EXPERIMENTAL FEDERATED MODELING OF AN OPTICAL DATA PATH

Roger D. Chamberlain, Jason E. Fritts, Praveen Krishnamurthy, and Hui Zhang
Computer and Communications Research Center
Department of Computer Science and Engineering
Washington University, St. Louis, Missouri, USA
roger@ccrc.wustl.edu, jefritts@cse.wustl.edu, praveen@ccrc.wustl.edu, huizh@ccrc.wustl.edu

ABSTRACT
This paper explores the federated modeling of an optical data path that interconnects processors and memory in a multicomputer system. By federated modeling, we mean the use of two pre-existing simulation models of distinct subsystems to form a resulting overall model of the entire system. In this experimental federation, we investigate mechanisms that facilitate the distinct models working together while explicitly minimizing the model integration effort required. This focus yields tradeoffs in both model execution time and (potentially) the fidelity of the overall performance results. These tradeoffs are investigated quantitatively in the context of a particular system under investigation, a multicomputer optical data path.

KEY WORDS
federated simulation models, optical data path, performance evaluation, processor/memory gap

1. Introduction

There has been a significant level of interest recently in federated simulation models. When constructing hierarchical systems, it is often the case that models are developed for the subsystems independently. Even more frequent is the circumstance where multiple previously existing systems (and their models) are combined into a larger system. In each of the above cases, one would like the ability to take two or more existing simulation models and use them to build an integrated simulation model of the entire entity. Often, these existing models are written by different teams, using different simulation tools, and model their respective systems at different levels of abstraction. Yet, the productivity benefits of federating the pre-existing models, rather than developing a new set of models, are significant.

Earlier work in the area of federated simulation models includes Distributed Interactive Simulation (DIS) [1] and, more recently, the High Level Architecture (HLA) [2] standard. Both of these systems require that the constituent federates be implemented conforming to the relevant standard, at which point they can effectively work together. Designing simulation models that explicitly support federated operation requires substantial effort, however, especially when development of the original model did not consider the requirements of the relevant standard. This is almost always the case for the many models that are already in existence, having been implemented prior to their use in federated modelling.

A vision of cooperating simulation models that does not require explicit conformation to an interoperability standard is described by Wilson [3]. This work is in its early stages though, and while it shows promise for the future, it does not solve the immediate practitioners' needs. Another example of a set of federated simulations that doesn't conform to an interoperability standard is the MILAN system out of USC [4]. They developed an integrated high-level model that automatically utilizes a pair of low-level detailed simulation models for fine-grained analysis of distinct subsystems.

We have previously investigated the performance implications of using optical technology in distributed memory multicomputers [5,6] and in the processor/memory data path of uniprocessor systems [7]. These two studies, however, were carried out using two separate simulation models, written in different languages, executing on different platforms, and modeled the systems at distinct levels of abstraction. The distributed memory multicomputer model is based upon the Interconnection Network Simulator (ICNS) framework [8], is written in MODSIM III, executes on a Sun workstation running Solaris, and provides a detailed model of the optical processor-to-processor interconnection network. The uniprocessor model is based upon the IMPACT system [9], is written in C, executes on a PC Linux platform, and provides a detailed model of the processor internals.

An interesting question that came out of this pair of earlier performance studies is whether or not a shared-memory multiprocessor that exploits optics in the processor/memory data path would show performance gains over an all-electrical system. To effectively address this question requires the capabilities of both existing simulation models. We need the ability to model both the processor and the interconnect with high fidelity. This caused us to pose the question, "Can we federate these pre-existing models with minimal effort?"

This paper provides an experimental investigation of an inexpensive (in terms of integration effort) federation of two existing models. We discuss the level of effort required to form the federation, as well as the implications on execution time requirements and overall model fidelity. Section 2 describes the optical technology used and the system that is to be modeled. Section 3 describes the two simulation models that we wish to federate, as well as the federation plan. Section 4 provides results of the federated modeling, including a discussion of the tradeoffs inherent in this type of modeling exercise. Section 5 summarizes the conclusions.

2. Description of Modeled System

Our system of interest is an optically-interconnected, shared-memory multiprocessor. As described below, optical technology is advancing to the point that chip-to-chip optical links will soon be economically viable, and we are interested in investigating the performance characteristics of several candidate architectures, including a shared-memory multiprocessor.

With the ever-increasing speed of processors, the gap between processor performance and memory performance continues to widen. This performance gap includes two principal factors, latency and bandwidth. The growing processor/memory latency gap has been the primary factor of concern to researchers, and has resulted in the proposal of numerous techniques for mitigating the impact of longer latencies, including lockup-free caches, data speculation, cache-conscious load scheduling, hardware and software prefetching, data reorganization, multithreading, value prediction, and instruction reuse, among others. Many of these techniques can substantially reduce latency penalties, but most of these techniques have the consequence of increasing the processor/memory bandwidth, which is also beginning to become a serious problem. Burger et al. [10] studied the impact of memory latency and bandwidth on overall performance and concluded that if aggressive latency tolerance techniques are implemented, limited off-chip bandwidth may seriously degrade system performance, and will likely soon become a critical bottleneck in many applications.

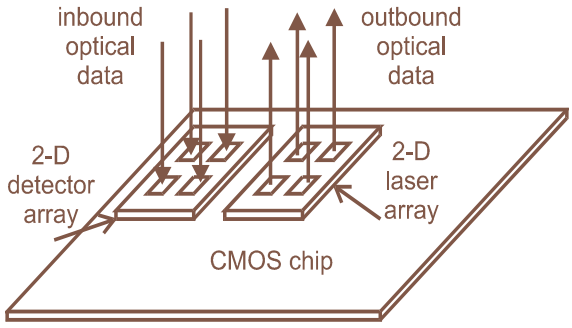

Figure 1. Optical I/O at the chip level.

One approach to dealing with the processor/memory performance gap is the use of optical technology in the data path between processor and memory. Optical technology is able to provide extremely high bandwidth communication, and recent advances in laser technology have significantly dropped the cost of constructing highly-parallel optical links. What still remains is the serious investigation of how this technology should be exploited architecturally, and this requires quantitative performance analysis.

2.1 Optical Technology

A primary enabling technology for chip-to-chip optical interconnects is the availability of 2-dimensional arrays of Vertical Cavity Surface Emitting Lasers (VCSELs) and photodetectors bonded to silicon circuitry [11]. The technique is illustrated in Figure 1, which shows a 2×2 array of VCSELs and a 2×2 array of detectors bonded to the surface of a CMOS chip. Unlike older edge-emitting lasers, the VCSELs transmit their light vertically, out of the plane of the chip. The data rate achievable in each light path is significant, and as the number of lasers and detectors grows, the result is an optoelectronic data pathway that provides orders of magnitude greater off-chip bandwidth than traditional electrical pins. With a 32×32 array operating at 4 Gb/s per laser, the aggregate data rate is 4 Tb/s (512 GB/s). Prototype chip-to-chip optical interconnects have been constructed by Plant et al. [12] with array sizes up to 32×16. Figure 2 illustrates a chip-to-chip optical interconnection in a ring topology using fiber image guides [13].

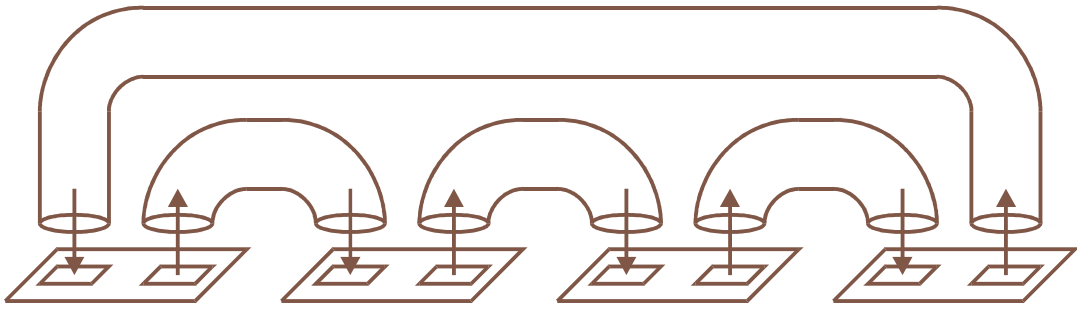

Figure 2. Chip-to-chip optical interconnection.

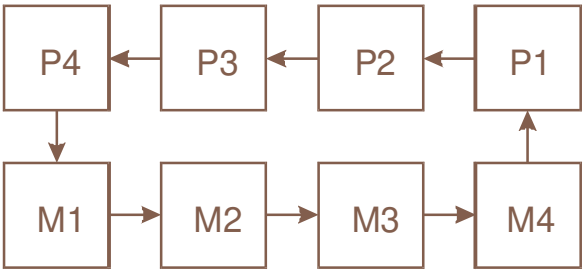

Figure 3. Eight-node optical ring.

2.2 System Architecture

Given the limited fan-in and fan-out capability of parallel optics, a ring architecture is reasonable. Here, we investigate the performance of an eight-node ring, which has 4 processing nodes (P1 through P4) and 4 memory nodes (M1 through M4), as illustrated in Figure 3.

Figure 4 shows in greater detail the design of the processor and memory nodes. Each processor node is a single-chip multiprocessor (CMP) with 4 processors, providing a total of 16 processors in this optical ring multicomputer system. The bus interface provides inter-processor arbitration to the optical processor/memory ring network and on-chip buffering of memory requests with the simplifying assumption that no processor is ever stalled due to insufficient buffering for memory requests.

Each of the 16 processors in the system is a 4-issue out-of-order superscalar processor with its own L1 and L2 cache memory system. We presume that optical processor/memory data paths will not become economically feasible for at least a few years, so our simulations model the anticipated processor technology of five years from now. The processor's frequency, capacity, and instruction latency estimations were based on the expected VLSI and processor technology trends as determined by the *2001 International Technology Roadmap for Semiconductors* [14], as well as the processor frequency versus performance studies by Agarwal et al. [15], and the optimal pipeline depth studies by Hrishikesh et al. [16]. From their analyzes, we predict that a high-performance processor designed five years from now, using 60-70 nm VLSI technology, and designed with a pipeline depth of 10 FO4 (fan-out-of-four inverter delays), will operate at 6 GHz with instruction latencies as given in Table 1.

The memory hierarchy for each of the processors comprises separate L1 instruction and data caches and a unified on-chip L2 cache. The L1 instruction cache is a 16 KB direct-mapped cache with 256-byte lines and a 20 cycle miss penalty. The L1 data cache is a 32 KB direct-mapped cache with 32-byte lines and a 15 cycle miss latency. It is non-blocking with an 8-entry miss buffer and an 8-entry write buffer. The L2 cache is a 256 KB

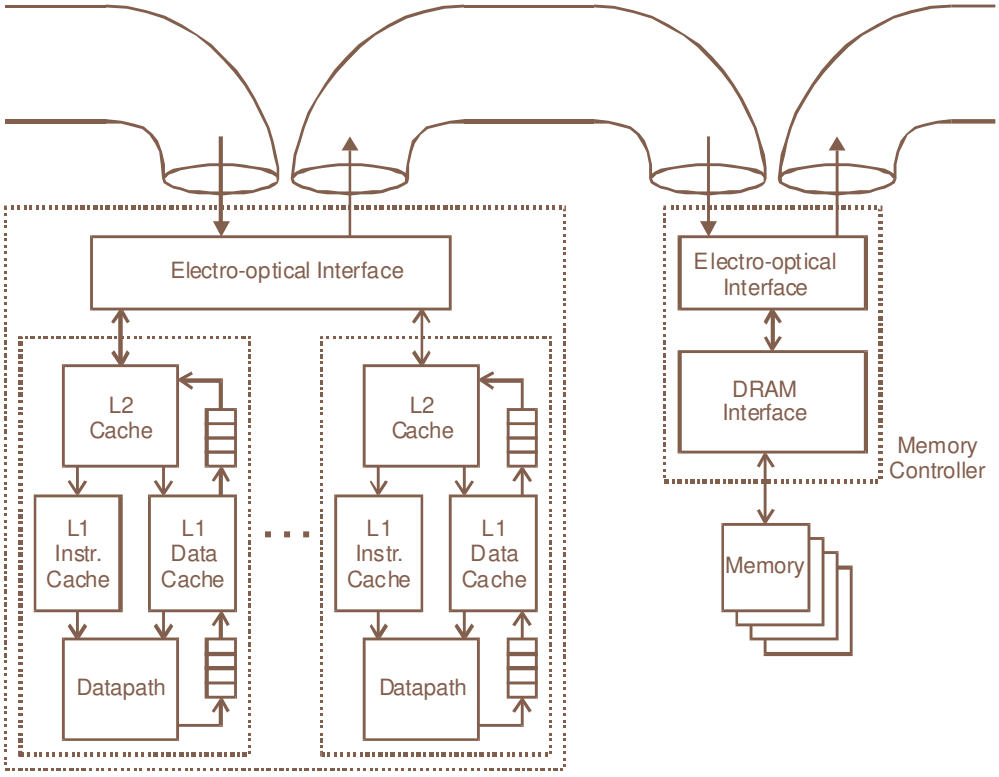

Figure 4. Multiprocessor chip and optical path to memory.

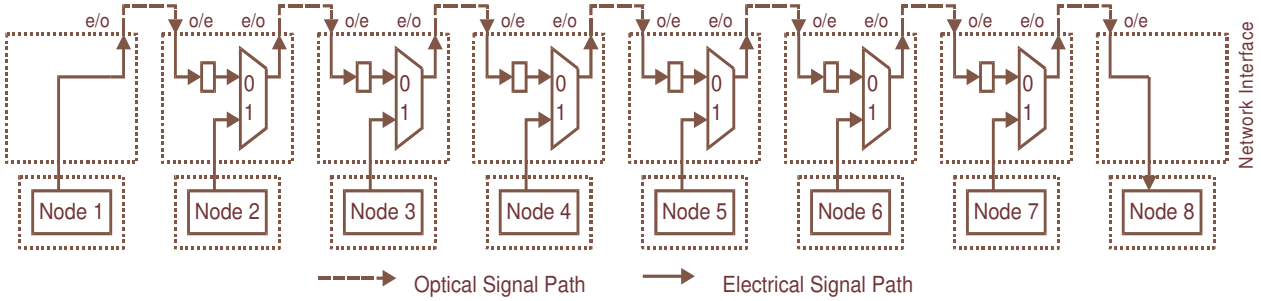

Figure 5. Subring associated with node 8, including network interface block diagram.

Instruction	Latency (clocks)
Branches	1
ALU	2
Store	3
Load	4
Multiply	10
FP Add/Mult	15
Divide	30
FP Div/Sqrt	30

Table 1. Instruction latencies for processor model.

4-way set associative cache[1] with 64-byte lines and a minimum miss latency of 288 cycles (240 cycles attributed to a 40 ns delay for DRAM access and memory controller overhead, and 8 ns for transferring the data around the 4 Gb/s optical ring network). The L2 cache is also non-blocking with an 8-entry miss buffer and an 8-entry write buffer.

The memory nodes are characterized by 16-way interleaved DRAM banks (indexed on bits 5 to 9 of the memory address). Each bank of DRAM is assumed to have a latency of 40 ns, and access requests are serviced in a first-come, first-serve fashion with the memory controller maintaining a queue for requests that have not yet been serviced. When any processor makes a memory request (e.g., triggered by a read miss in the L2 cache), the request first traverses the optical ring to the appropriate memory node (selected by bits 10 and 11 of the memory address). At the memory node, the request enters the queue for the selected DRAM bank. Upon service by the DRAM bank, the read response traverses the optical ring back to the originating processor node.

The optical interconnect is a physical ring that is organized as a multiring, with a subring associated with each destination. The logical organization of an individual subring (allocated to node 8) is illustrated in Figure 5. At each hop in the subring, the optical chip-to-chip interconnect is indicated via the e/o (electrical-to-optical) and o/e (optical-to-electrical) notations which indicate the appropriate signal conversion via lasers or detectors. The particular design features of the system under investigation are as follows. The laser and detector arrays are 32 × 32 in dimension. Each light path is assumed to operate at 4 Gb/s, yielding a total chip-to-chip bandwidth of 512 GB/s. This total bandwidth is uniformly allocated across the 8 subrings, giving a data rate of 64 GB/s to each subring. Data delivery is cell-based, with a fixed cell size of 64 bytes, implying a cell time of 1 ns. Within each subring, media access is enforced via the multiplexers shown within each network interface. Given the impracticality of providing sufficient buffer space within each node, priority is given to upstream nodes in the subring. This design decision results in an inherently unfair system, and an important performance question that needs investigation is whether or not this has negative performance implications on the system as a whole. Addressing questions such as this is the primary motivation behind the present modeling effort.

[1] L2 caches will likely be larger than this five years from now, but for the applications used in these simulations, larger L2 caches have negligible impact on execution time.

3. Federated Simulation Model

In this section, we describe the processor model, the interconnect model, and the federated model that is built from the two. Of particular interest here is the desire to minimize the integration effort (human effort), which implies tradeoffs that can impact both modeling accuracy and execution time.

3.1 Processor Model

The compilation and processor simulation tools for this architecture evaluation were provided by the IMPACT compiler, produced by Wen-mei Hwu's group at UIUC [9]. The IMPACT environment includes a trace-driven simulator and an ILP compiler. The ILP compiler supports many aggressive compiler optimizations including procedure inlining, loop unrolling, speculation, and predication. The IMPACT simulator is a parameterizable, emulation-based trace-driven simulator that enables both statistical and cycle-accurate simulation of a variety of microprocessor architecture models, including in-order superscalar, out-of-order superscalar, and VLIW data paths. The results for our experiments use aggressive global compiler optimizations, including the superblock optimization [17], and an out-of-order superscalar processor architecture.

Benchmark	Description
MPEG-4	Motion video compression codec supporting a variety of standards – used MPEG-4 standard in experiments; the MPEG-4 standard emplys an object-based representation of video; performs file I/O but no graphical display *mpeg4enc* – encoder, *mpeg4dec* – decoder
H.263	Very low bit-rate video decoder based on the H.263 standard; performs file I/O but no graphical display; provided by Telenor R&D *h263enc* – encoder, *h263dec* – decoder
JPEG	Lossy image compression decoder for color and gray-scale images, based on the JPEG standard; performs file I/O but no graphical display *jpegenc* – encoder, *jpegdec* – decoder
JPEG-2000	Image compression codec based on wavelets and multi-pass arithmetic entropy coding; performs file I/O but no graphical display *jp2kenc* – encoder, *jp2kdec* – decoder

Table 2. Descriptions of applications from MediaBench II video benchmark.

This study on federated model simulation uses memory traces generated from various video coding applications that derive from the MediaBench II benchmark suite [18]. Media benchmarks were selected for this evaluation for two reasons. First, media processing continues to dominate an increasing portion of computing workloads. Second, media applications are typically characterized by enormous amounts of streaming data. Limited bandwidth may become a significant bottleneck to such applications with the growing processor/memory gap.

Among the media benchmarks are four image and video compression methods: JPEG, JPEG-2000, MPEG-4, and H.263. Table 2 provides a short description of each of the compression methods. Since each coding method provides separate encoding and decoding procedures, a set of 16 unique processes, one for each of the 16 processors in this multicomputer system, is achieved using two separate input data sets (input-1 and input-2) for each of encoding/decoding procedures. The typical dynamic trace length for these applications and inputs is on the order of 500M instructions. However, it is generally not necessary to simulate the full trace, since the IMPACT processor simulator only needs to run long enough to produce the desired number of memory requests for simulation by the interconnection network simulator (ICNS).

3.2 Interconnect Model

The optical interconnect and memory subsystem models are built using the ICNS simulation framework [8], which has been used previously to model the optical multiring [5,6]. ICNS uses an individual cell time as the basic simulation timestep. Media access to the ring and delivery of data on the ring (including signaling overheads) are both modeled at the cell level, and given the cell size of 64 bytes we assume that a memory request containing an entire L2 cache line can fit within a single cell. The memory subsystem is modeled as a set of FIFO queues at the inputs to each memory bank, with a fixed access time of 40 ns to the memory bank itself. Each memory request progresses through the following steps in the ICNS simulator:

(1) originates at one of the 4 processor nodes,
(2) contends for access to the interconnect,
(3) is delivered to the appropriate memory node,
(4) contends for access to the memory bank,
(5) is serviced by the memory bank,
(6) contends for access to the interconnect, and
(7) is delivered to the originating processor node.

The total time required to perform the above list of actions is recorded as the memory service time for that individual memory request.

3.3 Federated Model

A primary guiding factor in the federation of the above two models is the desire to minimize the development effort required. This imposes severe limitations on the design choices that are feasible, and one of the goals of this work is to evaluate the appropriateness of even attempting such an effort. With this limitation in mind, we chose to define a collection of information that could be easily output from the processor model and used as input to drive the interconnect model, define an additional collection of information that could be easily output from the interconnect model and used as input to drive the processor model, and execute the two simulation models iteratively, passing information from one to another, until some measure of convergence is achieved. While there is no guarantee that this iterative process will converge, if convergence does occur then the result is a feasible solution to the logical fixed-point expression that is implied by the experiment.

Processor-to-Interconnect Federation via Trace – For the information flowing from the processor model to the interconnect model, we are interested in the memory requests generated by the processors during program execution. It is straightforward to generate a trace file with each record containing an individual memory request

and a timestamp indicating when (in the processor simulation) the request was generated. This trace file is then read into the interconnect simulator and drives the source model for memory traffic on the optical ring.

Interconnect-to-Processor Federation via Histogram – For the information flowing from the interconnect model to the processor model, we are interested in the service time realized by memory requests. Again, we generate a trace file with each record containing an individual memory request and the service time associated with that request. In the processor model, matching up generated memory requests with individual records from a trace file was considered impractical given the development time constraints imposed, so rather than matching each record in the trace file with its individual memory request, the trace file is used to generate a histogram of service times (modeling the empirical service time distribution). This histogram is then used during the subsequent iteration of the processor model to produce representative memory access times.

To simplify file manipulation overhead, a common trace file format was defined, which includes the information necessary for both information transfers (i.e., from processor model to interconnect model and from interconnect model to processor model). The records in the trace file contain the following fields:

<*processor ID*>, <*sequence number*>, <*memory addr*>, <*request timestamp*>, and <*service time*>

The timestamps and service times in the trace files are in the simulation time units of the processor model (the finer of the two models), and it is the responsibility of the interconnect model to perform time unit conversion.

4. Experiments and Results

In this paper, we are interested in assessing both the appropriateness and effectiveness of our federated modeling effort. We explore the first issue in section 4.1, comparing the analytic solution of a simplified processor/interconnect system versus the empirical results obtained through federated simulation of the same simplified system. In section 4.2, we examine the effectiveness of federated modeling in simulating a significantly more detailed multicomputer system.

4.1 Federated Model Validation

As an initial validation effort, we implemented a federated simulation model of a simplified system that has an analytic solution and compared the simulation model results to the analytic results. For this test, we chose to model the system illustrated in Figure 6, an open queuing network with a Poisson arrival process, FCFS queues, exponential service times, and fixed branching probabilities. The performance of this class of queuing network was analytically modeled by Jackson [19,20].

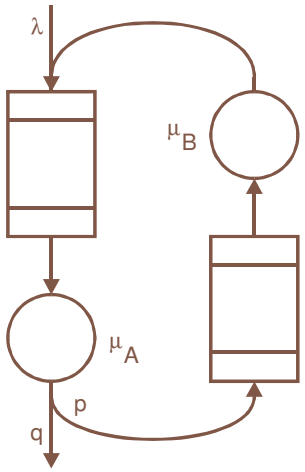

Figure 6. Simple queuing network.

Figure 7 shows the federated simulation model used to analyze the queuing network of Figure 6. The *server A* model includes the feedback path of *server B*, but does not include an accurate model of *server B*. Instead, a degenerate feedback delay (initially set to 0) is used in the initial execution of the *server A* simulation. This simulation generates a trace file of jobs destined for *server B*, which is used to drive an accurate simulation of *server B* (shown on the right side of the figure). A histogram representing transit time in the *server B* subsystem is generated from the *server B* simulation model execution. This histogram is used in the subsequent execution of the *server A* simulation to set the feedback delay of individual jobs.

There is a strong similarity between the form of this simple federated model and the form of the multicomputer federated model that is the true system of interest. *Server A* in the simple model corresponds to the processor in the primary system, and *server B* in the simple model corresponds to the interconnect and memory subsystems. Although the submodel complexities are significantly reduced, the two models are federated in a common way, with *model A* generating a trace for *model B* and *model B* generating a histogram for *model A*.

To illustrate the function of the federated model, the simulations were executed with the two service rates set to $\mu_A = 1$ and $\mu_B = 0.3$, an arrival rate of $\lambda = 2/3$, and branching probabilities of $p = 0.3$ and $q = 0.7$. The initial execution of model A used a delay of 0 in the feedback path, clearly an incorrect result. Table 3 gives both the analytical results for the mean number of jobs in the system and the mean delay in the system, as well as the simulation results after the initial execution of *model A* and after the federated execution of *model A* (which uses the histogram from *model B* in the feedback path). Clearly, the federated model results much more closely match the analytic results than the results from the initial execution, evidence that the conceptual mechanism we are using for federated modeling is a valid one.

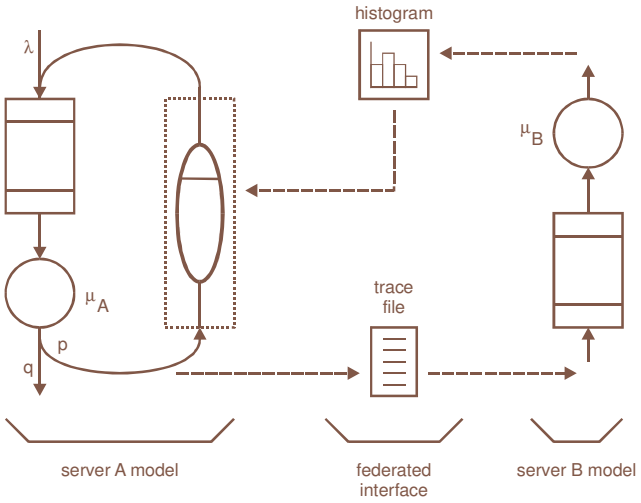

Figure 7. Federated simulation model of the simple queuing network.

4.2 Multicomputer Federated Model Assessment

Given the success of the federated simulation of the simple queuing network, we next assess the federated modeling of the target multicomputer system. To make this assessment, we performed three distinct simulation experiments. The first of these experiments mimics the operation of a write-back L2 cache by only passing the read memory accesses out of the processor simulator and into the interconnect simulator. The second and third experiments explicitly model the operation of a write-through L2 cache, including both read and write memory accesses in the trace file that communicates between federated simulations. In the first two simulation experiments, the memory request trace was limited to approximately 500,000 memory accesses, enabling us to evaluate the experiment quickly. For the third experiment, the trace length was increased by a factor of 2.

For all three experiments, 5 full iterations of the feedback loop were executed. The processor simulator is executed first, utilizing a fixed memory access time (corresponding to the minimum memory access time) to model memory service time. The memory access trace from this run is executed by the interconnect simulator, returning a histogram that represents the distribution of memory service times. This completes iteration 1. In the second and subsequent executions of the processor simulation, the time for memory service requests are drawn from the histogram that comes from the interconnect simulator. The execution of the processor simulation, followed by the execution of the interconnect simulation, then proceeds for 4 additional iterations.

Experiment 1 – The first set of results is from experiment 1. As a sanity check on the memory system design, a histogram that shows the distribution of read requests serviced by each memory bank is shown in Figure 8. With 4 memory nodes, each having 16 banks, there are a total of 64 memory banks in the system. The histogram shows that the memory requests are reasonably uniformly distributed across the available memory banks. Although only shown for experiment 1, this distribution of memory requests across memory banks is consistent across all three experiments.

The next set of graphs explores the aspect of correct federated modeling. Figure 9 shows the distribution of memory service times (this is the histogram produced by the interconnect simulator, which is subsequently input to the processor simulator) after each of the first 4 iterations. Each of these graphs is plotted using the same horizontal scale, and the extent of the tail of the distribution (i.e., the maximum histogram bin with a non-zero entry) is shown by the largest number indicated on the horizontal axis of the graph. Note that the difference between the prominent spikes in each of the graphs corresponds with the 40 ns memory access time.

Model	Mean number of jobs in system	Mean wait time in system
Initial execution	20.45	30.675
Federated execution	40.18	60.27
Analytic	40	60

Table 3. Federated model results for simple queuing network.

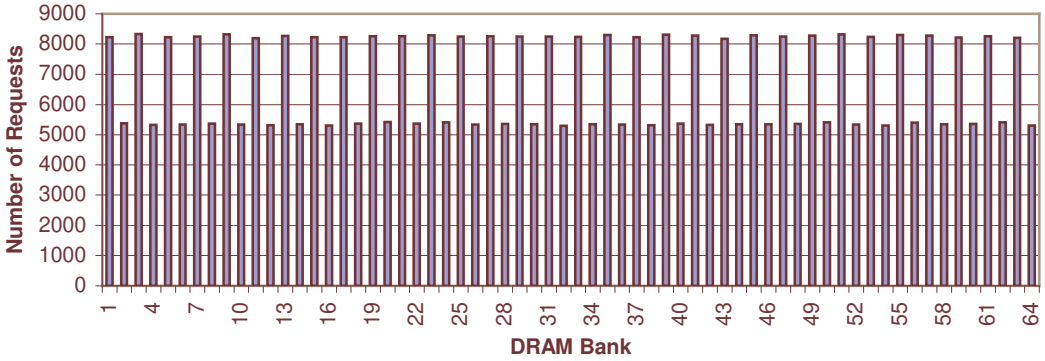

Figure 8. Distribution of requests across memory banks.

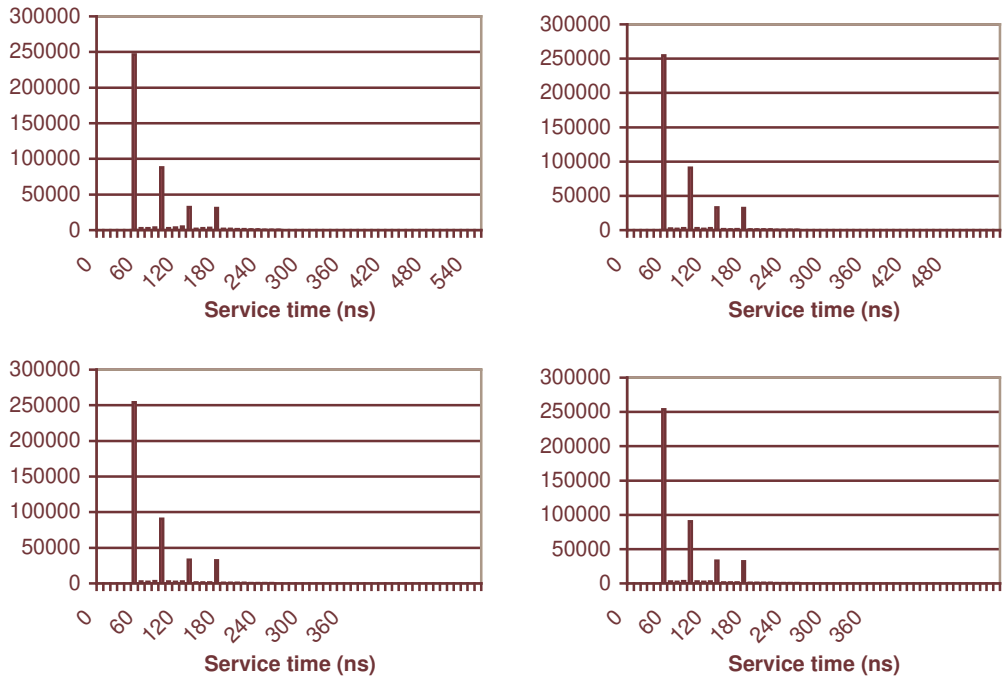

Figure 9. Histogram of service times for memory requests in experiment 1 (iterations are organized from left to right and top to bottom, so iteration 1 is the upper-left graph, and iteration 4 is the lower-right graph).

There is a clear convergence of the results between iterations 1 and 4. The most noticeable difference occurs between iterations 1 and 2, especially in the number of requests that were serviced at 50 ns, which increased from approximately 245,000 in iteration 1 to nearly 255,000 in iteration 2. It is also apparent that the histograms of service times for iterations 2, 3, and 4 are quite similar. So while there was some initial variability, this experiment still managed to converge very quickly towards a particular histogram for memory service times, further validating this federated modeling approach.

Experiment 2 – Experiment 2 examined the impact of including memory writes in the memory traces. Using a write-through policy for the L1 and L2 cache memories in each of the processors, all memory writes are immediately sent to main memory. The write-through policy is seldom used in on-chip L2 caches simply because of the enormous effect it has on the bandwidth of the processor/memory bus. However, with the optical bus, the combined bandwidth from reads and writes still consumes less only a small fraction of the network bandwidth on average. In this case, it is the bandwidth of main memory that is the major bottleneck in the system.

Even with the more intensive memory traces that include memory writes as well as reads, experiment 2 once again demonstrates a quick convergence towards a particular histogram for memory service times. Between iterations 1 and 2, there is an obvious increase in the number of memory requests with service times of 50 ns and 90 ns (the two large spikes in graphs in Figure 10), but the differences between iterations 2 and 3 are significantly smaller, and then the differences become negligible between iterations 3 and 4. Consequently, even as the memory trace complexity grows, this federated simulation methodology still manages to converge within a small number of iterations.

Experiment 3 – For experiment 3, we decided to significantly increase the load on the interconnect. To this end, the benchmark set was altered to emphasize the more memory-intensive applications, and the trace length (number of memory accesses simulated) was doubled. To diminish the impact of the memory itself, the interleaving was increased from 16-way to 32-way at each memory node. The histograms of service times for memory requests are shown in Figure 11. Note, for the histogram plot from iteration 1, the entire tail is not shown. The actual tail of the histogram extends out to 30,000 ns.

In experiment 3, the service time distributions clearly oscillate to a much larger degree than was the case with the earlier two experiments. Further evidence of this is shown in Figure 12, where the number of processor simulation cycles required to generate the appropriate number of memory requests oscillates significantly between iterations 1-3. While iteration 1 completes in 3.5 million cycles, iteration 2 jumps to 37 million cycles (more than an order of magnitude increase), and then drops back down to 5 million cycles in iteration 3. Iterations 3-5 oscillate much less, eventually converging on the resulting time of 5.7 million processor cycles (a processor cycle is 167 ps in a 6 GHz processor).

Figure 10. Histogram of service times for memory requests in experiment 2 (iterations are organized from left to right and top to bottom, so iteration 1 is the upper-left graph, and iteration 4 is the lower-right graph).

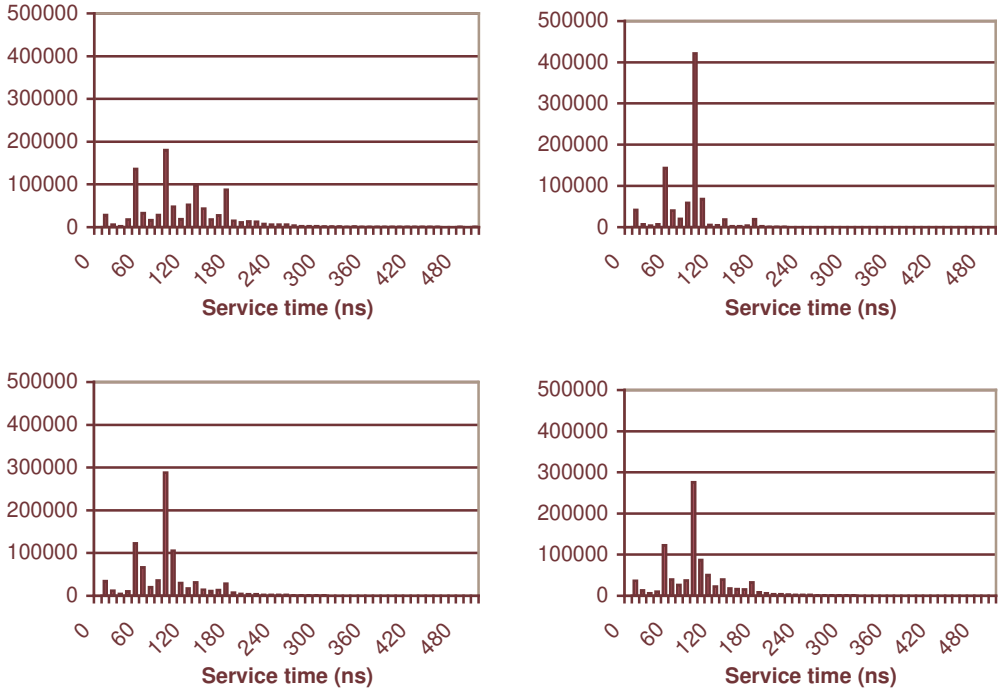

Figure 11. Histogram of service times for memory requests in experiment 3 (iterations are organized from left to right and top to bottom, so iteration 1 is the upper-left graph, and iteration 4 is the bottom-right graph).

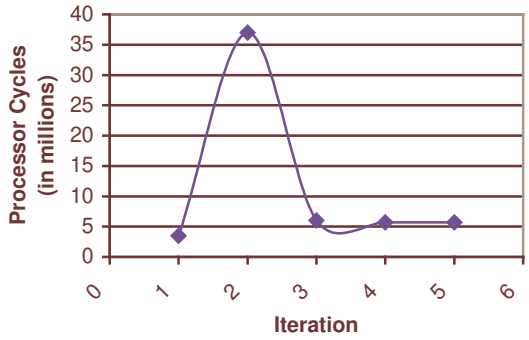

Figure 12. Variation in number of processor cycles (in millions) across runs in the federated simulation.

Detailed investigation of the interconnect simulator provides the explanation for these oscillations. In the processor simulation iteration 1, a minimum memory service time is used. This implies that the arrival rate for memory requests to the interconnect simulation for its execution of iteration 1 will be higher than the converged condition. The higher workload for the interconnect results in the unfairness described in section 2.2 above, in which individual sources can be starved for service (see [5] for a complete analysis of this phenomenon), giving a long tail on the service time distribution and necessitating a long execution time for the processor simulation in iteration 2. This effect swings the other direction in iteration 2 of the interconnect simulator, as the arrival rate is decreased (relative to iteration 1). We conclude from this effect that:

1. The fact that the federated model oscillates at all is evidence of correct modeling. Rather than simply reinforcing incorrect assumptions made by the other simulator, each federate is reacting correctly to the circumstances it perceived during each iteration.
2. The convergence of these oscillations implies that the results represent a valid set of performance results.

5. Conclusions

This paper has presented an experimental federated model that combines two pre-existing heterogeneous simulators that each model subsystems of an optically-interconnected shared memory multicomputer. The purpose was to investigate whether it is reasonable to build this type of federation with a minimum of human effort, and what tradeoffs are inherent in the endeavor. The modeling effort was quite successful, with a total development time consisting of two individuals, working part-time over a period of less than two weeks. This should be compared with our current estimates of the development time necessary to develop a fully integrated model, clearly at least many man-months of effort, particularly considering the great disparity in the simulators' languages.

The primary tradeoff present in this type of federated model is the need to execute the constituent component models iteratively, resulting in greater execution times than would be necessary if a fully integrated model were available. For the cases reported here, convergence was achieved consistently within 5 iterations (or less), implying that although the overall execution time is longer, the total calendar time dedicated to answering the desired performance questions was significantly diminished (i.e., we saved much more time in the development cycle than we expended in increased execution time of the simulation itself).

The performance conclusions for the system under investigation are positive as well. When modeling a candidate architecture that is quite unrealistic to implement with electrical interconnect technology, the optical interconnect used to support processor/memory data transfers is relatively lightly loaded. For experiment 3, which used a write-through policy in L1 and L2 cache, the memory traffic for the 16 processors resulted in a mean utilization of 12% for the optical ring, implying the *delivered* bandwidth to the set of applications was approximately 60 GB/s. This is eight times the peak capacity of an electrical bus supporting 128-bit data transfers running at 500 MHz. Consequently, much larger shared memory multicomputer systems could be feasibly constructed with the benefit of optical chip-to-chip communications, especially if a fairness mechanism is included in the interconnect. In the future, we plan to use this federated model to assess a whole host of architectural design issues that represent opportunities made available by the existence of chip-to-chip optics.

[1] IEEE Std. 1278.1-1995, *IEEE Standard for Distributed Interactive Simulation – Application Protocols* (New York, NY: Institute of Electrical and Electronics Engineers, Inc. 1995)

[2] Defense Modeling and Simulation Office, http://hla.dmso.mil

[3] L.F. Wilson, D.J. Burroughs, A. Kumar, and J. Sucharitaves, "A framework for linking distributed simulations using software agents," *Proceedings of the IEEE*, *89*(2), 2001, 186-200.

[4] A. Bakshi, V.K. Prasanna, A. Ledeczi, A. Agrawal, J. Davis, B. Eames, S. Mohanty, V. Mathur, S. Neema, G. Nordstrom, C. Raghavendra, and M. Singh, "MILAN: A model based integrated simulation framework for design of embedded systems," in *Proceedings of ACM SIGPLAN 2001 Workshop on Languages, Compilers, and Tools for Embedded Systems*, June 2001.

[5] R. Chamberlain, M.A. Franklin, and A. Mahajan, "VLSI photonic ring interconnect for embedded multicomputers: Architecture and performance," in *Proceedings of the 14th Conference on Parallel and Distribted Computing Systems*, August 2001, 351-358.

[6] Roger Chamberlain, Mark Franklin, and Praveen Krishnamurthy, "Optical Network Reconfiguration for Signal Processing Applications," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, July 2002, 344-355.

[7] Jason Fritts and Roger Chamberlain, "Breaking the Memory Bottleneck with an Optical Data Path," in *Proceedings of the 35th Annual Simulation Symposium,* San Diego, CA, April 2002, 352-362.

[8] R. Chamberlain, C. Shi Baw, M. Franklin, C. Hackmann, P. Krishnamurthy, A. Mahajan, and M. Wrighton, "Evaluating the Performance of Photonic Interconnection Networks," in *Proceedings of the 35th Annual Simulation Symposium*, April 2002, 209-218.

[9] P.P. Chang, S.A. Mahlke, W.Y. Chen, N.J. Warter, and W.W. Hwu, "IMPACT: an architectural framework for multiple-instruction-issue processors," *Proceedings of the 18th Annual International Symposium on Computer Architecture*, May 1991.

[10] D. Burger, J.R. Goodman, and A. Kagi, "Limited bandwidth to affect processor design," *IEEE Micro*, *17*(6), 1997, 55-62.

[11] Y. Li, E. Towe, and M. Haney, eds., Special issue on optical interconnections for digital systems, *Proceedings of the IEEE, 88*(6), 2000.

[12] D.V. Plant, M.B. Venditti, E. Laprise, J. Faucher, K. Razavi, M. Chateauneuf, A.G. Kirk, and J.S. Ahearn, "256-channel bidirectional optical interconnect using VCSELs and photodiodes on CMOS," *IEEE Journal of Lightwave Technology, 19*(8), 2001, 1093-1103.

[13] T. Maj, A.G. Kirk, D.V. Plant, J.F. Ahadian, C.G. Fonstad, K.L. Lear, K. Tatah, M.W. Robinson, and J.A. Trezza. "Interconnection of a two-dimensional array of Vertical Cavity Surface Emitting Lasers (VCSELs) to a receiver array by means of a fiber image guide," *Applied Optics*, *39*(5), 2000, 685-689.

[14] *"The International Technology Roadmap for Semiconductors", 2001 Edition* (Semiconductor Industry Association, 2001)

[15] V. Agarwal, M.S. Hrishikesh, S.W. Keckler, and D. Burger, "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures," *Proceedings of the 27th Annual International Symposium on Computer Architecture*, June 2000.

[16] M.S. Hrishikesh, Norman P. Jouppi, Keith I. Farkas, Doug Burger, Stephen W. Keckler, and Premkishore Shivakumar, "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," *Proceedings of the 29th Annual International Symposium on Computer Architecture*, June 2002.

[17] W.W. Hwu, S.A. Mahlke, W.Y. Chen, P.P. Chang, N.J. Warter, R. A. Bringmann, R.G. Ouellette, R.E. Hank, T. Kiyohara, G.E. Haab, J.G. Holm, and D.M. Lavery, "The Superblock: An Effective Technique for VLIW and Superscalar Compilation," *Journal of Supercomputing*, 1993, 229-248.

[18] J. Fritts and B. Mangione-Smith, "MediaBench II - Technology, Status, and Cooperation", presentation at *4th Workshop on Media and Stream Processors (held in conjunction with MICRO-35)*, Istanbul, Turkey, November 2002.

[19] J. Jackson, "Networks of Waiting Lines," *Operations Research*, *5*(4), 1957, 518-521.

[20] J. Jackson, "Jobshop-Like Queuing Systems," *Management Science*, *10*(1), 1963, 131-142.