NAM: Network Attached Memory

Juri Schmidt
Computer Architecture Group
University of Heidelberg
Mannheim, Germany
juri.schmidt@ziti.uni-heidelberg.de

***Abstract*—Power and energy increasingly move into focus for all types of computer systems, especially in High Performance Computing. The disparity in processor versus memory speed-up has been growing and still continues to grow at a moderate rate. The effects that can be observed are well known as the *memory wall*. Although a processor has resources to perform calculations it must wait for the data to become available. The idea of Processing in Memory (PIM), as one solution to overcome this issue, has been around for quite a long time. Only recently it has become technologically and economically available by the ability to create heterogeneous die stacks, i.e. DRAM and CMOS logic layers can co-exist in a single chip.**

As one approach we have developed the Network Attached Memory (NAM). The NAM is a research vehicle to explore the possibilities of PIM in an affordable way. The first prototype is implemented as an FPGA that is directly connected to a Hybrid Memory Cube (HMC). Since these are still two discrete devices we call it Near Data Computing (NDC). The NAM provides interfaces to connect to the EXTOLL high performance interconnection network. It is therefore a globally accessible, shared storage and compute node. As a first use-case the NAM serves as a dedicated device to generate the checkpoint parity for resiliency applications within the European funded DEEP-ER project. Since the NAM operates on data within the network domain we propose the term Processing In Network (PIN).

***Index Terms*—Network Attached Memory, NAM, Hybrid Memory Cube, HMC , openHMC, Processing In Memory, PIM, Near Data Computing, NDC, Processing In Network, PIN**

I. Introduction

Power and energy increasingly move into focus as technology trends dictate hard constraints on power and energy consumption. Not only the well known 'memory wall' is present (Fig. 1), but also recent work (e.g. [8]) observed that already today more energy is spent on data movement than for computation, and projections show that this trend will intensify in the future. As a result, computation overhead will be negligible in terms of energy while the overall energy footprint is dominated by data movement. In addition, energy consumption for data movement is highly dependent on the distance covered(see Fig. 2). In particular, one can observe a large increase when moving from on-chip to off-chip connectivity (basically a transition from R-C to a transmission line). Consequently developers recently started to investigate in Processing in Memory (PIM), i.e. bringing computation closer to the memory. Various early explorations have already been proposed, e.g. [2],[4],[1],[5].

Similarly, we propose a new device: the **Network Attached Memory (NAM)**. It is a PCI-Express form factor card that is

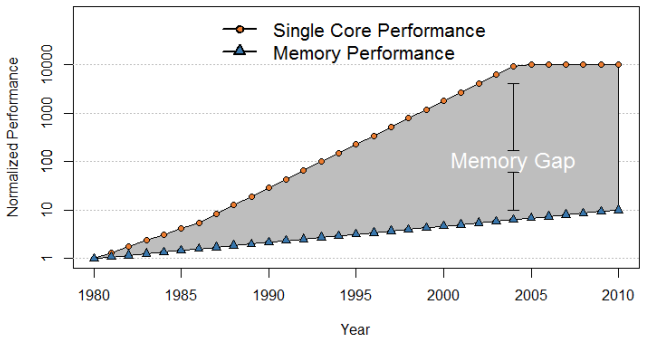

Fig. 1: The memory gap: Historical evolution of the processor to memory performance, also known as 'memory wall'

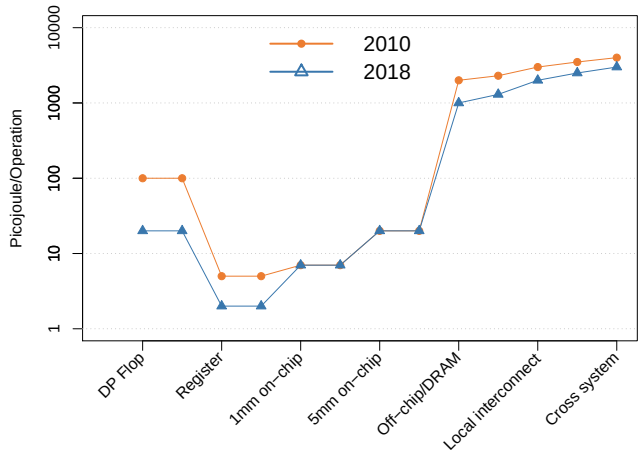

Fig. 2: Energy required to transport data plotted over several distances

equipped with an FPGA and Microns Hybrid Memory Cube (HMC), a 3D stacked DRAM device with integrated memory controllers and support for many outstanding requests. In fact, the NAM represents a class of Near Data Computing (NDC) nodes. The actual processing elements are not part of a heterogeneous die stack along with the memory as it is the case for PIM. Instead, the NAM provides a research vehicle to prototype and evaluate the advantages of processing data as close to the memory as possible. It is obvious that any processing element could also be implemented in the memory stack which results in a true PIM device.

The first project to use the NAM is the European funded

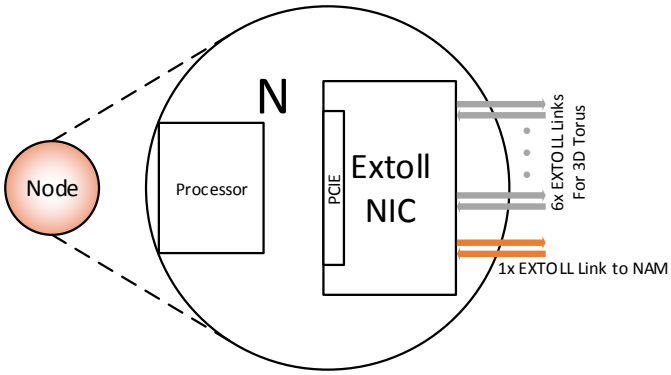

Fig. 3: A (co-)processor is connected to the network via an EXTOLL Tourmalet Network Interface Controller. The NAM can be attached to any of the available links

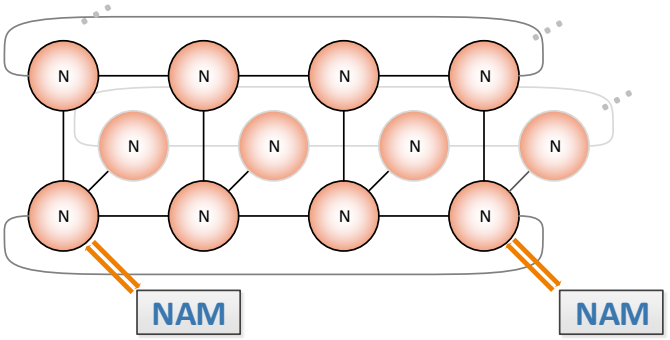

Fig. 4: Schematic representation of the DEEP-ER system with two NAMs connected

Dynamical Exascale Entry Platform - Extended Reach (DEEP-ER) project[1]. Here, the NAM is directly connected to the EXTOLL high performance interconnection network and serves as a compute node for global operations in general and/or as a dedicated device to perform an N+1 XOR parity checkpoint-restart mechanism. EXTOLL is a direct-connected, switch-less interconnect fabric and every Network Interface Controller (NIC) provides 6 links to create a 3D torus. A seventh link (or any of the others) can be used to attach a NAM, preserving scalability for any system size. Refer to Fig. 3 and Fig. 4 for abstract views of a node in the DEEP-ER scope and a schematic draw of the DEEP-ER system respectively.

A. Applications

The possible range of applications for the NAM can be generalized to any kind of global (collective) operations such as scatter or gather. In addition it can be used as a general purpose shared memory or a streaming processor. The first particular application, however, is a checkpoint/restart mechanism to support the resiliency features of DEEP-ER. Here, all nodes generate an application checkpoint from time to time. The NAM is able to retrieve these checkpoints and generate a corresponding X-OR parity. With this parity the system can be recovered upon a single node failure. The checkpoint/restart mechanism is described in more detail in Section III.

[1]The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013)

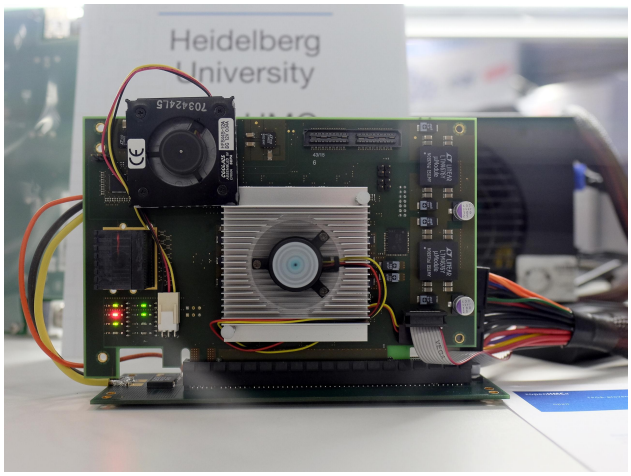

Fig. 5: The NAM Hardware Aspin v2

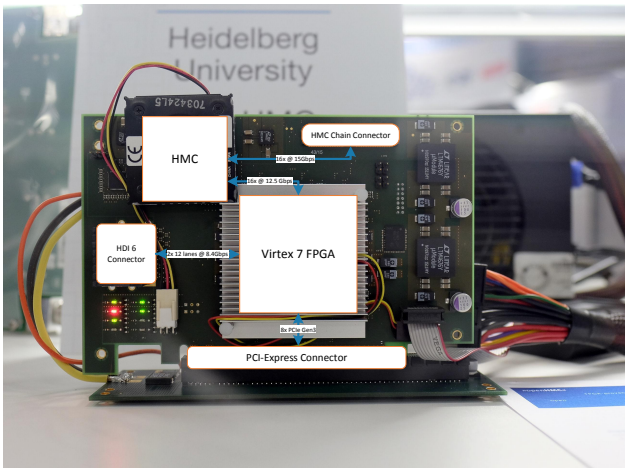

Fig. 6: Aspin v2 Component and Interface Block Diagram

B. Bandwidth and Capacity

The NAM provides two main interfaces: two EXTOLL links and one HMC link. Currently, both 12x EXTOLL links run at 8.4Gbps for a total of 48GByte/s bidirectional bandwidth. The HMC link provides 16x at 10Gbps (40GByte/s). The current capacity of the on-board HMC is 2GByte. Although the HMC capacity is not expected to grow for the current device generation, additional HMCs can be daisy-chained using an add-on card to further increase the capacity. The next generation HMC, however, will provide more capacity.

II. The NAM Hardware

The NAM hardware 'Aspin v2' integrates a Xilinx Virtex7 FPGA and an HMC on a single PCB. The FPGA can be accessed from PCI-Express (16 lanes) or two HDI-6 12x connectors over the EXTOLL network. Another connector on the top of the board is connected to the HMC and can be used to extend memory capacity by adding additional HMCs or to create a variety of FPGA-HMC topologies. The FPGA implements the actual NAM *intelligence*. Fig. 5 shows the

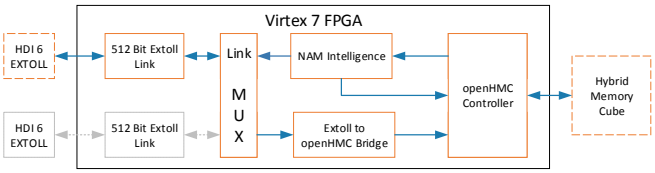

Fig. 7: NAM Design Block Diagram

fully functional NAM prototype and Fig. 6 highlights the individual components on the PCB. Note the additional HMC chain connector on the top of the card which can connect another NAM card or any other HMC based add-on cards.

A. Implementation Details

The structure of the FPGA design mainly consists of four building blocks as depicted in Fig. 7. We use the publicly available open-source HMC host controller openHMC [3] [6] which is our own development. The NAM requires an FPGA specific implementation of an EXTOLL link, mainly due to the limited clocking frequency in FPGAs compared to ASICs. A protocol bridge connects these two building blocks. Lastly, the NAM intelligence implements the application specific processing elements. Overall, the current implementation uses about 50% of the FPGA resources which leaves enough space for additional processing elements.

III. Improve Resiliency with Parity Checkpoints

One of the objectives in the DEEP-ER project is to speed-up and reduce overhead of checkpoint/restart mechanisms. As one approach the NAM serves as a dedicated resource to calculate and store the parity of checkpoints, also known as *N+1* parity. Fig. 8 and Fig. 9 illustrate the process of writing checkpoints and generating the parity, and the restart process respectively.

A. NAM Software

A software stack, the *libNAM*, closely derived from the EXTOLL RMA engine has been developed. The EXTOLL RMA is a high-throughput engine that communicates using simple put and get operations. libNAM only slightly modifies existing function calls and extends *libRMA* by application specific functionality such as checkpoint writing and reading. This approach ensures that application developers do not need to entirely redesign existing code but instead add a few new or replace some of the existing function calls. In addition a dedicated NAM manager (NM) provides the required configuration commands to allocate address space and administrate access rights to participating processes. It is also responsible to properly configure the NAM for any checkpoint/restart activities and provides remote control of the entire FPGA design.

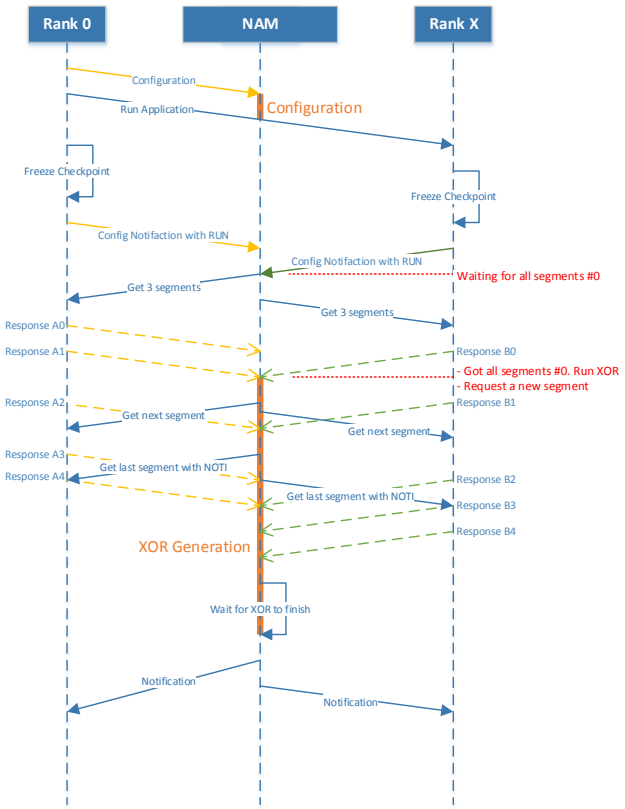

Fig. 8: Generate a X-OR checkpoint

1) Any rank configures the NAM. Configuration can also be performed at run-time
2) When a rank has finished its checkpoint it notifies the NAM
3) The NAM retrieves data in a sliding-window approach
4) When the XOR on all segments is finished, notifications are sent to the participating ranks

IV. NAM Status

All hardware components have been completed and verified. The operation of the openHMC host controller has been successfully demonstrated throughout the last years. Reading and writing to the NAM / the HMC using libNAM has been successfully tested. Yet outstanding are extensive tests for the checkpoint/restart implementation. This applies to hardware and software likewise. Overall it is expected that the NAM along with the developed software stack will provide a competitive alternative for current high-performance checkpoint/restart applications by the end of the PhD studies.

V. NAM Outlook

Although the checkpoint/restart application support is mostly implemented, performance measurements are yet outstanding. It is desirable but not mandatory to at least implement additional types of operations in the scope of the PhD studies. The probably most criticized characteristic of the NAM is the very limited capacity. It will be necessary to investigate in other memory technologies and to evaluate whether a hybrid-memory hierarchy approach is possible, e.g. HMC plus high-capacity, non-volatile memory such as NAND

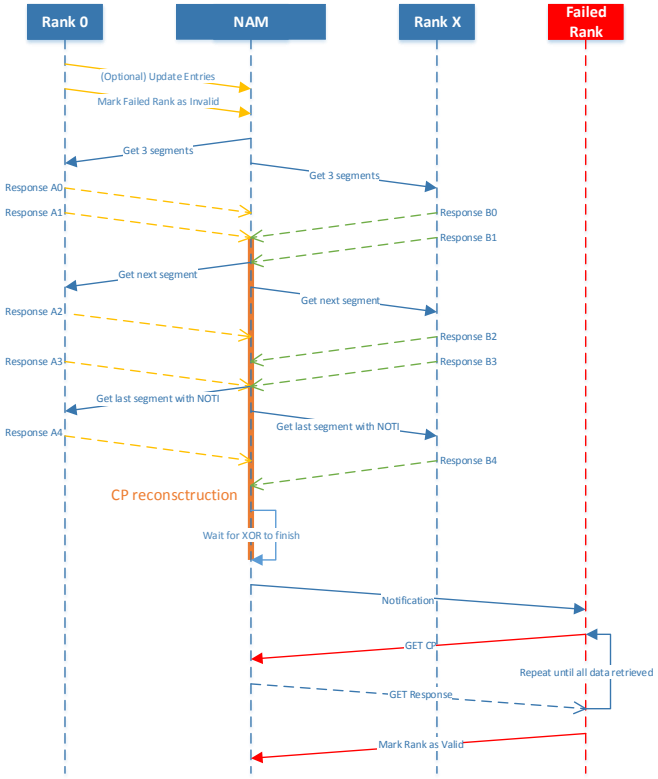

Fig. 9: Restore Checkpoint

1) Upon a rank failure the NAM must be notified
2) The NAM retrieves data from all remaining ranks, replacing the failed rank with the XOR parity stored in the HMC
3) The resulting data is the missing checkpoint. The failed rank is notified
4) The failed rank can now read its most recent checkpoint from the NAM

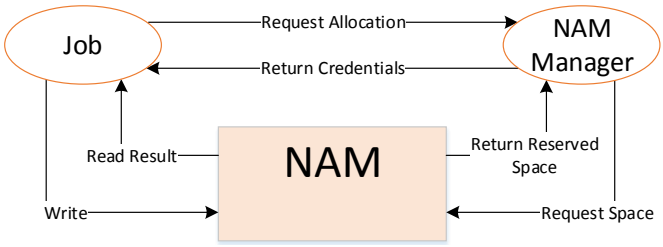

Fig. 10: NAM Manager: Address Space Allocation

or 3DXpoint. For the latter one, however, it remains unclear if the technology will become available for academic research.

VI. Publications

A first paper that described the architecture and implementation of openHMC has been presented at the 2015 International Conference on Reconfigurable Computing and FPGAs[6]. A second paper, which for the first time evaluates the key characteristics of the HMC, was presented at the 2016 International Symposium on Memory Systems in Washington D.C. in October 2016[7]. It is already planned to submit another paper which compares a checkpoint/restart application executed on the NAM to the Scalable Checkpoint Restart library currently used in the DEEP-ER project.
Additional conference activities include live demos of the openHMC controller and the NAM at the International Supercomputing Conference (ISC-HPC) 2015 and 2016 as well as SC15. Furthermore we hosted a booth at the Emerging Technologies track at SC16 and presented first NAM results at the PhD Forum at ISC-HPC 2016.

VII. About the Author

Juri Schmidt is a 3rd year PhD candidate at the Computer Architecture Group at the University of Heidelberg in Germany. He received his masters degree in Computer Engineering in 2013. Juri started his PhD studies with the development of the openHMC host controller and he is also part of the DEEP-ER project to contribute to the Exascale research using emerging memory technologies and advanced memory hierarchies.

References

[1] R. Balasubramonian et al. “Near-Data Processing: Insights from a MICRO-46 Workshop”. In: *IEEE Micro* 34.4 (July 2014), pp. 36–42. ISSN: 0272-1732. DOI: 10.1109/MM.2014.55.

[2] B. Black et al. “Die Stacking (3D) Microarchitecture”. In: *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*. Dec. 2006, pp. 469–479. DOI: 10.1109/MICRO.2006.18.

[3] University of Heidelberg Computer Architecture Group. *openHMC Home*. Apr. 2015. URL: www.uni-heidelberg.de/openhmc.

[4] G. H. Loh, Y. Xie, and B. Black. “Processor Design in 3D Die-Stacking Technologies”. In: *IEEE Micro* 27.3 (May 2007), pp. 31–48. ISSN: 0272-1732. DOI: 10.1109/MM.2007.59.

[5] Lifeng Nai and Hyesoon Kim. “Instruction Offloading with HMC 2.0 Standard: A Case Study for Graph Traversals”. In: *Proceedings of the 2015 International Symposium on Memory Systems*. MEMSYS ’15. Washington DC, DC, USA: ACM, 2015, pp. 258–261. ISBN: 978-1-4503-3604-8. DOI: 10.1145/2818950.2818982. URL: http://doi.acm.org/10.1145/2818950.2818982.

[6] J. Schmidt and U. Bruening. “openHMC - a configurable open-source hybrid memory cube controller”. In: *ReConFigurable Computing and FPGAs (ReConFig), 2015 International Conference on*. Dec. 2015, pp. 1–6. DOI: 10.1109/ReConFig.2015.7393331.

[7] J. Schmidt, H. Froening, and U. Bruening. “Exploring Time and Energy for Complex Accesses to a Hybrid Memory Cube”. In: *Memory Systems, 2016 International Symposium on*. Oct. 2016. DOI: 10.1145/2989081.2989099.

[8] John Shalf, Sudip Dosanjh, and John Morrison. “Exascale computing technology challenges”. In: *High Performance Computing for Computational Science–VECPAR 2010*. Springer, 2010, pp. 1–25.