iWarp-C-90 communication and a factor of 64 smaller for C-90-CM-2 communication) are large enough to allow efficient communication, and the number of blocks is large enough that pipeline fill and drain times are not significant.

Figure 5 shows the CPU times for the model analysis, data transfer and conversion computations on the C-90 for the 4k case. The data is read (ana1.read_data) from iWarp, converted into Cray floating point format (ana1.cf132c), and the statistical computations (apost) are performed to reduce the data set to cost means which are then sent (ana1.dhsc_write) to the CM-2. As shown in the figure, the read and write CPU times are relatively small. While the (vectorized) format conversion on the C-90 is quite fast (10 nsec per floating point number), it is nearly as costly as the statistical computations. This indicates that improvement in the conversion routines, or eliminating them by agreeing on a single representation, is highly desirable. Since CPU times (not elapsed times) were recorded for program *Ana* on the C-90, data pipelining has no effect.

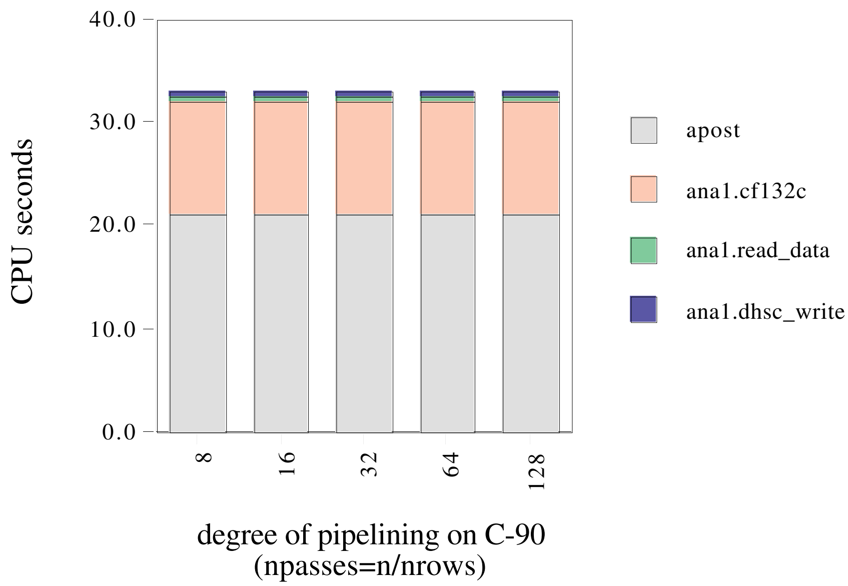


Figure 5 C-90 CPU time versus 'degree of pipelining' in analysis step on LAP (size 4k).

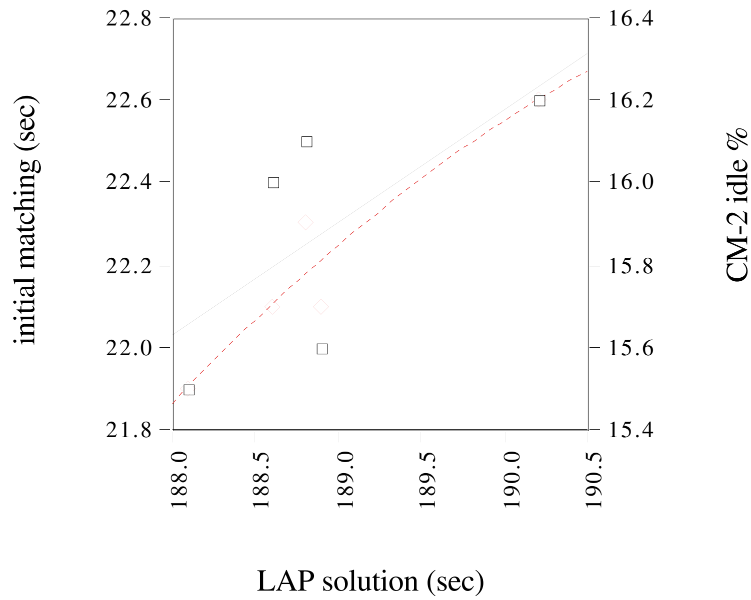


Figure 6 Initial matching and CM-2 idle % versus LAP solution times (size 4k).

The x axis of Figure 6 shows the total execution time of the second phase of the application (LAP solver) running on the CM-2 (*Lap*) and the C-90 (*Fem*) [7]. The LAP initial matching (elapsed) time on the C-90 (left - black full curve) and the CM-2 idle % (right - red dashed curve) are given versus the total LAP solution time for different runs of the solver. Since the CM-2 ran in fully dedicated mode, CM-2 times are identical for all runs of the same problem size, and any change in LAP solution time is a result of load changes on the C-90. The graph confirms this: there is a close correspondence between increasing initial matching times, waiting (idle) time on the CM-2, and increasing overall LAP solution times. An implication for distributed computing systems is that a load imbalance on one machine can affect the utilization of the other computing resources of the system. Running distributed applications efficiently will require careful allocation of system resources and a reasonably predictable response time from all systems used by the application.

5. Application Performance Model

To project the behavior of the application as a result of changes in the distributed computing system (i.e., number of compute nodes and network bandwidth) we developed a model of each application component. In the next section we use the model to examine the sensitivity of the application execution time to the network and computer system performance.

5.1 Computational Models

The computational model represents the execution time of each task in the application as a function of the problem size and, in the case of iWarp and CM-2, the size of the system; the parameters are summarized in Table 2. The models were derived using data from a large series of parameterized runs [5, 6, 7]. The models presented in this section are for the sparse approach; we discuss tradeoffs between the sparse and the dense approach in Section 6.4.

Function	Parameter	Value
Problem size	n	1k-4k
Number of data samples	N_{ts}	64
Range cost matrix values	r	10^5
Number of iWarp nodes	p_{iWarp}	64 measured
Number of CM-2 nodes	p_{CM2}	8k-32k measured
Bandwidth iWarp - C-90	BW_1	HIPPI
Bandwidth C-90 - CM-2	BW_2	HIPPI

Table 2 Model parameters used in the experiments.

Data Generation Model: (Gen running on iWarp)

Timing tests indicate that the 64 node iWarp array is capable of generating $1.4 \cdot 10^6$ samples per second. The rate of sample generation corresponds directly to the number of iWarp nodes: doubling the number of nodes doubles the rate. This results in the following elapsed-time performance model for generating the samples on iWarp:

$$t_{gen.x} = \frac{n^2 N_{ts}}{1.4 \cdot 10^6 (p_{iWarp} / 64)}. \tag{1}$$

Data Analysis Model: (*Ana* running on C-90)

Using a single node of the C-90 the elapsed times were closely correlated to the data size n^2. The model for the analysis on the C-90 obtained from regression of experimental data is:

$$t_{ana.x} = 2 \cdot (n / 1024)^2, \tag{2}$$

where only the sample means are computed.

LAP Solver Model: (*Lap* running on CM-2)

Using the data from a set of parametric tests, a statistical model was fit to the data giving the following performance model for the LAP solver:

$$t_{lap.x} = c_1 \left(\frac{n^2}{p_{CM2}} \right)^{c_p} r^{c_r}, \tag{3}$$

where the constants for (3) are: $c_1 = 0.046$, $c_p = 0.786$, and $c_r = 0.306$. There is a natural scaling in equation (3) between the number of cost elements and the number of CM-2 nodes. Also, the time to solve the weighted matching problem is weakly correlated to the degree of precision, characterized by the range (r).

LAP Initial Matching Model: (*Fem* running on C-90)

Using the data from a set of parametric tests, a statistical model was fit to the data from routine *Fem* running on the C-90 giving:

$$t_{fem.x} = c_1 n + c_2 n^2 + c_3 \left(n/r \right), \tag{4}$$

where the constants are: $c_1 = 4.508 \cdot 10^{-3}$, $c_2 = 1.046 \cdot 10^{-7}$, and $c_3 = 0.72$. In equation (4), the first and second order terms (i.e., n and n^2, respectively) characterize the LAP size, and the ratio n/r represents a natural indicator for the relative degree of precision [6].

5.2 Traffic Models

The traffic models are shown in Table 3. They are simpler than the computational models: there is a linear dependence between the data transfer time and the size of the data stream (Table 1) and the transfer time is inversely-proportional to the sustainable network bandwidth. Note that the sustainable network bandwidth is in practice limited by the sending and receiving host's ability to put data in, and to remove data from the network.

Transfer	Model	Equation
Gen output - iWarp to C-90	$t_{dt.gen} = \dfrac{4n^2 N_{ts}}{BW_1 \cdot 10^6}$	(5)
Ana output - C-90 to CM-2	$t_{dt.ana} = \dfrac{4n^2}{BW_2 \cdot 10^6}$	(6)
Lap output - CM-2 to C-90	$t_{dt.lap} = \dfrac{4\alpha n^2}{BW_2 \cdot 10^6}$	(7)
Fem output - C-90 to CM-2	$t_{dt.fem} = \dfrac{4n}{BW_2 \cdot 10^6}$	(8)

Table 3 Application traffic models.

Equation (7) requires further explanation. The program *Lap* running on the CM-2 computes the reduced cost matrix as part of the initialization procedure [5, 7]. The zero elements in this matrix are potential optimal assignments for the LAP. The program *Fem* running on the C-90 finds the optimal initial matching, given the zero element indices of the reduced cost matrix. The simplest way to transmit the elements is to have the CM-2 transmit the entire reduced cost matrix to the C-90, which then selects the zero elements; this is called the *dense* approach. However, we use a more efficient way, called the *sparse* approach: the CM-2 first locates all the indices for the zero elements and sends only the vector of indices to the C-90 for the initial matching. The advantage of the sparse approach is lower data transfer requirements (i.e., the number of indices to be sent to the C-90 is in general less than n^2). However, the tradeoff is that the CM-2 is less efficient than the C-90 at constructing the sparse incident matrix.

The parameter α in equation (7) is a reduction factor indicating the density of the incidence matrix. Setting α to 1 gives the traffic model for the dense approach. The scalar α is application dependent, but lies in general between $1/n$ and 1, and depends on the relative precision of the problem defined by the ratio n/r. For $r = 10^5$, as used in these experiments, α is inversely proportional to the problem size, such that $\alpha = 1.37 / n$ where $n \geq 128$.

6. Performance Analysis

Equations (1)-(8) represent a performance model of the distributed application. In this section we use the model to study the performance of the two phases of the computation individually and combined. The degrees of freedom in the model are: the LAP problem size, the number of nodes used on the iWarp and CM-2 machines, and the sustainable network bandwidths. We also use the model to compare the dense and sparse transfer options between the CM-2 and C-90, and we look at how the different structure of the two application phases influences the dependence of the execution time on the network bandwidth.

6.1 Data Generation and Analysis Performance Model

As a result of the pipelining of the generation and analysis phase, the throughput of the system is determined by the slowest component. Note that both the iWarp and the C-90 overlap communication with computation, so the communication phases are independent stages in the pipeline, and the elapsed time of the generation and analysis phase is the maximum of $t_{gen.x}$, $t_{dt.gen}$, $t_{ana.x}$ and $t_{dt.ana}$. For realistic parameters, the iWarp computation or the iWarp to C-90 communication limit the performance of this phase. Figure 7 shows the estimated generation and analysis time as a function of these two limiting parameters.

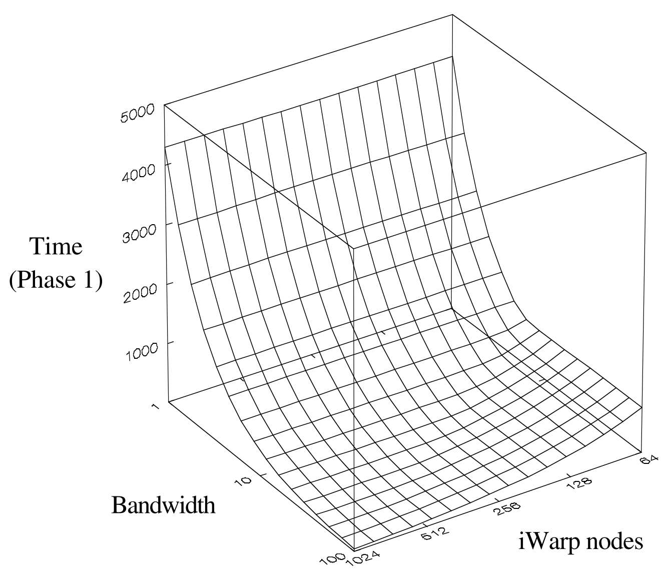


Figure 7 Estimated execution time (sec) for generation and analysis phase (size 4k) as a function of network bandwidth (MB/sec) and iWarp system size (number of nodes).

In the experiment we were using a 64 node iWarp system. The model predicts an execution time of 766 seconds, which closely matches the measured time (*Gen* in Figure 4). Note that this point is in the "flat" part of the graph with respect to the network bandwidth, indicating the generation phase was limited by the computation on iWarp and not by the network bandwidth. If we scale up the iWarp system to 256 nodes (equivalent to a 66 node Paragon system for this application), and 512 nodes (66 node Paragon with co-processors used for computation), then the execution time drops almost linearly. We have demonstrated sustained application throughputs of 40 MB/sec between iWarp and C-90 [11]. The 40 MB/sec network becomes the bottleneck when the iWarp grows above 460 nodes. Note that Paragon systems much larger than 66 nodes have been built, so 40-100 MB/sec network bandwidth is by no means excessive. The data analysis on the C-90 becomes the bottleneck of systems of size $p_{iWarp} \geq 1540$ nodes with the iWarp to C-90 bandwidth (BW_1) above 135 MB/sec.

6.2 LAP Solution Performance Model

The computation and communication occur sequentially in the LAP solution phase, so the elapsed time is the sum of $t_{lap.x}$, $t_{dt.lap}$, $t_{fem.x}$ and $t_{dt.fem}$. Figure 8 shows the execution time of the LAP solver as a function of the network bandwidth and the number of CM-2 nodes. The shape of the graph is different from the graph summarizing the generation and analysis phase. Specifically, there are no "flat" regions, and the impact of the network bandwidth on the (absolute) execution time is independent of the number of CM-2 nodes.

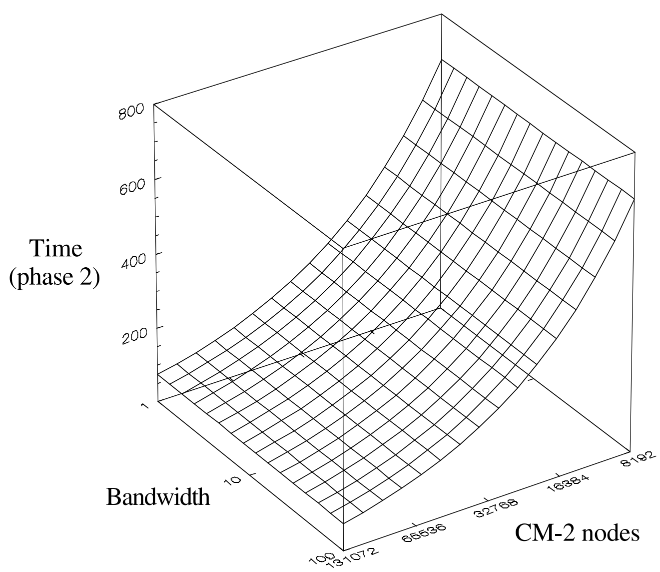


Figure 8 Estimated execution time (sec) for LAP solver phase (size 4k) as a function of the network bandwidth (MB/sec) and the size of the CM-2 system (number of nodes). Sparse incidence matrix transferred from CM-2 to C-90.

For the experiment, we used a CM-2 with 32k nodes, and the bandwidth between the C-90 and CM-2 was 12 MB/sec. The model predicts an execution time of about 200 seconds, which is close to the measured time (*Lap* plus *Fem* in Figure 4). Note that the C-90/CM-2 bandwidth would be 30 MB/sec without the need for presentation layer format conversion on the CM-2 [18]. With this higher throughput, the execution time would be reduced by about 10%.

6.3 Application Performance Model

Combining equations (1)-(8) gives the overall application performance model, accounting for the iWarp data generation and transfer overlap as discussed above. Figure 9 shows the sensitivity to network bandwidth and the number of iWarp nodes, given 64k CM-2 nodes and $BW_1 = BW_2$. The results are comparable to those of Figure 7, since the data generation and analysis phase of the application dominates the execution time. Again, the strongest sensitivity is network bandwidth, which limits the data flow between the iWarp and the C-90 when there is an adequate number of iWarp nodes to avoid the generation computation bottleneck. With relatively few iWarp nodes, the generation computations become more restrictive than the network, as shown on the graph and as observed in our experiments.

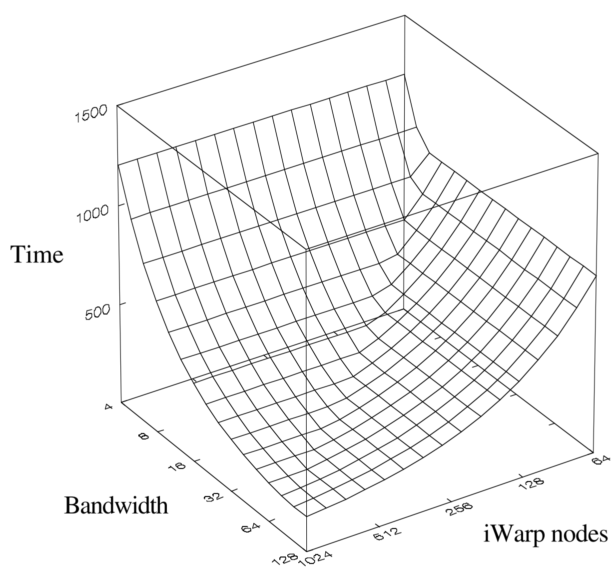
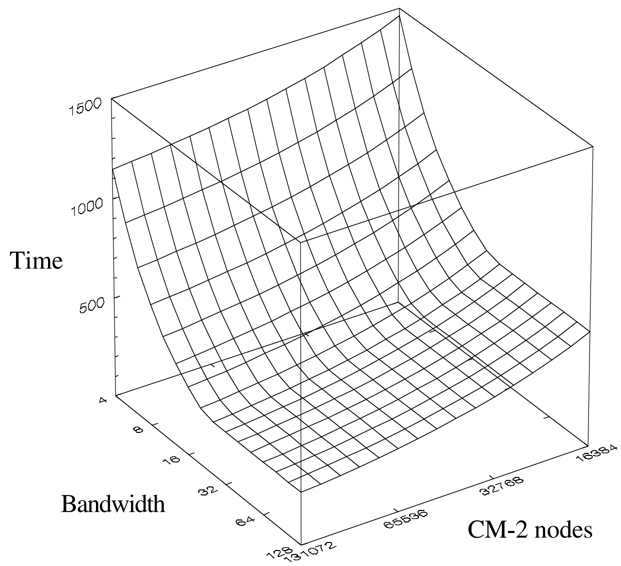
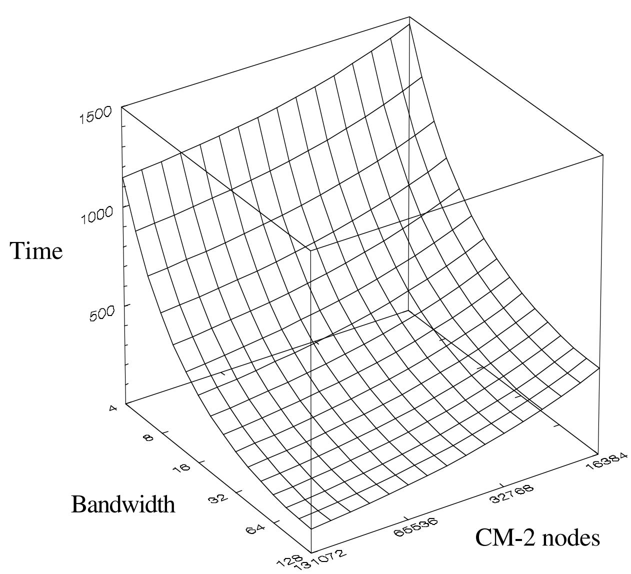


Figure 9 Estimated execution time (sec) for combined LAP application (size 4k) as a function of iWarp/C-90/CM-2 network bandwidth (MB/sec) and iWarp system size (number of nodes) – sparse approach.

Figure 10a shows the overall application performance sensitivity to network bandwidth and the number of CM-2 nodes, using 256 iWarp nodes and assuming $BW_1 = BW_2$. Because the data generation computations on iWarp are limiting at high bandwidths, no further improvements occur from increasing the bandwidth above the data generation rate. While there is a gain from increasing the number of nodes on the CM-2, the base line time remains above 200 seconds due to the data generation bottleneck. Figure 10b shows the comparable sensitivities, but using 1024 iWarp nodes. By increasing the number of iWarp nodes, the data generation bottleneck is removed and overall performance continues to improves as network bandwidth is increased.



(a) 256 iWarp nodes



(b) 1024 iWarp nodes

Figure 10 Estimated execution time (sec) for the LAP application (size 4k) as a function of the iWarp/C-90/CM-2 network bandwidth (MB/sec) and size of the CM-2 (number of nodes), for two iWarp system sizes.

6.4 Tradeoff Between Dense and Sparse Reduced Matrix Transfer

Two alternatives exist for transferring the initial reduced cost matrix zero-element indices from the CM-2 to the C-90, where the initial matching is computed (see Section 5.2). In the sparse approach, the CM-2 locates the zero-element indices for the initial reduced cost matrix, packs these into an incidence vector, and then sends this vector to the C-90 for the initial matching subproblem. In the dense approach, the CM-2 sends the entire initial reduced cost matrix to the C-90 which then forms the sparse incidence matrix. The advantage of the sparse approach is lower data transfer requirements, since the density term α in equation (7) is bound by $1/n \le \alpha \le 1$. However, the tradeoff is that the CM-2 is less efficient than the C-90 at constructing the sparse incident matrix.

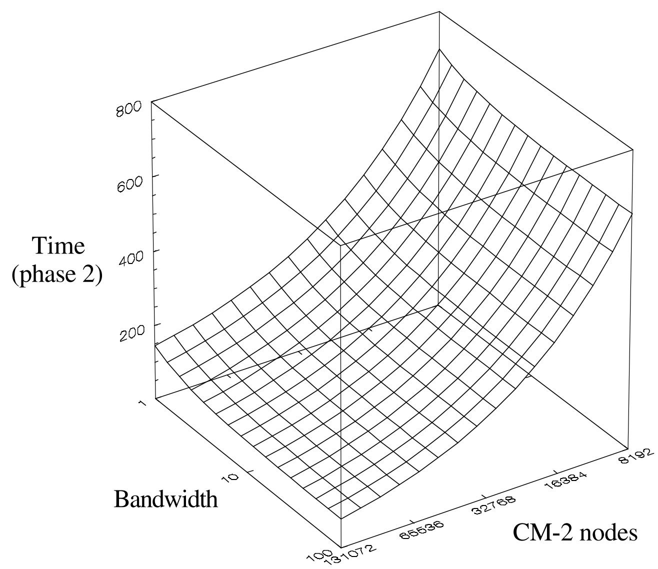


Figure 11 Estimated execution time (sec) for LAP solver phase (size 4k) as a function of the network bandwidth (MB/sec) and the size of the CM-2 system (number of nodes). Dense incidence matrix transferred from CM-2 to C-90.

In Figure 11 the time for the LAP solution phase is plotted as a function of the network bandwidth and the number of CM-2 nodes for the dense matrix transfer method and problem size of 4k. The sensitivity to the network bandwidth is minimal for the sparse method (Figure 8), while, as can be expected, the dense transfer method shows a higher sensitivity to the network bandwidth (Figure 11). Since the time to pack the incidence matrices on the CM-2 is relatively small compared to the overall LAP solution, the sparse method is superior in performance to the dense method for almost all cases–the dense method is sometimes faster for small data sets. However, the C-90 times for the dense method could be accelerated by using more than a single C-90 node. If multiple C-90 nodes were used, we expect that the dense method would be faster than the sparse method, if the network bandwidth were sufficiently high.

6.5 Impact of Network Throughput on Execution Time

The impact of network speed on overall application performance can be viewed in terms of percent of time in which the application was communications bound (vs. compute bound). The interconnection between compute speed (number of nodes) and network performance requirements for the LAP application is shown in Figures 12 and 13. In application phase 1 (Figure 12) the communications and computing operations are overlapped. Hence, so long as the network performance is adequate, the application can run completely compute bound. Increasing network bandwidth will not speed up the application. Decreasing the network bandwidth immediately slows the application down, to a complete stop at the limit.

In phase 2 (Figure 13) the communications and computing operations are serial. In this case, clearly the key to acceptable performance is to keep the communications time significantly below the computational time. Increasing or decreasing network performance has an direct effect on elapsed time. For this phase of the application, the communication times (binding or not) are practically insignificant due to the smaller data sets to be transferred (from the statistical reduction of the cost data).

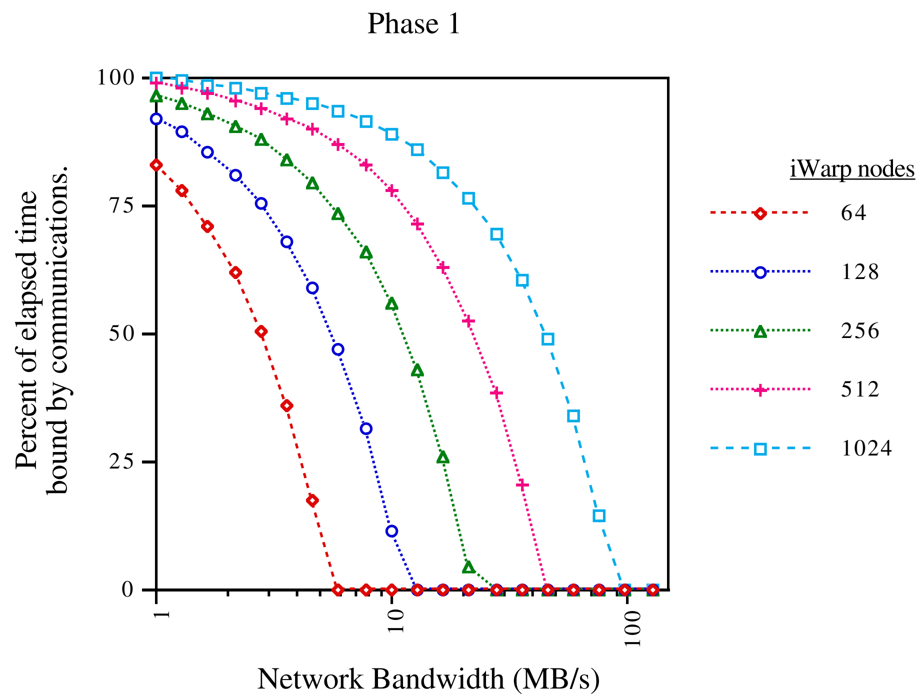


Figure 12 Phase 1 estimate of the percent of elapsed time where the application was bound by communications versus effective network bandwidth for various iWarp sizes. LAP size 4k.

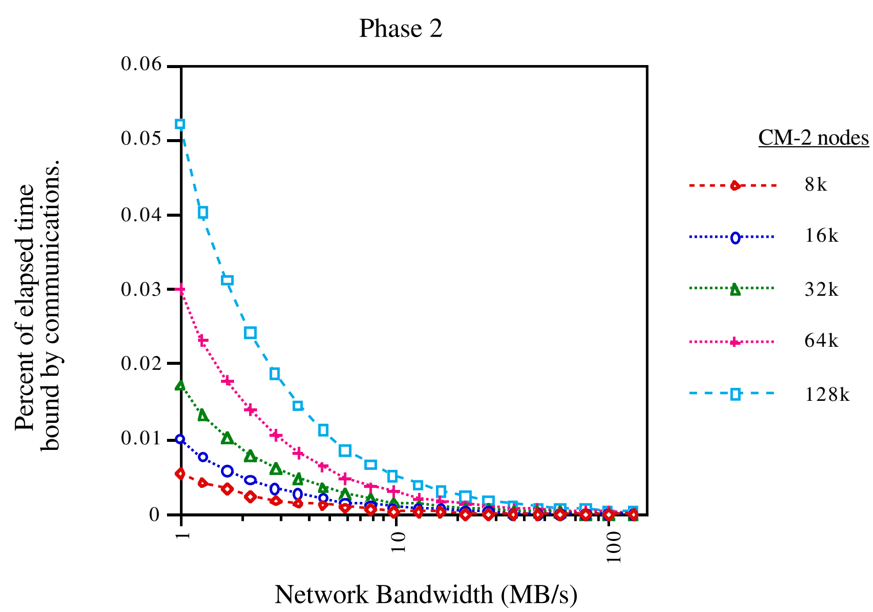


Figure 13 Phase 2 estimate of the percent of elapsed time where the application was bound by communications versus effective network bandwidth for various CM-2 sizes. LAP size 4k, sparse transfer.

7. Conclusions

In this paper we described the distribution of a chemical process optimization application across a heterogeneous system consisting of the Intel iWarp, Cray C-90 and Thinking Machines CM-2 computers connected by the Nectar gigabit testbed. The implementation demonstrates several benefits of heterogeneous computing: efficient execution and the ability to build applications by connecting existing, separately-developed application components across the network without having to port code. Note that some of these benefits, e.g. efficient system utilization, may only be realized if the different systems are simultaneously dedicated to the application and if sufficient network bandwidth is available; this creates an organizational problem for supercomputer centers. The implementation of the application also demonstrated the difficulty of distributing an application across heterogeneous computers since expertise in a number of different computer systems is required. Better programming tools are required that at the same time hide more details of the systems and integrate the systems better.

Our measurements and a performance model of the application show that the sensitivity of the execution time to the network bandwidth depends strongly on the structure of the distributed application. If computation and communication are serialized, as in the RPC-based solution phase of the application, any network delay, including increased latencies due to physical distance, will increase the execution time. Being able to burst at high rates is critical, and the average bandwidth requirements of the application are not very indicative of its communication requirements. In contrast, if communication and computation can be overlapped, as in the pipelined generation and analysis phase of the application, network bandwidth requirements are relaxed and speed of light latencies are hidden. Specifically, the average bandwidth requirements of the application represent the bandwidth that should be sustained by the network.

Acknowledgments

We would like to thank Michael Hemy, Jamshid Mahdavi and Todd Mummert for their help with the distribution of the optimization application, and Gregory J. McRae for his input into the conceptualization of the application. We also gratefully acknowledge the use of computational resources at the Pittsburgh Supercomputer Center, and especially the PSC applications support group. We are also grateful for the use of computational resources of Sandia National Labs in the final preparation of this manuscript.

The research was supported by the National Science Foundation and Defense Advanced Research Projects Agency under Cooperative

Agreement NCR-8919038 with the Corporation of National Research Initiatives, and by the Advanced Research Projects Agency (DOD) monitored by the Space and Naval Warfare Systems Command (SPAWAR) under Contract N00039-87-C-0251.

References

[1] D. Adams. *Cray T3D system architecture overview*. Cray Research Inc., September 1993. Revision 1.C

[2] J.R. Birge. *Models and Model Value in Stochastic Programming*. Technical Report 93-2, Department of Industrial and Operations Engineering, University of Michigan, 1993.

[3] Shekhar Borkar, Robert Cohn, George Cox, Sha Gleason, Thomas Gross, H. T. Kung, Monica Lam, Brian Moore, Craig Peterson, John Pieper, Linda Rankin, P. S. Tseng, Jim Sutton, John Urbanski, and Jon Webb. iWarp: An Integrated Solution to High-Speed Parallel Computing, In *Proceedings of Supercomputing '88*, pages 330-339, Orlando, Florida, November 1988. IEEE Computer Society and ACM SIGARCH

[4] N.K. Cheung and H.T. Kung. Gigabit/sec Wide Area Computer Networks: Potential Applications and Technology Challenges. In *OFC'91*, San Diego, February 1991, invited paper.

[5] R.L. Clay. *Scheduling in the Presence of Uncertainty: Probabilistic Solution of the Assignment Problem*, MS thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1991.

[6] R.L. Clay, J. Mahdavi, G.J. McRae. Scheduling in the Presence of Uncertainty: The Linear Assignment Problem, In *Proceedings of AIChE National Meeting*, Pittsburgh, Pennsylvania, August 1991.

[7] R.L. Clay, and G.J. McRae. Solution of Large-Scale Modeling and Optimization Problems Using Heterogeneous Supercomputing Systems, In *SuParCup 1991 Proceedings*, Mannheim, Germany, 1991.

[8] G.B. Dantzig. *Planning Under Uncertainty Using Parallel Computing*, Technical Report SOL 87-1, Department of Operations Research, Stanford University, 1987.

[9] G.A. Geist and V.S. Sunderam. The PVM System: Supercomputer Level Concurrent Computation on a Heterogeneous Network of Workstations, In *Proceedings of the Sixth Distributed Memory Computing Conference*, pages 258-261, IEEE, April 1991.

[10] I.E. Grossmann and D.A. Straub. Recent Developments in the Evaluation and Optimization of Flexible Chemical Processes, In L. Puigjaner and A. Espuna (editor), *Proceedings Computer-Oriented Process Engineering*, pages 49-59, Elsevier, 1991.

[11] Michael Hemy and Peter Steenkiste. Gigabit IO for Distributed-Memory Systems: Architecture and Applications, In *Proceedings of Supercomputing '95*, to appear, ACM/IEEE, December 1995.

[12] Intel Corporation. *Paragon X/PS Product Overview*, March 1991.

[13] M.Z. Iqbal, M. Stern, J. Young, H. Izadpanah, R. Standley, and J.L. Gimlett. A 2.5 Gb/s SONET Datalink with STS-12c Inputs and HiPPI interface for Gigabit Computer Networks, In *GLOBECOM '92 Conference Record*, pages 1196-1200, Orlando, Florida, December 1992. IEEE.

[14] R. Jonker, and A. Volgenant. A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems, *Computing 38*, pages 325-340, 1987.

[15] Karl Kleinpaste, Peter Steenkiste, and Brian Zill. Software Support for Outboard Buffering and Checksumming. In *Proceedings of the SIGCOMM '95 Symposium on Communication Architectures and Protocols*, to appear, ACM, August/September 1995.

[16] Thomas T. Kwan and Jeffrey A. Terstriep. Experiments with a Gigabit Neuroscience Application on the CM-2. In *Proceedings of Supercomputing '93*, pages 133-142, Oregon, November 1993. ACM/IEEE.

[17] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, 1976.

[18] J. Mahdavi, G. L. Huntoon, and M. B. Mathis. Deployment of a HiPPI-based Distributed Supercomputing Environment at the Pittsburgh Supercomputing Center, In *Proceedings of the Workshop on Heterogeneous Processing*, pages 93-96, March 1992. IEEE.

[19] Jamshid Mahdavi, Gwendolyn L. Huntoon, and Matthew B. Mathis. Deployment of a HiPPI-based Distributed Supercomputing Environment at the Pittsburgh Supercomputing Center, In *International Parallel Processing Symposium*, Los Angeles, April 1992. IEEE.

[20] S. Martello and P. Toth. Linear Assignment Problem, *Annals of Discrete Mathematics* 31, pages 259-282, 1987.

[21] L.F. McGinnis. Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem, *Operations Research* 31(2), March-April, 1983.

[22] Carlos R. Mechoso, John Farrara and Josheph Spahr. Running a Climate Model in a Heterogeneous, Distributed Computer Environment, In *Proceedings of the Third International Symposium on High-Performance Distributed Computing*, pages 79-84. IEEE, San Fransisco, August 1994.

[23] Special Report. Gigabit network testbeds. *IEEE Computer*, 23(9):77-80, September 1990.

[24] Peter A. Steenkiste, Michael Hemy, Todd Mummert, Brian Zill. Architecture and Evaluation of a High-Speed Networking Subsystem for Distributed-Memory Systems, In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, IEEE, pages 154-163, May 1994.

[25] Peter A. Steenkiste, Brian D. Zill, H.T. Kung, Steven J. Schlick, Jim Hughes, Bob Kowalski, and John Mullaney. A Host Interface Architecture for High-Speed Networks, In *Proceedings of the 4th IFIP Conference on High Performance Networks*, pages A3 1-16, Liege, Belgium, December 1992. IFIP, Elsevier.

[26] Jaspal Subhlok, Jim Stichnoth, Dave O'Hallaron, and Thomas Gross. Programming task and data parallelism on a multicomputer. In *Proceedings of the ACM Symposium on Principles and Practice of Parallel Programming*, pages 13-22, May 1993.

[27] R.J. Vetter, D.H.C. Du, A.E. Klietz. Network Supercomputing: Experiments with a Cray-2 to CM-2 HiPPI Connection, In *Proceedings of the Workshop on Heterogeneous Processing*, pages 87-92. IEEE, March 1992.

Appendix
Performance Model 2-D Plots

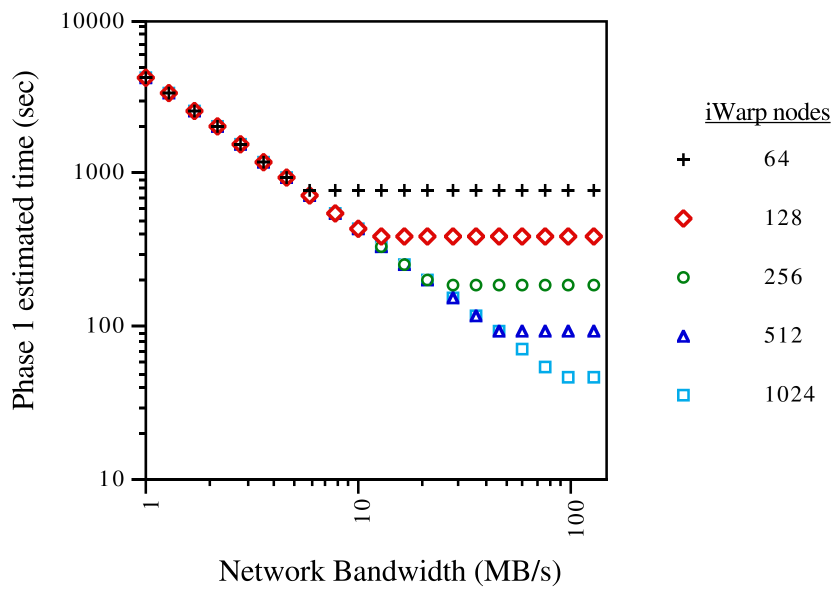


Figure A.1 Estimated execution time (sec) for generation and analysis phase (size 4k) as a function of network bandwidth (MB/sec) and iWarp system size (number of nodes).

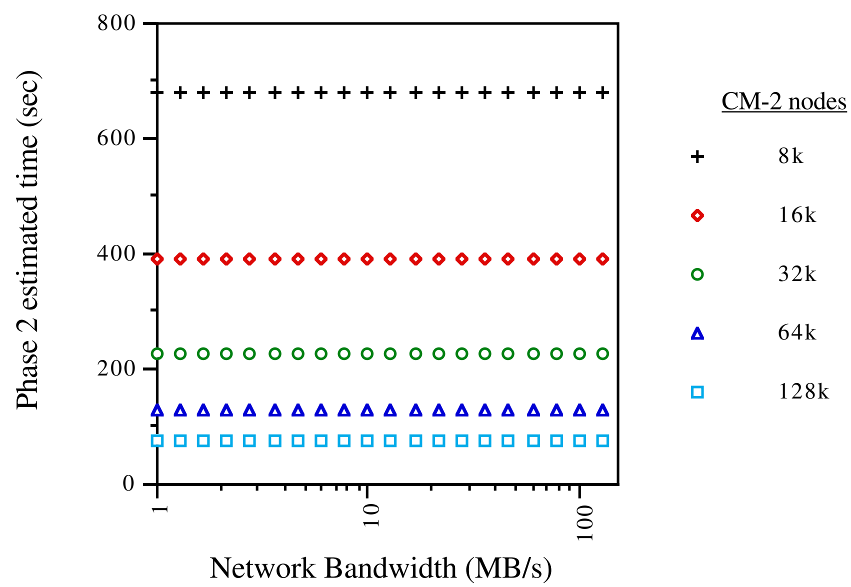


Figure A2 Estimated execution time (sec) for LAP solver phase (size 4k) as a function of the network bandwidth (MB/sec) and the size of the CM-2 system (number of nodes). Sparse incidence matrix transferred from CM-2 to C-90.

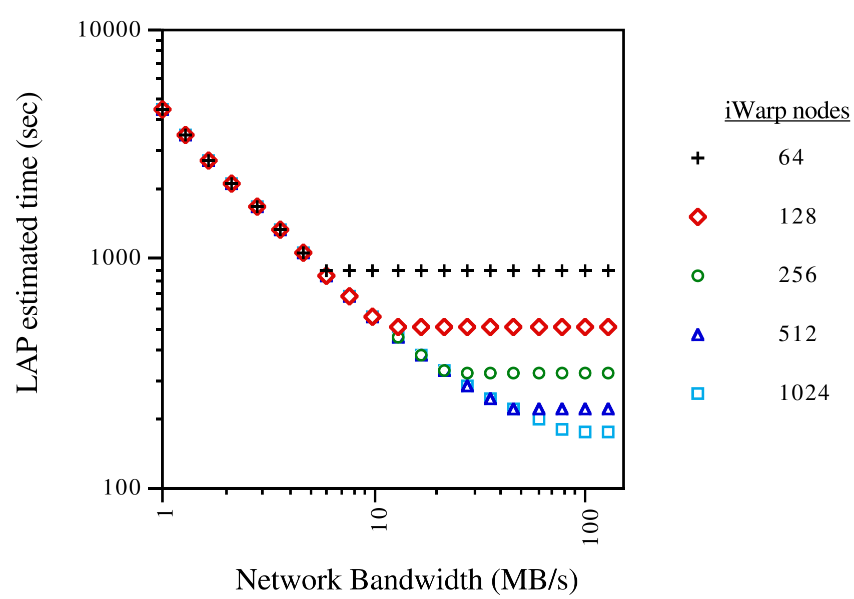
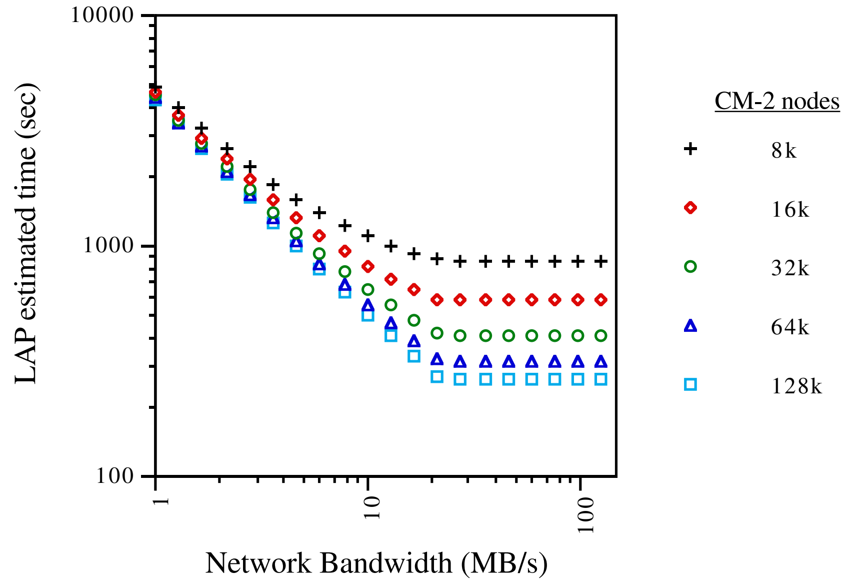
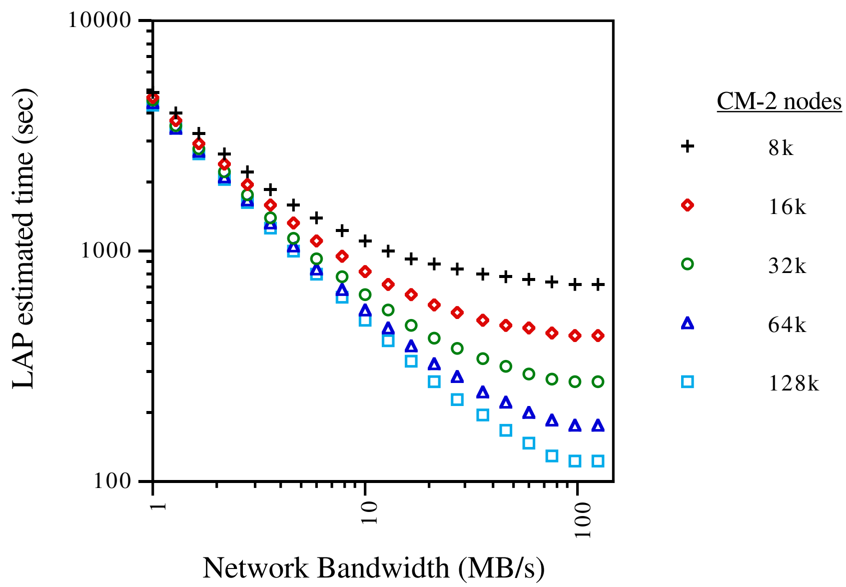


Figure A.3 Estimated execution time (sec) for combined LAP application (size 4k) as a function of iWarp/C-90/CM-2 network bandwidth (MB/sec) and iWarp system size (number of nodes) – sparse approach, 64k CM-2 nodes.



(a) 256 iWarp nodes



(b) 1024 iWarp nodes

Figure A.4 Estimated execution time (sec) for the LAP application (size 4k) as a function of the iWarp/C-90/CM-2 network bandwidth (MB/sec) and size of the CM-2 (number of nodes), for two iWarp system sizes, sparse incidence matrix transferred from CM-2 to C-90.

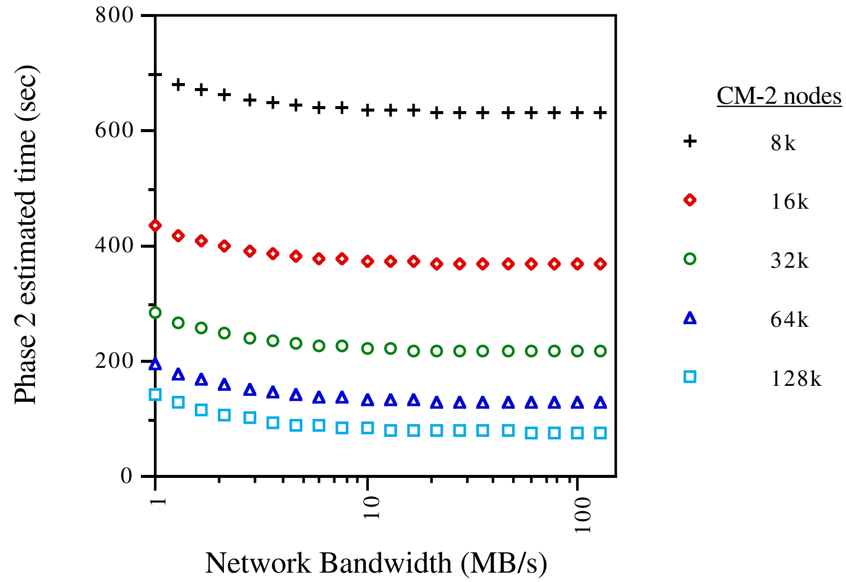


Figure A.5 Estimated execution time (sec) for LAP solver phase (size 4k) as a function of the network bandwidth (MB/sec) and the size of the CM-2 system (number of nodes). Dense incidence matrix transferred from CM-2 to C-90.

Author Biographical Sketches

Robert L. Clay is a Senior Member Technical Staff in the Distributed Systems Research Department at Sandia National Labs in Livermore, CA. His research interests and activities include: stochastic system characterization and optimization, large-scale modeling and optimization, heterogeneous supercomputing, scaleable numerical methods, and application of intelligent agents for distributed computing.

Robert holds a B.S. in Chemical Engineering from the University of Tennessee, Knoxville, and M.S. and Ph.D. degrees in Chemical Engineering from Carnegie-Mellon University, Pittsburgh, PA. Robert can be reached via (internet) email at rlclay@ca.sandia.gov; by phone at (510) 294-3193; by snail-mail at: Dr. R.L. Clay, MS 9011, P.O. Box 969, Sandia National Laboratories, Livermore, CA 94551-0969.

Peter Steenkiste is a Senior Research Scientist in the School of Computer Science at Carnegie Mellon University, Pittsburgh, PA. Peter's research interests involve high-performance networking and distributed computing. His research projects include:

- Gigabit Nectar (high-performance networking)
- Nectar testbed (distributed computing)
- Credit Net (ATM networking)
- Gems (Environmental modeling application).

Peter's mailing address is: School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891. He can be reached by phone at: (voice) 412-268-3261 : (fax) 412-268-5576, or via email at prs@cs.cmu.edu. The URL for Peter's web page is:

http://www.cs.cmu.edu/afs/cs/usr/prs/WWW/PeterSteenkiste.html