Introducing Kilo-instruction Multiprocessors

Marco Galluzzi†, Valentín Puente‡, Oliverio J. Santana†, Carmelo Acosta†, Adrián Cristal†, Ramón Beivide‡, José-Angel Gregorio‡, Mateo Valero†

I. INTRODUCTION

The gap between processor speed and memory latency is continuously widening, making the problem of the Memory Wall harder and harder [1]. In the multiprocessor design arena the Memory Wall reveals as an exacerbated problem, since remote memory accesses exhibit larger latencies and they have to share network bandwidth with coherency and synchronization traffic.

In this paper we will study the influence of Kilo-instruction processors on the performance of small-scale CC-NUMA multiprocessors. They are a very novel and promising processor architecture designed to deal with long latency instructions and consist in maintaining thousands of in-flight instructions [2] [3]. When running numerical applications, Kilo-instruction processors have demonstrated its ability to effectively maintain high values of IPC while increasing memory latencies.

Using a convenient simulation framework, which combines RSIM [4] and SICOSYS [5], we will analyze the behaviour of Kilo-instruction based CC-NUMA multiprocessors. As we will see, our first results using an ideal network show the enormous potential of Kilo-instruction processors when using them as computing nodes not only for hiding local DRAM latencies but also for the remote ones. A deeper analysis, using realistic networks, reveals the existence of heavy demands on packet throughput required by each node, since larger re-order buffers translate on higher density of remote accesses. Next, we show that current interconnection networks cannot cope with this high traffic levels, so newer and faster networks have to be designed.

The paper is organized as follows. In section II, we describe related work. In section III, we describe the simulator used and the benchmarks involved in our study. We show and analyze the results obtained in our simulations in section IV. In section V, we explain possible solution to the problems found in the analysis. In the last section we give the most relevant conclusions.

II. RELATED WORK

Many techniques have been proposed and used in the past to reduce the negative effects due to memory latency in uniprocessors. Cache memory [6] exploits the spatial and temporal locality exhibited by most programs. Prefetching fetches data from memory before it is requested [7] [8]. Later, the concept of having another thread analyzing more complex prefetching, bringing data from memory to L2 cache, has been introduced in [9].

In multiprocessors, we can also successfully use the same techniques to hide the communication latency. However, to reduce the memory latencies we also have to take care of the memory consistency model, where the use of a relaxed model can be useful [10]. To reduce communication latency, new interconnection network models and new routing algorithms and artefacts are being studied [11].

A recent technique to deal with long latency memory accesses is the mentioned Kilo-instruction processor. It proposes having thousands of in-flight instructions by combining different mechanisms that overcome the problem of up-sizing the critical resources: re-order buffer [2], instruction queue [12] and physical registers [13]. These mechanisms are feasible since critical resources are shown to be underutilized in present out-of-order processors [14].

III. SIMULATION ENVIRONMENT

A. EXPERIMENTAL TOOLS AND BENCHMARKS

A detailed network simulator called SICOSYS [5] has been integrated together with the RSIM multiprocessor simulator [4] providing a powerful execution-driven environment to emulate a complete CC-NUMA multiprocessor using local caches,
a directory-based coherency protocol and a switched interconnection network.

The microarchitecture modelled by RSIM is close to the MIPS R10000 processor, and it has been adequately configured in order to simulate a Kilo-instruction processor able to manage up to 2048 in-flight instructions.

Due to the limitations imposed by the complexity of the simulated system, 16KB L1 cache and 128-KB L2 cache have been used. The benchmarks employed in this study have been tuned according to these cache sizes. Other parameters of the simulated system are shown in Table I. The memory consistency model used is Release Consistency.

To carry out a realistic evaluation, we fed our simulation platform with three representative applications selected from the SPLASH-2 suite: Radix, FFT and LU. The problem size for FFT is 256K complex data points. This value correspond to the problem size established in [15]. Due to the high demand for computational resources, the problem size of LU has been reduced from its default size of 512x512 to 256x256. The problem size for Radix has been also reduced from one million integer keys to a half-million using the default radix of 1024.

Our study is based on assuming memory latencies of 250, 500 and 1000 cycles. The first value is used because is a reasonable number for current systems, and the last two values because we want to go a step further on and take into account future latencies.

### B. Interconnection Network Models

We are going to consider two types of interconnection networks. Firstly, an unrealistic network design will be used in order to establish an upper bound for the maximum achievable performance. This network, which we will denote as ideal network, will consist of a two-stages pipelined crossbar operating at the processor frequency. Secondly, a realistic network design will be considered for establishing practicable performance gains, so topology, router architecture and wire technology have to be carefully chosen.

Due to its good performance and its extended use [16] the selected topology is the bi-dimensional Torus in which four input/output links are used for communicating neighboring network nodes.

Taking into account the characteristics of the current wire technology we have set the router operation frequency at 666 MHz. In addition, links will be bidirectional supporting 32 data bits in each direction. It means in turn, that the phit size managed by the network (number of bits transmitted in parallel per port and network cycle) is set to 32 bits. Coherency protocol packets have been implemented to be 16 bytes long.

Finally, the router core employed is the Bubble Adaptive (BADA) router [17], so the realistic network will be denoted as BADA network. BADA router uses fully adaptive routing and Bubble Flow Control for avoiding packet deadlock [18]. It has a six-stages pipeline and two virtual channels (implemented as FIFO queues) per input port.

### IV. Experimental Results

#### A. Kilo-instruction Multiprocessors

In this subsection, we are going to evaluate a Kilo-instruction Multiprocessor system. This system will be made of 16 processors. Each processor employed will be that described in Table I and the network will be the ideal network cited in section III-B.

The following results are useful in three ways. First, we want to demonstrate that the desirable effects of a Kilo-instruction single processor can be extended to a multiprocessor system. Second, we want to know what the magnitude of those effects is. Third, they are also useful to establish an upper bound for possible performance enhancements.

Figure 1 shows, for each benchmark running on 1 or 16 processors, how the increasing amount of in-flight instructions, starting from 64 (the baseline), influences the resulting IPC. Observing the results for 1 processor, there are improvements between 170% and 370% approximately for all the benchmarks, using a 2048 entries ROB over the baseline. Having a look to the results for 16 processors we can also appreciate improvements. We have limited im-
provements for the LU benchmark, about 37%, and still considerable improvements for Radix and FFT benchmarks, 166% and 227% respectively.

The reason of having minor improvements, and better baseline IPCs, when running on 16 processors lies in the fact that there are less L2 cache missing loads, since we divide the same problem size between more processors, as shown in figure 2a. This is meaningful for our analysis, since we still observe significant improvements when using larger ROBs, as shown in figure 1, despite the decrement of the L2 miss rates, the important factor in Kilo-instruction processors.

Figure 2b shows the percentage of the total execution time in which the ROB has all of its entries occupied. These results are strongly related to the L2 miss rate, since the higher the miss rate the higher the percentage of the time the ROB is full. So, due to the lower multiprocessor L2 miss rates, lower percentages in which the ROB is full are achieved, what indicates an underutilization of larger ROBs.

Once studied the set of proposed benchmarks, we have demonstrated that the performance increase obtained by Kilo-instruction processors in uniprocessor systems is also achieved in multiprocessor systems.

\subsection*{B. The Impact of the Interconnection Network}

In this subsection we analyze the behaviour of Kilo-instruction Multiprocessors when using a realistic interconnection network: a bi-dimensional 4x4 Torus network with BADA routers. We will also employ an ideal network in order to establish a performance upper bound.

We start our analysis considering a high-end microprocessor running at 4 GHz with a 64 entries ROB and DRAM memory latency of 500 processor cycles. We consider three experiments: ideal network and memory (zero latency), only ideal network and BADA network. The results are shown in figure 3.

Under these conditions, the impact on system performance of the interconnection network is almost negligible. By using an ideal network, it is only possible to achieve, in the most favouring case, a performance gain under 10% in respect to a realistic network. On the one hand, DRAM latencies dominate over the remote accesses. Currently, the base-latency in a realistic network is only around 35 processor cycles. On the other hand, the network does not suffer from congestion because there are few in-flight remote accesses and hence, a low traffic volume to be managed.

The previous scenario totally changes when the number of ROB entries is progressively incremented. Figures 4a, 4b and 4c show the performance achieved for ROB sizes between 64 and 2048 entries and memory latency of 250, 500, and 1000 cycles for the FFT, LU and Radix benchmarks respectively.

When analyzing the FFT benchmark, we can clearly appreciate the benefits obtained when incrementing ROB sizes. It can be also seen that there are not important differences between 250 and 500 cycles memory latency. In both cases, the performance of an ideal network using processors with 2048 entries ROBs is quite close to that achieved when using a combination of an ideal network and an ideal memory. Latencies of 1000 cycles are more difficult to tolerate. There is still a margin of improvement around 15% in respect to the performance upper bound. When using a realistic network as BADA, we can see that we are obtaining just 65% of the maximum achievable performance (ideal mem. and net.). Performance differences between ideal and BADA networks are lower when using long memory latencies.

For the LU benchmark, maximum performance for an ideal network can be obtained by using smaller ROBs. No additional improvements can be appreciated by using ROBs bigger than 1024 entries. Although an ideal network provides a performance quite close to the upper bound for 250 cycles, it can be seen that the higher the latencies the lower the performance gains. BADA networks only achieve around 80% of the maximum achievable performance for all the considered latencies. As in the FFT case, the differences between ideal and BADA networks
Fig. 4. Performance of three SPLASH-2 benchmarks running on 16 processors, using different types of network and assuming three different memory latencies.

Fig. 5. Distribution of the execution time among different tasks for the FFT and a memory latency of 500 cycles.

decrease when augmenting memory latencies.

Although performance gains are achieved with Radix when augmenting the ROB sizes, a different behaviour can be observed. The performance exhibited by an ideal network is quite sensitive to memory latencies for a 2048 entries ROB. For 250, 500 and 1000 cycles, an ideal network just obtain around 80%, 72% and 53% of the maximum achievable performance. In contrast, the system performance remains, more or less, stable when varying memory latencies in the case of realistic BADA networks. BADA only can achieve around 50% of the benefits provided by a combination of an ideal network and an ideal memory. This application could clearly benefit from having bigger ROBs.

We can see that, under a realistic scenario, the ability to achieve higher values of IPC by using big ROBs will depend on the interconnection network effectiveness. As stated before, the use of standard instruction windows generates a relatively low traffic volume and the interconnection network can cope with it. Nevertheless, bigger instruction windows implies higher rates of remote accesses which in turn, brings a higher pressure over the network. In such situations, contention among packets arises and consequently, remote accesses exhibit longer latencies.

In order to obtain a better understanding of these phenomena, figure 5 shows how the execution time is expended among different system activities for a FFT execution and 500 cycles memory latencies. The execution time has been divided into four different parts: "busy" for processing instructions, "local mem" for intra-node memory accesses, "remote mem" for inter-node memory accesses and "sync" for synchronization primitives. Figure 5 plots values for each analyzed run when they are normalized in respect to the slowest run. It can be seen how an ideal network with 2048 entries ROBs can nearly behave the same that a system having also an ideal memory. Consequently, their execution times are the same. It is also clear the ability of big ROBs to hide local memory latencies when using both an ideal network and a BADA network. Nevertheless, it is easy to see the performance degradation caused by the latency increase of the remote memory accesses. These long latencies are clearly caused by contention among packets in a saturated network, which constitutes the system bottleneck. This bad operation causes that, in the best of the realistic cases, processors are stopped almost half of the execution time waiting for data.

In order to have a better insight about this congestion we are going to analyze two snapshots of the network behaviour. Set of figures 6 shows the traffic throughput and packet latencies supported by the network when executing the benchmark. In some execution phases, which correspond to all-to-all collective communications, the network throughput is around 14 phits consumed every network cycle, close to the maximum of 16 phits. Hence, in such execution phases the network is saturated regardless the window size. Although in such phases the network is able to manage a slightly higher load by using big ROBs, it is not translated into a superior performance when consider the whole application. In contrast, intensive computing phases in which no remote accesses exist can benefit from the ability of big ROBs for hiding more local latencies.

Having a look to figure 6b we can observe how the network contention negatively impacts on the remote access latencies. As we increment the number of in-flight instructions, remote accesses are also in-
cremented but the network is unable to cope with them. In such cases, an important part of the traffic generated by the computing nodes cannot be injected in the network causing a clear increment of the latency experienced by packets. As the system is saturated, the latency provoked by the network itself remains nearly the same regardless the ROB size. Nevertheless, the waiting times at the interface injection queues experience a notable increase leading to higher overall latencies. Fortunately, using big ROBs can hide local latencies and, in some cases, compensates the degradation caused by network congestion. It is very illustrative to compare the bars corresponding to BADA routers for 1024 and 2048 entries ROBs.

V. Reducing the Interconnection Network Bottleneck

Two approaches can be considered for improving the network, which has been proved to be a bottleneck in Kilo-instruction Multiprocessors. They are based on providing the network with either technological or architectural enhancements. Although both solutions are not mutually exclusive, we are going to analyze them separately.

On the one hand, employing technological approaches, network performance can be clearly improved by using wider communication links or making them work at higher frequencies. By augmenting frequency, the network load will decrease because of the reduction of the ratio between processor and network cycles. All the previous analysis was done using 32-bits parallel links operating at 666 MHz, which can be considered a high value for current off-chip wire technologies. Nevertheless, that frequency can be increased in the future by using either new wire technology or higher integration levels. A similar effect could be achieved by widening the network links. The designer may have to choose between both approaches depending on design and cost constrains. In order to simulate a practicable design we are going to employ a simpler router which facilitates the implementation of links operating at higher frequencies. For this purpose, we will use a router with deterministic Dimension Order Routing (DOR) and Bubble Flow Control for packet deadlock avoidance [18]. This router, which is denoted as BDOR, consists of a five-stages pipeline and employs FIFO queues located at the input ports.

On the other hand, employing architectural approaches, network performance can be improved by using more aggressive router architectures. For this purpose, we will consider an evolved router architecture denoted as HPAR (High-Performance Adaptive Router) [11] with 32-bits parallel links operating at 666 MHz. HPAR inherits most of the characteristics of BADA routers but it employs multiport output memories in order to reduce the negative effects of the so-called Head-of-Line (HOL) blocking.

In order to evaluate the performance improvements provided by each previous enhancement, we are going to analyze the FFT benchmark for a 2048 entries ROB when varying the router clock cycle, the link width and when employing HPAR routers.

Figures 7a and 7b show the obtained IPC values when consider BDOR and HPAR routers for different memory latencies. In most cases, we can observe that long memory latencies are worst tolerated. Nevertheless, there are some cases (slow BDORs and HPAR) in which slower memories can return higher benefits, since fast memories can easily flood the network, which causes higher contention.
On-Chip Kilo-instruction Multiprocessors

Analyzing the general behaviour, we can see that performance gains are noticeable but even using network links operating at one third of the processor frequency (1.33 GHz), the network contention still provokes a performance degradation around 25% in respect to the ideal case. Moreover, the cost of having 32 parallel data wires clocked at such frequencies can be prohibitively high if not impracticable. Furthermore, the flexibility required by medium to high scale multiprocessors able to accommodate a considerable number of nodes makes that frequency simply unreachable. Nevertheless, integrating all the network nodes together with the network itself in the same die should make possible the attainment of such a clock cycle. This is a clear indication for exploring On-Chip Kilo-instruction Multiprocessors.

In addition, it is possible that on-chip multiprocessor designs allow the use of wider data links. We have done some experiments varying the link width as reflected in figure 7b. We can conclude that the effect of widening the links is quite similar to augmenting their frequencies. In this way, doubling the link width (from 32 bits to 64) returns the same benefits that doubling their frequencies (from 666 MHz to 1.33 GHz). It should be mentioned here that widening the network links can provoke an undesirable added network contention. This phenomenon is related to the fact that smaller packets originate a higher number of collisions among their headers requiring more arbitration cycles in which bubbles can appear in the router pipeline.

As stated before, an alternative solution for improving network performance relies on architectural enhancements. Some proposals consist in improving the router local memories [19] and avoiding HOL blocking [20], among others. In order to have a clue about the effectiveness of such solutions, both figures 7a and 7b show the system performance when using HPAR networks. When comparing its performance against the one obtained by BDOR we can see that by using HPAR we can obtain benefits which have nearly the same effect that either doubling the network frequency or doubling the link width.

VI. Conclusions

From previous works done on Kilo-instruction processors we can conclude that a clear way to tolerate increasing memory latencies is to maintain a large number of in-flight instructions. With this paper, a first evaluation of what we call Kilo-instruction Multiprocessors, we have demonstrated that this is also true for multiprocessor systems. Therefore, employing Kilo-instruction processors as computing nodes in CC-NUMA multiprocessors can provide significant performance improvements. The reason is that not only the memory latency is tolerated but also the communication latency, what appears when performing remote memory accesses and sending coherence and synchronization messages. An other conclusion from our work is that the increase of communication traffic, due to the aggressive speculation imposed by Kilo-instruction processors, make the interconnection network a relevant factor. The ability of the multiprocessor system to achieve high IPC rates will depend on the effectiveness of the interconnection network. In short, Kilo-instruction Multiprocessors are a novelty in the multiprocessor design arena that open new directions where more research can be useful. These directions include new multiprocessor models and new interconnection networks.

VII. Acknowledgments

This work was supported by the Ministry of Science and Technology of Spain under contracts TIC-2001-0995-C02-01, TIC-2001-0591-C02-01, and grant AP2003-0539 (M. Galluzzi). This work has been also supported by the HiPEAC European Network of Excellence. The authors would like to thank Francisco J. Cazorla for his valuable comments.

Referencias