

Traffic Pattern Analysis for Reconfigurable Interconnects in Shared-Memory Systems

W. Heirman, J. Dambre, C. Debaes*, J. Van Campenhout, H. Thienpont*

Universiteit Gent, ELIS
Sint-Pietersnieuwstraat 41
9000 Gent, Belgium

{wheirman, jdambre, jvc}@elis.ugent.be

* Vrije Universiteit Brussel, TONA
Pleinlaan 2
1050 Brussel, Belgium

{christof.debaes, hthienpo}@vub.ac.be

Abstract— New advances in reconfigurable optical interconnect technologies will allow the fabrication of cheap, fast and run-time adaptable networks for connecting processors and memory modules in large shared-memory multiprocessor machines. Since the switching times of these components are typically high compared to the memory access time, reconfiguration can only take place on a time scale significantly above individual memory accesses. In this paper, we present preliminary results of our investigation into the exploitability of the space and time locality of address streams by a reconfigurable network.

Keywords— reconfiguration, interconnection network, shared-memory, access patterns

I. INTRODUCTION

The electrical interconnection networks, connecting the different processors and memory modules in a modern large-scale multiprocessor machine, are running into several physical limitations [9]. In shared-memory machines, where the network is part of the memory hierarchy [6], the ability to overlap memory access times with useful computation is severely limited by inter-instruction dependencies. Hence, a network with high latencies causes a significant performance bottleneck.

It has been shown that optical interconnection technologies can alleviate this bottleneck [4]. Mostly unhindered by crosstalk, attenuation and capacitive effects, these technologies will soon provide a cheaper, faster and smaller alternative to electrical interconnections, on distances from a few centimeters upward. Massively parallel interchip optical interconnects [1], [3] are already making the transition from lab-settings to commercial products.

Optical signals may provide another advantage: the optical pathway can be influenced by components like steerable mirrors, liquid crystals or diffractive elements. In combination with tunable lasers and photodetectors these components will enable a runtime reconfigurable interconnection network [5], [2] that

supports a much higher bandwidth than that allowed by electrical reconfiguration technology. From a viewpoint higher in the system hierarchy, this would allow us to redistribute bandwidth or alter the network topology such that node-pairs with high communication between them have a high-bandwidth, low-latency connection.

However, the switching time for these components is such that reconfiguration will necessarily have to take place on a time scale that is significantly above that of individual memory accesses. Therefore we need to have an idea of the locality in both time and space of the traffic flowing over the network. To this end, a simulation platform has been set up, based on the Simics multiprocessor simulator [7], that will allow us to study the network traffic in great detail. The execution of a collection of benchmarks on a shared-memory multiprocessor was simulated, and the resulting network traffic stream was analyzed for temporal locality. This way, we have identified the time scale at which reconfiguration should take place, in order to sufficiently follow the changing requirements made by the application.

In the remainder of this paper the different steps that were taken to obtain and analyze the data are presented. Section II details the architecture of both the shared-memory machine and the reconfigurable network that were chosen for this study. In section III the methodology that was followed to obtain the communication patterns is described. The analysis of temporal locality is performed in section IV. Section V will discuss some future work, the conclusions are summarized in section VI.

II. SYSTEM ARCHITECTURE

A. Multiprocessor architecture

Multiprocessor machines come in two basic flavors: those that have a strong coupling between the different processors and those with a more loose coupling.

Both types consist of a number of nodes, each containing a processor, some memory and a network interface, and a network connecting the different nodes to each other. In the loosely coupled type, commonly referred to as the *Beowulf cluster* class [11], the network consists of a commodity technology such as Ethernet. This simplistic interconnection network results in relatively low throughput (1 Gbps per processor) and high latency (several ms). These machines are necessarily programmed using the message passing paradigm, and place a high burden on the programmer to efficiently schedule computation and communication.

Strongly coupled machines usually have proprietary interconnection technologies, resulting in much higher throughput (tens of Gbps per processor) and very low latency (down to a few hundred nanoseconds). This makes them suitable for solving problems that can only be parallelized into tightly coupled subproblems (i.e., that communicate often). It also allows them to implement a hardware-based shared-memory model, in which communication is initiated when a processor tries to access a word in memory that is not on the local node, *without programmers interference*. This makes shared-memory based machines relatively easy to program, but also makes them vulnerable to increased network latencies. Modern examples of this class of machines range from small, 2- or 4-way SMP server machines, over mainframes with tens of processors (Sun Fire, IBM iSeries), up to supercomputers with hundreds of processors (SGI Altix, Cray X1). Since the capabilities of electrical networks are evolving much slower than processor frequencies, the larger types of these machines make very likely candidates for the application of reconfigurable optical interconnection networks.

B. Network architecture

Previous studies concerning reconfigurable networks have mainly dealt with fixed topologies (usually a mesh or a hypercube) that allowed swapping of node pairs, incrementally evolving the network to a state in which processors that often communicate are in neighboring positions [10], [12]. However, algorithms to determine the placement of processors turned out to converge slowly or not at all when the characteristics of the network traffic change rapidly. Moreover, an efficient way to implement such a network is not available with the current generation of optical components.

Therefore, we assume a different network architecture in this study. We start from a base network with

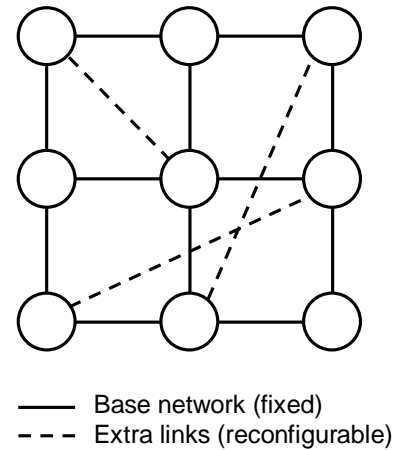


Fig. 1. Reconfigurable network topology.

fixed topology. In addition, we provide a second network that can realize a limited number of connections between arbitrary node pairs. In reality, some limitations will of course apply. Since the number of neighbors for each node is limited (due to maximum pin-counts per chip or per backplane connector, or the rising complexity of a router with a large number of in- and outputs) only a few extra links can at the same time connect to one node, and probably not *every* arbitrary node pair can be connected using only one link. However, for the current study we do not take these limitations into account. An advantage of this setup, compared to other topologies that allow for more general reconfiguration, is that the base network is always available, which is most important during periods where the extra network is undergoing reconfiguration and may not be usable. Routing and reconfiguration decisions are also simplified because it is not possible to completely disconnect a node from the others – the connection through the base network will always be available.

To make optimal use of the extra connections, they should speed up memory accesses that are in the critical path of the application. Since it is very hard, if not impossible, to determine which accesses are in the critical path of any given application, we will place the extra links between the node pairs where communication is highest. This way, congestion – and the resulting latency – can be avoided, and a large fraction of the traffic, hopefully including most of the critical accesses, can be given a single-hop pathway, minimizing routing and arbitration delays and resulting in the lowest possible latency. The remaining traffic will use the base network, possibly being routed over several intermediate nodes, and hence will experience a higher

latency. Since the network traffic changes over time, the node pairs with the most intense communication will change and thus we will need to revise the position of the extra links over time. Because reconfiguration is not immediate, the interval between decision times will be one of the most important parameters. This interval should be long enough to amortize on the cost of reconfiguration, during which the extra links are unusable, but it must not be too long otherwise the changing demands made by the application can not be followed fast enough. As a first step in analyzing this trade-off, we have chosen to measure the length of communication bursts between node pairs. This will be done in the following sections.

III. METHODOLOGY

We have based our simulation platform on the commercially available Simics simulator [7]. It was configured to simulate a multiprocessor machine based on the Sun Fire 6800 Server, with 16 UltraSPARC III processors at 1 GHz running the Solaris 9 operating system. Stall times for caches and main memory are set to the values of the real machine. The interconnection network is a custom extension to Simics, and models a 4x4 torus network with contention and cut-through routing. Both coherence traffic (resulting from a directory based MSI-protocol) and data are sent over the network, and result in realistic memory access times. Source, destination and size of each network packet are saved to a log file for later analysis.

Since the simulated caches are not infinitely large, the network traffic will be the result of both coherence misses and cold/capacity/conflict misses. This way, the network traffic is not underestimated as is done in other studies that do not include references to private data in the simulation. To make sure that private data transfer does not become excessive, a first-touch memory allocation was used that places data pages of 8 KiB on the node of the processor that first references them.

The SPLASH-2 benchmark suite [13] was chosen as the workload. It consists of a number of scientific and technical applications and is a good representation of the real-world workload of large shared-memory machines. Because the default benchmark sizes are too big to simulate their execution in a reasonable time, smaller problem sizes were used. Since this influences the working set, and thus the cache hit rate, the level 2 cache was resized from an actual 8 MiB to 512 KiB, resulting in a realistic 80 % hit rate. Line size, associativity and hit penalty for both cache levels were kept

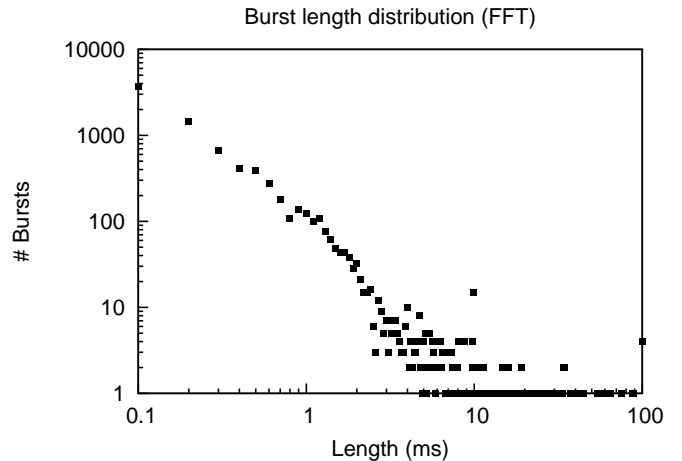


Fig. 2. Burst length distribution for the FFT application.

at the values they have in the real machine.

The simulation slowdown (simulated time versus simulation time) was a factor of 50,000 resulting in execution times of roughly 2 hours per benchmark on a Pentium 4 running at 2.6 GHz with 2 GiB RAM.

IV. TEMPORAL ANALYSIS

A. Traffic burst length distribution

In section II-B it was assumed that the network can make n connections between arbitrary node pairs, and that we would choose those node pairs that communicate the most. To see if this technique allows us to capture a sizable fraction of all network traffic – for a given switching time – we will look at the duration of *traffic bursts* (periods of elevated communication) between node pairs.

We start by dividing time in intervals of length Δt . For the interval starting at time t , the traffic that flows between nodes i and j is accumulated into $T_{i,j}(t)$. Only traffic that has the processor at node i as source and the processor at node j as destination (or vice versa) is considered, not traffic that passed through both nodes en route to somewhere else. This makes the analysis independent of the current topology. Furthermore we consider all links to be bidirectional and symmetric¹, so i is always smaller than j .

¹An optical interconnection (light source \rightarrow waveguide \rightarrow detector) is unidirectional, so a *link* consists of two such assemblies. In theory it is not necessary for the extra links to be bidirectional, however since the implementation of a shared-memory model uses a request-response protocol it is not considered very useful to speed up the request but not the response or vice versa – especially if the links are highly parallel so the possibly larger size of the response does not have a big influence on latency.

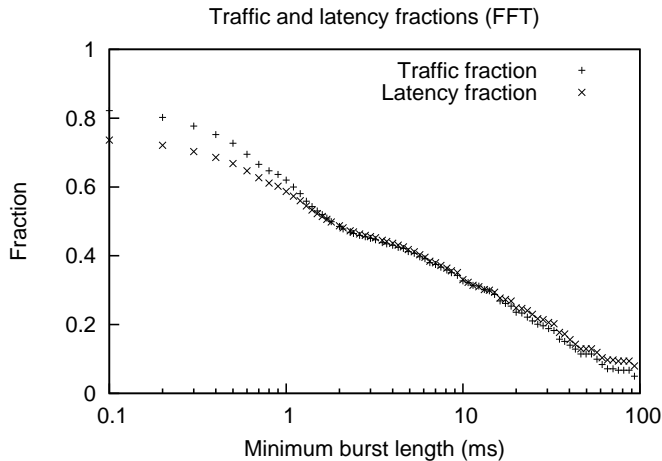


Fig. 3. Traffic size and latency fractions per minimum burst length for FFT.

At the end of each interval, all node pairs are sorted according to the traffic that has flown between them. The top n of those pairs are marked for this time interval. Next we count the number of consecutive intervals a certain node pair was marked. Multiplied by Δt this gives us the length of a burst of communication that took place between that node pair. This burst will be represented by the 4-tuple (i, j, t, l) , indicating the nodes that were involved (i and j), the starting time t and the length l . The distribution of the burst lengths (over all node pairs at any starting time) for one of the benchmark applications (the FFT kernel) is shown in figure 2. This distribution closely resembles a power law for short time scales, augmented with some very long bursts that sometimes span the length of the entire program. This should not come as a surprise, since the power-law (also described as fractal or self-similar) behavior of memory access patterns has been shown before [8]. The long bursts are due to references to global data such as configuration parameters and synchronization variables that are needed throughout the program.

B. Size and latency fractions

In the next step, we determine how much traffic is contained in bursts of a given minimal length. First, we sum the total amount of traffic in each of the bursts (i, j, t, l) such that $T(i, j, t_0, l) = \sum_{t=t_0}^{t_0+l} T_{i,j}(t)$. Next we calculate the total traffic size per burst length: $T(l) = \sum_{i,j,t} T(i, j, t, l)$. Finally we compute the inverse cumulative distribution as $T_C(l_0) = \sum_{l=l_0}^{\infty} T(l)$. With T the total amount of traffic that was sent across the network during the benchmark execution, we now

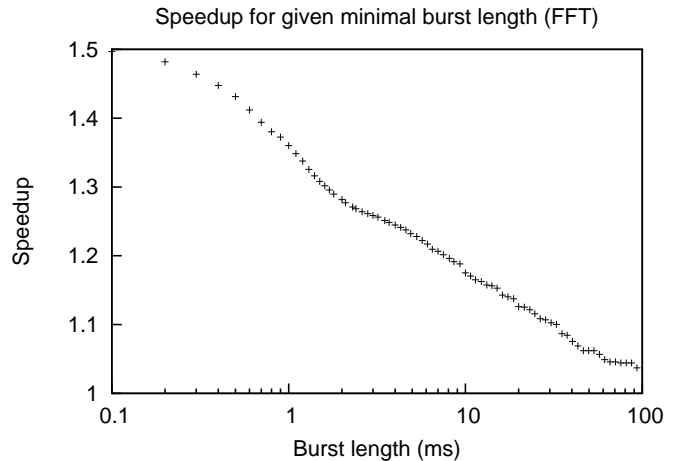


Fig. 4. Maximum achievable speedup for FFT with different minimal burst length.

know that a fraction $T_C(l)/T$ of all traffic is part of a burst of length l or longer. This distribution is given in figure 3 for the FFT application. The intervals Δt are $100 \mu\text{s}$ long, there are 16 processors and 16 extra links at any time.

To translate this into an application speedup, we look at the memory access latencies involved. Instead of summing the number of bytes in a packet, we can sum its latency: the time between transmission at node i (or j) and reception at node j (or i), including serialization, propagation, routing and congestion delays. This yields $L_{i,j}(t)$. Using the same derivation as for the packet size and L for the total memory access latency, we find the fraction $L_C(l)/L$ which is also plotted in figure 3.

From this analysis we can make the following conclusion: “Assuming we have a reconfiguration method that can detect all bursts of length l and longer, and a technology that allows the network to be reconfigured sufficiently fast to provide a fast path for a significant portion of the traffic in these bursts, at most a fraction $L_C(l)/L$ of total memory access latency can be reduced”. For the FFT application and a minimum burst length of 1 ms this would result in a 59 % fraction, which is about 35 % of the total execution time. Assuming a fourfold speedup for the affected traffic, this would allow for a 1.36 times speedup of the execution of FFT. The same calculation has been done for all SPLASH-2 programs, the results can be found in table I. The estimated upper bound on achievable speedup ranges from 1.02 to 1.86, depending on the application.

Notice however that, unless perfect prediction of

Code	Problem size	Speedup
Barnes	8K particles	1.15
Cholesky	tk15.O	1.19
FFT	256K points	1.36
FMM	8K particles	1.30
LU	512 × 512 matrix	1.72
Ocean	258 × 258 ocean	1.33
Radiosity	test	1.86
Radix	1M integers, 1024 radix	1.16
Raytrace	teapot	1.12
Volrend	head	1.08
Water-Nsq	512 molecules	1.02
Water-Sp	512 molecules	1.06

TABLE I

APPLICATIONS, PROBLEM SIZES AND ESTIMATED UPPER BOUNDARIES ON ACHIEVABLE SPEEDUPS.

burst location and starting times is possible, the switching time needs to be significantly faster than l . For instance, if burst detection takes 100 μ s and re-configuration takes the next 100 μ s, we would be able to affect only 80 % of the traffic in a 1 ms burst, reducing the actual speedup. Mistakes in burst prediction will cause some of the extra links to be placed on locations where they provide little gain, again resulting in an actual speedup that is less than the projected upper bound calculated above. Moreover, a 100 μ s switching time is already stretching the possibilities of the projected optical technologies. For slower technologies, figure 4 shows the achievable speedup for minimal burst lengths other than 1 ms.

On the other hand, moving traffic away from a contended link in the base network will not only speed up the traffic that was moved, but also the remaining traffic on the now uncontended link. This effect has not been taken into account in the previous discussion, and might provide for a speedup higher than the reported estimates.

V. FUTURE WORK

The current work is based on the assumption that the extra links can only be used by their direct endpoints, not by traffic that could use the extra link as only a part of its path. Allowing this is of course possible, but requires that the topology and node placement of the base network are known in advance, whereas the current discussion is independent of this.

Secondly the variability of the estimates given should be checked under influence of a range of pa-

rameters, such as network topology, bandwidth and latency, cache size and processor behavior. Modifying these parameters can for instance cause the presence or absence of congestion, which has a very large effect on latency. An aggressive out-of-order processor can overlap some of the memory access times, affecting the possible speedup.

Also a wide variety of algorithms deciding where to place the extra links can be considered. Optimizations based on the expected speedup from an extra link compared to the base network could improve efficiency by not placing them where their effect is minimal. For instance, if the endpoints of a large burst are already neighbors in the base network and they do not suffer from contention, no extra link should be placed between them. More information about the past could also help when bursts between a certain pair are interspaced with short periods of lower communication, it might not be worthwhile to remove the extra link between them and having to place it back a short while later.

Finally the choice of benchmarks should be extended. When considering large shared-memory machines, a large and growing fraction of them is being used as commercial server machines, rather than as scientific computers. Commercial applications, like databases and web services, will therefore be added to the benchmark suite. Because these types of applications usually have less regular access patterns and can be more dependent on remote miss latencies, differing conclusions may apply.

VI. CONCLUSIONS

Using detailed simulation to obtain accurate traces of network traffic and subsequent analysis of these traces, we have shown that communication bursts occur between node pairs, and that the duration of these bursts can be several milliseconds. A reconfigurable network that can provide a fourfold speedup for a significant fraction of traffic in bursts of at least 1 ms long will be able to achieve speedups ranging from 1.02 to 1.86 for the range of applications considered. According to our results, the switching time for the components needed to realize such a network would need to be significantly less than 1 ms, which is near the limit of the projected optical technologies.

VII. ACKNOWLEDGMENTS

This paper presents research results of the PHOTON Inter-university Attraction Poles Program (IAP-Phase V), initiated by the Belgian State, Prime Min-

ister's Service, Science Policy Office. C. Debaes is indebted to the FWO for his post-doctoral fellowship.

REFERENCES

- [1] M. Brunfaut, W. Meeus, J. Van Campenhout, R. Annen, P. Zenklusen, H. Melchior, R. Bockstaele, L. Vanwassenhove, J. Hall, B. Wittman, A. Nayer, P. Heremans, J. Van Koetsem, R. King, H. Thienpont, and R. Baets, "Demonstrating optoelectronic interconnect in a FPGA based prototype system using flip chip mounted 2D arrays of optical components and 2D POF-ribbon arrays as optical pathways," in *Proceedings of SPIE*, vol. 4455, Bellingham, July 2001, pp. 160–171.
- [2] K. Bui Viet, L. Desmet, J. Dambre, K. Beyls, J. Van Campenhout, and H. Thienpont, "Reconfigurable optical interconnects for parallel computer systems: design space issues," in *VCSELS and Optical Interconnects*, vol. 4942. Brugge: SPIE, October 2002, pp. 236–246.
- [3] L. Chao, "Optical technologies and applications," *Intel Technology Journal*, vol. 8, no. 2, May 2004.
- [4] J. Collet, W. Hlaylel, and D. Litaize, "Parallel optical interconnects may reduce the communication bottleneck in symmetric multiprocessors," *Applied Optics*, vol. 40, pp. 3371–3378, 2001. [Online]. Available: <http://www.laas.fr/collet/A02001.pdf>
- [5] C. Katsinis, "Performance analysis of the simultaneous optical multi-processor exchange bus," *Parallel Computing*, vol. 27, no. 8, pp. 1079–1115, 2001.
- [6] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. L. Hennessy, M. Horowitz, and M. S. Lam, "The Stanford DASH multiprocessor," *IEEE Computer*, vol. 25, no. 3, pp. 63–79, March 1992.
- [7] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *IEEE Computer*, vol. 35, no. 2, pp. 50–58, February 2002.
- [8] B. McNutt, *The Fractal Structure of Data Reference: Applications to the Memory Hierarchy*. Kluwer Academic Publishers, 2000.
- [9] D. A. B. Miller and H. M. Ozaktas, "Limit to the bit-rate capacity of electrical interconnects from the aspect ratio of the system architecture," *Journal of Parallel and Distributed Computing*, vol. 41, no. 1, pp. 42–52, 1997.
- [10] T. M. Pinkston and J. W. Goodman, "Design of an optical reconfigurable shared-bus-hypercube interconnect," *Applied Optics*, vol. 33, no. 8, pp. 1434–1443, 1994.
- [11] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer, "Beowulf : A parallel workstation for scientific computation," in *Proceedings of the International Conference on Parallel Processing, Boca Raton, USA*. CRC Press, August 1995, pp. 11–14.
- [12] J. L. Sanchez, J. Duato, and J. M. Garca, "Using channel pipelining in reconfigurable interconnection networks," in *6th Euromicro Workshop on Parallel and Distributed Processing*, January 1998.
- [13] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proceedings of the 22th International Symposium on Computer Architecture*, Santa Margherita Ligure, Italy, 1995, pp. 24–36. [Online]. Available: <http://citeseer.nj.nec.com/woo95splash.html>