First International Workshop on
HyperTransport™ Research and Applications

Proceedings

of the

1st International Workshop on HyperTransport Research and Applications

WHTRA 2009

Editors
Holger Fröning
Mondrian Nüssle
Pedro Javier García García

ISBN: 978-3-00-027249-3

February 12th, 2009, Mannheim, Germany

University of Heidelberg, Computer Architecture Group

Proceedings of the
First International Workshop on HyperTransport Research and Applications (WHTRA2009)
Feb. 12th, 2009, Mannheim, Germany

PGAS Model for the Implementation of Scalable Cluster Systems*

Juan A. Villar, Francisco Andújar
Francisco J. Alfaro, José L. Sánchez
DSI – Univ. of Castilla–La Mancha
02071 – Albacete, Spain
{*juanan, fandujar, falfaro, jsanchez*}*@dsi.uclm.es*

José Duato
DISCA – Tech. Univ. of Valencia
46022 – Valencia, Spain
jduato@disca.upv.es

Abstract

This paper introduces an extended version of the traditional Partitioned Global Address Space (PGAS) model, for the implementation of scalable cluster systems, that the HyperTransport Consortium Advanced Technology Group (ATG) is working on. Using the Simics and GEMS simulators, we developed a software module that approximates the behavior of a PGAS cluster. This approach mainly provides the simplest mechanism to evaluate how much the PGAS infrastructure will affect overall the application performance. The aim of this work is to study the feasibility of the ATG's PGAS model for running applications with high memory requirements. Such a model, will let manufacturers build clusters that enable the execution of these applications, in such a way that it will be impossible to run them in a single processor, or in a multi–processor.

1. Introduction

Traditionally, shared memory systems have been used to run applications requiring a high memory space. However, such as systems do not scale more than tens of processors. As memory requirements of applications and the number of applications that run concurrently on computers have been increasing, designers have made proposals to partially solve the lack of memory on computer systems. In this way, modern operating systems provide advanced virtual memory managers that solve the lack of memory. These managers utilize secondary devices for freeing contents of memory whenever it is necessary. This approach provides a simple way of running applications that have a running image size bigger than available physical memory size. When an application is running and the system is low of memory, the virtual memory manager can evict it to a special device called a "swap device", or swap, to free memory. This technique is called swapping [17], and there are several approaches in the literature about it. Just to mention a few, Unix–based systems use a separate swap partition type that is hosted in the user file system. In contrast, Windows uses a user–space file that is hosted inside the file system, while the MacOS X operating system can use partitions and files.

Nevertheless, swapping has several drawbacks. The first one comes from the access time of the swapping device, which is usually a hard disk, therefore, one or more orders of magnitude higher than the memory access time. The second drawback is thrashing, which occurs when the memory manager evicts parts of the running image of a process and, after a time, it reallocates those parts in memory again. Thus, solving the lack of memory always involves a high run time.

The AMD's Opteron processor can be used as a commodity to build clusters. This processor includes the memory controller on–die, in such a way that all memory is accessible from one memory controller. The Opteron processors use the AMD's HyperTransport protocol [8] for communicating with each other. Moreover, the HyperTransport protocol enables CPUs to directly connect the Opteron HyperTransport link to add–in card subsystems via the HTX connector [5], which is placed in the motherboard.

The HyperTransport Consortium Advanced Technology Group (ATG) is working on an extended version of the traditional Partitioned Global Address Space (PGAS) programming model [3]. Such a model will let manufacturers build clusters with PGAS native support. In this paper we carry out an assessment of "rough" PGAS model through Simics [10] and GEMS [11] simulators. However, this work has not attempted to conduct a study using a hardware implementation, because the ATG has not completed the specification of its PGAS model yet, and therefore it is impossible to achieve that kind of evaluation. The behavior of the final system will be similar (bridging the gap) to the behavior in a cluster with PGAS native support.

*This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants Consolider "Ingenio–2010 CSD2006–00046" and "TIN2006–15516–C04–02"; and by Junta de Comunidades de Castilla–La Mancha under grant PCC08–0078–9856 and Beca Predoctoral de Investigación 07/096.

In addition, the increasing use of interconnection networks to intercommunicate the processor and memory, such as AMD HyperTransport, might allow the construction of scalable systems, from hundreds to thousands of nodes composed of multicore processors. Therefore, the Opteron processors will provide the basis to build the clusters with PGAS native support.

The remainder of this paper is structured as follows: In the next section we present a summary of the related work. The details of the proposed model are explained in Section 3, while Section 4 details the simulation scenarios and the results obtained. In the last Section, we conclude and provide a brief overview of the future work.

2. Related Work

HyperTransport is an interconnection technology which enables connecting the processors among each other and with the I/O devices. It provides an extremely low latency, high bandwidth and excellent scalability. Moreover, the definition of the HTX connector allows co–processing and acceleration based on ASIC or FPGA technologies. In particular, it is receiving a highlighted interest of the community because it makes easy to reduce execution time by the use of accelerators.

Partitioned Global Address Space languages combine a Single Program Multiple Data (SPMD) programming model with a global address space, which is logically partitioned to give each thread a portion of shared memory to which it has affinity [19]. In the SPMD model, a fixed number of threads are created at program startup, and every thread runs the same program. Each thread has both a space for private local memory and some partition of the shared space to which it has affinity. A private object may only be accessed by its corresponding thread, whereas all threads can read or write any object in the shared address space. The partitioning of the shared space into regions with logical affinity to threads allows programmers to explicitly control data layout, which is then used by the runtime system to map threads and their associated data to processors: on a distributed memory machine, the local memory of a processor holds both the thread's private data and the shared data with affinity to that thread.

The HTC Advanced Technology Group (ATG) [1] is working on developing proposals for HyperTransport to define an address space globally and dynamic partitioning (PGAS) for using in scalable clusters. The idea is not new, but it comes from the existing PGAS models [3]. In the bibliography several PGAS applications can be found, for example, Unified Parallel C [4] to define models of programming languages. In addition, developers of these languages have tools like GASnet [2] which is a communication interface for programming languages such as Unified Parallel C. GASnet is a language independent of the network that allows the definition of libraries providing global addressing. GASnet is inspiring the work of HTC Advanced Technology Group for the implementation of PGAS [18] in a native way.

The ATG is proposing the mechanisms and abstractions that will allow the construction of clusters using Opteron processors. The motherboard containing Opteron processors will support the HTX connector. By plugging extension cards on the HTX connector will allow the formation of a cluster of motherboards. The memory controller of each Opteron will divide the whole range of physical addresses in regions and distribute them among the memory of other Opterons. Subsequently, the memory controllers will be able to access remote regions of memory transparently. Following this approach, the Opteron processors can avoid the use of a device for the lack of memory, since access to remote memory controller will be lower than access to a local secondary storage device, and hence system performance will be enhanced.

In such systems, the physically addressable memory in all nodes is part of a global address space with non–uniform access time from any specific node. From the perspective of a node, the global address space is composed of local partitions and remote partitions where the former can be accessed with the lowest latency and the latter can be accessed with larger and possibly non–uniform latencies (across distinct partitions). In this model, a local partition refers to DRAM accessed through a tightly integrated memory controller. The latency of remote memory accesses will be non–uniform because it will depend on interconnection performance and the load and contention of the network.

Specific details of the implementation of the PGAS model cannot be provided in this paper because they are still confidential, and some aspects of it are still under deliberation of the ATG group.

The Computer Architecture Group at the University of Heidelberg in Germany has the expertise to design complex hardware/software systems. The HTX–Board [7], a contribution of this group, provides a convenient and efficient way to evaluate user specific devices connected to the Hypertransport connector standardised under the name of HTX–Connector. In [16] they published the architecture and mechanisms of the HTX–board. Subsequently, in [9] they have introduced a novel communication engine in combination with the HyperTransport interface. They provide an excellent prototype to get real–world measurements connecting two Opteron through two HTX–boards. Moreover, they also show the initial latencies of sending a HyperTransport packet on a HyperTransport link and propose some optimisations that can minimise that latency (e.g. doubling the HyperTransport clock frequency or migrating FPGA to ASIC technology).

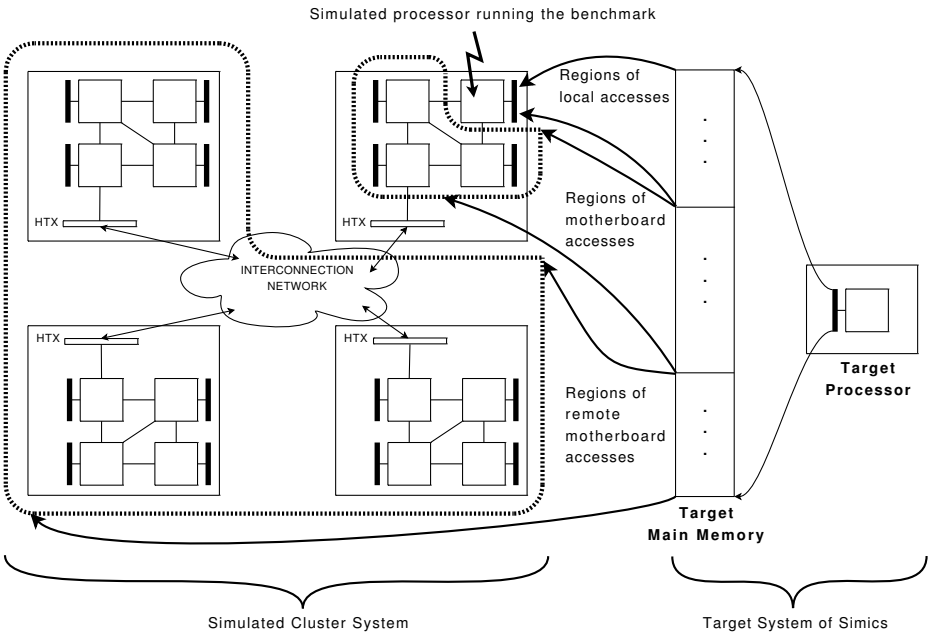


Figure 1. Diagram of the simulated cluster.

3. Model

Because of the fact that the ATG has not completed the specification of its model PGAS yet, it is impossible to carry out an evaluation using a hardware system. However, it is feasible to carry out an approximate evaluation of the PGAS model by simulation.

In order to simulate the cluster, Simics 2.1 and GEMS 2.2.19 simulators were used. In the Simics context, two fundamentals terms are always used:

- The computer on which we are running Simics is referred to "host system".
- The computer simulated by Simics is referred to "target system". Specifically, it simulates the cluster with PGAS support.

Our approach consists in simulating the execution of one sequential application as if it would be running on a cluster with PGAS support just using the execution–driven Simics simulator. In that hypothetical cluster, any processor might issue requests of the global address space. The global address space will consist of the joint of every memory in the cluster. From the processors point of view, most of the physical memory in the cluster can be accessed, except some private areas that will not be allowed to access. The Figure 1 explains how a cluster can be simulated starting from a simulated processor in Simics.

Without lossing generality, we run one sequential application in the target system and we process every message going inside the system. The messages are deliverated by the simulator as they are in a usual execution, but the delay of every message is customised depending on the type, source and destination of the message.

In the AMD's whitepaper [14] a suite of benchmarks is examined to illustrate their performance and scalability in single, multi–processor, and cluster configurations. Its results clearly show the exceptionally responsiveness of an Opteron–based NUMA support system. Specifically, it shows the cost for one processor for accessing to the shared memory in a four AMD Opteron motherboard. The latencies are given depending on the distance between the transmitter and receiver processors.

Regarding the application, it must be a benchmark that makes an intense use of memory. Also, a sequential application is preferred in order to avoid any dependency produced by a parallel execution. Stream [12] is a well–known benchmark that measures bandwidth sustainable by ordinary user programs, and not the theoretical peak bandwidth that vendors advertise [13]. Moreover, it is used by AMD to measure the performance of the memory of their processors [15].

The benchmark performs functions with matrices that are stored in memory. The functions are executed several times. When the benchmark concludes, it returns the rate of traffic data of memory in MB/second and execution time (average, minimum and maximum) in seconds, for each function. Both performance indexes are the most relevant in this kind of benchmarking. The execution time gives the global performance measure and the traffic rate offers the real load of the memory system.

4. Evaluation

In this section, we start describing the simulation model we have used to carry out our experiment. Then, we present the results we have obtained and some comments about them.

4.1. Simulation Model

In all the simulations, our customized GEMS module is loaded. It is responsible for managing all the messages sent by the memory system. We assume that the target processors utilised in this work are used to build systems with a coherent shared memory, similar to the Opteron processors. Therefore, the GEMS module has to manage the messages caused by the memory coherence protocols. The study of memory coherence protocols is out of the scope of this paper. Thus the simulations have been carried out with a single processor in order to reduce the influence of these protocols.

The host system is a SUN W2100Z workstation that has two Opteron processors at 2.4 GHz and a DDR–400 memory of 4 GB. SuSe 10.2 is used as operating system. The target system is the *sarek* preinstalled system of Simics which is a UltraSPARC processor at 75 MHz. Solaris is used as operating system and an amount of 512 MB of memory is configured.

The assumption that the whole memory of the cluster is 512 MB seems initially nonsense. However, the aim of this work has never been to propose a detailed PGAS simulation model because the ATG has not finished its PGAS model yet. Considering an increase of the target memory size, that is the memory size of the cluster, requires to increase the Stream benchmark size and then it causes an exponential simulation time growth that would be unaffordable. Even though the results remain representative because the benchmark spreads the accesses out the memory address space.

Additionally, the parameters of the Stream benchmark have been set to achieve the following behavior:

1. The target host is running a unique process of Stream benchmark in absence of processes that interfere with Stream. Meanwhile, the memory accesses are controlled by the customized GEMS module.
2. Stream runs two series of functions (copy, scale, add and triad). Previous tests had proved that increasing the number of series does not alter the final outcome in the absence of processes that interfere with Stream.
3. The size of Stream is 460 Mbytes. This size was chosen because it represents 90% of the available simulated memory (512 MB).
4. The GEMS simulator requires to set the latencies in the target processor cycles, so a 2.4 GHz processor was considered for the translation from real nanoseconds to simulated cycles.

It must be noticed that the latencies in our simulations are considered as an approximation. However, we have selected values that reflect real systems:

- When an access is destined for memory allocated in the same motherboard, a latency of 115 ns for each access has been considered [14]. In that case, this value is the mean time for both read and write accesses.
- When an access is destined for memory allocated in a remote motherboard, the latency has been deduced from the proposed delays in [9, 16]. We assume all the improvements suggested by [9, 16] like ASIC technology instead of using FPGA technology, doubling the HyperTransport link frequency, and a 16–bit HyperTransport link width. In this case, the calculation of the latencies is:
  - Due to all the technology improvements, [9] claims a total latency of 130 ns for the transmission and it expects a fixed latency (for local CPU and remote memory controller) of about 300 ns. The functionality of [9] is exactly the behavior of a remote write operation (transmision of data from source node to destination node). Hence we assume a latency of 430 ns for a write operation of 64 bytes payload.
  - The read is much more costly. The Opteron used in [16] only issues 32 bit read operations. The read operation consists of transmiting the read request to the destination node, and then transmiting the result back to the source node. Because of the access granularity of 8 bytes (it is a limitation of the Opteron K10 architecture) a simple 8 bytes read operation costs 610 ns. Therefore, a series of 8 consecutive operations have to be issued to retrieve the total amount of 64 bytes. This is the reason for assuming the read operation takes 4880 ns.
- When an access is destined for memory mapped in a swap device, we assume a latency of 45,600 ns as

it is suggested in [6] for enterprise class harddisks. In that case, this value is the mean seek time for a variable 512 bytes sector size. We assume this time as a representative, so we do not consider neither the sector size effect nor cache implications.

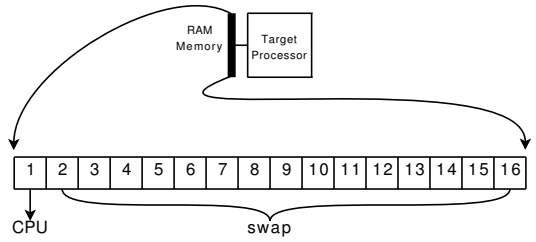
Regarding to simulation scenarios, we have considered the following scenarios:

- Local scenario (1P): It represents a desktop system with just one processor and one motherboard.
- Shared scenario (4P): It represents a server system, commonly known as a shared memory multiprocessor. There are four processors assembled in one motherboard.
- Remote scenario (16P): It represents a enterprise system or a cluster. A total of sixteen processors are distributed between four motherboards of four processors per motherboard that are interconnected using HTX connectors.

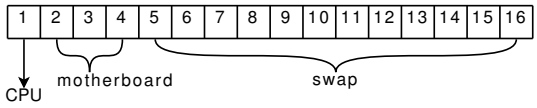
A group of extra three scenarios have been selected to evaluate the loss of performance due to the utilisation of swapping. Figure 2 shows the distribution of the target memory into regions and the access type that is associated with each region. Each region corresponds to a contiguous physical memory partition controlled by a single node. In our test, we assumed all regions are of the same size. These extra scenarios are based on the previous scenarios and they consist in distributing the regions of memory between swapping devices, as if they were accesses to secondary devices and therefore such accesses suffer an extra delay. All the extra scenarios assume a partitioning of the memory in 16 regions. Specifically, these three extra scenarios are:

- 1P–SW: It is basically the 1P scenario, but the accesses from the second to the last regions are accesses to a swap device (see Figure 2(a)).
- 4P–SW–U: It is similar to the 4P scenario. The swap regions are assigned uniformly (see Figure 2(b)).
- 4P–SW–D: Similar to the 4P–SW–U scenario, but the swap regions are interleaved on the memory (see Figure 2(c)).

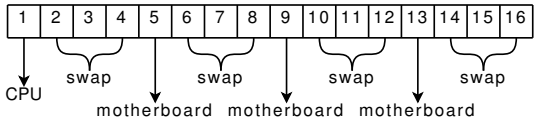
The previous partitioning of the memory is quite extreme because it assigns up to 90% of the target memory to swap. However, that partitioning represents a suitable configuration for checking the influence of the target operating system and how its target virtual memory manager allocates the target memory (e.g. how the target memory is allocated to the applications).



(a) 1P–SWAP



(b) 4P–SWAP–U



(c) 4P–SWAP–D

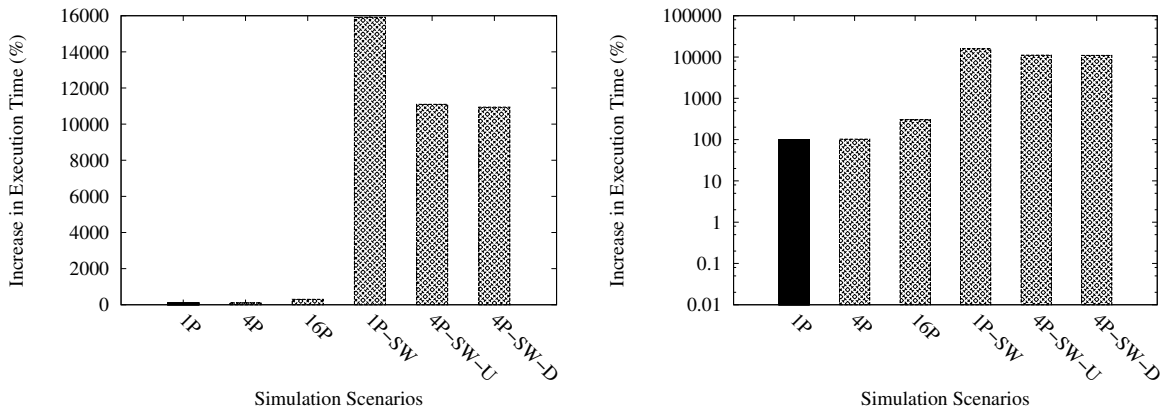
Figure 2. Distribution of target memory into regions and their access type.

4.2. Simulation Results

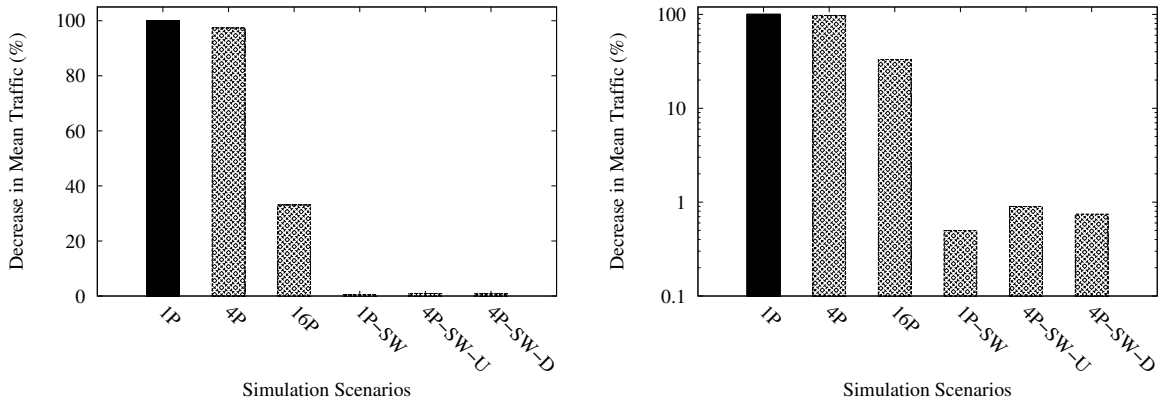
The aim of this work has never been focused on the performance of the application, but the behavior of a cluster with native PGAS support (scenario 16P) to run applications. Note that it will be impossible to run these applications in a single processor (scenario 1P) or multi–processor (scenario 4P) due to the memory requirements of these applications. Note also that the only alternative in these cases is the use of swapping devices, which is considered in the 1P–SW and 4P–SW scenarios.

Mainly, it is interesting to know how the execution time evolves. Figure 3 depicts that the execution time increases for 4P and 16P on average 0.54 seconds (2.68%) and 40.76 seconds (202.68%) with regard to the 1P scenario. Because of it is a memory benchmark, it is also interesting to know how the traffic memory evolves, so Figure 3 depicts that the performance of the memory for 4P and 16P decreases 0.53 MB/s (2.63%) and 13.46 MB/s (66.93%), regarding the performance achieved by scenario 1P. When extra scenarios are studied, the results are even worse, as it is expected. Both, the execution time and the memory traffic, fall dramatically for all scenarios.

In Figure 4 we show the same results, but only for the 1P, 4P and 16P scenarios in order to improve the readability of the Figure 3.

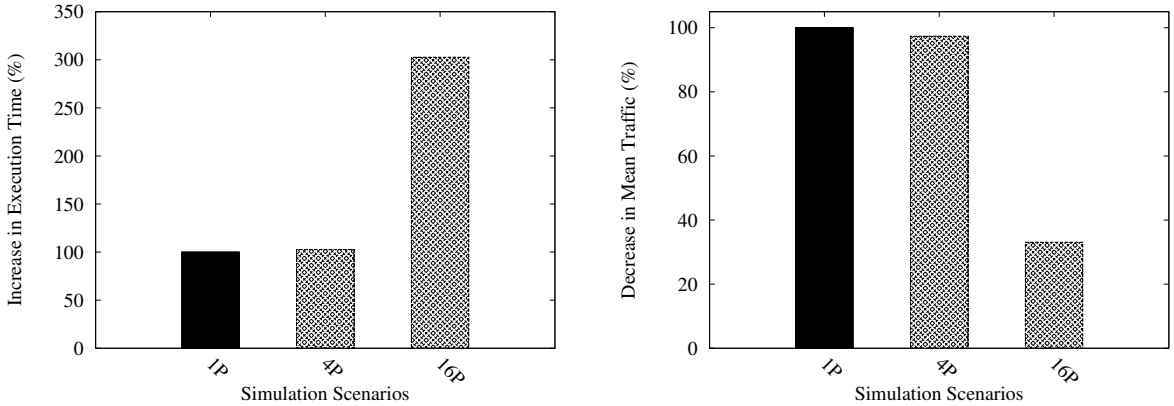


(a) Increase in execution time with regard to scenario 1P (numerical and logarithmic scale).



(b) Decrease in mean traffic with regard to scenario 1P (numerical and logarithmic scale).

Figure 3. Performance results of the Stream application in each scenario.



(a) Increase in execution time with regard to scenario 1P.

(b) Decrease in mean traffic with regard to scenario 1P.

Figure 4. Detailed performance results of the Stream application in 1P, 4P and 16P scenarios.

The results have shown that scenario 16P using the PGAS model is always a better choice than extra scenarios that implement swapping. Of course, the performance of the unfeasible scenarios 1P and 4P is much better than the performance of the 16P scenario, but this one is the best option for those applications with high memory requirements.

5. Conclusions and Future Work

This paper presents the results of the preliminary assessment of the work in progress that is made by the ATG. Because the ATG has not completed the specification of its PGAS model yet, it is not possible to carry out a hardware evaluation. However, we have performance a simulation–driven study.

Firstly, we have developed a module of the GEMS simulator for tracking the memory requests and customizing their latency. By this module, we could simulate approximately the behavior of an application running in a cluster with PGAS support. This cluster would run any application with high memory requirements if they do not exceed the whole physical memory of the cluster, because the application will be spread out into the DRAM memory of the processors in the cluster.

As it was explained, swapping can solve the lack of memory, but the application performace falls dramatically as the lack of memory increases. This paper has introduced the PGAS model as one alternative to swapping. Results have showed that the PGAS model would never be a better option than having enough memory in the processor that runs the applications, because a lot of time would be spent in accesses to remote memories. However, it will be always better than using swapping, because the latencies of the inter–memory communications will be lower than accesses to swapping.

As future work it is interesting to keep updated of all the work that is done by the ATG, for example, the real implementation of the PGAS support, the future improvements of the HTX–board, and the specification of the HTX connector.

Acknowledgements

The authors want to give special thanks to Dr. Holger Fröning of the University of Heidelberg for his helpful comments and assistance for the development of this work.

References

[1] HyperTransport Consortium Advanced Technology Group. `http://www.hypertransport.org`.

[2] D. Bonachea. Gasnet Specification, version 1.1, Report No. UCB/CSD–02–1207, October 2002.

[3] P. Charles, C. Grothoff, V. Saraswat, and et. al. X10: An object–oriented approach to nonuniform cluster computing (OOPSLA'05). In *Proceedings of the 20th annual ACM SIGPLAN Conference on object oriented programming, systems, languages, and applications*, 2005.

[4] T. El-Ghazawi, W. Carlson, T. Sterling, and K. Yelick. *UPC: Distributed Shared Memory Programming*. John Wiley and Sons–May, 2005.

[5] D. Emberson and D. O'Flaherty. HTX Specification for HyperTransport 3.0 Daughtercards and ATX/EATX Motherboards. Technical report, HyperTransport Consortium, June 2008.

[6] Enterprise–class versus Desktop class Hard Drives, Revision 1.0, April 2008.

[7] H. Fröning, M. Nüssle, D. Slogsnat, H. Litz, and U. Brüning. The HTX–Board: A Rapid Prototyping Station. In *Proceeding of 3rd Annual FPGAWorld Conference, Stockholm, Sweden*, November 2006.

[8] HyperTransport Consortium. Hypertransport Specification, Revision 3.0, April 2007.

[9] H. Litz, H. Froening, M. Nuessle, and U. Bruening. VELO: A Novel Communication Engine for Ultra–Low Latency Message Transfers. In *Proceedings of 37th International Conference on Parallel Processing (ICPP–08), Portland, Oregon, USA*, September 2008.

[10] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A Full System Simulation Platform Computer. *Computer*, 35(2), February 2002.

[11] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's General Execution–driven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News (CAN)*, September 2005.

[12] J. D. McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Computers. Technical report, University of Virginia (USA), 2007. `http://www.cs.virginia.edu/stream`.

[13] S. A. McKee. Reflections on the Memory Wall. In *Conference Computing Frontiers*, 2004.

[14] D. O'flaherty and M. Goddard. AMD Opteron Processor Benchmarking for Clustered Systems. Technical report, Advanced Micro Devices, July 2003.

[15] Second–Generation AMD Opteron Processor Industry Standard Server Benchmarks. `http://www.amd.com`.

[16] D. Slogsnat, A. Giese, M. Nüssle, and U. Brüning. An open–source HyperTransport core. *ACM Trans. Reconfigurable Technol. Syst.*, 1(3):1–21, September 2008.

[17] W. Stallings. *Operating Systems: Internals and Design Principles (6th Edition)*. Prentice Hall, April 2008.

[18] S. Yalamanchili, J. Young, J. Duato, and F. Silla. A Dynamic, Partitioned Global Address Space Model for High Performance Clusters. Document GIT–CERCS–08–S01, School of Electrical and Computer Engineering (Georgia Institute of Technology (USA); Universidad Politecnica de Valencia (Spain), 2008.

[19] K. Yelick and et. al. Productivity and Performance Using Partitioned Global Address Space Languages. In *Proceeding of International Workshop on Parallel Symbolic Computation (PASCO'07)*, July 2007.