Disaggregated Computing. An Evaluation of Current Trends for Datacentres

Hugo Meyer[1], José Carlos Sancho[1], Josue V. Quiroga[1], Ferad Zyulkyarov[1], Damián Roca[1], and Mario Nemirovsky[2]

[1] Barcelona Supercomputing Center (BSC), Barcelona, Spain
[2] ICREA Senior Research Professor at Barcelona Supercomputing Center (BSC), Barcelona, Spain
hugo.meyer@bsc.es, jose.sancho@bsc.es, josue.quiroga@bsc.es, ferad.zyulkyarov@bsc.es, damian.roca@bsc.es, mario.nemirovsky@bsc.es

Abstract
Next generation data centers will likely be based on the emerging paradigm of disaggregated function-blocks-as-a-unit departing from the current state of mainboard-as-a-unit. Multiple functional blocks or bricks such as compute, memory and peripheral will be spread through the entire system and interconnected together via one or multiple high speed networks. The amount of memory available will be very large distributed among multiple bricks. This new architecture brings various benefits that are desirable in today's data centers such as fine-grained technology upgrade cycles, fine-grained resource allocation, and access to a larger amount of memory and accelerators. An analysis of the impact and benefits of memory disaggregation is presented in this paper. One of the biggest challenges when analyzing these architectures is that memory accesses should be modeled correctly in order to obtain accurate results. However, modeling every memory access would generate a high overhead that can make the simulation unfeasible for real data center applications. A model to represent and analyze memory disaggregation has been designed and a statistics-based queuing-based full system simulator was developed to rapidly and accurately analyze applications performance in disaggregated systems. With a mean error of 10%, simulation results pointed out that the network layers may introduce overheads that degrade applications' performance up to 66%. Initial results also suggest that low memory access bandwidth may degrade up to 20% applications' performance.

Keywords: remote memory, disaggregated computing, simulation, optical networks, datacenters

1 Introduction

Next generation servers in data centers may be based on the emerging paradigm of disaggregated function-blocks-as-a-unit[7]. This new paradigm is departing from the current server architecture where the server mainboard functions as a unit. In a disaggregated architecture, functional blocks (bricks) such as compute, memory and peripheral will be spread through

the entire system and interconnected together through high speed networks. This paradigm promises to bring new advantages to data centers such as: fine-grained technology upgrade cycles, fine-grained resource allocation and access to a larger amount of memory and peripherals.

When researching on new architectures, simulation is a must for exploration and evaluation of new systems and architectures. Simulation of disaggregated memory architectures is complex because it requires knowing the memory access pattern of the applications. Accurate simulators that models every application instruction such as Gem5 [1] could be used to simulate disaggregated memory systems. However, due to the fact that the typical workload in data centers usually operates in large databases, an execution may be very time consuming. In addition, this problem becomes exacerbated as the number of cores used by the application increases. Analytical models may be used for initial designs and evaluations, but similar to other coarse grained approaches, the amount of information and detail they provide is limited. In case a detailed model is constructed, the analytical models often become complex.

Queue theory was used successfully (Zilan et al. [17]) to get the best of both full system simulators and analytical models when modeling multicore processors. Queue model-based techniques allow faster analysis while preserving accuracy and offer a flexible granularity scope to include the most relevant components and techniques that form a modern architecture. Queue model-based simulators are desired to conduct design space exploration and bottleneck identification in a reasonable time frame.

This paper presents an extensive analysis of applications performance in systems with disaggregated memory architecture. It makes the following contributions: a) Design and implementation of a multicore queue-based simulator; b) Profiling and characterization of real datacenter workloads; c) Performance analysis and identification of potential bottlenecks; d) Insights for designing future systems with disaggregated memory.

Experimental results suggest that the accuracy of our simulator ($\sim$ 10% of mean error) is within the acceptable range for driving an exploratory design space analysis. Simulation results are produced in a very short time ($\sim$ 30 secs.). The analysis of the results suggest that network layers and communication protocols involved in accessing the disaggregated memory may introduce non-negligible overhead as high as 66%. Aditionally, memory bandwidth can be another source of bottleneck causing up to 20% performance degradation.

2 Background and Related Work

The concept of disaggregation in computing servers has the intent to break the boundaries of current compute, memory, network and storage components built in a hard, unique and tighly connected unit. The Machine project from HPE calls for reinventing the traditional CPU-centric architecture of nowdays to a new Memory-Driven"Computing with the use of Memristors [2]. Disaggregation of the memory has an important focus of attention. Memory footprint of applications is increasing and the interconnection between the disaggregated memory and processing components[9] adds more latency to the already complicated memory system.

The full-system simulators are good for detailed high-accuracy simulations and validating individual components, but they are cumbersome in dealing with the initial stages of design space exploration due to the lengthy simulation time and substantial development effort involved.

To solve the problem of the simulation execution time, Genbrugge et al. developed a technique called interval simulation [4], which abstracts certain processors components and the simulation is guided by certain events such as cache misses and branch miss-predictions.

Other works focus on trace-driven simulators [11] [5] in an effort to reduce the execution time while maintaining accuracy. This technique is based on obtaining an instruction trace from

a previous execution of an application. However, managing these traces is not easy especially considering that applications with large working sets and long execution times can lead to significant storage problems. As a way to improve the trace-based simulations, Wunderlich et al. make use of representative reduced traces [15] which are capable of capturing and evaluating the behavior of the real programs while executing fewer instructions. However, the complexity of the processor model is the same as with the full system simulators.

Analytical models may be used for initial designs and evaluations, but similar to other coarse-grained approaches, the amount of information and detail they provide is reduced. In case a detailed model is constructed, the analytical models often become complex. To improve the outputs of these models, some researchers have proposed the use of queuing theory to construct models from superscalar [6] and multithreaded processors [13] to multiprocessor systems [17] and data center workloads. Queuing modeling techniques allow faster analysis while preserving accuracy and offer a flexible granularity scope to include the most relevant components and techniques that form a modern architecture. Queuing model based simulators complement to more detailed simulators and simplify the design space exploration as well as bottleneck identification for architects when evaluating how to improve current systems.

3 Disaggregated Architectures: dReDBox Proposal

The dReDBox project [7] focuses on designing a highly modular software-defined architecture for datacentres, where SoC-based microservers, memory modules and accelerators will be placed in separated modular trays interconnected through opto-electronic fabric. dReDBox aims to deliver a full-fledged integrated datacentre-in-a-box prototype to show the benefits of disaggregation in terms of scalability, efficiency, reliability, performance and energy reduction.

Figure 1 introduces the architecture of a disagregated memory system used in the dReDBox project. In this example, two racks interconnected through a reconfigurable Optical Circuit Switch (OCS) are shown. Each rack is housing multiple interconnected bricks. Compute bricks are dedicated to run application code whereas the memory brick is where the main memory is placed. Compute bricks contain FPGAs that are in charge of remote memory addressing. Memory requests are captured and the FPGAs use them to create a packet containing the necessary information to access the specific memory segment required. Once the packet is generated, an network endpoint (Port) is used to transmit the packet passing through a switch (in this case an OCS, but could be an electrical switch as well). Once a memory request arrives to a Memory Brick, it is decoded in the FPGA and the requested memory is returned

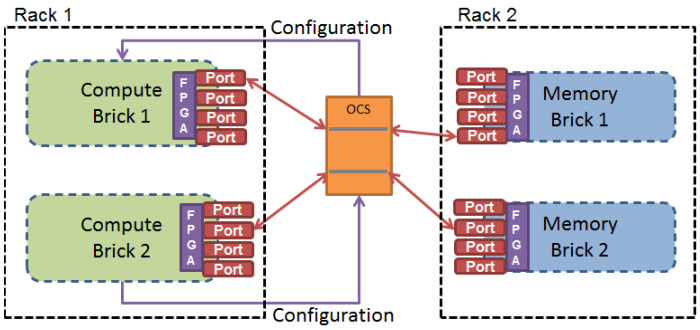

Figure 1: Overview of the disagregated memory architecture in the dReDBox project.

(considering a Load operation).

The packetization process (in the critical path) and the available bandwidth have a major influence in the memory accesses. Current datacenter applications are written considering high memory bandwidth and delays in the order of nanoseconds. However, in disaggregated architectures, the available bandwidth would depend on the number of ports (and port bandwidth) used to transmit requests to a Memory Brick and delays would highly depend on the transmission protocols used. If a Compute Brick is connected to several Memory bricks (accessing a huge amount of memory), there may be not enough ports to provide a bandwidth similar to current memory bandwidths. Taking this facts into account, the rest of the paper focuses on modeling and analyzing memory access and their influence in applications' performance when executing in the dReDBox architecture.

4 Simulation framework for disaggregated processor architectures

Queue models are based on queue structures, message passing, and latency accumulation that allow to model systems. A message is received at the tail of a queue structure, propagated until it reaches the head of the queue and then a delay is added to emulate the amount of time that the action for that particular message or component takes. For example, for the case of an Integer ALU, the queue models the ALU input queue where the message passed represents a particular arithmetic instruction such (add, substract, equals) and the delay added will depend on the amount of time the ALU unit typically takes to execute that arithmetic instruction.

The simulation occurs as a collection of discrete events which in our model is represented as one computational cycle. Execution progress (including instruction generation, propagation, waiting, execution, and retirement as well as resource usage and branch and memory misses) is simulated within the queuing model for every event (i.e every cycle). Total performance is measured as a collection of processed messages per total events, i.e., instructions per cycle. The event driven queuing model we are proposing uses a modular combination of various queue structures, dependency tracking, and probabilistic execution flow to simulate particular systems.

The queue-model based methodology (called iQ [12]) emulates processor components by abstracting the implementation details into modular components composed of queue structures, delay parameters and probabilistic driven message generation and event control.

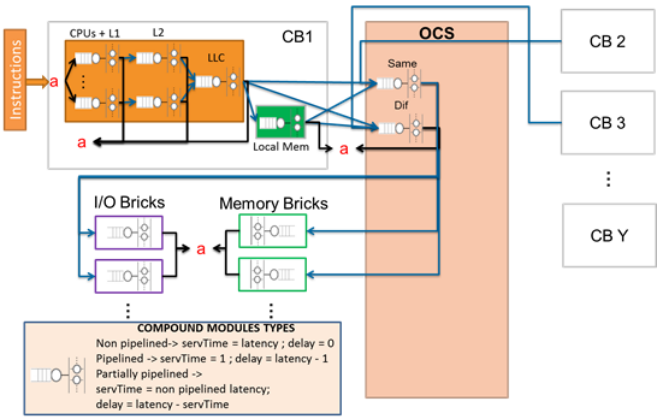

Figure 2: iQ model that represents different components of the dReDBox Infrastructure.

To model the behavior of the dReDBox system, we developed a simulation queue-based model. Figure 2 shows the bricks' interconnection through an optical switch for simplification purposes. In order to represent processors, memories and other components using queuing models, we have implemented a modular queue structure that models different behaviors through a set of variable configurations (the red a letter indicates the beginning of instruction processing). At the left-bottom side of Figure 2 is depicted the module that is used to represent each component. The module is formed by a queue, a server and a delay. The queue length and delays required to process instructions are flexible and configurable. The length parameter is used to model resource contention and availability. The service time (*servTime*) represents the time needed to process an instruction until the following instruction may start to be processed. The instructions total execution time inside a compound module will be the sum of its own *servTime* plus the service time of all previous executed instructions (pipelining). The lower the service time, the higher level of pipeline and vice versa. The delay (*delay*) parameter is used to complement the *servTime* to ensure the appropriate total delay is added to the instruction.

Instructions are generated according to the statistical information collected during the profiling stage and they are introduced at the entry point of the compute bricks (CB) as shown in Figure 2. Then, depending on the probability values the instructions will move from one queue to the other or to the sink (point a in the Figure 2). Instructions move from the different levels of cache and the local memory. In the case that instructions need to access remote bricks (I/O, Memory, etc.), they may need to go through the Optical Circuit Switch (OCS).

The model shown in Figure 2 has been used to design and implement a multicore iQ using the Omnest simulation framework [14]. Dependencies between instructions as well as between pipeline stages or within computation resources (e.g. ALUs, branch predictors, Out-of-Order tracking) are also accounted, in order to increase the accuracy of simulations.

5 Analysis and Evaluation

Since memory accesses in disaggregated systems rely on the interconnection network between compute bricks and memory bricks, here we evaluate how bandwidth and latency would impact applications' performance. We analyze and profile real datacenter applications focusing on Load/Store instructions and memory accesses, since this information is critical to analyze the impact of disaggregated architectures in applications performance. The profiling information is used to feed the simulator and evaluate its confidence, by comparing real results with simulation results. Finally, we present a disaggregated memory access model and an extensive analysis on how bandwidth and packetization times may affects datacenter applications' performance.

5.1 Application Profiling and Simulator Validation

Several data center applications were evaluated in the disaggregated memory architecture. Profiling information about their execution were obtained in a 6-core Intel Xeon CPU X5675 running at 3.07GHz server. Each core has 64KB of L1 cache (32 KB data and 32 KB instruction), 256 KB of L2 cache, and 12 MB of LLC cache shared across cores.

Table 1 summarizes the profiling information for the selected workloads. The workloads which we used are described below:

- FFMPEG [3]: it is a popular video processing application, able to perform various operations on a video file such as decode.

Table 1: Application Profiling Information.

Metrics	FFMPEG	Video Analytics	Network Analytics	NFV
Input Size	2.2 GB	2.2 GB	2 GB	3 GB
Number of Cores/Apps. per Core	4/1	1/1	1/1	1/1
Number of Instructions	6.30E+12	2.98E+11	44,236,862	47,111,048
Cycles	3.68E+12	1.95E+11	57,483,561	65,344,290
Branch (%)	7.97	8.64	19	19.39
Branch Miss(%)	4.33	3.39	3.24	2.76
Load (%)	14.69	31.68	29.99	25.04
Store (%)	4.82	15.40	20.58	18.20
L1 Load Miss (%)	4.50	1.68	4.73	4.12
L1 Store Miss (%)	4.87	1.14	2.02	1.57
L2 Load Miss (%)	16.60	45.78	39.84	39.67
L2 Store Miss (%)	29.24	0.005	5.95	4.05
LLC Load Miss (%)	38.53	46.08	7.74	18.31
LLC Store Miss (%)	23.52	23.21	16.52	24.29
Execution Time (Seconds)	1175	1800	27.01	481

- Video Analytics [8]: this application performs video content analytics and indexing on video files and streams.
- Network Analytics [16]: the DetectPro is a network analytics application that performs two critical tasks for network monitoring: Packet parsing and flow statistics generation.
- Network Function Virtualization (NFV) [10]: this application uses a key server for collaborative encryption functions. The Key Server is a NFV in charge of generating a session Key in a collaborative way with an edge server.

Table 2 depicts the results obtained with the multicore version of the iQ simulator. The real values shown were obtained during the profiling step (Table 1). We validated our simulator using profiling information obtained in systems where the memory is not disaggregated. By contrasting the simulated IPC with the real IPC we are able to determine the error percentage (which ranges between 5.19% and 20.83%). The IPC is used as a performance indicator in all the experiments, and it helps to analyze the impact of disaggregation in the applications.

Table 2: Simulator Validation. Comparison of Real IPC against Simulated IPC.

Applications	Input Size	No. of Cores/Apps. per Core	Simulation Ex. Time (Secs.)	Real IPC	Simulated IPC	Error(%)
FFMPEG	2.2 GB	4/1	29.75	1.71	1.57	-8.29
Video Analytics App.	2.2 GB	1/1	27.77	1.53	1.44	5.88
Network Analytics App.	2 GB	1/1	20.18	0.77	0.81	5.19
NFV App.	3GB	1/1	31.90	0.72	0.87	20.83

5.2 Memory Access Analysis

Figure 3 presents the model that we used to simulate how memory bricks are accessed from compute bricks. *MNwTimeIn* represents the time that takes from preparing the request and transmitting it over the network. *MQueueTimeIn* represents the time that the request spent

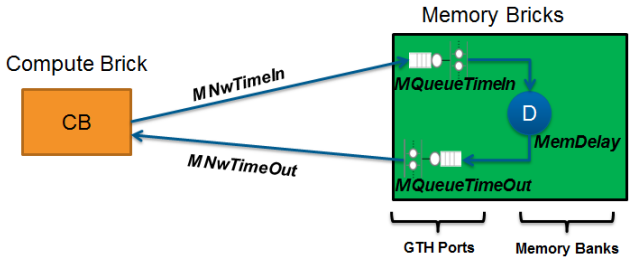

Figure 3: Model that represents accesses between Compute Bricks and Memory Bricks.

before starting to be serviced by the specific memory bank. *MemDelay* is the time taken to move the requested data to the memory bank buffer. *MQueueTimeOut* represents the time spent in the output queue of the memory brick. Finally, *MNwTimeOut* is the time spent in preparing and transmitting the output packet to the compute brick. In order to model the memory access time, we have used the described model which is represented in the next equations:

$$TotalMemTime = MemServiceTime + MemDelay \quad (1)$$
$$MemServiceTime = MemNwTime + MemQueueTime \quad (2)$$
$$MemNwTime = MNwTimeIn + MNwTimeOut \quad (3)$$
$$MemQueueTime = MQueueTimeIn + MQueueTimeOut \quad (4)$$
$$MNwTimeIn = 2 \times pkPrep + NwDelay + PacketSize/NwBandwidth \quad (5)$$
$$MNwTimeOut = 2 \times pkPrep + NwDelay + PacketSize/NwBandwidth \quad (6)$$

The *MemServiceTime* includes the times spent in the network and in the queues. *MNwTimeIn* and *MNwTimeOut* include the time spent in preparing/processing the packet in the compute and memory brick (2 * *pkPrep*) and the time spent in transmitting the packet over the network. *MemQueueTime* is divided in *MQueueTimeIn* and *MQueueTimeOut* since the time in the queues may vary when arriving to the memory brick and when leaving it.

The presented equations allow us to carry out sensibility analysis and design exploration analysis by modifying different parameters taking into account current datacenters infrastructures as baselines and compare them with expected values in the disaggregated architectures.

Below we present an example of the model using default values found in todays datacenter machines (no disaggregated memory architectures). We are modeling a Load operation, and assuming, for simplicity, that cache line matches request size as well (64B). We consider the next values for the equations:*CPU Clock=2.6 Ghz.; MemDelay=20 cycles; pkPrep=10 cycles; MQueueTimeIn = MQueueTimeOut=15 cycles; PacketSize=64 Bytes (Cache Line Size); NwBandwidth=102.4 Gbps (DDR3 bandwidth);* and *NwDelay=26 cycles (10 ns.).*

$$\begin{aligned} MemNwTime = (2 \times 10 + 26 + \frac{64B}{102.4Gbps} \times 2.6GHz) \\ +(2 \times 10 + 26 + \frac{64B}{102.4Gbps} \times 2.6GHz) \end{aligned} \quad (7)$$

$$MemNwTime = 118cycles \quad (8)$$
$$MemServiceTime = 118 + (15 + 15) = 148cycles \quad (9)$$
$$TotalMemTime = 148 + 20 = 168cycles \quad (10)$$

The *NwBandwidth* and *pkPrep* time are used in the queues that represent the network ports (Figure 3). The service time (*servTime*) of the queue model (Section 4) represents the *pkPrep* time and the *delay* represents the delay proportional to the bandwidth plus the memory bank delay (*MemDelay*).

When applying our analytical model using dReDBox expected values, the main value changes would be in *MemNwTime* due to the bandwidth options available in the system and the *pkPrep* times. The *MemQueueTime* can be also affected when several requests from different compute bricks are going to the same memory brick, and even the same memory bank. Table 3 depicts how the variation in certain values affects the *MemServiceTime* and in consequence the *TotalMemTime*. We increase the *NwBandwidth* and the *pkPrep*. *TotalMemTime* is presented in cycles in the inner cells. The highlighted cells have been selected in order to perform simulations and evaluate the impact in the performance of applications.

Table 3: Projection of *TotalMemTime* increasing *NwBandwidth* and *pkPrep*.

NwBandwidth (Gbps)	**16**	**32**	**48**	**64**	**80**	**96**	**112**	**128**
pkPrep (cycles)								
15	308.40	225.20	197.47	183.60	175.28	169.73	165.77	162.80
50	448.40	365.20	337.47	323.60	315.28	309.73	305.77	302.80
100	648.40	565.20	537.47	523.60	515.28	509.73	505.77	502.80
150	848.40	765.20	737.47	723.60	715.28	709.73	705.77	702.80
200	1048.40	965.20	937.47	923.60	915.28	909.73	905.77	902.80
250	1248.40	1165.20	1137.47	1123.60	1115.28	1109.73	1105.77	1102.80
300	1448.40	1365.20	1337.47	1323.60	1315.28	1309.73	1305.77	1302.80

Table 4 presents results of the 4 applications running each one in 1 core. FFMPEG application was executed using 4 cores, while the others were run using 1 core. We analyzed how *NwBandwidth* and *pkPrep* variation affects the IPC. In a dReDBox-like system the main values that may vary are bandwidth and the packet preparation time.

The analysis presented above takes into account worst case scenarios, so we can see what can be the lower performance expected. It is important to highlight that the number of network ports used are going to limit the memory access. For example, if 1% of the total instructions go to main memory and a memory access takes 84ns (using a single endpoint of 16Gbps, assuming roundtrip packets of 512 bits and *NwDelay* of 10ns), then, 100 instructions will be executed in no less than 84ns, just considering the memory access. Then, it can be important to set more than one port in order to avoid a high impact in performance.

However, the biggest source of overheads for disaggregated architectures is probably going to be the *pkPrep* time. If requests are queued until other packets are processed, and the *pkPrep* time is around 300 cycles, the overheads could be as high as 66.88%, as observed with the FFMPEG application. Then, it is critical to maintain a low *pkPrep* time, or to have several modules that can work in parallel to process packets.

6 Conclusions

Taking into account that disaggregated architectures are an emerging paradigm that most likely will replace current mainboard-as-a-unit systems, in this paper we have presented an extensive analysis and evaluation of the disaggregated architecture proposed in the dReDBox project. These architectures will bring major benefits such as fine-grained technology upgrade cycles, fine-grained resource allocation, and access to a larger amount of memory. However,

Table 4: Application results with different *NwBandwidth* and *pkPrep* time.

App./Input Size/No of Cores	NwBand-Width (Gbps)	pkPrep (cycles)	Normal Datacenter Simulated IPC Real IPC	Simulated IPC Disaggregated Mem	Overhead (%)
FFMPEG /2.2GB/4	16	15	1.57/1.71	1.34	14.65
	16	15	1.57/1.71	0.80	49.04
	16	300	1.57/1.71	0.52	66.88
	64	15	1.57/1.71	1.55	1.27
	64	150	1.57/1.71	0.80	49.04
	64	300	1.57/1.71	0.58	63.06
	128	15	1.57/1.71	1.57	0.00
	128	150	1.57/1.71	0.85	45.86
	128	300	1.57/1.71	0.64	59.24
Video Analytics /2.2GB/1	16	15	1.44/1.53	1.14	20.83
	16	150	1.44/1.53	0.64	55.56
	16	300	1.44/1.53	0.52	63.89
	64	15	1.44/1.53	1.34	6.94
	64	150	1.44/1.53	0.76	47.22
	64	300	1.44/1.53	0.52	63.89
	128	15	1.44/1.53	1.36	5.56
	128	150	1.44/1.53	0.74	48.61
	128	300	1.44/1.53	0.48	66.67
Network Analytics /2 GB/1	16	15	0.81/0.77	0.8	1.23
	16	150	0.81/0.77	0.67	17.28
	16	300	0.81/0.77	0.58	28.40
	64	15	0.81/0.77	0.81	0.00
	64	150	0.81/0.77	0.70	13.58
	64	300	0.81/0.77	0.62	23.46
	128	15	0.81/0.77	0.81	0.00
	128	150	0.81/0.77	0.70	13.58
	128	300	0.81/0.77	0.58	28.40
NFV /3 GB/1	16	15	0.87/0.72	0.83	4.60
	16	150	0.87/0.72	0.63	27.59
	16	300	0.87/0.72	0.5	42.53
	64	15	0.87/0.72	0.87	0.00
	64	150	0.87/0.72	0.68	21.84
	64	300	0.87/0.72	0.57	34.48
	128	15	0.87/0.72	0.89	-2.30
	128	150	0.87/0.72	0.66	24.14
	128	300	0.87/0.72	0.52	40.23

there are many challenges regarding low-latency memory access and applications performance implications.

Remote memory addressing would require packetization protocols that would be in the critical path of memory access, and also packets may be transmitted over a network with different bandwidths. The implication of these variables should be explored in order to foresee applications performance and to drive software/hardware decisions when building disaggregated systems. In this work we have introduced the iQ multicore simulator that in the order of seconds can accurately predict the performance of applications when migrating them from current datacenters to disaggregated systems. With a mean error of 10%, simulation results demonstrate that high packetization times may degrade applications performance up to 66% and low memory access bandwidths can degrade up to 20% performance.

7 Acknowledgements

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 687632 (dReDBox project) and TIN2015-65316-P - Computacion de Altas Prestaciones VII.

References

[1] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[2] Rachel Courtland. Can hpe's" the machine" deliver? *IEEE Spectrum*, 53(1):34–35, 2016.

[3] FFmpeg Developers. FFMPEG tool (version be1d324). `http://ffmpeg.org/`, 2016.

[4] D. Genbrugge, S. Eyerman, and L. Eeckhout. Interval simulation: Raising the level of abstraction in architectural simulation. In *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12, Jan 2010.

[5] Hugo Meyer, Jose Carlos Sancho, Milica Mrdakovic, Wang Miao and Nicola Calabretta. Optical packet switching in HPC. An analysis of applications performance. *Future Generation Computer Systems*, 2017.

[6] Tejas S Karkhanis and James E Smith. A first-order superscalar processor model. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pages 338–349. IEEE, 2004.

[7] K. Katrinis, D. Syrivelis, D. Pnevmatikatos, G. Zervas, D. Theodoropoulos, I. Koutsopoulos, K. Hasharoni, D. Raho, C. Pinto, F. Espina, S. Lopez-Buedo, Q. Chen, M. Nemirovsky, D. Roca, H. Klos, and T. Berends. Rack-scale disaggregated cloud data centers: The dredbox project vision. In *2016 Design, Automation Test in Europe Conference Exhibition*, pages 690–695, March 2016.

[8] Kinesense. Kinesense kes. `http://www.kinesense-vca.com/product/`, 2016.

[9] Kevin Lim, Jichuan Chang, Trevor Mudge, Parthasarathy Ranganathan, Steven K Reinhardt, and Thomas F Wenisch. Disaggregated memory for expansion and sharing in blade servers. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 267–278. ACM, 2009.

[10] Javier Gusano Martnez. Key server that implements the tls session key interface (ski). `http://github.com/mami-project/KeyServer`, 2016.

[11] A. Rico, A. Duran, F. Cabarcas, Y. Etsion, A. Ramirez, and M. Valero. Trace-driven simulation of multithreaded applications. In *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, pages 87–96, April 2011.

[12] Damian Roca. High level queuing architecture model for high-end processors. In *Master Thesis.*, 2014.

[13] Thin-Fong Tsuei and Wayne Yamamoto. Queuing simulation model for multiprocessor systems. *Computer*, 36(2):58–64, 2003.

[14] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[15] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling. *SIGARCH Comput. Archit. News*, 31(2):84–97, May 2003.

[16] Jose Fernando Zazo, Marco Forconesi, Sergio López-Buedo, Gustavo Sutter, and Javier Aracil. TNT10G: A high-accuracy 10 gbe traffic player and recorder for multi-terabyte traces. In *2014 International Conference on ReConFigurable Computing and FPGAs, ReConFig14, Cancun, Mexico, December 8-10, 2014*, pages 1–6, 2014.

[17] R. Zilan, J. Verdu, J. Garcia, M. Nemirovsky, R. A. Milito, and M. Valero. An abstraction methodology for the evaluation of multi-core multi-threaded architectures. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 478–481, July 2011.