Emerging Memory Technologies for Improved Energy Efficiency

Martin Wenzel
Department of Computer Engineering
University of Heidelberg
Germany, 68131 Mannheim

***Abstract*—Power consumption is the dominating constraint for processor design, and analyses and projections show that data movements outshine computations in terms of energy consumption. In addition, multi- and many-core architectures are dramatically limited by the memory wall, which makes an increasing amount of applications memory-bound. Possible solutions to this problem are near-memory architectures and processing-in-memory. Both gear to overcome the memory bandwidth bottleneck and reduce the physical distance for data movements. This paper will provide an overview of current related work in this area, and which prospects come with such solutions.**

***Index Terms*—Memory, Memory Wall, Processing in Memory, PIM, Hybrid Memory Cube, HMC, Offloading, Tesseract**

I. Introduction

In the last decades it was sufficient to speed up the processors to archive performance improvements. Current processors are highly energy efficient and deliver high performances. But nowadays the performance of a computer is not only limited by the processor performance. For more and more workloads the memory bandwidth is the limiting factor. This means the time it takes to get a value from memory gets significant compared to the time it takes to process the element. This is called memory wall. For further performance increases it is necessary to overcome this memory wall.

The memory bandwidth in a computer is limited by following points:

- package pinout is limited
- Signal frequency is limited by distance
- Interface power budged is limited

To increase the bandwidth between two chips either the Signal frequency or the link width can be increased. Unfortunately the amount of balls underneath a package is limited to less than 8000 Pins. In advance to this, the distance between a memory module and a processor is typically between 3 and 7cm. This distance has a significant impact on the reachable frequency of the signal between memory and the processor.

To sum this up, the power needed by the interface is strongly related to the interface width and even more to the interface frequency. In fact a floating point operation in a current processor consumes up to 10 times less energy than a memory access (Figure 1). Memory interfaces became a main power consumer.

Obviously the solution to bring down the memory wall is to cut down the distance between memory and processor. The shortest distance between memory and processor is Processing-in-Memory (PIM) which is proposed as highly efficient since the late 90s [1].

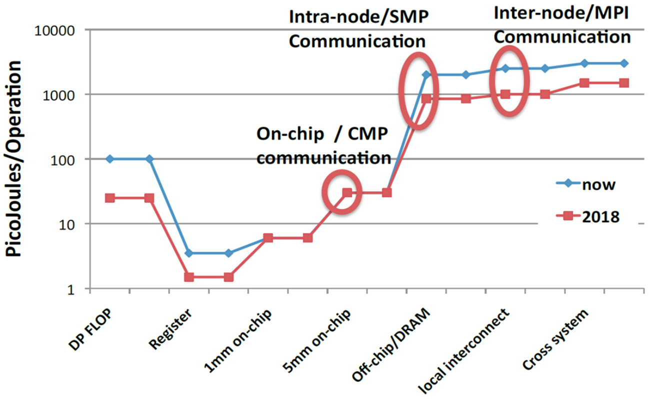


Fig. 1. Sources of power consumption [2]

This paper focuses on PIM to overcome the memory wall. The discussed papers use the Hybrid Memory Cube (HMC) as starting point for offloading techniques. To give a short background 3D DIE stacking and the HMC are introduced in section II and section III. In section IV PIM is discussed. Section V gives a short conclusion to PIM.

II. 3D DIE Stacking

The process of placing more than one die on a package is called die stacking. It is not necessary that the dies are produced in the same process or have the same dimensions. For example logic dies (blue) and memory dies (green) can be stacked on the same package as shown in Figure 2.

Every die is connected with μbumps to the package. These are up to 100 times smaller as the balls used to solder the package to the pcb [3] [4]. Soldering more than one die next to each other is called 2.5D stacking. 2.5D permits using these smaller bumps to realize wide interfaces. Stacking also reduces the signal length between two dies from about 5 to 7 cm to less than 1 cm. Currently stacked memory is available in form of HBM, Wide I/O and HMC [5]. 3D die Stacking is a special form of die stacking. Under normal circumstances a die has only μbumps on top of it. Through Silicon Vias (TSV) are used to add μbumps to the bottom of the die. This allows stacking multiple dies on top of each other. 3D stacking allows even

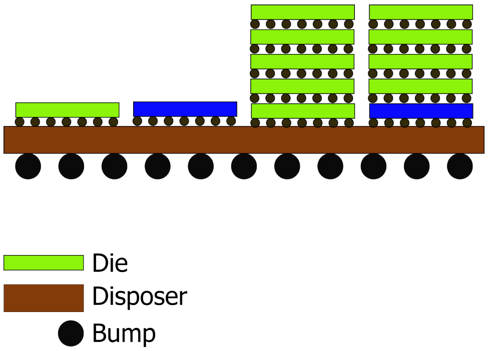


Fig. 2. 2.5D and 3D IC Stacking on a single Package

wider interface and extreme short signal lengths of about 20 to 100 μm [6].

But adding a die to a 3D stack also increases the power density. In addition, stacking dies on top of each other adds thermal resistance which leads to higher temperatures within the stack. Thus, the power wall is lower and more challenging than without stacking [6].

III. Hybrid Memory Cube HMC

The Hybrid Memory Cube is a 3D stacked memory architecture [7]. several layers of DRAM are stacked on top of a logic die (Figure 3). The memory cube is prepared to be stacked next to a processor die (2.5D stacking). A memory cube can hold up to 8 GB of data. The HMC is vertically parted in 32 such called vaults. Each vault has its own vault controller. Inside of this vault controller there is a DRAM controller and an additional processing unit for atomic operations implemented. Following atomic operations are allowed:

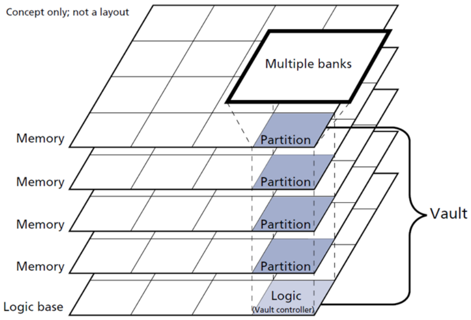


Fig. 3. HMC Memory Cube [7]

- Integer arithmetic
- Compare and Swap
- Bitwise swap / write
- Boolean operations

Each vault controller has a packet based interconnect.

The HMC inherits a packet based crossbar switch as shown in Figure 4. The HMC provides up to eight independent serial links with a link bandwidth of 20 GB/s per direction.

Each link can be used either to interconnect with a host or with a second HMC. The host interface bandwidth can reach up to 160 GB/s per direction.

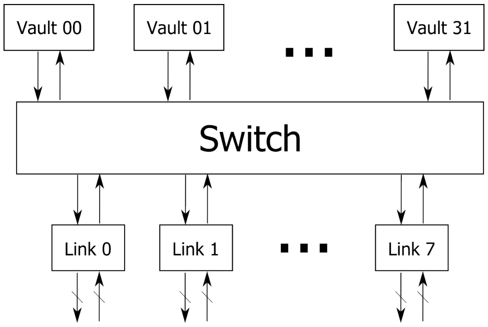


Fig. 4. HMC Network Interconnect [7]

IV. Processing-in-Memory

Data processing inside the memory itself is known as Processing-in-Memory (PIM). Since the distance between memory and processing units is minimized to several μm the internal memory bandwidth could expect to be maximal. Nevertheless, the host memory bandwidth as shown in Figure 5 is still limited by the known problems. PIM is supposed to greatly increase the efficiency of the off-chip interface bandwidth usage. This result in higher performance and also in higher energy efficiency. Adding additional PIM modules will not only increase the amount of Memory but also deliver a speedup.

A. Demonstration Workload

To demonstrate the full potential of PIM a workload which is heavily limited by the memory wall is necessary.

Graph processing is such a workload. A graph consists of several nodes. Relations between these nodes are represented via links as shown in Figure 6. Each node inherits only a small amount of data. For Example a social graph might store a name and a telephone number in each node. In general the processing amount for each node is very low. Often checking if a specific node was processed yet is the most time consuming part. Nodes which are connected via links are distributed through the whole memory. Only a low amount of locality can be found in most graphs.

Some frequently used algorithms for graph computing are:

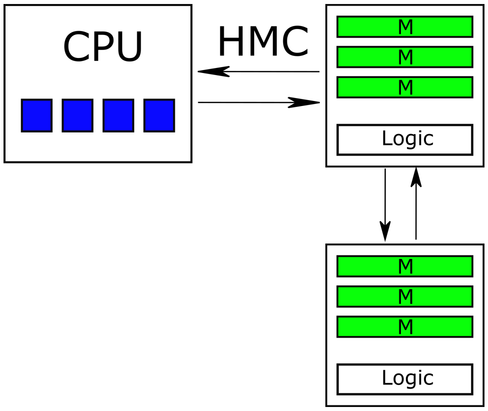


Fig. 5. HMC Interface

- Page Rank
- Shortest Path
- Graph Search

To sum this up, graph processing is a workload which is dominated by low locality and low compute intensity. Both examples focus on graph processing as workload.

B. *Instruction Offloading*

Often most computational operations are used to determine if a node was already processed. A specific node might be loaded from memory multiple times. Thus, the memory interface is used very ineffective. Even worse modern processors load cache lines to exploit locality. A common cache line has a length of 64 Bytes and inherits multiple data values. Even if a node was not processed in before, loading a complete cache line to change a single value wastes memory bandwidth.

Lifeng Nai and Hyesoon Kim use the atomic Compare and Swap (CAS) Operations available in the HMC to offload instructions to the HMC [7].

In fact they optimize the breadth-first search which is part of many graph algorithms.

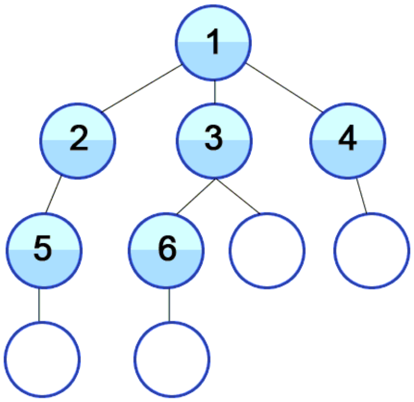
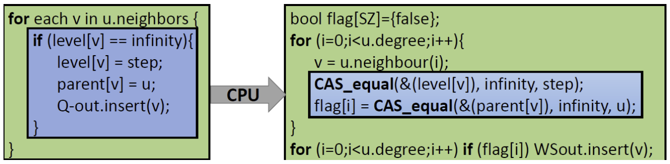


Fig. 6. Breadth-first Search [8]

A graph search tries to find a specific node. The breadth first search starts at a random node called root node. As shown in Figure 6, the algorithm processes the nodes directly related to the root node (distance 0) first. If all directly related nodes are processed the algorithm checks all nodes with a distance of 1.



```
for each v in u.neighbors {
    if (level[v] == infinity){
        level[v] = step;
        parent[v] = u;
        Q-out.insert(v);
    }
}
```

```
bool flag[SZ]={false};
for (i=0;i<u.degree;i++){
    v = u.neighbour(i);
    CAS_equal(&(level[v]), infinity, step);
    flag[i] = CAS_equal(&(parent[v]), infinity, u);
}
for (i=0;i<u.degree;i++) if (flag[i]) WSout.insert(v);
```

Fig. 7. Breadth-first search algorithm conversion to CAS operations [9]

As shown in Figure 7, the check part of a node can be converted in two CAS operations. These CAS operations can be processed on the HMC. The maximum bandwidth reduction will be reached if the cache hit rate (graph property hit rate) reaches zero and each node has a high amount of links to other nodes. As shown in Figure 8 the necessary bandwidth for the conventional algorithm descents if the hit rate rises. The optimized approach shows no benefit of a high cache hit rate. Since graphs in general have low locality the hit rate can be assumed as very low.

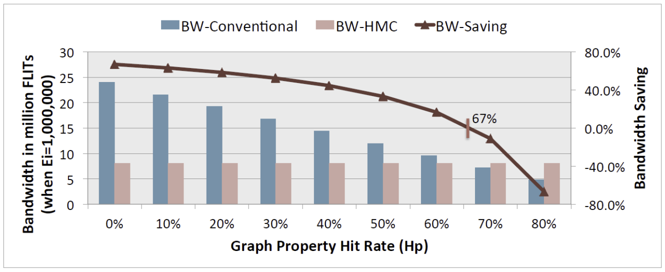


Fig. 8. Instruction Offloading resulting Bandwidth reduction [9]

C. *Application Offloading*

PIM allows scaling processing performance and memory bandwidth proportional to the memory size. The host interface bandwidth does not scale as well. If instruction offloading is used, the scaling of the PIM system is limited to the host interface bandwidth. Offloading of an application instead of single instructions allows the PIM system to scale beyond current host interface bandwidth limitations.

1) Tesseract Architecture: The Tesseract architecture is an approach to build an application accelerator to demonstrate the scalability of PIM. The architecture consists of multiple modified HMCs (Tesseract Cubes). A small in-order cpu core is added to each vault controller (see Figure 9) inside of an HMC. Message passing is used instead of a cache coherence protocol. Cache coherence would be difficult and inefficient in such a manycore environment. Since instruction offloading is more efficient than fetching data, the message pathing interface is used to pass function calls. To improve the efficiency of the in-order core two prefetchers are used.

- Stride Prefetcher
- Message triggered Prefetcher

Since each node in graph computing consists of several values including the links between nodes, a stride prefetcher can exploit this kind of locality. The stride prefetcher prefetchs cache blocks based on a reference prediction table to hide memory latency.

As shown in Figure 9, the message triggered prefetcher is used to prefetch any data which is needed to process a remote function call. Thereby fetching-stalls of the in-order core could be prevented.

2) Tesseract Benchmark: To benchmark the Tesseract architecture a simulator is used. In the following benchmark results the Tesseract architecture is compared to 3 traditional processing centric architectures. An architecture with a low amount of high performance cores is used with DDR3 memory (DDR3-OoO) and with HMCs (HMC-OoO). In addition, an architecture with a high amount of low performance cores is used with HMCs (HMC-MC)

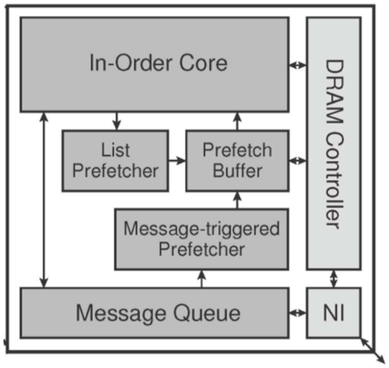


Fig. 9. Tesseract core with additional prefetching units [10]

Following graph algorithm are used as benchmark:

- Average Teenager Follover (AT)
- Conductance (CT)
- PageRank (PR)
- Shortest Path (SP)
- Vertex Cover (VC)

Following graphs are used for the benchmarks:

- ljournal 2008 from the LifeJournal social site (LJ) [11]
- enwiki-2013 from the english Wikipedia (WK) [11]
- indochina-2004 from the country domains of Indochina (IC) [11]

Each of the input graphs is several times bigger than the total cache size to avoid caching effects.

Figure 10 shows the benchmark results normalized to the DDR3-OoO architecture. The Tesseract without the prefetching units reaches a speedup between 4 and 16. Using the prefetching units of the Tesseract cores increases the speedup up to 43. The reached speedup depends highly on the workload.

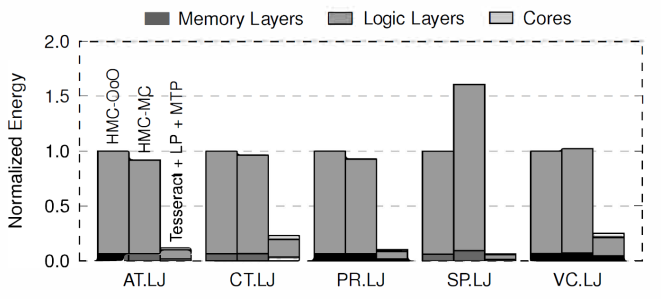


Fig. 11. Benchmark results Power Consumption [10]

During the benchmarks the power consumption of the different architectures was also simulated. As shown in Figure 11 the Tesseract consumes up to 80% less power than the processing centric architectures. Since the former memory interface is used as off chip communication interface the shorter execution times take a big part of the power savings. The Tesseract architecture offers high energy efficiency in comparison to processing centric architectures.

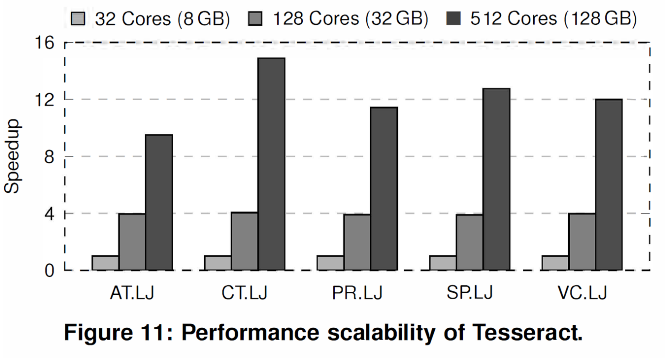


Fig. 12. Benchmark results Scalability [10]

As shown in Figure 12, the Tesseract architecture scales nearly linear from 1 Tesseract cube to 4 cubes. However, the scaling of the Tesseract architecture seems to be limited. 16 Tesseract cubes do not reach a speedup of ≈ 16. The scaling limit of the Tesseract system is the external memory bandwidth between the Cubes. Increasing the external bandwidth will also increase the scaling limit.

The Tesseract architecture was proposed as a memory centric architecture to show scaling with memory capacity. The simulated workloads show high speedups and higher energy efficiency compared to conventional architectures. The scalability of the Tesseract is limited to the external bandwidth.

D. Comparing Instruction Offloading with Application Offloading

Instruction offloading is easy to implement. Depending on a specific algorithm it may only be necessary to convert small parts. Using application offloading means to convert the complete algorithm in a specific order to match the PIM system.

However, instruction offloading only increases the effective bandwidth. Only the higher effective memory bandwidth can be used to reach a speedup.

Depending on the algorithm application offloading highly increases the efficiency and the performance of the system. The reached performance of an PIM accelerator scales with the amount of memory installed.

To sum this up, instruction offloading has fewer benefits and is easier to implement. Application offloading offers high performance and is harder to implement.

V. Conclusion

Future PIM systems will overcome the current memory wall. Due to the small distance between memory and processing units extreme high memory bandwidth can be archived. Optimized accelerators offer high speedups and great efficiency improvements compared to current designs.

Although PIM systems offer several improvements compared to conventional systems, there is still a lot of work to do.

Since the scalability of PIM accelerators is only limited by the off-chip bandwidth the old memory wall will be resurrected as a communication wall. Though graph computing is a well performing workload on PIM systems, other workloads must

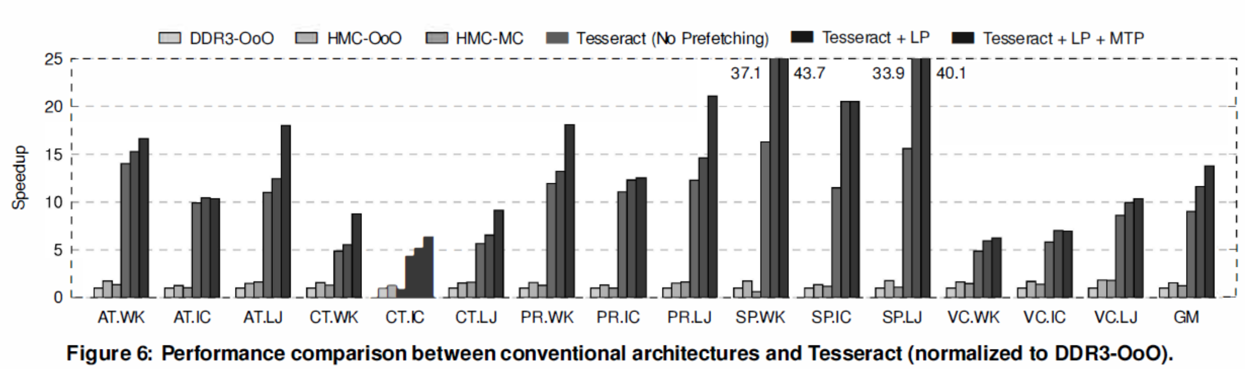


Fig. 10. Benchmark results Speedup [10]

be examined as well. Compute intensive workloads might even show less performance on a PIM system. Future work is necessary to determine the type of the processing units in a PIM system. A small amount of different approaches are:

- General Purpose Processing Units
- Fixed Function Processing Units
- FPGAs

References

[1] C. Kozyrakis, S. Perissakis, D. Patterson, T. Anderson, K. Asanovic, N. Cardwell, R. Fromm, J. Golbus, B. Gribstad, K. Keeton, R. Thomas, N. Treuhaft, and K. Yelick, “Scalable processors in the billion-transistor era: Iram,” *Computer*, vol. 30, no. 9, pp. 75–78, Sep 1997.

[2] R. Sevens, A. White, S. Dosanjh, and et al., “Scientific grand challenges: Architectures and technology for extreme-scale computing report,” 2011.

[3] W. Koh, “Memory device packaging - from leadframe packages to wafer level packages,” in *High Density Microsystem Design and Packaging and Component Failure Analysis, 2004. HDP ’04. Proceeding of the Sixth IEEE CPMT Conference on*, June 2004, pp. 21–24.

[4] S.-Y. Huang, C.-J. Zhan, Y.-W. Huang, Y.-M. Lin, C.-W. Fan, S.-C. Chung, K.-S. Kao, J.-Y. Chang, M.-L. Wu, T.-F. Yang, J. Lau, and T.-H. Chen, “Effects of ubm structure/material on the reliability performance of 3d chip stacking with 30 μm-pitch solder micro bump interconnections,” in *Electronic Components and Technology Conference (ECTC), 2012 IEEE 62nd*, May 2012, pp. 1287–1292.

[5] J. Hruska. (2015) Beyond ddr4: The differences between wide i/o, hbm, and hybrid memory cube. [Online]. Available: http://www.extremetech.com/computing/197720-beyond-ddr4-understand-the-differences-between-wide-io-hbm-and-hybrid-memory-cube

[6] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCauley, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, “Die stacking (3d) microarchitecture,” in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, Dec 2006, pp. 469–479.

[7] “Hybrid memory cube specification 2.0,” Hybrid Memory Cube Consortium, Tech. Rep., 2014.

[8] Wikimedia Foundation Inc. [Online]. Available: https://de.wikipedia.org/wiki/Breitensuche

[9] L. Nai and H. Kim, “Instruction offloading with hmc 2.0 standard - a case study for graph traversals,” in *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, ser. MEMSYS’15, 2015.

[10] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, “A scalable processing-in-memory accelerator for parallel graph processing,” in *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, June 2015, pp. 105–117.

[11] L. for Web Algorithmics. [Online]. Available: http://law.di.unimi.it/datasets.php