Architectural and Communication Issues in Designing Heterogeneous Parallel Systems with Optical Interconnection *

Ravi Prakash **Dhabaleswar K. Panda**
Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210.
Tel:(614)-292-5199, Fax: (614)-292-2911
E-mail: {prakash, panda}@cis.ohio-state.edu

Abstract

This paper investigates architectural and communication issues in designing heterogeneous parallel systems. As the speed of processors and parallel systems keep on increasing over years, electronic interconnections like HIPPI and FDDI are reaching their limit to interconnect several parallel systems to provide heterogeneous parallel computing environment. This paper explores the suitability of passive star-coupled optical interconnection using wavelength division multiplexing as the system interconnect to provide high bandwidth (Gbits/sec) communication demanded by heterogeneous systems. Several different communication strategies (combinations of communication topologies and protocols) over *Wavelength Division Multiplexed* (WDM) communication media like optic fiber are investigated. A representative master-slave computational model is used to evaluate and determine suitable communication architecture for such systems. The interplay between system speed, network speed, task granularity, and degree of parallelism are studied. It is shown that a hierarchical ALOHA-based communication strategy between the master and the slaves, implemented on top of the passive star-coupled network, leads to a considerable reduction in channel contention. Compared to direct point-to-point communication, it is demonstrated that hierarchical interconnection provides $50\% - 80\%$ reduction in task completion time for applications with medium to high degrees of parallelism. Comparable reduction in channel contention is also shown to be achieved by using tunable acoustooptic filters at master nodes. These two strategies demonstrate significant potential for building future generation heterogeneous parallel systems capable of delivering high performance.

Keywords: Heterogeneous parallel systems, hierarchical networks, optical interconnection, meta-parallelism, task scheduling, time division multiplexing, tunable filters, wavelength division multiplexing.

*A preliminary version of this paper [24] has been presented and received an *Outstanding Paper Award*, at the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks, Kanazawa, Japan, December 14–16, 1994. This research is supported in part by the National Science Foundation Grant # MIP-9309627.

Contents

1 Introduction **1**

2 Heterogeneous Parallel Systems **3**
2.1 A Typical Heterogeneous Parallel Architecture 3
2.2 A Representative Master-Slave Computation Model 4
2.3 Task Scheduling Strategies 6
2.3.1 Subtask Code Migration 6
2.3.2 No Code Migration 7
2.3.3 Qualitative Comparison of the Two Strategies 9

3 Architectural and Communication Requirements **9**
3.1 Processing Speed 10
3.2 Disk I/O Bandwidth 10
3.3 Communication Bandwidth and Latency 11

4 Using Optical Interconnection Architecture **12**
4.1 Design Components 12
4.2 Medium Access Protocol 13
4.2.1 Communication with Tunable Transmitters 14
4.2.2 Communication with Tunable Receivers 14
4.2.3 Qualitative Comparison 15
4.3 Contention Resolution Protocol 15
4.4 Communication Topology 16
4.4.1 Direct Communication Topology 16
4.4.2 Hierarchical Communication Topology 17
4.4.3 Tunable Optic Filters at Master Nodes 19

5 Simulation Experiments and Results **20**
5.1 Simulation Environment 20
5.2 Effect of Varying Data Size 23
5.3 Effect of Varying Degree of Parallelism 25
5.4 Inflexibility of TDMA Protocol 28
5.5 Effect of Intermediate Processing of Data 29
5.6 Comparison between subtask-code-migration and no-code-migration 30

6 Summary of Evaluations and Design Choices **32**

7 Conclusions **34**

1 Introduction

A large number of computational tasks exhibit coarse-grain heterogeneity. Such tasks can be divided into several subtasks such that each subtask has a different kind of parallelism embedded in it. Previous efforts towards parallelizing the execution of such tasks have involved running all the subtasks on a set of identical processing units. This has led to a poor exploitation of the inherent parallelism in the subtasks.

Heterogeneous processing [8, 14, 16, 20, 21] aims to exploit the coarse-grain heterogeneity of an application task by executing it on a suite of heterogeneous parallel computers. In such systems, parallel computers with diverse architectures (SIMD/MIMD/vector, etc.) are connected through a network. The basic idea is to execute a subtask on the parallel computer that is most suitable for it among those available in the network.

Previous studies in heterogeneous computing have concentrated mainly on the software aspects: determining the nature of parallelism inherent in the subtasks, finding a suitable match between a subtask and a parallel computer, and the penalties of non-suitable subtask assignments [8, 14, 21, 31]. However, very little research has been done to study the architectural and communication aspects of such systems. Since heterogeneous systems emphasize coarse-grain meta-parallelism, they involve high volume of communication that is bursty in nature. Also, innovations in the area of host network interfaces, as done in the Nectar project [22], can enable nodes of a heterogeneous system to pump data of the order of 1 gigabyte/second between their memory and the external network. Hence, there is a need for high bandwidth and low latency communication networks to build heterogeneous platforms.

Recently, high speed communication networks like HIPPI [30], FDDI [25], and Nectar [4, 22] are being used to interconnect high speed parallel computers. These networks use optic fiber as the communication medium to transmit data between the nodes. The constituent nodes in a typical heterogeneous system can pump data into the network at a rate greater than the bandwidth provided by HIPPI (100 Mbytes/s) and FDDI (100 Mbits/s). Obviously, the increase in communication bandwidth has not kept pace with the increase in processing speed. The theoretically available bandwidth of an optic fiber cable is orders of magnitude greater than the bandwidth achieved by HIPPI and FDDI. This raises some interesting issues: What prevents HIPPI and FDDI from achieving a higher bandwidth? Is it possible to design *communication protocols* to achieve higher bandwidths? Can the state-of-the-art technology

support such protocols? Finally, how suitable are these protocols for the communication patterns exhibited by tasks executing on heterogeneous parallel systems?

This paper explores such challenging issues and shows that existing high speed networks have high latency and limited achievable bandwidth because they employ electronic network protocols. It also shows that higher bandwidth can be achieved by employing *optical switching*. The recent developments in fiber-optics networks promise data transfer rate of the order of 1 Gbits/sec. One such development is *passive star-coupled interconnection* using *wavelength division multiplexing* [5, 7, 10]. This technology is currently being explored to provide high performance interconnection for parallel systems [6, 29]. However, it has not been studied in the context of heterogeneous parallel systems.

In this paper, we develop guidelines for building heterogeneous parallel systems with passive star-coupled interconnections. Unlike studying the performance of an interconnection network in isolation, we consider an integrated approach in deriving the best communication protocol and topology for heterogeneous parallel systems using optical interconnection. We consider the programming model, task scheduling strategies, application characteristics and network performance together. The commonly used *master-slave* computation model is used to evaluate our system. In this model, the subtasks executing on different slave computers communicate with each other through message passing via the master node. As an alternative to the slaves directly sending the results of their subtasks to the master, which is known as *direct communication*, we propose two new approaches using optical interconnections: a tree-like *hierarchical interconnection* topology and a communication scheme that employs *tunable optic filters* [7] at the communicating nodes. The passive star-coupled interconnection can support a variety of communication protocols for master-slave communication, like $ALOHA$ [3] or time-division multiplexed access ($TDMA$) to the shared channel. We analyze these protocols and topologies from the standpoint of *medium access control* as well as *contention resolution*.

Task completion time in a heterogeneous parallel system is also influenced by the I/O bandwidth of the disks and the availability of disk space at the nodes. If sufficient disk space is available at a slave node, the slaves can invoke suitable library routines for a subtask and fetch their object code from local disks. This scheduling strategy is defined as *no-code-migration*. In this strategy, the master needs to send only the data for the subtask. An alternative strategy, call *subtask-code-migration*, is also possible where both the object code

and data for the subtask are sent from the master to the slave. In this paper we consider both these task scheduling strategies in our evaluation. Various combinations of communication protocols, topologies and task scheduling strategies are evaluated qualitatively as well as quantitatively through simulations. Is it shown that hierarchical interconnection and tunable optical filters at the master node can reduce channel contention in the network in a significant manner.

The paper is organized as follows: In Section 2 we provide an overview of heterogeneous parallel system, master-slave computation model and two task scheduling strategies. Section 3 evaluates the processing, communication and disk I/O requirements of heterogeneous systems. In Section 4 we describe the proposed optical interconnection architecture and communication strategies. Section 5 quantitatively evaluates the performance of the proposed communication schemes through simulation. The suitability of different communication strategies for building practical heterogeneous systems is discussed in Section 6. Finally, the conclusions are presented in Section 7.

2 Heterogeneous Parallel Systems

A heterogeneous parallel system consists of a suite of autonomous parallel computers connected to each other by a high-speed communication network. Such a system is suitable for executing applications that exhibit coarse grain meta-parallelism. In this section, we provide an overview of heterogeneous parallel architectures, task graphs with precedence relationships, and a master-slave execution model to execute task graphs on heterogeneous systems. Two different task scheduling approaches, *subtask-code-migration* and *no-code-migration*, are introduced and evaluated with respect to their computation, communication and I/O overheads.

2.1 A Typical Heterogeneous Parallel Architecture

A heterogeneous parallel system consists of a set of parallel computers, each designed to exploit a different kind of parallelism (SIMD/MIMD/vector, etc.). Each computer has its own disk. The computers can communicate with each other through a high speed communication network connecting them. An example of such a system is shown in Fig. 1.

Applications targeted for heterogeneous systems exhibit coarse granularity [13]. They

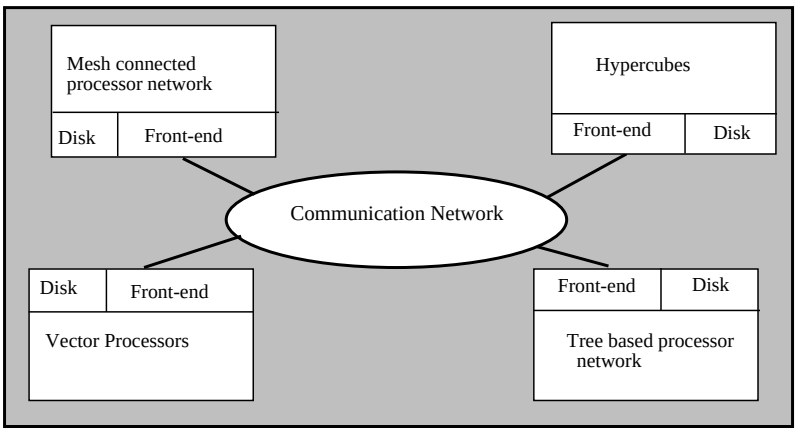

Figure 1: An example heterogeneous parallel system with diverse architectures.

can be divided into a number of subtasks having different kinds of parallelism embedded in them (SIMD/MIMD/vector). These subtasks also have certain control and data dependencies amongst them. Based on these dependencies, the execution of the application task can be expressed as a precedence relationship among the subtasks, as shown in Fig. 2. Two sub-

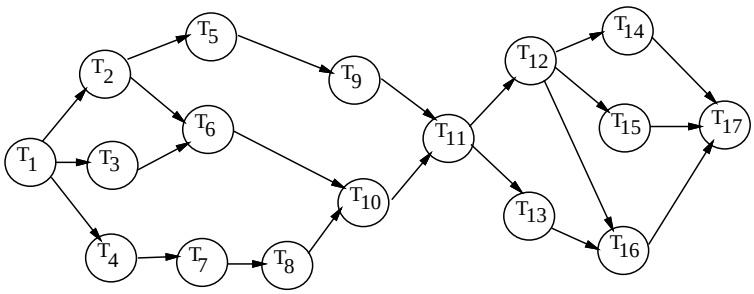

Figure 2: A coarse grained task graph showing precedence relations between subtasks.

tasks with no precedence relation between them can be executed concurrently, typically on separate parallel computers. As the subtasks exhibit different kinds of parallelism, executing all of them on a homogeneous parallel computer or on a network of homogeneous parallel computers will not yield optimum performance. Hence, a heterogeneous set of parallel computers are generally used to execute such tasks leading to better performance [13].

2.2 A Representative Master-Slave Computation Model

Tasks suitable for heterogeneous computing are large scientific computations in diverse areas like medical imaging, fluid dynamics, finite-element analysis, combinatorial analysis, protein sequencing, etc. [21, 23]. Based on the precedence relations, some of the subtasks can be executed concurrently on different computers in the network. Let there be n_1 subtasks

that execute concurrently on a set of n_1 computers and produce their respective results. Let there be n_2 subtasks, mapped onto a different set of computers, that use the results produced by all these n_1 subtasks as their input data. Using a commonly used multiple-phase master-slave model as shown in Fig. 3, the master computer spawns n_1 subtasks on

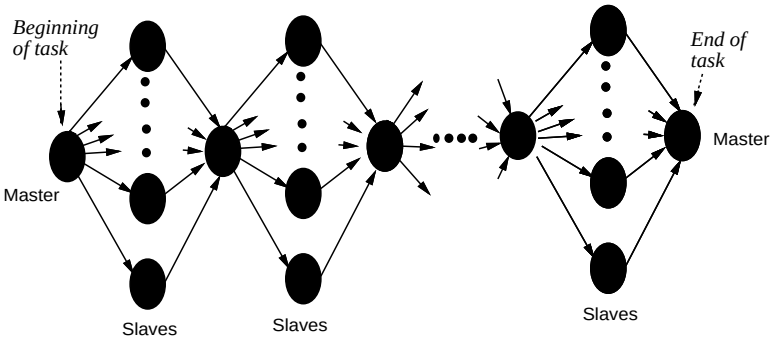

Figure 3: A multiphase master-slave model of task execution.

different slave computers in the beginning of a phase. Each slave sends its result to the master on completion. The phase ends when the master has received results from all the n_1 subtasks. In the next phase the master spawns n_2 subtasks on different slaves and sends the appropriate input data to each of them. Thus, the number of messages required to distribute the results of the n_1 subtasks to the n_2 subtasks succeeding them is $n_1 + n_2$. This compares favorably with the $n_1 \times n_2$ messages required for point-to-point communication between these two sets of slaves. Such master-slave model, with centralized information about all computers, also helps in achieving load balancing in a heterogeneous system [17]. It is to be noted that the number of subtasks spawned concurrently in a phase is known as the *degree of parallelism* of that phase.

The task graph shown in Fig. 2 is quite suitable for multiple-phase master-slave execution. Initially, the master spawns T_1 on a slave. On its completion, subtasks T_2, T_3 and T_4 are spawned concurrently on three different slaves. The completion of these subtasks marks the end of the second phase. The master collects their results and then concurrently spawns T_5 and T_6 on two different slaves and schedules T_7 followed by T_8 on a third slave computer. The completion of these four subtasks marks the end of the third phase. Similarly, T_9 and T_{10} correspond to the fourth phase, and so on.

2.3 Task Scheduling Strategies

Besides the communication overheads associated with the master-slave model of execution, the completion time of a task depends on the scheduling strategy employed to assign its subtasks to the different nodes of the heterogeneous system. A good strategy is one that schedules a subtask on a machine that exploits the kind of parallelism embedded in the subtask. It also considers the capacity and the bandwidth of the main and auxiliary memory at the nodes of the system as they can influence the task completion time. In heterogeneous systems all the tasks are generally submitted at one particular node which acts as a front-end for the entire system. First, *code profiling* [21] is done to identify the different kinds of parallelism in a task. Based on this information, the task is broken down into several subtasks, each exhibiting a different kind of parallelism. Then *analytical benchmarking* [21] is done to determine the suitability of a particular computer for a given type of parallelism. Subsequently, the operating system schedules the subtasks on the different computers in the network.

Without loss of generality, let us assume that there are always a sufficient number of computers available and each subtask can be allocated to the computer which best exploits the nature of parallelism in that subtask. This assumption is based on the *Optimal Selection Theory* [8, 21, 31]. It can also be assumed that only one task is being executed by the system at any given time. This avoids external interference from any other task. Also, the main memory at a slave computer is assumed to be big enough to store the executable code of any subtask and the data that it works with. Now, let us consider two possible strategies for scheduling tasks on heterogeneous parallel systems.

2.3.1 Subtask Code Migration

In this scheduling strategy, both the executable code and the data corresponding to a subtask are migrated from the front-end computer of the master to the front-end of the slave, where it is to be executed. The motivation behind this strategy is that sometimes a task cannot be split into subtasks corresponding to routines in the software libraries of the slave computers. Since the master sends code and data to multiple slaves, these transfers get queued up at the master node. While the master is migrating the code and data for one subtask, other subtasks, belonging to the same phase, that were migrated earlier, may be executing at the corresponding slaves. This leads to an overlap between computation and communication in

execution of the application, as shown in Fig. 4.

Let us consider that i subtasks in a phase have been migrated by the master before the migration of subtask $i+1$ starts. So, the time elapsed between the beginning of a phase and the instant the master starts migrating the code and data for the $i+1^{st}$ subtask can be expressed as:

$$\sum_{j=1}^{i}[t^{setup} + t^{transfer} \times (\mid code_j \mid + \mid data_j \mid)]$$

where t^{setup} is the communication set-up time and $t^{transfer}$ is the time to transfer one unit of code or data between a pair of computers.[1] After the code and data for the $i+1^{st}$ subtask have been migrated to the appropriate slave, the slave executes the subtask and sends the result back to the master. Therefore the time at which the result from the $i+1^{st}$ subtask reaches the master's queue is given by the following expression :

$$T_{i+1} = \sum_{j=1}^{i+1}[t^{setup} + t^{transfer} \times (\mid code_j \mid + \mid data_j \mid)] + t_{i+1}^{execute} + t^{setup} + t^{transfer} \times \mid result_{i+1} \mid$$

where $t_{i+1}^{execute}$ is the time to execute the $i+1^{st}$ subtask at the corresponding slave. Figure 4 shows the execution of a phase consisting of n subtasks. The thicker ovals represent actions taken by the master. In this model *transmit* t_i indicates the transmission of the executable code and data for the i^{th} subtask to the appropriate slave. The completion time of a phase is equal to $Max_{i=1}^{n}(T_i)$.

2.3.2 No Code Migration

In this strategy, the master computer sends only the input data (not the executable code) for the subtask to the appropriate slave computer. It is assumed that on receiving the data, the slave invokes the routine corresponding to the subtask from its library. This is similar to a remote procedure call [28]. The actual implementation of the subtask at a slave is optimized to exploit the architectural features of the slave, and to provide the best possible performance for the subtask at that computer. Once again, Fig. 4 represents the execution of a phase consisting of n subtasks. It is to be noted that in this *no-code-migration* strategy, *transmit* t_i corresponds to the transmission of only the data for the i^{th} subtask from the master to the slave. Thus, the completion time of a phase consisting of n subtasks can be

[1] For the sake of simplicity we assume that $t^{transfer}$ is the same between every pair of computers.

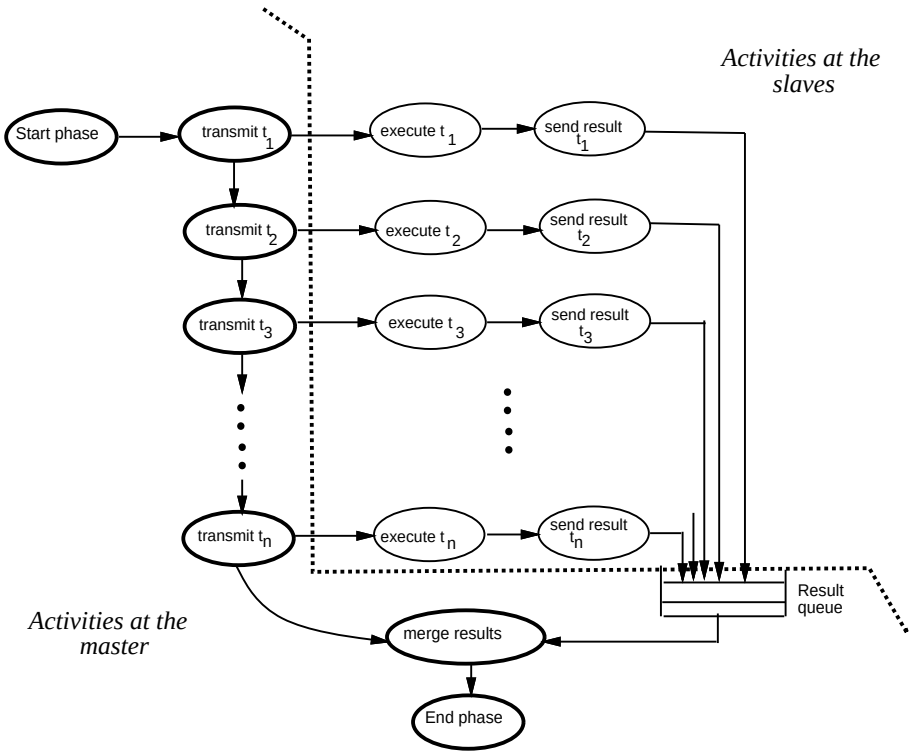

Figure 4: Temporal representation of master-slave activities in a phase (t_i: subtask i).

expressed as:

$$Max_{1 \leq i \leq n}[t_i^{start} + t_i^{comm} + t_i^{execute} + t_i^{paging}]$$

In this expression, t_i^{start} is equal to the time elapsed between the start of the phase and the instant that the master starts sending the data for subtask i to the corresponding slave. This is equal to:

$$\sum_{j=1}^{i-1}[t^{setup} + (t^{transfer} \times \mid data_j \mid)]$$

The component t_i^{comm} is the time taken to communicate the data and result for subtask i between the master and the slave, and is equal to:

$$2 \times t^{setup} + t^{transfer} \times (\mid data_i \mid + \mid result_i \mid)$$

The component t_i^{paging} is the time spent fetching pages of the executable code for subtask i into the main memory of the slave's front-end processor, from the library in auxiliary storage. If the code size of the subtask is equal to S_i, there will be $\lceil S_i/P \rceil$ page faults, where P is the size of a page. If the time to fetch a page from auxiliary memory is equal to t^{disk}, t_i^{paging} is equal to $\lceil \frac{S_i}{P} \rceil t^{disk}$. As stated earlier, we assume that the main memory of a slave is big enough to store the code and data of its subtask. A smaller main memory may lead to

additional page-faults, whose number would depend on the relative size of the memory and the subtask code and data, and the pattern of reference. Considering the rate at which the size of the main memory of computers is increasing we can safely assume that the executable code, data, and result for a subtask will fit in the main memory. So, we do not consider the additional page-faults mentioned above.

2.3.3 Qualitative Comparison of the Two Strategies

Both the strategies have their own strengths and weaknesses. When the executable code is not migrated, great demands are placed on the auxiliary storage of the slaves as the software libraries have to be maintained at all the sites. The appropriate routines have to be copied from the software library, thus incurring the paging overheads. But the only communication required between the master and the slaves is to pass the input data and results. Thus, the network traffic is low.

When *subtask-code-migration* is employed, software libraries do not have to be maintained at the slaves. Hence, a large auxiliary storage is not required. But, the volume of network traffic is high. The resultant communication time contributes to the overall execution time of the task. It is to be noted that for *subtask-code-migration* to be unconstrained by the nature of the computers in the system, machine-independent and portable parallel programming languages and tools are needed. This requires *cross-parallel compilers* for all the machines and appropriate debuggers [21].

Over the years, disk I/O bandwidth and network communication speed have not grown at the same rate as the speed of processors. With such a trend in mind, as the processing speed increases, it is obvious that the fraction of the task completion time corresponding to subtask migration and page fault penalty will grow over time. This has the potential to limit the possible speedup in a heterogeneous system. Hence, these costs need to be minimized while designing a high-performance heterogeneous system.

3 Architectural and Communication Requirements

The subtasks of an application task can be scheduled using either the *subtask-code-migration* or *no-code-migration* strategy. The processing speed of the nodes on which the subtasks are scheduled will have a significant impact on the task completion time. If the communication

network between the nodes has a low bandwidth and/or high latency the task completion time will increase. As the problem size grows, greater demands are also placed on auxiliary memory like disks to store data and provide it to the processors at a fast rate and with low latency. In this section we briefly discuss these requirements.

3.1 Processing Speed

The performance of the currently available parallel computers and of those expected to be available in the next few years has been reported in [1, 2, 15, 27]. The processing speeds of the nodes that can constitute a heterogeneous system is growing at a fast pace. A 1024 node CM-5, with a peak performance of 131 gigaflops is an example of the processing power currently available, and an indicator of things to come. Even if the sustainable performance is assumed to be 10% of the peak, 13.1 gigaflops can be executed every second. So, it is realistic to assume that each node in a heterogeneous system will have a comparable computational rating. In the master-slave computation model, each slave will correspond to such a node. There is an implicit assumption that different slaves exploit different kinds of parallelism (SIMD, MIMD, vector, etc.) and a subtask gets assigned to the most suitable slave.

The time required to copy the code and data from the front-end of the slave to each of the constituent processors of the slave before the computation starts, and the time to gather the result at the front-end will contribute to the total execution time at the slave. We assume that the subtask code is broadcasted to each constituent processor while a small and distinct fraction of the data is sent to individual processors. Emerging high-speed DRAMs like RamBus and RamLink can provide memory bandwidth of up to 500 megabytes/second [1]. Modern designs used in SCI-based local area multiprocessors [2, 15] are also capable of supporting a transfer rate of 1 gigabyte/second between the front-end and the computational processors at a slave computer.

3.2 Disk I/O Bandwidth

With the increase in processing power, greater demands are placed on the storage disks. In the *no-code-migration* strategy the code for a subtask is initially fetched from the disk at the slave. In the *subtask-code-migration* strategy the code for all the subtasks has to be fetched from the disk at the master. The I/O bandwidth provided by the disk storage

of the currently available supercomputers is low, compared to their data processing rate. For example, the basic disk storage node and the data vault of CM-5 provide peak transfer rates of 17 megabytes/second and 25 megabytes/second, respectively. The Intel Paragon's disk provides a peak transfer rate of 100 megabytes/second [19]. Disk transfer rates can be further increased through architectural innovations such as connecting an array of disks to the high speed network mentioned above and employing interleaved disk access.

However, over the years, disk *access time* has not decreased at the same rate as the transfer rate has increased. The disk access time can be hidden in the *no-code-migration* approach in the following manner: Let the first few bytes of the message sent by the master containing the data for a subtask carry information about the library routine to be invoked at the slave. As soon as the slave gets this information it initiates a request to fetch the code from the disk. So, the disk access time gets overlapped with the data transfer time from the master to the slave. As soon as this data transfer finishes, the slave can start receiving the subtask code from the disk. For the *subtask-code-migration* approach, while the master is transmitting the code for one subtask from an output buffer to the slave, the disk may be sending the code for the following subtasks to a different buffer. Due to this overlap, the disk access time is not a factor in determining the overall task completion time.

3.3 Communication Bandwidth and Latency

The nodes in a heterogeneous system require high bandwidth communication links between them. The communication issue is complicated by the distance separating the nodes. Most of the previous solutions for handling inter-node communication in parallel systems have assumed that these nodes are on different boards inside a cabinet, or are computers sitting next to each other in a room. However, in a heterogeneous system, the nodes may be as far apart from each other as different corners of a city.

Local area networks are currently used to connect computing nodes a few kilometers apart. The twisted pair and coaxial cable communication media currently used by these networks have a limited bandwidth over distances of a few kilometers [28]. They cannot sustain the traffic produced by a heterogeneous system, which can be of the order of several gigabits per second, over the distances separating the constituent nodes. A single coaxial cable cannot carry several data channels of 1-gigabit bandwidth, even over short distances. Establishing dedicated end-to-end coaxial cable links between every pair of nodes is not

economical. Besides, several repeaters may be required along the path and each repeater will contribute towards communication latency.

The high performance parallel interface (HIPPI) physical layer standard is being suggested as the backbone for connecting high performance computers [30]. HIPPI defines full duplex parallel interfaces that can run at speeds up to 200 megabytes/second [27, 30]. It has an Ethernet connection in its data path which becomes a bottleneck. The high overhead associated with Ethernet communication also limits the data transmission rate that can be sustained. Establishing and terminating a HIPPI connection also takes a long time due to round-trip delays across the Ethernet [30].

On the basis of the preceding discussion, it is obvious that the communication media and protocols currently in use are not suitable for building future heterogeneous systems. There is a definite need for multiple high bandwidth, low latency communication channels. In the following sections, we propose solutions to this problem by using optical interconnection and a set of new communication strategies.

4 Using Optical Interconnection Architecture

As discussed above, currently available electronic switching hardware cannot support multiple low latency communication channels, each with a sustainable data rate of 1 gigabyte/second. However, technological advancements in the area of optical switching, routing and data transmission [7, 12] have enabled optic-fiber networks to meet these requirements. In this section, we show how optic-fiber cables and an entirely new set of communication protocols and topologies can be used to support high bandwidth and low latency communication requirements of heterogeneous parallel systems. Diverse issues are considered in the design of efficient optical interconnection architectures.

4.1 Design Components

Three major issues (shown in Fig. 5) should be considered while designing optical networks. When multiple nodes try to use an optic-fiber cable to send data, the available bandwidth can be accessed either in a time-division multiplexed (TDMA) fashion or in a wavelength-division multiplexed fashion (WDMA). Irrespective of the medium access strategy employed, there is a likelihood of multiple nodes trying to use the communication medium at the

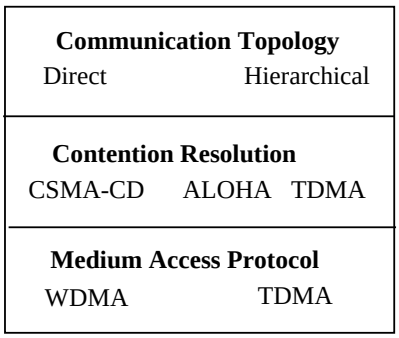

Figure 5: Design components for optical interconnection architecture.

same time, leading to contention. Hence, having settled on an appropriate medium access protocol, a suitable contention resolution protocol should be adopted. The contention for the communication medium and the achievable bandwidth are also significantly influenced by the communication topology. Let us, now, discuss each of these issues in some detail.

4.2 Medium Access Protocol

Recent technological advancements have made it easier to achieve higher transmission bit-rates using wavelength-division multiplexing instead of time-division multiplexing [7]. Also, the bandwidth of the fiber is most easily accessed in the wavelength domain than in the time domain [7]. Hence, em wavelength division multiplexed access (WDMA) is the most suitable medium access protocol for optical interconnection architectures.

WDMA protocols are being developed to transmit several 1-gigabit data streams through a single optic-fiber cable [5, 7, 10]. These protocols assume passive optical star-coupled networks with *preassigned* channels (wavelengths). Figure 6 illustrates such a network. The individual signal streams, along different wavelengths, from the source nodes are combined in a passive star coupler. The combined signal is sent on each outgoing fiber. At the destination node optical filters can select the desired signal stream and reject the remaining streams. The protocols have low complexity, which is especially important since there is still a great mismatch between the speeds of the optical network interfaces and the bandwidth of the fiber [5]. The protocols are also scalable since large and energy efficient passive star couplers with low latency can be designed [9]. We present brief descriptions of two WDM access mechanisms. Based on their relative merits we will select one of the access mechanisms for use in heterogeneous systems.

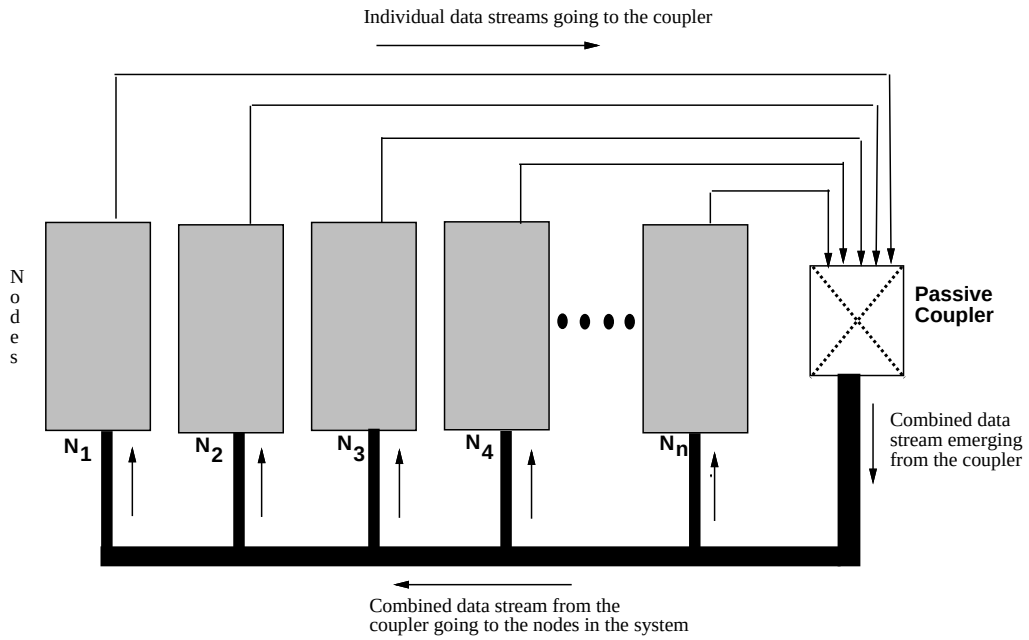

Figure 6: A passive star-coupled optical network using wavelength division multiplexing.

4.2.1 Communication with Tunable Transmitters

In this access mechanism, each node is assigned a *home frequency* (channel) at which it can receive data. The senders tune their transmitters to the receiver's channel. In the beginning of a phase of the master-slave execution model, the master sends data, one at a time, to the slaves along their respective channels. Each slave attempts to send its result to the master, on the master's channel. If two or more slaves send their results to the master at the same time collision results. The impact of collisions on the performance of the system depends on the number of slaves contending for the master's channel and the duration of their transmissions.

4.2.2 Communication with Tunable Receivers

In this access mechanism, each node can transmit along only one channel – its home channel. The receivers have to tune to the sender's channel to receive the data. So, once the slaves have tuned to the master's channel, the master can send data to the slaves one after the other. On finishing its computation, a slave tries to send its result to the master by requesting the master to tune to the slave's transmission frequency. If the master is tuned to some other frequency the slave waits until it gets a positive acknowledgment from the master. For master-slave communication, using tunable receivers and fixed frequency transmitters there are no collisions.

4.2.3 Qualitative Comparison

If tunable receivers are used, a separate control channel is needed to carry the tuning requests and acknowledgments between nodes. The transmitters should be able to transmit at two frequencies — their home frequency and the frequency of the control channel. Besides, tuning requests can collide on the control channel. Also, advances in the areas of fast tunable transmitters exceed those in fast tunable receivers, and the tuning times for receivers are relatively long as compared to the packet transmission times and their tuning range is small [10]. Thus a tunable receiver protocol will have significant overheads that may offset the advantages due to the absence of contention. Tunable receivers are suitable for broadcasting or multicasting data, i.e., if the input data to several slaves is the same. In most heterogeneous applications different subtasks, being executed concurrently on different computers, will be working on separate data. Hence, the broadcasting capability of the *communication with tunable receivers* access mechanism cannot be exploited. *Hence, we choose the tunable transmitter access mechanism.* Having decided upon tunable transmitters for heterogeneous systems, appropriate contention resolution strategy and communication topology need to be selected.

4.3 Contention Resolution Protocol

As mentioned in Section 4.2.1, using tunable transmitters will lead to contention for the master's channel and result in collisions. In order to maximize the achievable bandwidth in the network, such collisions have to be minimized.

If a carrier sense strategy, like CSMA-CD [18], is employed the slaves will send data by tuning to the master's channel, and at the same time *sense* the channel for collisions by *listening* to the channel. A slave will stop transmitting as soon as it detects a collision. Then it will wait for a random period of time before retrying to send the result. Such a strategy is not suitable for the tunable transmitter medium access protocol because the receivers at the slaves are tuned to sense signals traveling along their channel only. Tuning these receivers to sense the presence of traffic along the master's channel will introduce excessive delays.

Since the slaves' receivers are not tuned to the master's channel, a slave can know about a successful transmission to the master only on receiving an acknowledgment from the master. Hence, the CSMA-CD protocol will deliver a performance that is hardly better than the

comparatively simpler ALOHA protocol [3] which does not have to continuously sense the channel. The ALOHA protocol transmits the entire data and relies on the feedback properties of the network to detect collisions. So, due to its simplicity, using the ALOHA protocol for communication with the master during result transmission is advisable.

Channel contention can also be reduced by employing time-division multiplexed access (TDMA) to the master's channel. In this strategy, an interval of time, called a TDMA cycle time, is divided into slots of finite duration. Each slave is assigned a time slot and can directly send its result to the master, along the master's channel, only during this slot. If the slave misses its time slot, it has to wait for its slot in the next cycle. If the number of slaves in a phase is less than or equal to the number of slots in a cycle, each slave gets its own slot and there is no contention among the slaves.

However, TDMA lacks flexibility because the number of slots in a cycle and the duration of each slot are fixed *a priori*. If the number of slots is more than the number of slaves in a phase, several time slots remain *unutilized*. Fewer slots than the number of slaves in a phase will lead to *contention* among the slaves for some slots. If the slot duration is greater than the time needed by a slave to send its result to the master, the slot is *underutilized*. So, TDMA running on top of the WDM medium access protocol will have a good performance only for a small subset of applications whose degree of parallelism and slave result sizes match well with the number of slots in a cycle and the slot duration, respectively.

Therefore, the ALOHA protocol and the TDMA protocol appear to be suitable for use with tunable transmitters and will be quantitatively evaluated through simulation experiments.

4.4 Communication Topology

The ability of a contention resolution strategy to minimize collisions depends in great part on the logical communication topology and the number of nodes contending for a given communication channel in a phase (degree of parallelism). Let us evaluate communication topologies which can deliver better performance for communication with tunable transmitters.

4.4.1 Direct Communication Topology

In the master-slave model, this communication topology implies that all the slaves directly send their results to the master node, at the end of a phase, on the master's channel by

tuning their transmitters to that channel. This is the simplest topology and can be blended well with the ALOHA protocol. Each slave will send its result along the channel without checking the status of the channel. There may be collisions between the data streams. So, corresponding slaves will have to back-off and retry after a random period of time. As the degree of parallelism of a task increases, more slaves in a phase are concurrently trying to send their results to the master. Hence, the chances of collision rise. If this problem is not addressed, the expected decrease in task completion time, due to increased parallelism, will be offset by the contention penalty. If a single communication channel is being shared by all the slaves, there will be great contention for the channel. The contention will also increase with increase in the size of the result that the slaves have to send to the master.

4.4.2 Hierarchical Communication Topology

As the degree of parallelism is a feature of the task being executed, it is beyond the control of a designer of a heterogeneous parallel system. Thus a communication topology capable of delivering good performance over a wide range of degrees of parallelism in an application is needed. We propose a new *hierarchical communication topology* to deliver better performance by reducing the impact of contention.

In this topology, additional nodes, called sub-masters, are introduced into the system. The slaves in a phase are assigned to mutually disjoint sets with each set corresponding to a sub-master. The slaves in a set send their results to their sub-master. Once a sub-master gets the results of all its slaves, it forwards them to the master. Thus, each slave has to contend for the communication channel leading to the sub-master with only those slaves that belong to its set. In this hierarchical communication topology, the number of sub-masters contending with each other for a channel to send the results to the master is significantly less than that in the direct communication topology.

Such hierarchical communication topology, with tunable transmitters, using WDM for use in heterogeneous systems is shown in Fig. 7. The master directly sends the input data for all the subtasks to the corresponding slaves without the involvement of the sub-masters. At the end of their computations, half the slaves send their results to one sub-master and the other half to the other sub-master. The assignment of slaves to sub-masters is done *a priori*. Once a sub-master gets the results corresponding to all its slaves, it combines and sends them to the master. The phase ends when the master has received results from both

the sub-masters. Only half the slaves contend for the channel of each sub-master, and only two sub-masters contend for the master's channel. Both TDMA and ALOHA protocols can be used in conjunction with the hierarchical topology. The slaves can have time multiplexed access to the channels of their sub-masters or they can employ the ALOHA protocol. Each sub-master can send the results it collects from its slaves to the master using the ALOHA protocol. In Fig. 7, the channels for which contention takes place are represented like buses for simplicity.

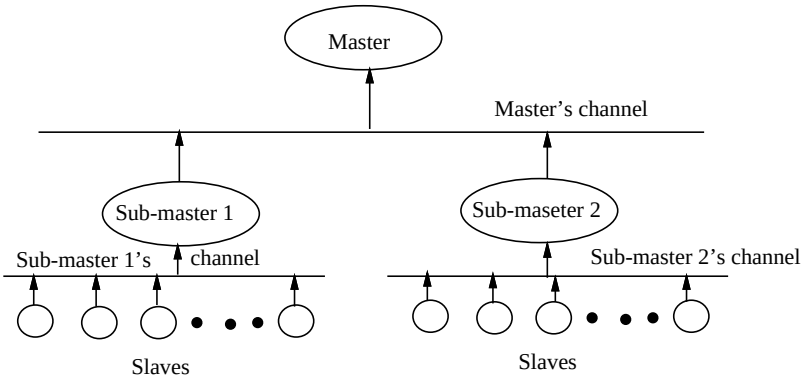

Figure 7: A hierarchical representation of channel contention during the result-receive period using two additional sub-masters.

The hierarchical topology promises a much better performance than the direct communication topology. The extent of channel contention between the contending nodes will depend on two factors: the number of nodes in contention and the duration of time that each node needs to use the channel.

For very large tasks with many subtasks, large size of data and result, and high degrees of parallelism, additional levels in the hierarchy between the master and the slaves may be needed. The communication topology may look like a tree with the number of children of interior node increasing as one goes further away from the master.

In some applications like computer vision, the sub-masters in the hierarchical topology can carry out some intermediate processing on the results received from the slaves. The size of the processed result sent by a sub-master to the master is smaller than the sum of the sizes of the results received from its slaves. We can model this situation of *data compaction* as follows:

- when a sub-master receives results as an $N \times N$ matrix from a slave, the computational complexity at the sub-master for intermediate processing is $O(N^2)$.

- the size of the result sent by a sub-master to the master is a fraction of the sum of the result sizes received from the slaves. If the application is capable of compacting results from slaves to $x\%$, the sub-master will send $x\%$ of the sum of the results to the master.

4.4.3 Tunable Optic Filters at Master Nodes

Using tunable optic filters is an elegant alternative to hierarchical topology. A tunable optic filter can be tuned to receive signals along one or more channels. Acoustooptic tunable filters can be tuned to receive signals on the desired channel by changing the acoustic drive frequency [7]. They have a broad tuning range and the drive power to tune such filters is quite low, 100–200 milliwatt. Also, acoustooptic filters can select more than one wavelength simultaneously. This enables a node to receive data along multiple channels concurrently.

Let us consider a communication topology where the master node is equipped with an acoustooptic filter that enables it to receive data from the slaves along *two* home frequencies. Half the slaves send their results to the master along one home frequency, and the other half along the other home frequency, using the ALOHA protocol.

Thus the sub-master nodes are no longer needed — a saving in terms of hardware cost. Moreover, the contention between the sub-masters for the master's channel, in the hierarchical topology, is completely eliminated by using such acoustooptic filters. Therefore, if the direct communication topology is used along with tunable acoustooptic filters, it is expected to have a better performance than the hierarchical topology. The relative advantage will be small for applications with smaller result sizes and lower degrees of parallelism. This is because the inter-submaster contention will be small in such situations. However, using the acoustooptic filters should be much more advantageous as the degree of parallelism increases and the result sizes increase.

Based on the above discussion, five different combinations of communication topology and contention resolution protocol appear to be interesting, and worth evaluating, for passive star-coupled optical interconnections that employ tunable transmitters. In the next section, we evaluate these combinations through simulation experiments. These combinations/strategies are:

1. *Direct, with ALOHA*: during each phase of execution, the slaves send their results

directly to the master node on the master's channel using ALOHA protocol.

2. *Direct, with TDMA*: the slaves send their results directly to the master node employing time division multiplexed access to the master's channel.

3. *Direct, with 2 channel tunable filter*: the master is equipped with a tunable filter enabling it to simultaneously receive data along two *home channels*. Each channel is used by half of the slaves to send their results directly to the master. The ALOHA protocol is used for both the channels.

4. *Hierarchical, with ALOHA+ALOHA*: half the slaves send their results along one sub-master's channel, while the other half send their results to another sub-master. ALOHA protocol is used by the slaves. Both the sub-masters send the collected results to the master, along the master's channel, using ALOHA protocol.

5. *Hierarchical, ALOHA+TDMA*: similar to the previous combination except that the slaves used TDMA protocol, instead of ALOHA protocol, to send their results to the sub-masters.

5 Simulation Experiments and Results

In Sections 3 and 4 we described the architectural and communication requirements of a heterogeneous system and proposed new communication strategies using passive star-coupled optic interconnection architecture and wavelength division multiplexing. The proposed communication strategies are designed to reduce communication latency and increase the communication bandwidth for heterogeneous parallel systems. Improvements in communication characteristics of the system should be manifested as a decrease in the overall task completion time. In order to observe the impact of the proposed strategies, their performance was studied through simulation experiments. A process-oriented discrete-event simulator using CSIM [26] was used.

5.1 Simulation Environment

The salient features of the simulation environment are described below:
System characteristics: We simulated the performance of a heterogeneous system consisting

of a set of parallel computers (nodes). Each node in the system was assumed to be have computational power equivalent to a 1024 node CM-5. The sustainable computational rating of each slave node was equivalent to 13.1 gigaflops (10% of the peak performance of a 1024 node CM-5). A sustainable data transfer rate of 100 megabytes/second between the local disk and the main memory of a node was assumed.

Communication characteristics: We assumed that the nodes were connected by a passive star-coupled network with fixed frequency receivers and tunable transmitters. The home channel corresponding to each node was assumed to have a sustainable bandwidth of 1 gigabyte/second. The mean distance between a master and a slave was assumed to be about 10 kilometers. The speed of light in a medium being inversely proportional to its refractive index, the speed of light in the fiber is 2×10^8 meters/second, giving a mean message propagation delay of 50 microseconds. We added to that about 10 microseconds as the message set-up time. Hence, a total of 60 microsecond delay was assumed between the time the first bit of a message is sent by the transmitting node and the time instant the corresponding bit arrives at the destination node. For TDMA communication, we assumed each time slot to be of 1 millisecond duration which is the time required to transmit a 400×400 matrix over the 1 gigabyte/second links we simulated. When tunable optic filters at the master were simulated, we assumed that the filter could be tuned to simultaneously receive data along two different home channels. Each home channel of the master was assigned to half of the slaves in a given phase *a priori*.

Application characteristics: The master-slave computation model was considered. Each task consisted of 1000 subtasks executed in multiple phases, as shown in Fig. 3. Two sets of simulations were done:

- The first set studied the impact of result size on channel contention and overall task completion time. It was assumed that the number of subtasks executed concurrently in a phase was uniformly distributed in the range 7–13 giving an average of 10 subtasks per phase. The data and result size was varied. For this set of experiments using TDMA protocol, the TDMA cycle was divided into 13 slots, each of 1 millisecond duration.

- The second set studied the impact of degree of parallelism on channel contention and overall task completion time. The data and result size was kept fixed while the number

of subtasks executing concurrently in a phase was varied. For these simulations, the number of slots in a TDMA cycle was not fixed at 13. Instead, it was set to be equal to the maximum number of slaves spawned in any phase of that run. So, for example if the number of slaves in a phase was uniformly distributed between 22 and 28, with the mean number of slaves per phase equal to 25, the TDMA cycle had 28 time slots of 1 millisecond each. This ensured that there was no contention between slaves for a time slot.

Each subtask in our model problem consisted of matrix operations. We used matrix operations just as a tool for modeling purposes. Subtasks may consist of a variety of different operations, depending on the nature of parallelism embedded in them. We assumed the subtask executable code size to be normally distributed with a mean of 8 megabytes and a standard deviation of 0.8 megabytes – a realistic assumption because usually the object code for a subtask has to be linked with several software libraries that make a significant contribution to the size of the executable code. Both the input data and result of a subtask were assumed to be dense matrices of equal size, with normal distribution. If the mean number of rows/columns of the matrices was N, then the standard deviation was assumed to be $0.3N$.

A significant fraction of numerically-intensive computations using $N \times N$ matrices involve LU-decomposition, finding the inverse, multiplication, or some iterative operations (for convergence to a result). The computation complexities of such operations are between $O(N^2)$ and $O(N^3)$. Hence, we modeled the time complexity of a subtask to be either $c_1 N^3$ or $c_2 N^2$, where c_1 and c_2 are integers, selected from a uniform distribution with ranges 5–10 and 20–100, respectively. The computational complexity of each subtask (N^2 or N^3) was also selected randomly.

Properties studied: We simulated the performance of the five communication topology–protocol combinations listed at the end of Section 4. For these different topology–protocol combinations, we studied the following effects on channel contention and task completion time:

1. effect of varying data size

2. effect of varying degree of parallelism

3. effect of data compaction

4. effect of *subtask-code-migration* and *no-code-migration* strategies

To study the impact of data size, different matrix sizes were simulated by varying N from 50 to 1200, with each element of the matrix being a 64-bit floating-point number. The impact of the degree of parallelism on task completion time was studied by keeping the data size fixed ($N = 800$) and varying the average number of slaves in a phase between $5 - 40$.

5.2 Effect of Varying Data Size

The relationship between subtask data size and task completion time was evaluated for both the *subtask-code-migration* and *no-code-migration* strategies and the five communication topology-protocol combinations mentioned in Section 4. The simulation results obtained are shown in Fig. 8.

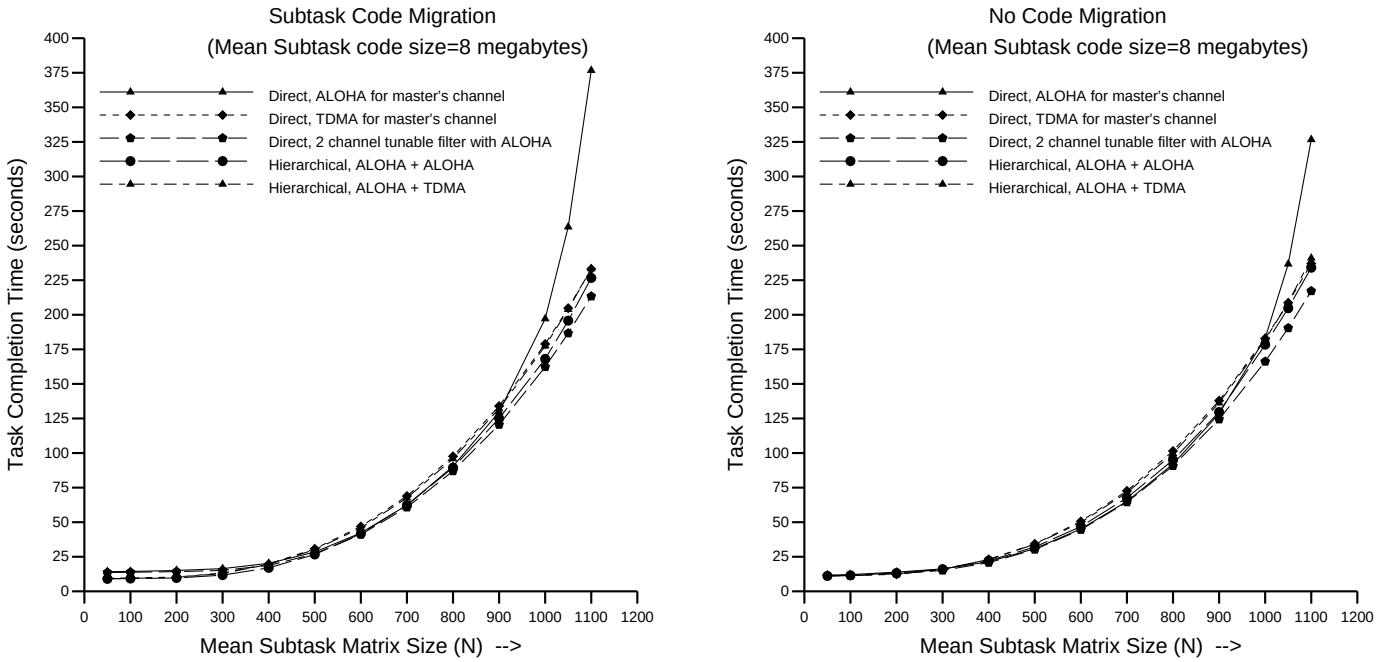

Figure 8: Effect of channel contention with varying data size on task completion time for the *subtask-code-migration* and *no-code-migration* approaches. Number of slaves per phase = 7–13 (mean = 10).

Both the *subtask-code-migration* and *no-code-migration* strategies exhibit similar impacts of variations of subtask data/result size on task completion time. When the size of the subtask results (assumed to be of the same size as the subtasks' input data) is small the

communication time is small. Hence, the collision probability for the ALOHA protocol is low and channel contention does not have a significant impact on the task completion time. For small result sizes ($N \leq 900$) all communication strategies have similar impacts on task completion time. But for larger result sizes ($N \geq 900$) the ALOHA protocol has many more collisions. Consequently, the task completion time becomes higher. It should be noted that the result size (the number of elements in the matrix) is proportional to N^2. Hence, a sharp rise in the task completion time is observed for higher values of N.

Direct-TDMA and hierarchical-ALOHA+TDMA strategies have almost identical performance and they are both better than the direct-ALOHA strategy for large values of N. This improvement in performance is due to the reduction/elimination of channel contention. As mentioned earlier, for the direct-TDMA protocol, the maximum number of slaves in a phase was 13 and the number of slots in each cycle was kept at 13. So, there is no contention between slaves for the same time slot. However, with the number of slaves in a phase always being less than or equal to 13 some slots are not utilized. Also, some slots are underutilized if there is not enough data to fill the slots. Therefore for intermediate data sizes ($400 \leq N \leq 900$) the direct-TDMA strategy has a slightly higher task completion time than the direct-ALOHA strategy.

The hierarchical-ALOHA+TDMA strategy eliminates contention between the slaves for the sub-master's channel. Also, only two sub-masters contend for the master's channel. Thus channel contention is significantly reduced. For smaller result sizes ($N \leq 900$) channel contention is low to begin with. So, the reduction in contention is not significant. However, the results of subtasks have to be transmitted twice: once from the slaves to the sub-masters, and then from the sub-masters to the master. Hence, the hierarchical-ALOHA+TDMA strategy has a slightly higher task completion time than the direct-ALOHA strategy for intermediate data sizes ($400 \leq N \leq 900$). But, for large data sizes the elimination of contention between slaves for each sub-master's channel more than offsets the cost of transmitting the results twice.

The hierarchical-ALOHA+ALOHA strategy has a slightly better performance than the three strategies mentioned above. This is due to the fact that only half the slaves in a phase are contending for a sub-master's channel and only two sub-masters are contending for the master's channel. Also, this strategy does not suffer from the non-utilization or underutilization of time slots seen in direct-TDMA and hierarchical-ALOHA+TDMA strategies.

The direct communication strategy using two channel tunable filter at the master has the best performance. This is due to three reasons. First, due to the availability of two home channels for the master the contention between the slaves is reduced just as in the hierarchical-ALOHA+ALOHA strategy. Second, the contention between the sub-masters for the master's channel, seen in the hierarchical strategies, is absent because the slaves send their results directly to the master. Third, due to direct communication of the results from the slaves to the masters the results do not have to be transmitted twice.

5.3 Effect of Varying Degree of Parallelism

The results presented above assumed that the degree of parallelism (number of slaves spawned in each phase) varies uniformly from 7 to 13, i.e., the mean number of slaves in every phase is 10. There are a sizable number of applications that exhibit a higher degree of parallelism. If the system has an adequate number of suitable computers to run these mutually independent subtasks concurrently, we expect the task completion time to be reduced. But, with increased parallelism there is a greater demand on the communication network at the end of each phase, when all the slaves try to send their results to the master.

The relationship between degree of parallelism and task completion time was evaluated for both the *subtask-code-migration* and *no-code-migration* strategies and the five communication topology-protocol combinations. The subtask data/result size was fixed ($N = 800$). Executions of a task with 1000 subtasks were simulated with each run having a different value for the mean number of slaves in a phase, varying from 5 to 40. The simulation results obtained are shown in Fig. 9.

Keeping the mean subtask data size (and consequently, the mean subtask execution time) fixed, an increase in the degree of parallelism initially leads to a decrease in the task completion time for all the five communication strategies, both for the *subtask-code-migration* and *no-code-migration* approach. Subsequent increases in parallelism lead to an increase in the completion time for all the communication strategies except the direct-TDMA protocol which continues to show a decrease in task completion time. This increase in task completion time with increasing parallelism is because of increased contention for communication channels at the end of a phase. The reduction in time due to increased concurrency is more than offset by the increase in contention.

Channel contention with increase in parallelism becomes so severe for the direct-ALOHA

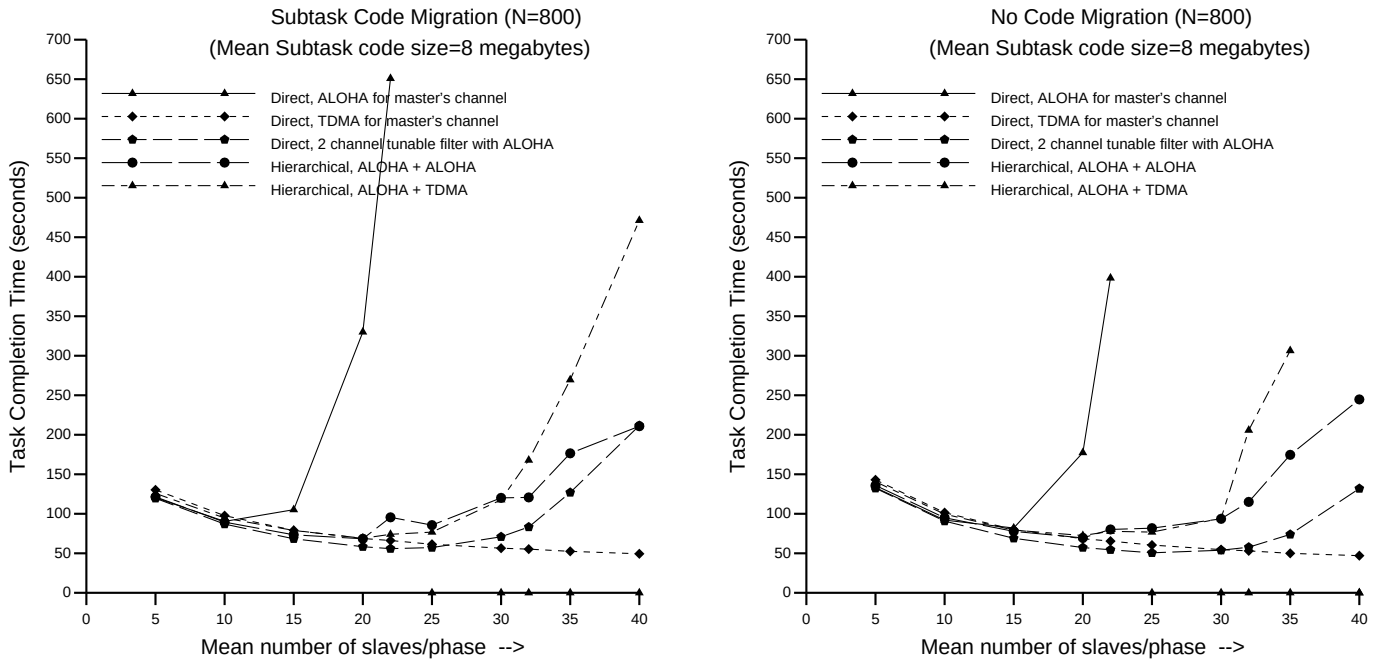

Figure 9: Effect of degree of parallelism on task completion time for the *subtask-code-migration* and *no-code-migration* approaches. Mean data size: N = 800.

strategy that the task completion time increases at a very fast rate. So, its value is not plotted beyond the 20 slaves per phase point.

The hierarchical communication strategy reduces channel contention significantly which is reflected in lower task completion times. In the hierarchical-ALOHA+TDMA strategy some slots may remain unutilized in each cycle. Also, once a slave is ready to send its results to its sub-master it may have to wait for its slot. As the degree of parallelism increases, the number of slots in a cycle increases, thus increasing the waiting time. The hierarchical-ALOHA+ALOHA strategy does not have these drawbacks. So, it performs better than the hierarchical-ALOHA+TDMA strategy.

The direct communication strategy using two channel tunable optical filter at the master has similar advantages as the hierarchical-ALOHA+ALOHA strategy in terms of reducing contention between the slaves for sub-master's channel. As slaves directly send their results to the master, duplication of result transfer (once from slaves to sub-masters and then from sub-masters to master) is avoided. Hence, using direct communication with tunable optic filters is a better strategy than the hierarchical-ALOHA+ALOHA strategy.

The direct-TDMA strategy shows the best performance as the degree of parallelism in-

creases. This is due to a total absence of channel contention between the slaves. Also, the overheads associated with the TDMA protocol decline with increasing parallelism. As already mentioned, for any simulation run the number of slots in a TDMA cycle is equal to the maximum number of slaves that can be spawned in any phase of task execution. With the parameters assumed, for a given simulation run, the difference between the maximum and minimum number of slaves spawned in different phases can be at most six. So, at most six time slots in a TDMA cycle can remain unutilized if all the slaves are trying to send their results simultaneously. As the degree of parallelism increases this number becomes a smaller fraction of the total number of time slots in a cycle. So, overall channel utilization increases with increasing parallelism. An added advantage of the direct-TDMA and the tunable filter strategies is that they do not require extra nodes to act as sub-masters.

The inflexibility of the TDMA protocol is illustrated by the fact that for low degrees of parallelism the overheads due to unutilized slots are significant and the direct-TDMA strategy has the worst performance of the five strategies. For high degrees of parallelism the overheads decline relatively. So, direct-TDMA shows the best performance. Also, all the simulations were carried out for results being 800×800 matrices. Matrices of this size require exactly four time slots to be transmitted. So, the problem of slot underutilization is non-existent.

The hierarchical-ALOHA+TDMA strategy appears to suffer from the drawbacks of both the TDMA and ALOHA protocols, namely increased cycle time and greater contention, respectively at higher degrees of parallelism. Hence, its performance is poorer than both hierarchical-ALOHA+ALOHA and direct-TDMA strategies. We also simulated a hierarchical communication protocol that used time division multiplexing at both the levels. Its performance was almost identical to that of the direct-TDMA strategy, and hence, has not been reported here. Such identical results further emphasize the effect of contention between the two sub-masters for the master's channel.

From these simulation results, it can be observed that the direct-ALOHA and the hierarchical-ALOHA+TDMA strategies demonstrate poor performance for varying degrees of parallelism and data size. Thus, these combinations are not suitable for building general-purpose heterogeneous parallel systems. So, in the remaining part of the paper, we eliminate these two strategies and restrict our study to the remaining three communication strategies.

5.4 Inflexibility of TDMA Protocol

In order to verify the fact that the performance of the direct-TDMA strategy is highly dependent on the result size, we compared this strategy with the hierarchical-ALOHA+ALOHA and the direct-with two channel tunable filter at master strategies for smaller N. These results are illustrated in Fig. 10 for $N = 500$. The hierarchical-ALOHA+ALOHA strategy has a better performance than the direct-TDMA strategy for degree of parallelism up to 32. For higher degrees of parallelism, the direct-TDMA strategy shows better performance than the hierarchical-ALOHA+ALOHA strategy. The direct communication strategy that uses two channel tunable filter at the master with ALOHA has the best performance for the entire range. The results of Fig. 10 can be explained as follows: a slave needs approximately 1.56

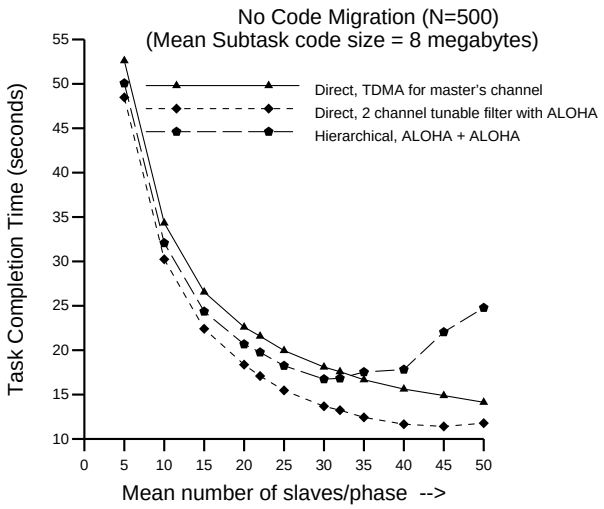

Figure 10: Performance of Hierarchical-ALOHA-ALOHA, Direct-TDMA and Direct-2 channel tunable filter with ALOHA strategies for smaller data/result sizes (N=500).

TDMA time slots to send its result (a 500×500 matrix) to the master. So, every other time slot that a slave uses to send results is only 56% utilized, thus reducing the communication efficiency. The other two strategies do not have the problem of underutilized slots. Also, the hierarchical strategy leads to a reduction in the contention between the slaves when they have to send their results. Using tunable filters has the added advantage of the total absence of contention between the sub-masters for the master's channel that is present in hierarchical strategies.

Therefore, even though the direct-TDMA strategy may have the best performance in some situations, its performance is highly dependent on the nature of the application and the

data size. The direct-TDMA strategy lacks flexibility as the slot duration and the number of slots in a TDMA cycle are fixed *a priori* at the time of protocol design. In contrast, the hierarchical-ALOHA+ALOHA and the direct-with two channel tunable filter strategies consistently perform well in all situations. Hence, the latter two strategies are suitable for general purpose heterogeneous parallel systems while the direct-TDMA strategy is not. So, the direct-TDMA strategy is eliminated from further simulation studies.

5.5 Effect of Intermediate Processing of Data

As mentioned in Section 4.4.2, for some parallel applications, the results produced by the slaves in one phase have to be processed before they can be forwarded as input to the slaves in the next phase. In the hierarchical strategies the multiple sub-masters can concurrently perform intermediate processing on the results received from their respective slaves. The size of the processed result sent to the master is usually a fraction of the sum of the sizes of the results received from the slaves. We refer to this reduction in size as *data compaction*. Thus the sum of the sizes of input data sent out by a master to the slaves in a phase is greater than the sum of the sizes of the *processed* results received by the master from the sub-masters in that phase. Such a reduction in traffic is not possible with direct communication. When the sub-masters are eliminated by using the acoustooptic filters, all the slaves send their results directly to the master and all the intermediate processing has to be done by the master. Therefore, concurrency of intermediate processing is lost.

The simulation results of the previous section showed that the hierarchical-ALOHA-+ALOHA and direct-with two channel tunable filter–ALOHA communication strategies are the most suitable communication strategies. Hence, in order to study the impact of intermediate processing of data on communication performance, we simulated the performance of these two strategies. Different compaction ratios (sum of the size of processed results sent by sub-masters to the master/sum of the size of results sent by the slaves to sub-masters), from 25% to 90%, were used and the results are shown in Fig. 11.

When direct communication with tunable optic filters is employed intermediate processing of data at the master has no impact on the volume of communication. This is because the slaves still have to send the same volume of results to the master for intermediate processing. Hence, channel contention, and consequently, the communication costs remained unchanged. The computation overheads involved in the intermediate processing were too

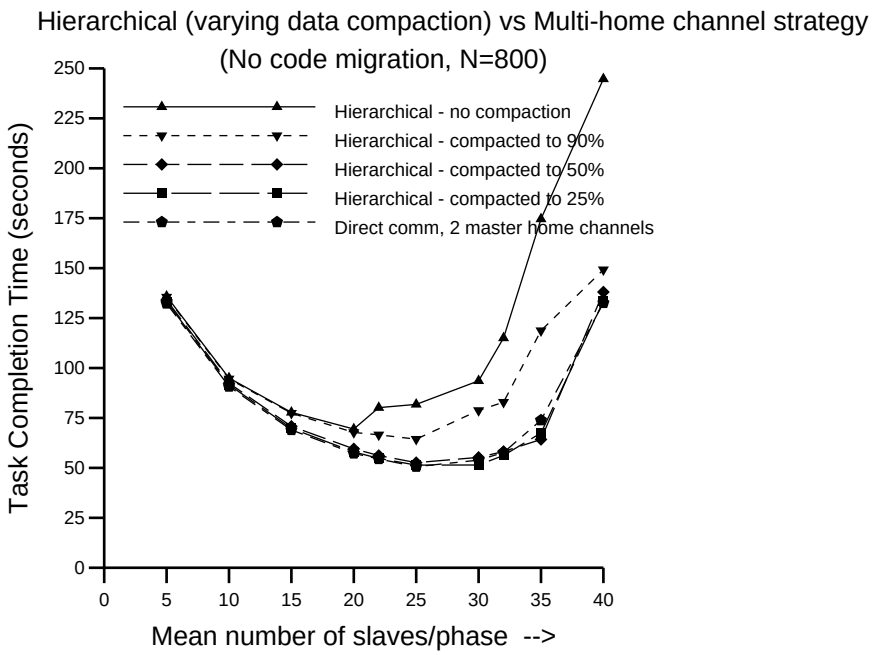

Figure 11: Comparison between the direct (with 2 channel tunable filter) and hierarchical-ALOHA+ALOHA strategies if slaves' result is compacted to different extents at sub-masters.

small a fraction of the total task completion time to have a significant impact. However, with intermediate processing and data compaction, the hierarchical-ALOHA+ALOHA strategy showed a significant reduction in task completion time over the no-compaction situation. The extra computation time needed by the sub-masters for data compaction was more than offset by the significant reduction in communication time between the sub-masters and the master. This is due to significant reduction in channel contention between the sub-masters as they are trying to send a smaller amount of data over the master's channel. When the size of the result sent by a sub-master to the master was 50% (or lower) of the sum of the result sizes received from the corresponding slaves, the hierarchical-ALOHA+ALOHA strategy performed as well as the direct communication protocol using tunable acoustooptic filters.

5.6 Comparison between subtask-code-migration and no-code-migration

Next we studied the impact of task scheduling strategies on the two topology–protocol combinations. Figure 12(a) shows that for a low degree of parallelism (an average of 10 subtasks per phase) the *subtask-code-migration* and the *no-code-migration* approaches have comparable performance for the hierarchical-ALOHA+ALOHA strategy as well as the direct–2

channel tunable filter with ALOHA strategy. The two approaches lead to comparable task completion times regardless of the subtask data/result size.

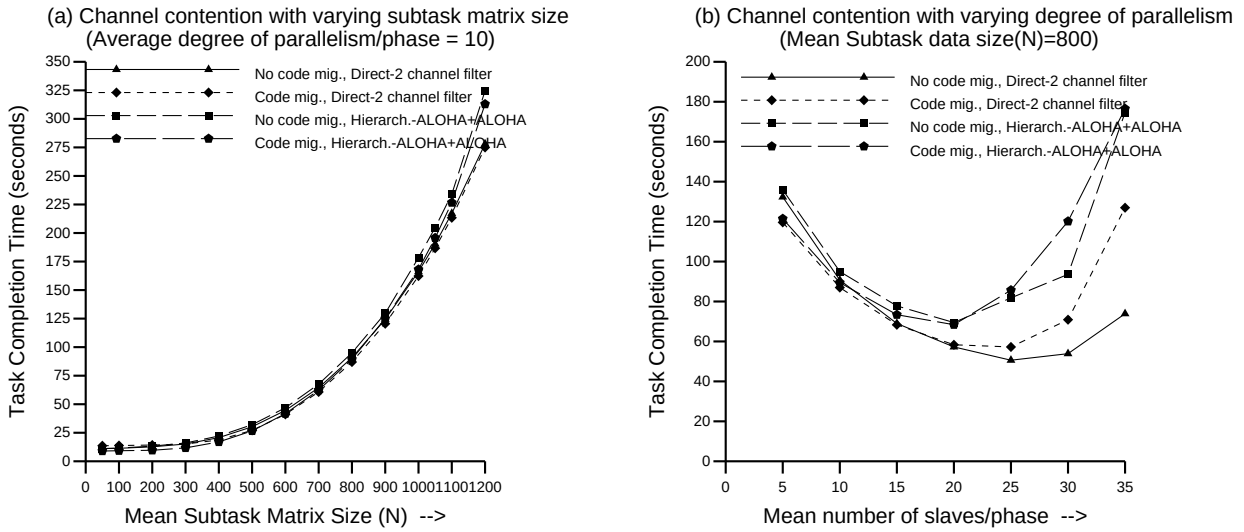

Figure 12: Comparison between the performance of the *no-code-migration* and *subtask-code-migration* approaches: (a) varying data size with fixed average degree of parallelism/phase – 10 and (b) varying degree of parallelism with fixed data size (N=800).

Figure 12(b) also shows the effect of the degree of parallelism on channel contention, and consequently the task completion time. For higher degrees of parallelism the *subtask-code-migration* approach has a higher completion time than the *no-code-migration* approach for both the hierarchical-ALOHA+ALOHA and the direct–2 channel tunable filter with ALOHA strategies. The opposite is true for lower degrees of parallelism.

The higher completion time of the *subtask-code-migration* approach for high degrees of parallelism is not due to greater contention alone. As the number of subtasks per phase increases, the start times of the slaves get more and more skewed in the *subtask-code-migration* strategy, because a slave can start only after the code *as well as the data* for itself and all the preceding slaves in its phase have been communicated by the master. The finish times of the slaves, too, are highly skewed. As a result, the finish time of each phase is increased. Hence, the task completion time increases. This increase appears to be more due to the skewed start and finish times of the subtasks than due to increase in contention. Therefore, the *subtask-code-migration* strategy is not able to fully exploit higher degree of parallelism. In the *no-code-migration* strategy, the skewing is much less for higher degrees of parallelism as only the data for the subtasks has to be migrated. Hence, contention for the master's

channel plays a greater role in determining the task completion time. For low degrees of parallelism, it appears that the slight skewing of subtask start times leads to a small skewing of the subtask finish times at the slaves, for the *subtask-code-migration* approach. This seems to result in reduced contention among the slaves and smaller task completion times for the *subtask-code-migration* approach.

The effects of the skewed starting time and channel contention are further demonstrated by the difference in the points at which the two approaches achieve their best performance for the direct–2 channel tunable filter with ALOHA strategy. The *no-code-migration* approach achieves its best performance when the degree of parallelism is close to 25. Beyond that, increased contention offsets the increase in parallelism. But, the *subtask-code-migration* approach achieves its best performance for a lower degree of parallelism (somewhere close to 20). Beyond that, both the skewing in subtask start-up times and increased contention offset the performance gain due to higher parallelism.

6 Summary of Evaluations and Design Choices

Based on the simulation results, the communication strategies suitable for different subtask data sizes and different degrees of parallelism in the applications are shown in Fig. 13. The direct-TDMA, direct-with 2 channel tunable filter for master-ALOHA and the hierarchical-ALOHA+ALOHA strategies are found to be suitable for applications with larger data/result size. However, as already shown in Fig. 10 and described in Section 5.3, the TDMA protocol lacks flexibility because the number of slots in a TDMA cycle and slot duration are fixed *a priori*. Hence, the direct-TDMA strategy is not suitable for building general-purpose parallel systems if the size of the result is variable.

The two-level hierarchical-ALOHA+ALOHA and the direct with 2 channel tunable filter for master strategies are independent of applications and result sizes. So, they have the flexibility that the direct-TDMA strategy lacks. The performance of the hierarchical-ALOHA+ALOHA strategy may degrade for high degrees of parallelism and large data sizes due to a dramatic rise in contention for the channels of the sub-masters and the master. This contention can be reduced by using multiple levels in the hierarchy with the number of children of interior nodes increasing as one goes further away from the master. However, unlike fat-trees [11], the communication bandwidth required by the channels at all the levels would

Fixed degree of parallelism (low) and varying subtask data size		
Data/Result size		
small	medium	large
all	1, 3, 4	3, 4

Fixed subtask data size and varying degree of parallelism		
Degree of parallelism		
low	medium	high
all	2, 3, 4, 5	2, 3, 4

Figure 13: Suitability of various communication topology–protocol combinations for different data sizes and degrees of parallelism. 1 = Direct-ALOHA for master's channel, 2 = Direct-TDMA for master's channel, 3 = Direct-2 channel tunable filter for master-ALOHA, 4 = Hierarchical-ALOHA+ALOHA, 5 = Hierarchical-ALOHA+TDMA.

be the same. This is because at higher levels of the hierarchy, the increase in the volume of results to be transferred would be compensated by the reduced contention as fewer nodes will probably be contending for their parent's channel. This uniformity in the bandwidth of the home channels of the master, sub-masters, and slaves makes the hierarchical interconnection scalable and attractive.

The direct-with 2 channel tunable filter for master strategy has comparable performance to the hierarchical-ALOHA+ALOHA strategy. However, the advantages of using direct communication with tunable optical filters diminishes if intermediate processing of slave results is involved. As described in Section 5.5, the master has to do all the intermediate processing unlike the hierarchical strategy where the sub-masters can do the intermediate processing concurrently.

Figure 14 summarizes the comparison between the *subtask-code-migration* and *no-code-migration* task scheduling strategies. For two topology-protocol combinations, namely hierarchical-ALOHA+ALOHA and direct-2 channel tunable filter at master, *no-code-migration* strategy has better performance for medium to high degrees of parallelism. Hence, based on the simulation results, the hierarchical-ALOHA+ALOHA strategy with *no-code-migration* appears to be most suitable for heterogeneous parallel systems capable of supporting a wide range of applications.

Appropriate task scheduling strategy for varying degree of parallelism			
	Degree of parallelism		
	low	medium	high
Hierarchical-ALOHA+ALOHA	*subtask-code-migration*	*no-code-migration*	*no-code-migration*
Direct-2 channel tunable filter	*subtask-code-migration*	*no-code-migration*	*no-code-migration*

Figure 14: Suitability of two task scheduling strategies for different degrees of parallelism and different topology-protocol combinations.

7 Conclusions

In this paper we have studied issues pertaining to computation and communication in heterogeneous parallel systems. Current high-speed networks cannot fully satisfy the high-bandwidth and low latency requirements of heterogeneous systems because they employ electronic network protocols. Developments in the area of optical networking, especially passive star-coupled optical interconnection, promise to meet the bandwidth and latency demands of heterogeneous systems. In this paper, we considered medium access strategy, contention resolution protocols and communication topology in an integrated manner. We developed five different communication topology–protocol combinations to work with passive star-coupled optical interconnection. All these topology–protocol combinations were evaluated for heterogeneous parallel systems with commonly used master-slave computation model. Two variations of task scheduling strategies, namely subtask-code-migration and no-code-migration, were also considered.

It was observed that three topology–protocol combinations are capable of significantly reducing channel contention and delivering the best performance. These combinations are hierarchical interconnection with ALOHA protocol, direct communication with TDMA protocol and direct communication with tunable optical filters at the nodes. However, the TDMA protocol was found to be suitable for only a limited set of applications. The remaining two combinations were observed to deliver comparable performance for a wide range of applications. With intermediate processing of results, the hierarchical interconnection was found to be preferable than using direct communication with tunable filters. The no-code-migration task scheduling strategy generally performed better than the subtask-code-migration strategy. The analysis with respect to these scheduling strategies also showed that increase in

capacity and I/O bandwidth of local disks and memory can have a great impact on reducing task completion time in a heterogeneous parallel system. In summary, results of our analysis indicate that the proposed hierarchical interconnection on a passive star-coupled optical network with ALOHA protocol and no-code-migration task scheduling strategy is most suitable for heterogeneous parallel systems.

Laboratory prototypes of several key components required for building heterogeneous systems with optical interconnection are currently available. Considering the projected advancements in optic-fiber communication technology and the rate at which processor speeds, and memory and disk I/O bandwidths are increasing it should be feasible to build large-scale heterogeneous parallel systems with optical interconnection in the near future. The analysis and results presented in this paper provide guidelines for building such a system to deliver high performance while minimizing communication overheads and channel contention.

References

[1] *IEEE 1596.4 IEEE Standard for High Bandwidth Memory Infrastructure based on SCI Signaling Technology (RamLink)*, December 1993.

[2] *Scalable Coherent Interface: ANSI/IEEE Standard 1596-1992*, August 1993.

[3] N. Abrahamson. Development of the ALOHANET. *IEEE Transactions on Information Theory*, IT-31:119–123, March 1985.

[4] E. A. Arnould, F. J. Bitz, E. C. Cooper, H. T. Kung, R. D. Sansom, and P. A. Steenkiste. The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers. In *Proceedings of the* 3^{rd} *International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS III)*, pages 205–216. ACM/IEEE, April 1989.

[5] K. Bogineni, K. M. Sivalingam, and P. W. Dowd. Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks. *IEEE Journal on Selected Areas in Communication*, 11(4):590–604, May 1993.

[6] M. Borella, B. Mukherjee, F. Jia, S. Ramamurthy, D. Banerjee, and J. Iness. Optical Interconnects for Multiprocessor Architectures Using Wavelength-Division Multiplexing. In *Proceedings of* 27^{th} *HICSS*, pages 499–508, January 1994.

[7] C. A. Brackett. Dense Wavelength Division Multiplexing Networks: Principles and Application. *IEEE Journal on Selected Areas in Communication*, 8(6):948–964, August 1990.

[8] S. Chen, M. M. Eshaghian, A. Khokhar, and M. E. Shaaban. A Selection Theory and Methodology for Heterogeneous Supercomputing. In *Proceedings of the* 2^{nd} *Workshop on Heterogeneous Processing*, pages 15–22. IEEE, 1993.

[9] D. M. Chiarulli, S. P. Levitan, R. P. Melhem, J. P. Teza, and G. Gravenstreter. Multiprocessor interconnection networks using partitioned optical passive star (pops) topologies and distributed control. In *Proceedings of the* 1^{st} *International Workshop on Massively Parallel Processing using Optical Interconnection*, pages 70–80, Cancun, April 1994. IEEE.

[10] R. Chipalkatti, Z. Zhang, and A. S. Acampora. Protocols for Optical Star-Coupled Network Using WDM: Performance and Complexity Study. *IEEE Journal on Selected Areas in Communication*, 11(4):579–589, May 1993.

[11] Thinking Machines Corporation. *The Connection Machine CM5 Technical Summary.* Cambridge, MA, 1992.

[12] N. R. Dono, Jr. P. E. Green, K. Liu, R. Ramaswami, and F. F.-K. Tong. A Wavelength Divison Multiple Access Network for Computer Communication. *IEEE Journal on Selected Areas in Communication*, 8(6):983–994, August 1990.

[13] R. F. Freund, S. Natarajan, and V. K. Prasanna. Experiences in Using Heterogeneous Computing for Image Understanding. In *Proceedings of the Heterogeneous Computing Workshop*, pages 53–60. IEEE, April 1995. Held in conjunction with IPPS.

[14] R. F. Freund and H. J. Siegel. Heterogeneous Processing. *IEEE Computer*, pages 13–17, June 1993.

[15] D. Gustavson and Q. Li. Local Area MultiProcessor: the Scalable Coherent Interface. In S. F. Lundstrom, editor, *Defining the Global Information Infrastructure: Infrastructure, Systems, and Services*, volume 56, pages 131–160. SPIE Press, 1994.

[16] E. Haddad. Load Distribution Optimization in Heterogeneous Multiple Processor Systems. In *Proceedings of the* 2^{nd} *Workshop on Heterogeneous Processing*, pages 42–47. IEEE, 1993.

[17] D. A. Hensgen, L. Moore, T. Kidd, R. F. Freund, E Keith, M. Kussow, J. Lima, and M Campbell. Adding Rescheduling to and Integrating Condor with SmartNet. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–11. IEEE, April 1995. Held in conjunction with IPPS.

[18] The Institute of Electrical and Electronics Engineers. *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications – American National Standard ANSI/IEEE Standard 802.3*, 1990.

[19] Paragon XP/S Product Overview. Intel Corporation, 1991.

[20] M. A. Iqbal. Partitioning Problems in Heterogeneous Computer Systems. In *Proceedings of the* 2^{nd} *Workshop on Heterogeneous Processing*, pages 23–28. IEEE, 1993.

[21] A. Khokhar, V. K. Prasanna, M. Shaaban, and C.-L. Wang. Heterogeneous Supercomputing : Problems and Issues. In *Proceedings of the Workshop on Heterogeneous Processing*, pages 3–12. IEEE, March 1992.

[22] H.T. Kung, R. Sansom, S. Schlick, P. Steenkiste, M. Arnould, F. J. Bitz, F. Christianson, E. C. Cooper, O. Menzilcioglu, D. Ombres, and B. Zill. Network-Based Multicomputers : An Emerging Parallel Architecture. In *Supercomputing*, pages 664–673, 1991.

[23] D. J. Lilja. Experiments with a Task Partitioning Model for Heterogeneous Computing. In *Proceedings of the* 2^{nd} *Workshop on Heterogeneous Processing*, pages 29–35. IEEE, 1993.

[24] R. Prakash and D. K. Panda. Architectural Issues in Designing Heterogeneous Parallel Systems with Passive Star-Coupled Optical Interconnection. In *Proceedings of the IEEE International Symposium on Parallel Architecture, Algorithms and Networks*, pages 246–253, Kanazawa, Japan, December 1994.

[25] F. E. Ross. An Overview of FDDI: The Fiber Distributed Data Interface. *IEEE Journal on Selected Areas in Communication*, 7(7):1043–1051, September 1989.

[26] H. Schwetman. *CSIM Revision 16*. Microelectronics and Computer Technology Corporation, 3500 West Balcones Center Drive, Austin TX 78759, U.S.A., June 1992.

[27] J.E. Smith and W.-C. Hsu C. Hsiung. Future General Purpose Supercomputer Architectures. In *Supercomputing*, pages 796–804. IEEE, November 1990.

[28] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1988.

[29] A. Varma, V. Sahai, and R. Bryant. Performance Evaluation of a High-Speed Switching System Based on the Fibre Channel Standard. In *Proceedings of the Second International Symposium on High Performance Distributed Computing*, pages 144–151, Spokane, Washington, July 1993. IEEE.

[30] R. J. Vetter, D. H.C. Du, and 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.

[31] M.-C. Wang, S.-D. Kim, M. A. Nichols, R. F. Freund, H. J. Siegel, and W. G. Nation. Augmenting the Optimal Selection Theory for Supercomputing. In *Proceedings of the Workshop on Heterogeneous Processing*, pages 13–21. IEEE, March 1992.