PERFORMANCE EVALUATION OF LARGE RECONFIGURABLE INTERCONNECTS FOR MULTIPROCESSOR SYSTEMS

Wim Heirman, Iñigo Artundo*, Joni Dambre, Christof Debaes*,
Pham Doan Tinh†, Bui Viet Khoi†, Hugo Thienpont*, Jan Van Campenhout

ELIS Department, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
* TONA Department, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussel, Belgium
† Hanoi University of Technology, 01 Dai Co Viet Str, Hanoi, Vietnam

ABSTRACT

Communication has always been a limiting factor in making efficient computing architectures with large processor counts. Reconfigurable interconnects can help in this respect, since they can adapt the interprocessor network to the changing communication requirements imposed by the running application. In this paper, we present a performance evaluation of these reconfigurable interconnection networks in the context of shared-memory multiprocessor (SMP) machines. We look at the effects of architectural parameters such as reconfiguration speed and topological constraints, and analyze how these results scale up with higher processor counts. We find that for 16 processors connected in a torus topology, reconfigurable interconnects with switching speeds in the order of milliseconds can provide up to 20% reduction in communication delay. For larger networks, up to 64 processors, the expected gain can rise up to 40%. This shows that reconfigurable networks can help in removing the communication bottleneck from future interconnection designs.

1. INTRODUCTION

Traffic patterns on an interprocessor communication network are far from uniform. This makes the load over the different network links vary greatly across individual links, as well as over time, when different applications are executed or even when the same application goes through different phases. Most fixed-topology networks are therefore a suboptimal match for realistic network loads. These problems magnify as the network size increases since even a single-link congestion can quickly spread to slow down the whole network.

One solution to this problem is to employ a reconfigurable network, which has a topology that can be changed at run-time. This way the traffic pattern can, at each point in time, be accommodated in the most efficient way (i.e., with the highest performance, the lowest power consumption, or a useful tradeoff between them). We have previously introduced a generalized architecture in which a fixed *base network* with regular topology is augmented with reconfigurable *extra links* that can be placed between arbitrary node pairs [1]. While the network traffic changes, the extra links are repositioned to locations were contention on the base network is most significant. An implementation using optical interconnection technologies and tunable lasers to provide the reconfiguration aspect can be found in [2].

In this paper we extend our previous work in reconfigurable network evaluation to larger size networks (from 16 to 32 and 64 nodes) and observe the improvement in communication performance as we scale up the shared-memory multiprocessor system.

2. SYSTEM ARCHITECTURE

2.1 Multiprocessor architecture

We have based our study on a multiprocessor machine that implements a hardware-based shared-memory model. This requires a tightly coupled machine, usually with a proprietary interconnection technology yielding high 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 sub-problems (i.e., that communicate often). Since network communication here is largely hidden from the programmers, they can do little to hide communication latency which makes the performance of such machines very vulnerable to increased network latencies.

Modern examples of this class of machines range from small, 2- or 4-way SMP server machines (including multi-core processors), over mainframes with tens of processors (Sun Fire, IBM iSeries), up to supercomputers with hundreds of processors (SGI Altix, Cray X1). For several important applications, the performance of the larger types of these machines is already interconnect limited [3]. Also, their interconnection networks have been moving away from topologies with uniform latency, such as busses, into highly non-uniform ones where latencies between pairs of nodes can vary by a large degree. To get the most out of these machines, data and processes that communicate often should be clustered to neighboring network nodes. However, this clustering problem often cannot be solved adequately when the communication pattern exhibited by the program has a structure that cannot be mapped efficiently (i.e., using only single-hop connections) on the base network topology. Also communication requirements can change rapidly, this limits the use of a software approach which would move processes and data around on the network. Therefore, SMP systems are very likely candidates for the application of reconfigurable interconnection networks.

For this study we consider a shared-memory machine in which cache coherence is maintained through a directory-based coherence protocol. In this computing model, every processor can address all memory in the system. Accesses to non-local words are intercepted by the network interface, which generates the necessary network packets requesting the corresponding word from its home node. Since processors are allowed to keep a copy of remote words in their own caches, the network interfaces also enforce cache coherence which again causes network traffic. This way, memory access requests may stall the processor for one or more network

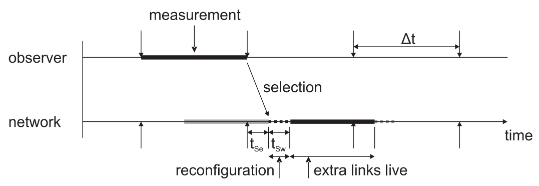

Fig. 1 Extra link selection and reconfiguration.

round-trip times (in the order of hundreds of nanoseconds). This is much more than the time that out-of-order processors can occupy with other, non-dependent instructions, but not enough for the operating system to schedule another thread. System performance is therefore very much dependent on network latency.

2.2 Reconfigurable network architecture

Our network architecture starts from a base network with fixed topology. In addition, we provide a second overlapped network that can realize a limited number of connections between arbitrary node pairs – these will be referred to as *extra links* or *elinks*. To keep the complexity of the routing and elink selection algorithms acceptable, packets can use a combination of base network links and at most one elink on their path from source to destination.

The elinks are placed such that most of the traffic has a short path (low number of intermediate nodes) between source and destination. This way a large percentage of packets will end up with a correspondingly low (uncongested) latency. In addition, congestion is lowered because heavy traffic is no longer spread out over a large number of intermediate links. A heuristic is used that tries to minimize the aggregate distance traveled multiplied by the size of each packet sent over the network, under a set of implementation-specific conditions: the maximum number of elinks n, the number of elinks that can terminate at one node (the *fan-out*, f), etc. After each execution time interval of length Δt (the reconfiguration interval), a new optimum topology is computed using the traffic pattern measured in the *previous* interval, and the elinks are repositioned (Figure 1).

The reconfiguration interval must be chosen short enough so that traffic doesn't change too much between intervals, otherwise the elink placement would be suboptimal. On the other

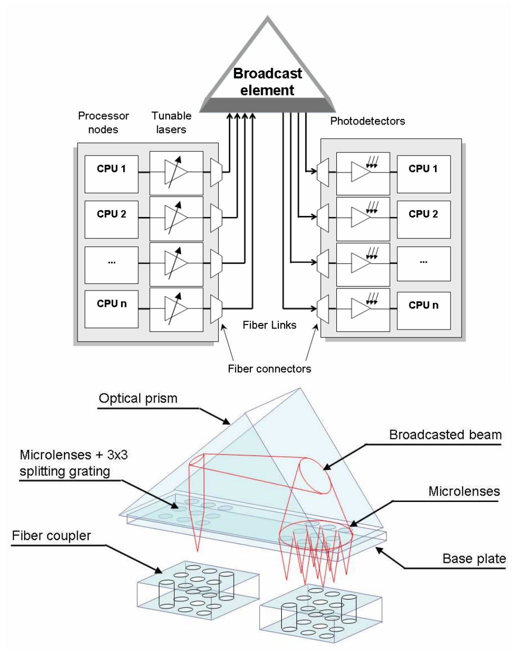

Fig. 2 Implementation using a Selective Optical Broadcast (SOB) component

hand, the elinks are unavailable for routing traffic while they are being reconfigured (the duration of this reconfiguration we call the *switching time*, t_{Sw} in Figure 1). This limits the choice of the reconfiguration technology to one that has a switching time much shorter than the reconfiguration interval. In this work, we realistically assume this to be the case and further ignore the switching times, bearing in mind however that the selection of a certain technology places a lower limit on the reconfiguration interval (of for instance 10× the switching time).

2.3 Implementation

The physical implementation of the reconfigurable optical network we envision can be done by using low-cost tunable laser sources, a broadcast-and-select scheme for providing the extra optical links, and wavelength selective receivers on every node (Figure 2). By tuning the laser source, the correct destination is addressed. On the transmission side, tunable Vertical Cavitiy Surface Emitting Lasers (VCSELs) are preferred for their characteristics, with a tuning range of a few tens of channels and a switching speed between 100 µs and 10 ms.

The broadcasting could be done through the use of a starcoupler-like element that reaches all the nodes. When scaling up to tens of nodes or more this is no longer feasible: the number of available wavelengths is finite, also such a wide broadcast would waste too much of the transmitted power. In this case a component like a diffractive optical prism can be used, which broadcasts light from each node to only a subset of receiving nodes. Note that the routing to and from the broadcast element will be such that nodes will have different neighbors on the broadcast element than those on the base network. This way the elinks will span a distance on the base network that is larger than one hop.

On the receiving side, Resonant Cavity Photodetectors (RCPDs) make each node susceptible to just one wavelength. Integration of all these optical components has been proven and (non-reconfigurable) optical interconnects are currently arriving to the midrange servers. More information about this envisioned implementation can be found in [2].

3. SIMULATION METHODOLOGY

3.1 Simulation platform

We have based our simulation platform on the commercially available Simics simulator [4]. It was configured to simulate a multiprocessor machine based on the Sun Fire 6800 server, with 16, 32 or 64 UltraSPARC III processors clocked at 1 GHz and running the Solaris 9 operating system. Stall times for caches and main memory are set to realistic values (2 cycles access time for L1 caches, 19 cycles for L2 and 100 cycles for SDRAM). Both the caches with directory-based coherence and the interconnection network are custom extensions to Simics. The network models a 4×4, 4×8 or 8×8 torus with contention and cut-through routing. A number of extra point-to-point (optical) links can be added to the torus topology at any point in the simulation.

The network links in the base network are 16 bits wide and are clocked at 100 MHz. In the reported experiments, the characteristics of an elink were assumed to be equal to those in the base network, yielding a per-hop latency that is the same for an elink as for a single base network link. Both coherence traffic (resulting from the directory-based protocol) and data (the actual cache lines) are sent over the network, and result in remote memory access times representative for a Sun Fire server (around 1 µs on average).

To avoid deadlocks, dimension routing is used on the base network. Each packet can go through one elink on its path, after that it switches to another virtual channel to avoid deadlocks of packets across elinks. For routing packets through the elinks we use a static routing table: when reconfiguring the network, the routing table in each node is updated such that for each destination it tells the node to route packets either through an elink starting at that node, to the start of an elink on another node, or straight to its destination.

The SPLASH-2 benchmark suite [3] was chosen as the workload. It consists of a number of scientific and technical applications using a multi-threaded, shared-memory programming model, and is representable for the expected workload on a real machine of this scale.

3.2 Network architecture

To both determine the expected performance of our reconfigurable network implementation using the selective optical broadcast component, and to predict the performance of other, future implementations, we run our simulations with a hypothetical parameterized architecture that provides the infrastructure to potentially place an elink between any two given nodes. Two constraints are made on the set of elinks that are active at the same time:

- a maximum of *n* elinks can be active concurrently,
- the *fan-out* (the number of elinks that terminates at any one node) is limited to *f*.

The reconfiguration interval *Δt* is the third parameter, all simulation results in this paper will be based on different sets of values for these three parameters. The network from Section 2.3 using the Selective Optical Broadcast (SOB) element can be modeled using *f = 1*, *n* equal to the number of processors, and an additional constraint on which destinations can be reached through an elink from each source node. In our

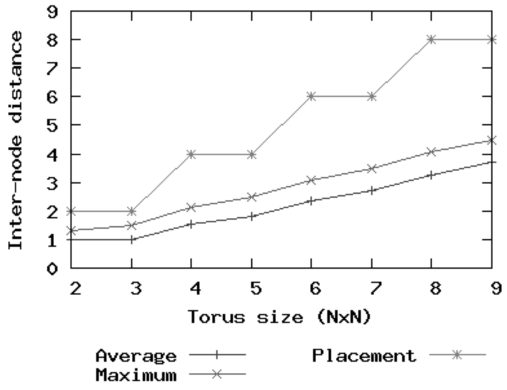

Fig. 3 Average and maximum inter-node distances for torus networks of differing size, and the new average distance after adding elinks using the optimal node placement.

case, only the surrounding neighbors on the prism will be available as destinations (at most 9 out of *n*). This significantly limits the connectivity of the design. The placement of nodes on the prism, which determines which 9 nodes each node can reach through an elink, is therefore crucial. More information on how this placement is determined is provided in Section 3.4.

3.3 Extra link selection

At the start of each reconfiguration interval, a decision has to be made on which elinks to activate, within the constraints imposed by the architecture, and based on the expected traffic during that interval (with, in our current implementation, the expected traffic being the traffic as measured during the previous interval). As explained in section 3.2, we want to minimize the number of hops for most of the traffic. We do this by minimizing a cost function that expresses the total number of network hops traversed by all bytes being transferred.

The time available to perform the elink selection (called the *selection time*, t_{Se} in Figure 1) is from the same order of magnitude as the switching time, because both need to be significantly shorter than the reconfiguration interval. Since the switching time will typically be in the order of milliseconds, we need a fast heuristic that can quickly find a set of elinks that satisfies the constraints imposed by the architecture and has a near-optimal cost. This algorithm is described in detail in [5].

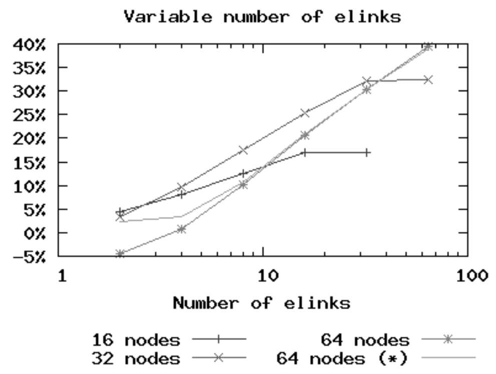

Fig. 4 Performance scaling for increasing the number of extra links, *f = 2* and *Δt = 100 µs*.

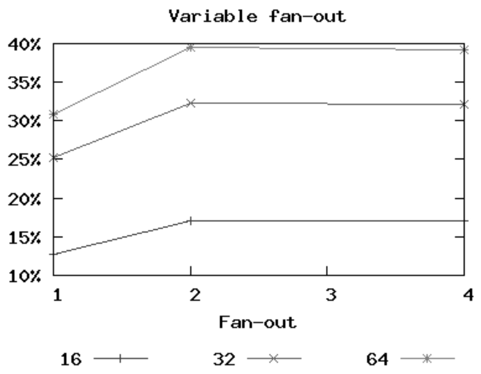

Fig. 5 Performance scaling for increasing the fan-out, *n = #CPUs* and *Δt = 100 µs*.

3.4 Node placement on the prism

To provide all nodes with a reconfigurable connection, each will be connected to the SOB prism. This mapping of the source and receiving nodes will be crucial, as it directly determines the possible destinations for every transmitting node through the SOB. This mapping results in the *placement matrix*, which needs to be optimized such that the resulting inter-node distances are minimized.

The number of possible permutations of all placements increases in a factorial way with the network size. A heuristic approach is therefore needed to solve the problem in a realistic time. We used simulated annealing to optimize the node placement algorithm, with the average inter-node distance as the evaluation metric. We tried to maximize the distance saved using the elinks by surrounding every node on the prism with nodes that are distant on the base network. As a result, we obtained close-to-optimal placements resulting in an ideal average distance saving of up to 46.5% (for an 8×8 torus, relative to a base network only implementation). Figure 3 shows this, plotting, for a number of network sizes, the maximum and average inter-node distances for a torus network, and the resulting distance using the optimal node placement.

4. RESULTS

Figures 4 through 7 show our simulation results. Each plots the improvement of the network latency relative to a base network only implementation, averaged over the execution of all benchmark applications.

In Figure 4 the effect of adding more elinks is shown. The fan-out is always *f = 2*, the reconfiguration interval *Δt* is fixed at 100 µs. Clearly, adding more elinks increases performance, up to a point where there are as many elinks as there are nodes where the performance levels out. Further simulations will therefore be done with *n = #CPUs*, this is also the case for our implementation with the SOB prism. Also visible is a negative improvement in the 64-node case with only 2 elinks. Here, too many nodes want to use the elinks causing highly increased congestion which has a large negative impact on performance. The *64 nodes (*)* case avoids this extra congestion by allowing only a few nodes to use the elinks.

Next we study the influence of the fan-out. If one node communicates frequently with a large set of other nodes, it would be interesting to connect multiple elinks to this first node. This is however not technologically feasible, indeed, in our prism implementation the fan-out is just one. Figure 5 shows the effect of different fan-out limitations, all for *n = #CPUs* and *Δt = 100 µs*. As can be expected, improving the fan-out from 1 to 2 increases performance. Higher fan-outs however do not result in more performance gains.

Figures 6 and 7 show the effect of faster reconfiguration, here we vary *Δt* between 1 µs and 10 ms. In Figure 6 we used *f* = 2 and *n = #CPUs*, Figure 7 is for the prism implementation which can be expressed as *f = 1,*

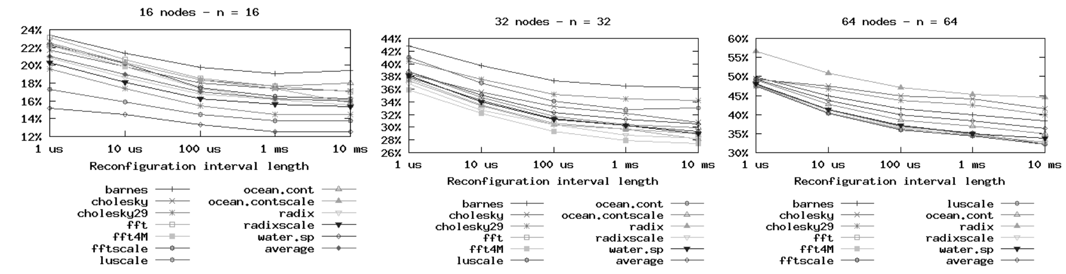

Fig. 6 Performance trends for varying reconfiguration intervals, all with *f* = 2 and *n* = *#CPUs*.

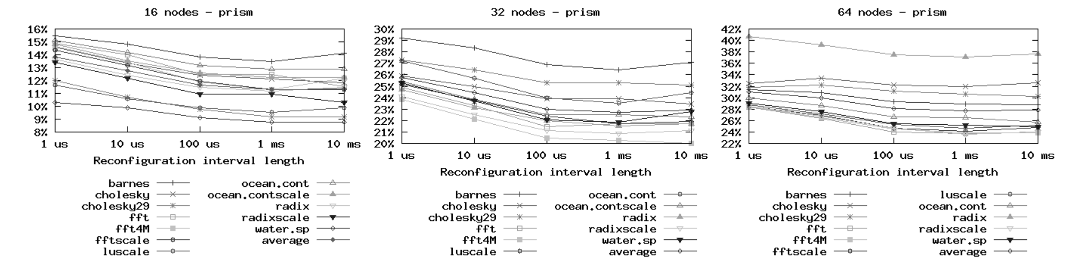

Fig. 7 Performance trends for varying reconfiguration intervals, SOB implementation.

n = *#CPUs* and an extra limitation on which nodes are reachable from each processor. Here we have plotted the different benchmarks separately, indicating that the behavior of the program being executed on the multiprocessor machine highly influences the performance improvement obtained. An extreme case for instance is the *radix* application running on 64 nodes, which suffers much less performance degradation when moving from an *f = 2* network to a prism implementation whereas the limited connectivity of the second network causes a more significant drop in performance for all other benchmarks.

5. CONCLUSIONS

It is clear that, as processor counts increase in SMP systems, the performance bottleneck moves to the interconnection network. Reconfiguration of the interconnect can speed up communications in shared-memory multiprocessors by up to 40%. We have characterized this speedup for large networks, and found how it depends on the application, network size, and the topological constraints of the reconfigurable design. Our optical implementation has been validated and further results can be extracted from this model.

ACKNOWLEDGEMENTS

This paper presents research results of the Inter-university Attraction Poles Programs PHOTON (IAP-Phase V) and photonics@be (IAP-Phase VI), initiated by the Belgian State, Prime Minister's Service, Science Policy Office. C. Debaes is indebted to the FWO for his post-doctoral fellowship. We also acknowledge the VLIR-UOS program to support our close collaboration with the Hanoi University of Technology.

REFERENCES

1. W. Heirman, J. Dambre, I. Artundo, C. Debaes, H. Thienpont, D. Stroobandt and J. Van Campenhout. Integration, the VLSI Journal, Vol. 40(2007), pp. 382-393.
2. I. Artundo, L. Desmet, W. Heirman, C. Debaes, J. Dambre, J. Van Campenhout and H. Thienpont. JSTQE, Vol. 12(2006), pp. 828-837.
3. S.C. Woo, M. Ohara, E. Torrie, J.P. Singh and A. Gupta, in ISCA 22, Santa Margherita Ligure, Italy, (1995), pp. 24-36.
4. P.S. Magnusson et. al. IEEE Computer. Vol. 35(2002), pp. 50-58.
5. W. Heirman, J. Dambre, I. Artundo, C. Debaes, H. Thienpont, D. Stroobandt and J. Van Campenhout. Photonic Network Communications, 2007, *to appear.*