Latency and Bandwidth Requirements of Massively Parallel Programs: FFT as a Case Study

Fabrizio Petrini and Marco Vanneschi

Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy,
tel +39 50 887248, fax +39 50 887226
e-mail: {petrini,vannesch}@di.unipi.it

Abstract. In this paper we compare three routing algorithms for massively parallel architectures, each offering an increasing degree of adaptivity: a *deterministic* algorithm, a minimal adaptive based on *Duato*'s methodology and a non-minimal adaptive, the *Chaos* routing. Rather than using a synthetic benchmark, the comparison is done with a real application, the transpose FFT algorithm. The simulation results collected on bi-dimensional tori with up to 256 processing nodes show that both adaptive algorithms suffer from post-saturation problems that degrade the network throughput.

1 Introduction

In many existing works in the literature routing algorithms are compared using synthetic benchmarks. These include uniform, transpose, complement and other communication patterns. In this paper we adopt a detailed simulation model, that describes both the interconnection network and the internal structure of the processing nodes, to compare three routing algorithms using a real application, the transpose FFT algorithm.

The remainder of this paper is organized as follows. Section 2 presents some parallel FFT algorithms. Section 3 describes the relevant details of the simulation model that we use in Section 4 to compare the routing algorithms. An overview of the experimental results and some concluding remarks are given in section 5.

2 FFT algorithms

The most commonly used parallel FFT algorithm maps each row r of an n-input butterfly on processor $\lfloor \frac{rP}{n} \rfloor$, where P is the number of processors[1]. With this mapping, processors need to communicate with each other during the computation of the first $\log P$ columns. The remaining $\log \frac{n}{P}$ columns are available on the same processor. While this mapping is effective on the hypercubes because

[1] We will assume that both n and P are powers of two.

generates a *normal* algorithm, low-dimensional cubes cannot make efficient use of a large number of processors [7].

An alternative formulation, often named *transpose* algorithm, divides the computation into two computational phases separated by a global communication phase [6]. Each processor computes in the first phase an $\frac{n}{P}$-input butterfly available locally, allocated according to a cyclic mapping. In the second computational phase each processor has assigned $\frac{n}{P^2}$ butterflies, whose size is P. There is a direct connection between any two butterflies belonging to different computational phases. This implies that the communication phase is an *all-to-all personalized broadcast* (AAPB) where each processor during the first computational phase has to send the values of $\frac{n}{P^2}$ output butterfly nodes to the remaining $P-1$ processors.

3 The simulated model

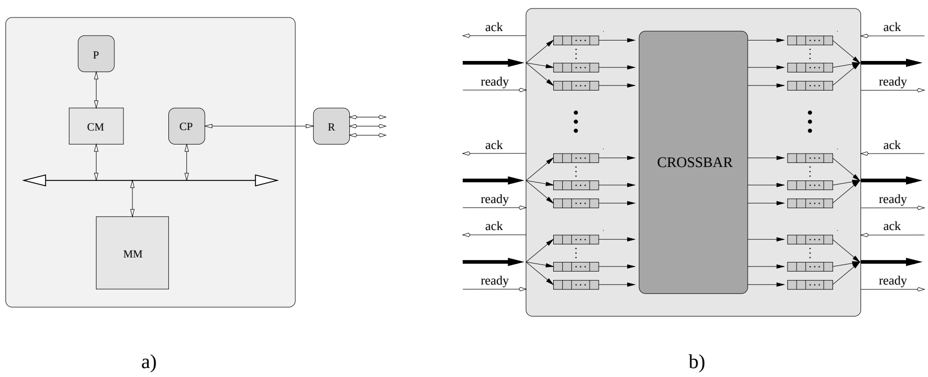

Fig. 1. a) The internal structure of the processing node. b) The internal structure of the router. P processor, CM cache memory, MM main memory, CP communication processor, R router.

This section presents the simulation model of parallel architectures that is used to analyze the behavior of the routing algorithms. The internal structure of the processing node is shown in Figure 1 a) and is centered around the main bus. We distinguish the following units: the *main processor*, its *cache memory*, the *main memory*, the *communication processor* and the *router*.

The computational part of the algorithm is executed on the main processor. The communication processor is dedicated to message handling and acts as a Message Driven Processor [3]. It fragments outgoing messages in packets that are enqueued in a FIFO directed to the router and receives incoming packets from another FIFO. The communication processor can perform DMA-like activities and store the content of incoming messages in proper memory locations. There is a simple coherency mechanism between the cache memory and the communication processor. The cache controller snoops on the main bus, and every time a memory location is overwritten by the communication processor, it invalidates the corresponding cache line, if present in the cache memory. Symmetrically, when the communication processor tries to read a memory location replicated on the cache memory, it gets the updated copy.

Figure 1 b) outlines the internal structure of a router. We can distinguish the external channels or links, the input and the output buffers or lanes that implement the buffer space of the virtual channels and an internal crossbar.

3.1 Routing algorithms

We compare three algorithms, each offering a different degree of adaptivity: deterministic, minimal adaptive based on Duato's methodology and non-minimal adaptive, the Chaos routing.

The deterministic algorithm is a dimension order routing based on a static channel dependency graph [4]. Packets are sent to their destination along a unique, predetermined and minimal path. The potential deadlocks caused by the wrap-around connections are avoided doubling the number of virtual channels and creating two distinct virtual networks. Packets enter the first virtual network at the beginning and switch to the second network upon crossing a toroidal connection. Our version uses two virtual channels for each physical link.

Rather than using a static channel dependency graph, Duato's methodology only requires the absence of cyclic dependencies on a connected channel subset. The remaining channels can be used in almost any way. We associate four virtual channels to each link: on two of these channels, called *adaptive* channels, packets are routed along any minimal path between source and destination. The remaining two channels are *escape* channels where packets are routed deterministically when the adaptive choice is limited by network contention. A static channel dependency graph is used in the channel subset [5]

The Chaos routing is a non-minimal cut-through routing algorithm. Under normal conditions, it routes packets from its input buffers to the output buffers as in minimal adaptive routing. The Chaos router has a central multiqueue to avoid blocking and deadlocks. A packet is moved into the multiqueue if it is potentially in a deadlock loop or it has stalled at an input buffer. When an output buffer becomes available, packets in the multiqueue have priority over packets in the input buffers and, if more than a packet in the multiqueue desires the channel, it is allocated using a fair policy. If a packet must be moved into the multiqueue when it is full, another packet in the queue is chosen at random and derouted to the next available output buffer along a potentially non-minimal path. As suggested in [1], we utilize a multiqueue with five packet buffers.

3.2 SMART

This model is evaluated in the SMART (Simulator of Massive ARchitectures and Topologies) environment. Implemented in C++, SMART is an object-oriented discrete-event simulation tool for evaluating massively parallel architectures. Configuring some shell scripts, it is possible to select the node internal structure, the network topology and the internal router policies. The simulator allows the definition of the network topology, routing algorithm, packet length, number of virtual channels and buffers for both input and output lanes. Also, it is possible to monitor several metrics and time-dependent events, that are gathered in trace files.

The computation of the non-input nodes of the butterfly requires 4 clock cycles when the operands are on chip. A read hit is served in 3 cycles and a read miss in 26 cycles. The message marshaling overhead in the communication processor is 40 cycles, for both incoming and outgoing packets. The flit size is 16 bits and the link delay to transmit a flit across a physical link is 4 cycles.

Adaptive algorithms have more degrees of freedom but require larger crossbars and more complex arbitration [2]. For this reason the routing delay is normalized with the router complexity. The routing delay is 4 cycles for the deterministic algorithm, 8 cycles for the Duato algorithm and 12 cycles for the Chaos routing.

4 Experimental results

In our experiments we mapped a 65536-input butterfly [2] on machines with up to 256 processors. In Figure 2 a) we can see that the speedups for the three routing algorithms are very close. It is worth noting that there is a superlinear speedup with up to 64 processors. This is due to the favorable computation/communication ratio of the transpose algorithm and to smaller working sets that lead to a better memory hierarchy utilization. The computation with 256 processors is bandwidth-limited: the Chaos routing provides a slightly better speedup of 116, followed by the Duato's algorithm with 110 and the deterministic with 107. The number of active processors is shown in Figure 2 b). We can easily distinguish the two local phases separated by the global communication.

Though similar in terms of accepted bandwidth, the three algorithms show widely different behaviors, as can be seen in Figure 2 c) and d). While the network utilization, i. e. the fraction of active links, of the deterministic algorithm is stable around 22%, in the Duato's algorithm it oscillates between 5% and 42%. In the Chaos routing the network utilization reaches 63%, even if the communication pattern is executed with a comparable amount of time.

The Duato's algorithm suffers from post-saturation problems. In a saturated network incoming packets tend to flood the adaptive channels and this limits the network throughput. The lowest network utilization is reached around 27000

[2] A 65536-input butterfly is the minimum data set size supported by the transpose algorithm on a 256 processors architecture.

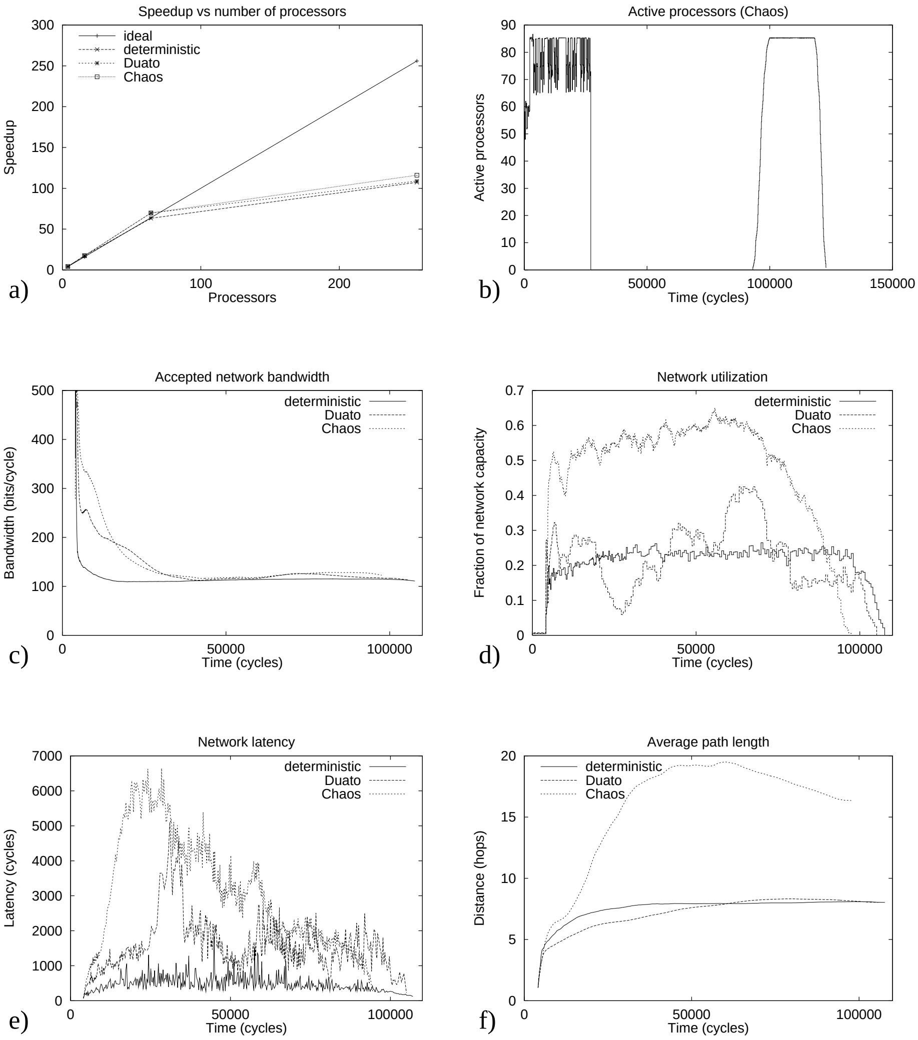

Fig. 2. Performance results on a 256 processors bi-dimensional torus.

cycles, at the end of the first computational phase. When the pressure at the source nodes begins to decrease, there is a corresponding increase in the accepted bandwidth, which reaches 42%.

The gap between the network utilization and throughput in the Chaos routing can be explained looking at the graphs in Figure 2 e) and f). At saturation, many packets are derouted and temporally sent away from their minimal path. In fact the average path length reaches 20 hops, more than twice the topological average distance. This is confirmed by the sharp increase of the network latency[3] at over 6000 cycles.

5 Conclusion

In this paper we have analyzed the behavior of three routing algorithms using a real application, the transpose FFT algorithm. The experimental results have shown that the performance of both adaptive algorithms is very close to the deterministic one.

The algorithm based on Duato's methodology becomes unstable when the network is saturated, with a network utilization that falls down to 5% of the capacity. The usage of a source throttling mechanism to limit the injection of new packets when the network is close to saturation could be a viable solution to solve this problem.

In the Chaos routing many packets are derouted from their destination when the network is saturated. Even if the network utilization is high, above 60%, the global network throughput is low. Also, the network latency in the initial phase of the global communication pattern experiences a sharp increase.

From these results we argue that adaptive algorithms should provide a better and more stable post-saturated throughput to be competitive with the simpler deterministic ones.

References

1. Kevin Bolding. *Chaotic Routing: Design and Implementation of an Adaptive Multicomputer Network Router*. PhD thesis, University of Washington, Department of Computer Science and Engineering, Seattle, WA, July 1993.
2. Andrew A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Hot Inteconnects '93*, Palo Alto, California, August 1993.
3. William J. Dally et al. The Message-Driven Processor. *IEEE Micro*, pages 23–39, April 1992.
4. William J. Dally and Charles L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.
5. José Duato. A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, October 1995.
6. Anshul Gupta and Vipin Kumar. The Scalability of FFT on Parallel Computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(8):922–932, August 1993.
7. F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.

This article was processed using the LaTeX macro package with LLNCS style

[3] The network latency is the time spent by a packet in the network, without including source queuing delay