UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

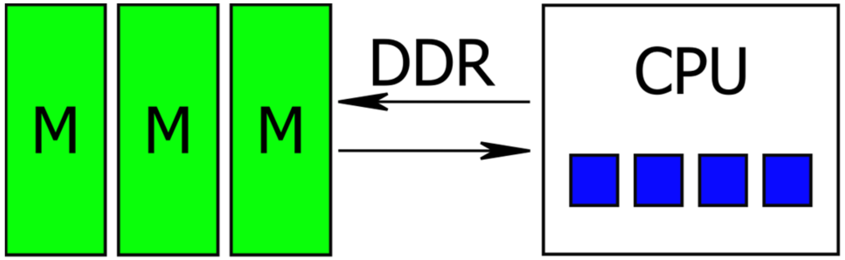
Emerging memory technologies for improved energy efficiency

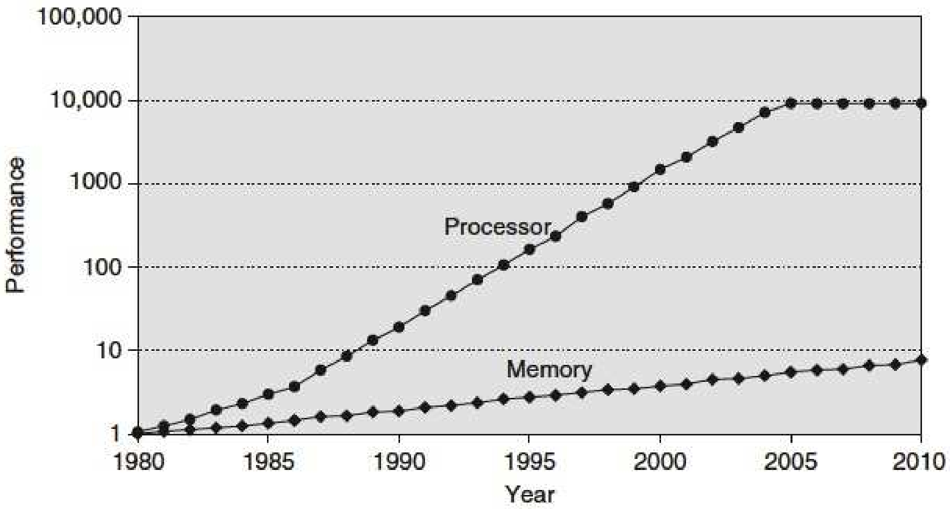
Martin Wenzel

Advanced Seminar

WS2015

Memory Bandwidth

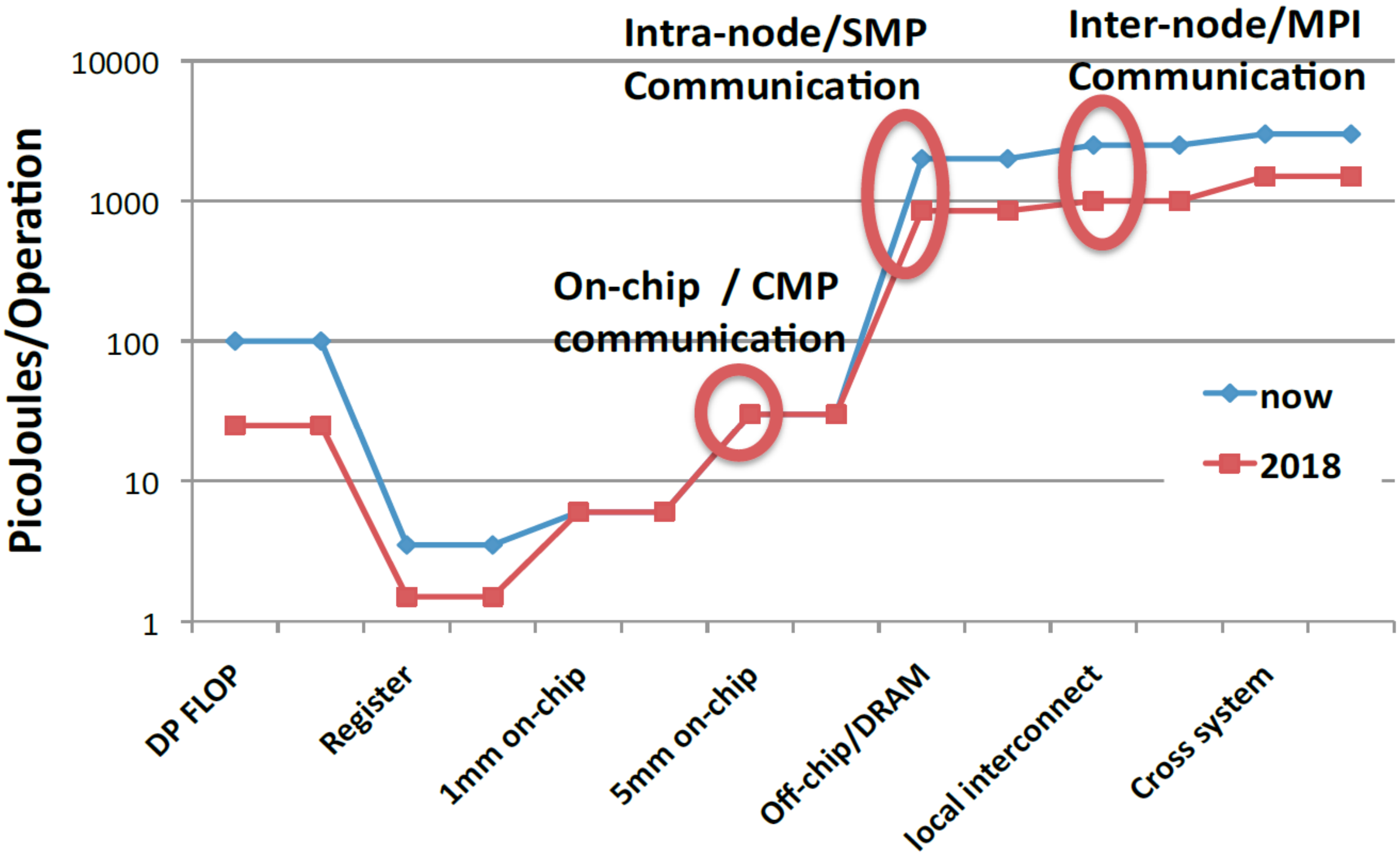




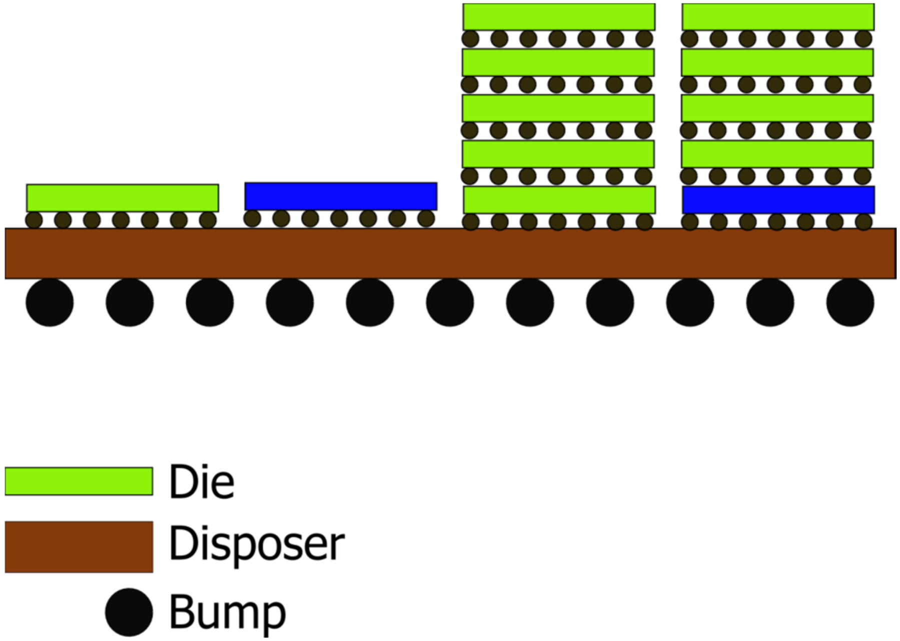
Technology	BW GB/s
DDR3-1333 2GB	10,66
DDR4-2667 4GB	21,34

Hennessy, Patterson, Computer Architecture, A quantitative Approach
http://www.extremetech.com/computing/197720-beyond-ddr4-understand-the-differences-between-wide-io-hbm-and-hybrid-memory-cube

Power Consumption

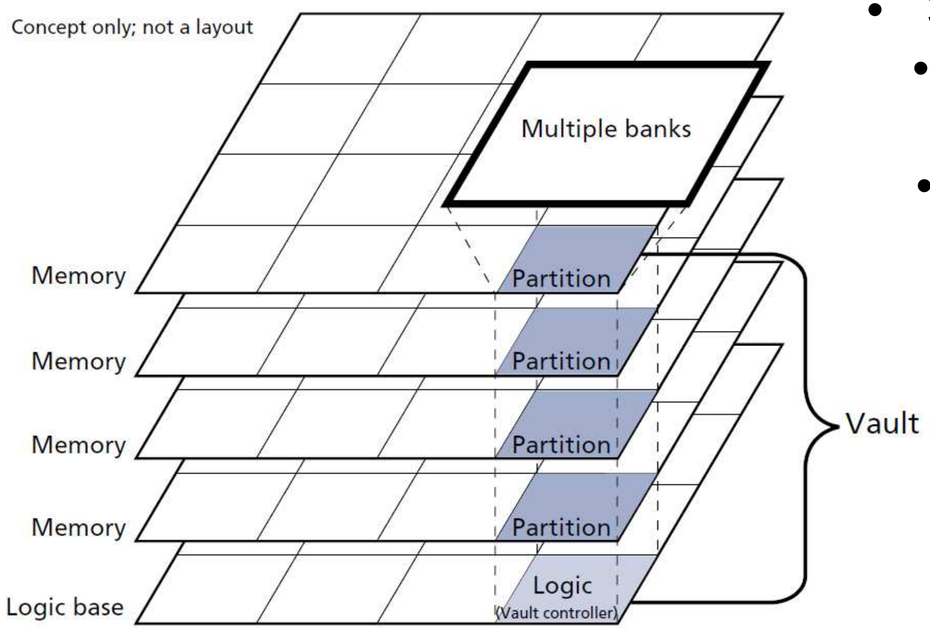


Stacking



- Pricey
- Thermal Resistance
- High Density
- Low Interconnect Length
- High Internal Interconnect Width
  - $\sim 400 \frac{\mu Bumps}{mm^2}$
  - Package limited < $4 \frac{Bumps}{mm^2}$

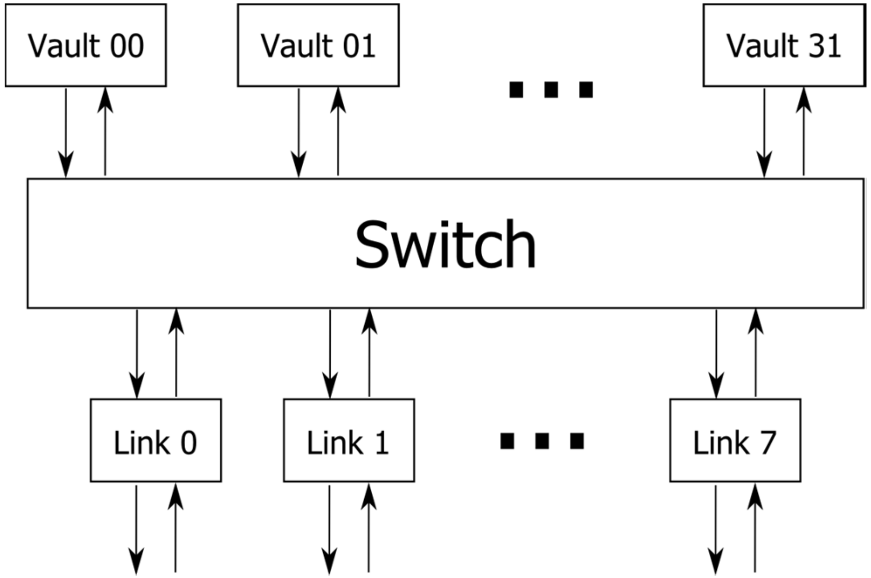
Stacked Memory
Hybrid Memory Cube



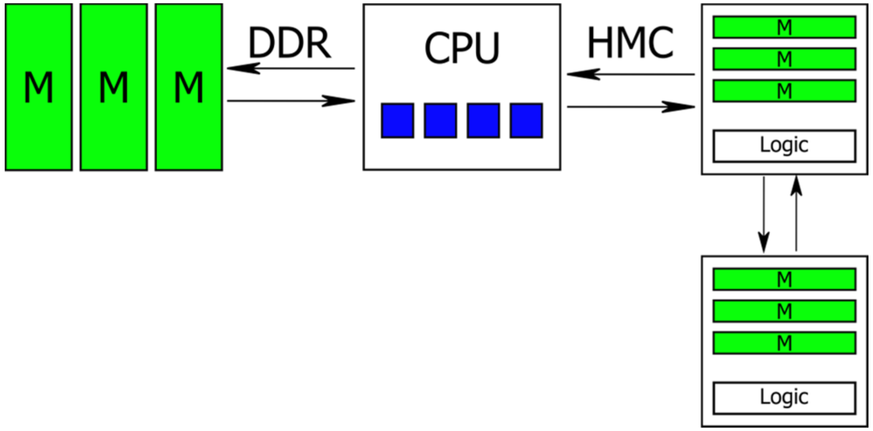
- 32 Vaults
  - Vertical Memory partitions

  - Vault Logic
    - DRAM Controller
    - Packetized Interconnect
    - Support for Atomics
      - Arithmetic
      - Bitwise swap / write
      - Boolean
      - Compare and Swap

Hybrid Memory Cube Interconnect

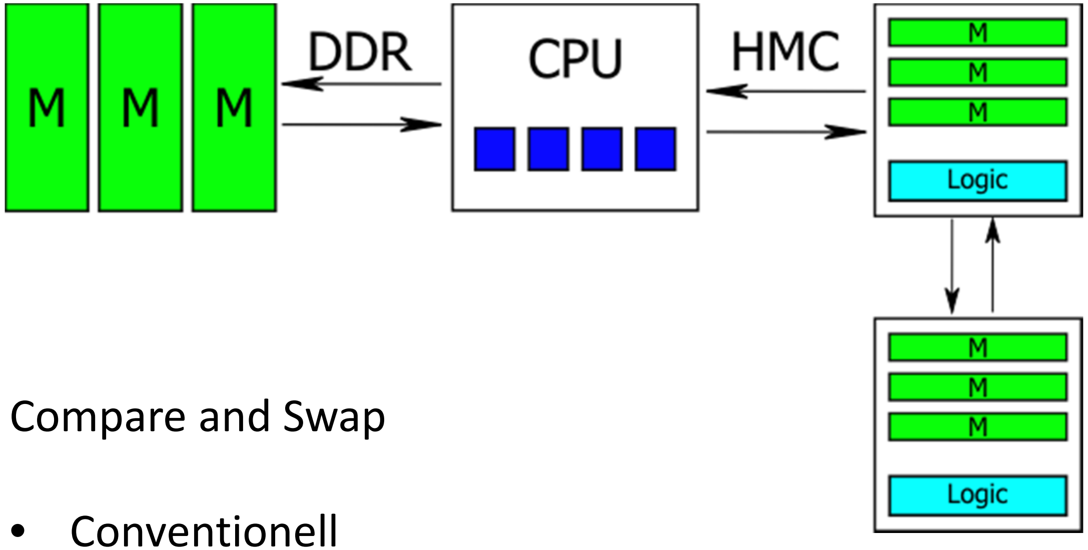


- Packet based Interconnect
- 20GB/s Per Link
- 8 Links per HMC
  - Aggregate Link Bandwidth
  - Connect additional HMCs



Technology	BW GB/s
DDR3-1333 2GB	10,66
DDR4-2667 4GB	21,34

Processing in Memory (PIM) Instruction Offloading



- Problematic Workload
  - Low Computation Intensity
  - Low Locality

- Expectation
  - Efficient Bandwidth Usage

Compare and Swap

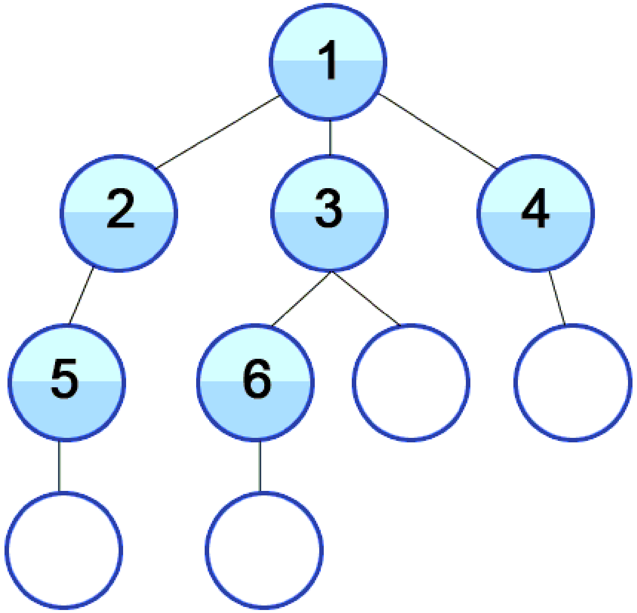
- Conventionell

ReadCacheline(PTR)	64B Data
CAS(PTR,CompVal,New)	
WriteCacheline(PTR)	64B Data

- Atomic CAS

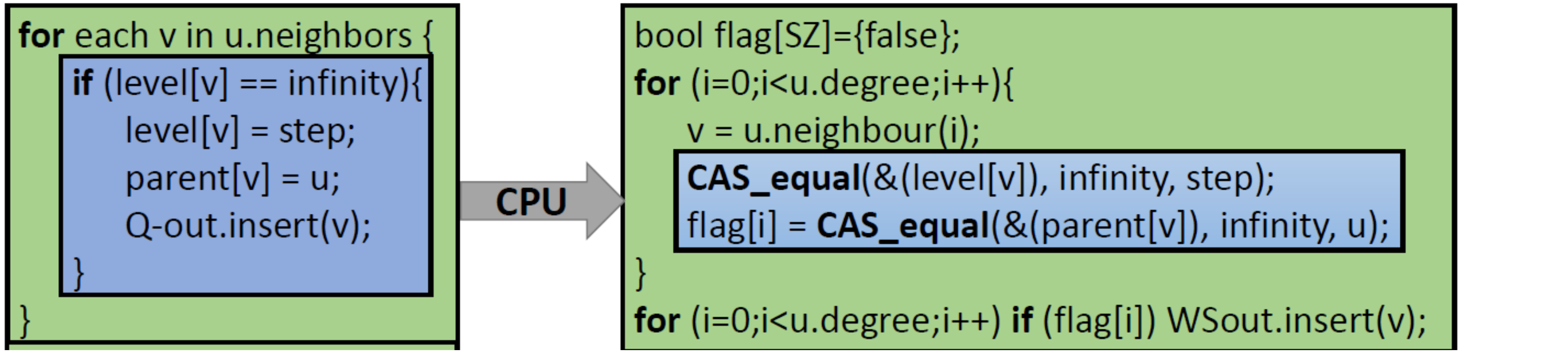
Request_CAS(PTR, CompVal, New)	16B Data
Response	16B Data

Example Workload: Graph Computing
Graph Search



- Breadth-first Search
  - Check all Neighbors
  - Move to the next level

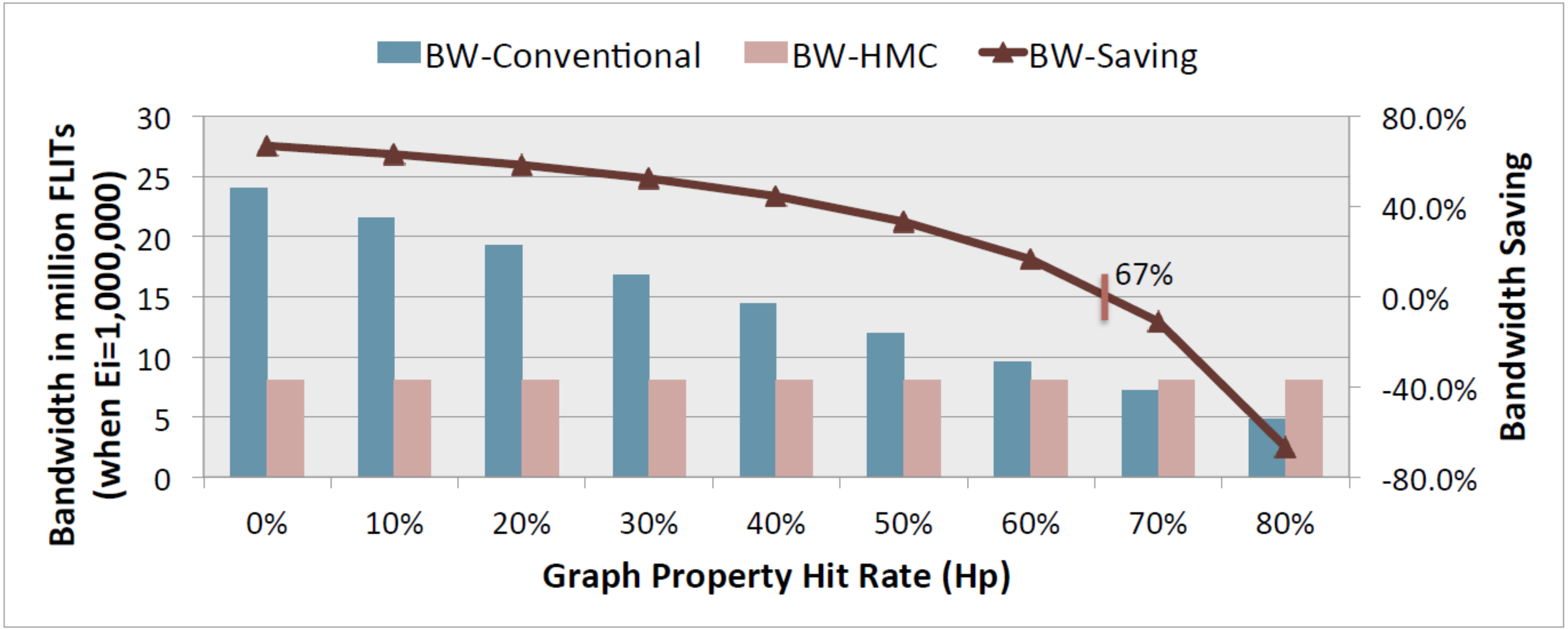
Processing in Memory Offloading



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

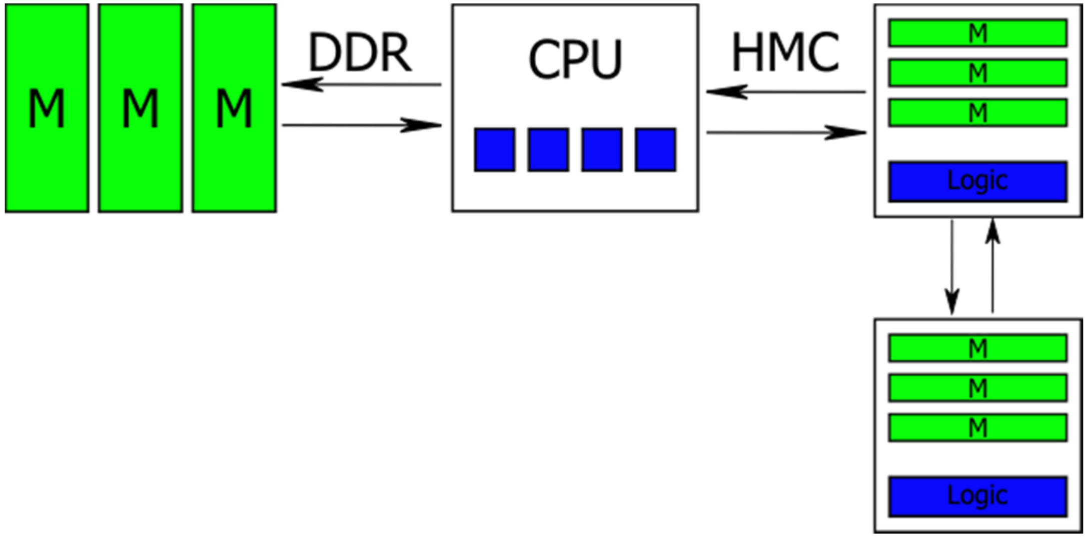
```
bool flag[SZ]={false};
for (i=0;i<u.degree;i++){
    v = u.neighbour(i);
    CAS_equal(&(level[v]), infinity, step);
    flag[i] = CAS_equal(&(parent[v]), infinity, u);
}
for (i=0;i<u.degree;i++) if (flag[i]) WSout.insert(v);
```

Nai, Kim, 2015, Instruction Offloading with HMC 2.0 Standard – A Case Study for Graph Traversals



Nai, Kim, 2015, Instruction Offloading with HMC 2.0 Standard – A Case Study for Graph Traversals

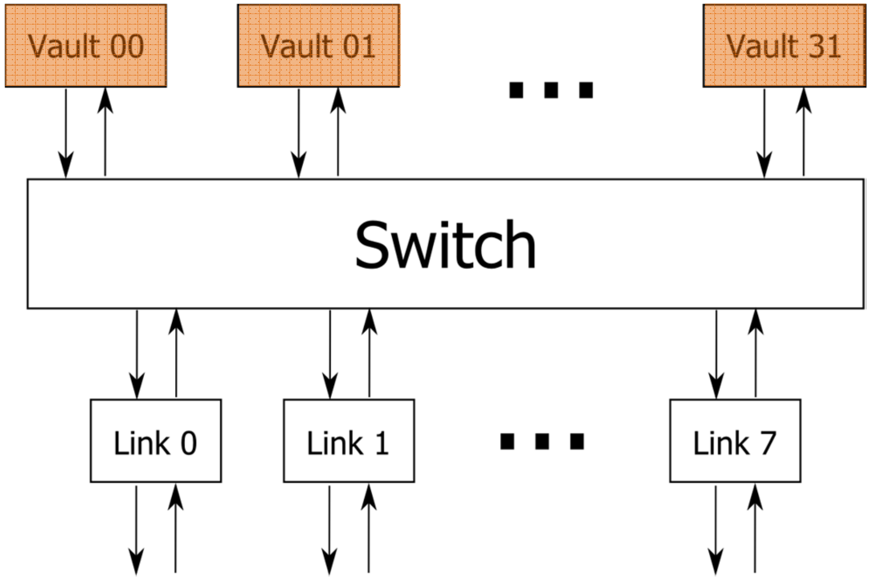
Processing in Memory
Application Offloading – Tesseract



- Problematic Workload
  - Low Computation Intensity
  - Low Locality
- Expectation
  - Efficient Bandwidth Usage
  - High Energy Efficiency
  - Scalability

Processing in Memory
Tesseract

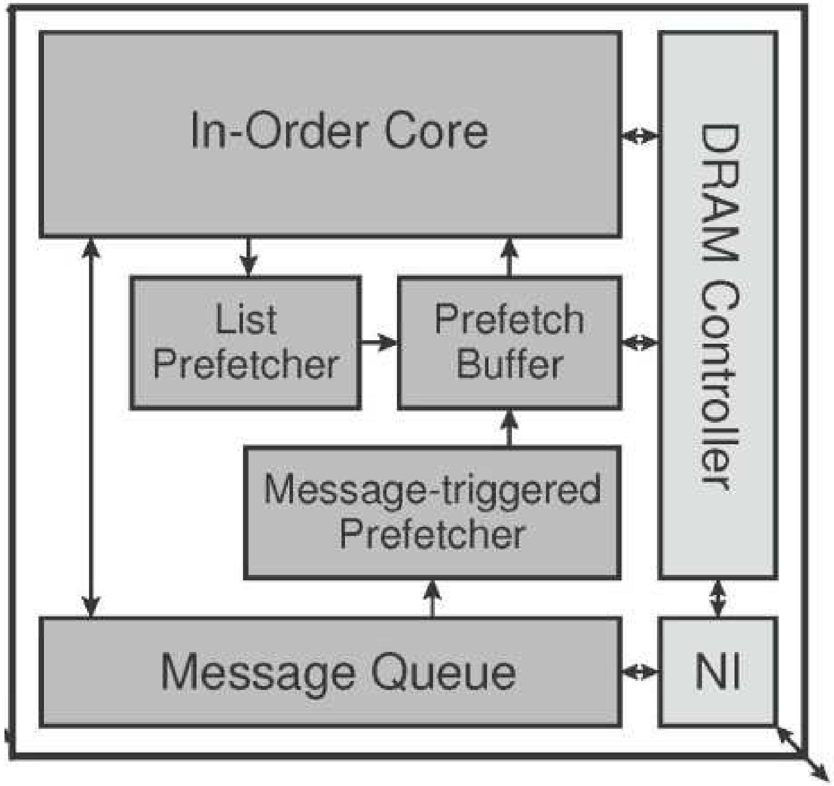
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386



- Single HMC
  - Max Interconnect Bandwidth: 160 GB/s
  - Max Memory Bandwidth: 256 GB/s
- Tesseract
  - PU in every Vault
  - 16 HMC in Network
  - Max Interconnect Bandwidth: 160 GB/s
  - Max Memory Bandwidth: 4 TB/s

HMC Specification V1.0
Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Processing in Memory
Tesseract Core Architecture



- Distributed Memory Architecture
  - No Cache Coherence
  - Remote Function Call
- List Prefetcher
  - Prefetch Stride (Cache Lines)
- Message Triggered Prefetcher
  - Preload Data before Message handling

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Processing in Memory
Tesseract – Speedup

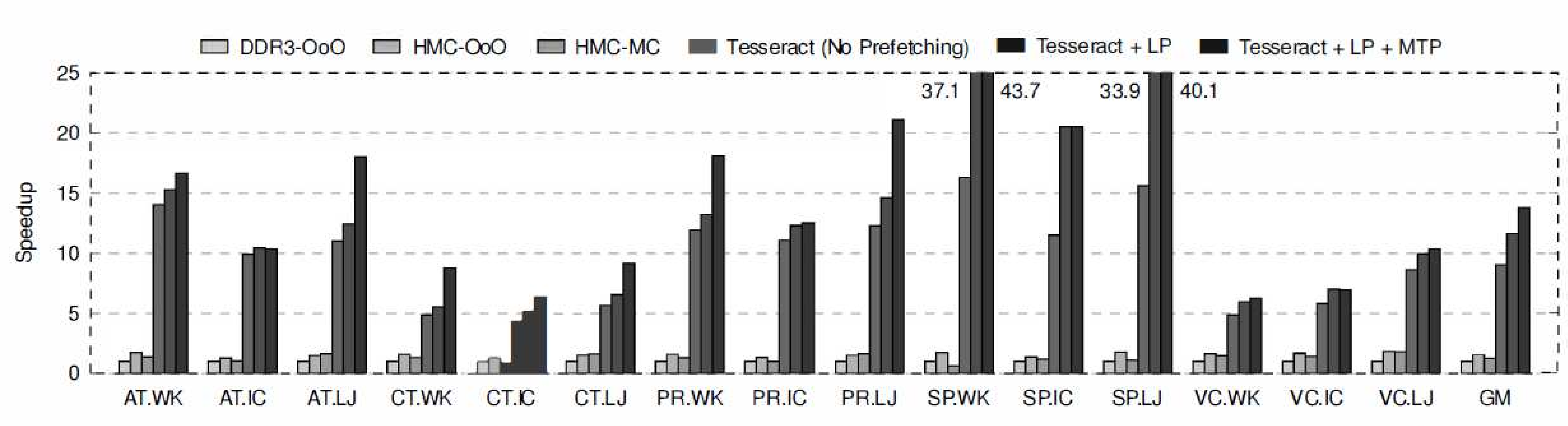
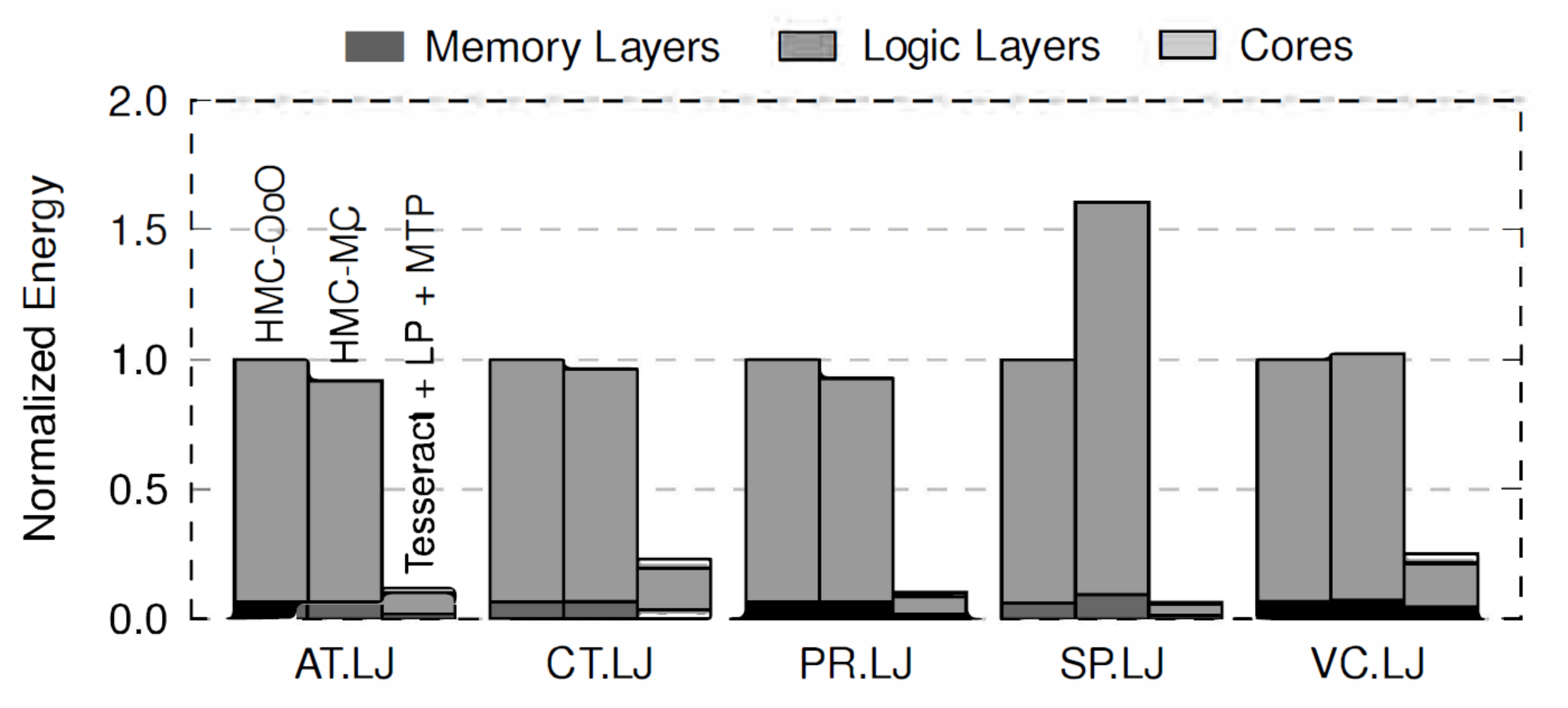


Figure 6: Performance comparison between conventional architectures and Tesseract (normalized to DDR3-OoO).

- HMC-OoO Architecture
  - 32 Performance Cores
  - 16 HMCs
  - 320GB/s Memory Bandwidth
- HMC-MC Architecture
  - 512 low-power Cores
  - 16 HMCs
  - 320GB/s Memory Bandwidth
- Tesseract
  - 512 low-power Cores
  - 16 HMCs
  - 4TB/s Memory Bandwidth

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Processing in Memory
Tesseract – Energy Efficiency



Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Processing in Memory
Tesseract – Scalability

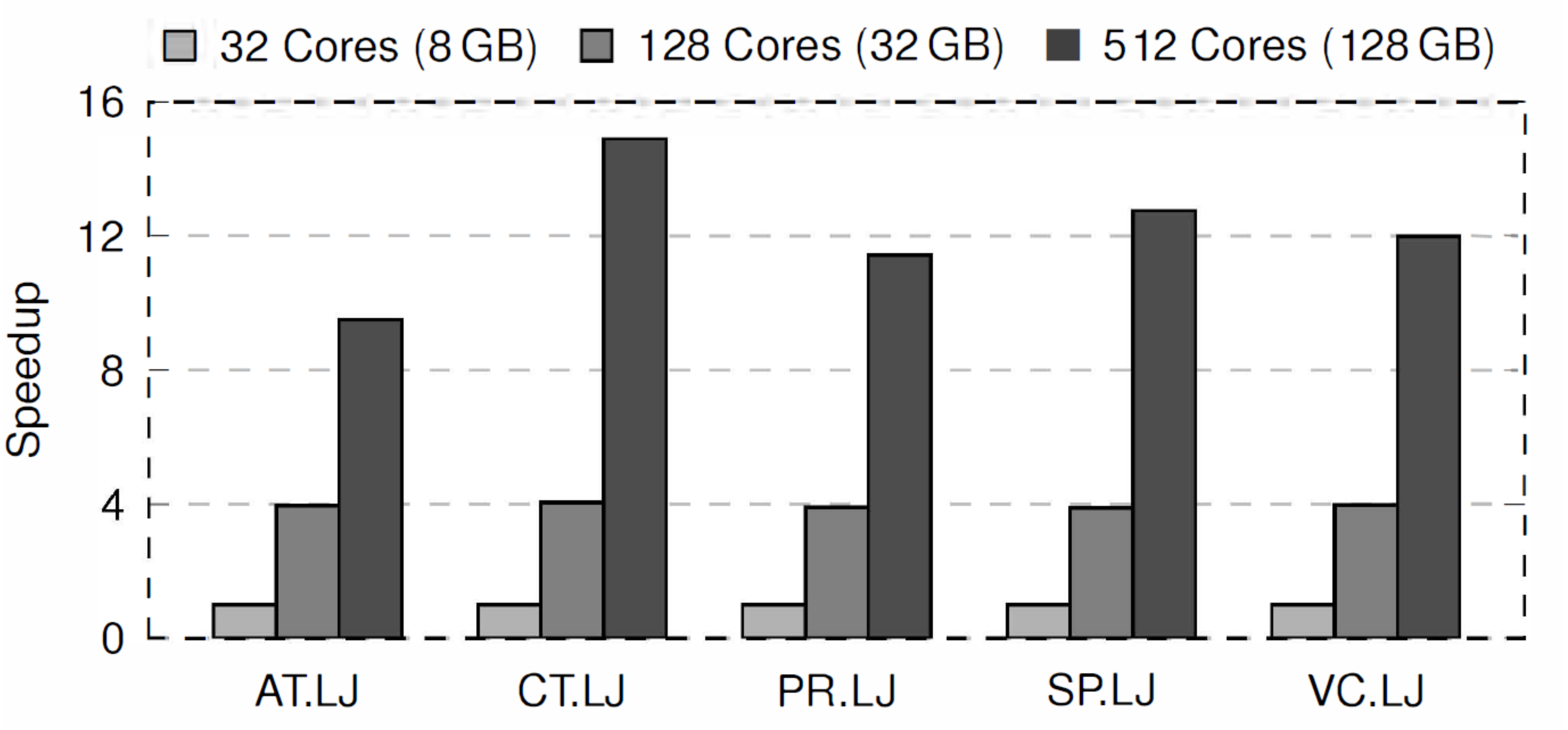


Figure 11: Performance scalability of Tesseract.

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Conclusion Processing in Memory

- High Speedup
- Highly Energy Efficient
- Scales proportional to Memory Capacity
- Currently usable via Instruction Offloading

- Current Designs optimized for Graph Computing

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Future Work

- Additional Workloads

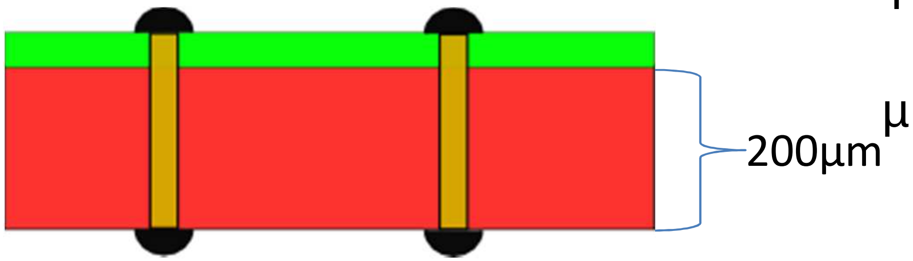
- Processing Units
  - Internode Communication
  - Application specific
  - General Purpose
  - FPGA technology?

Further Information

MEMSYS International Symposium on Memory Systems

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Through – Silicon Via



µBumps on top Metal Layer

~ 50 µm pitch

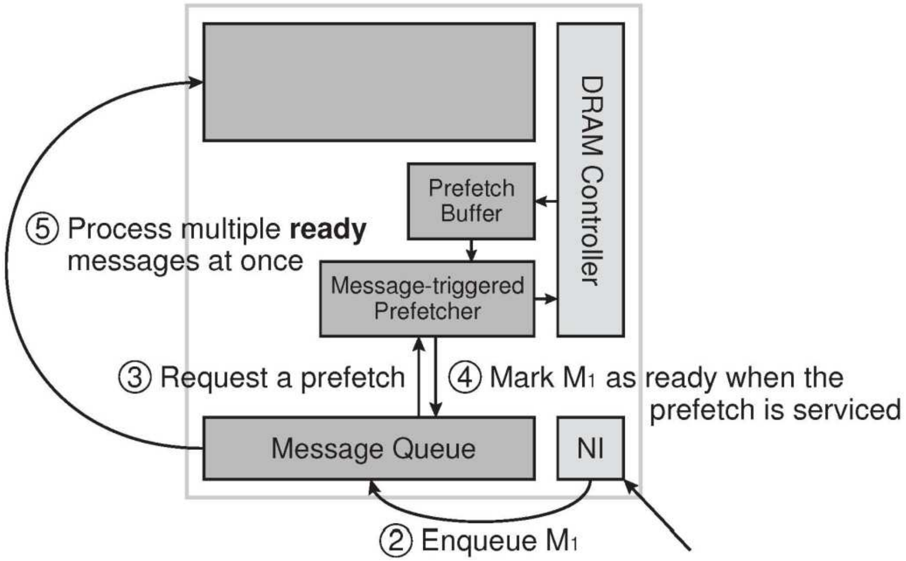
Through – Metal Via

~ 2 - 50 µm

µBumps under Substrate

~ 50 µm pitch

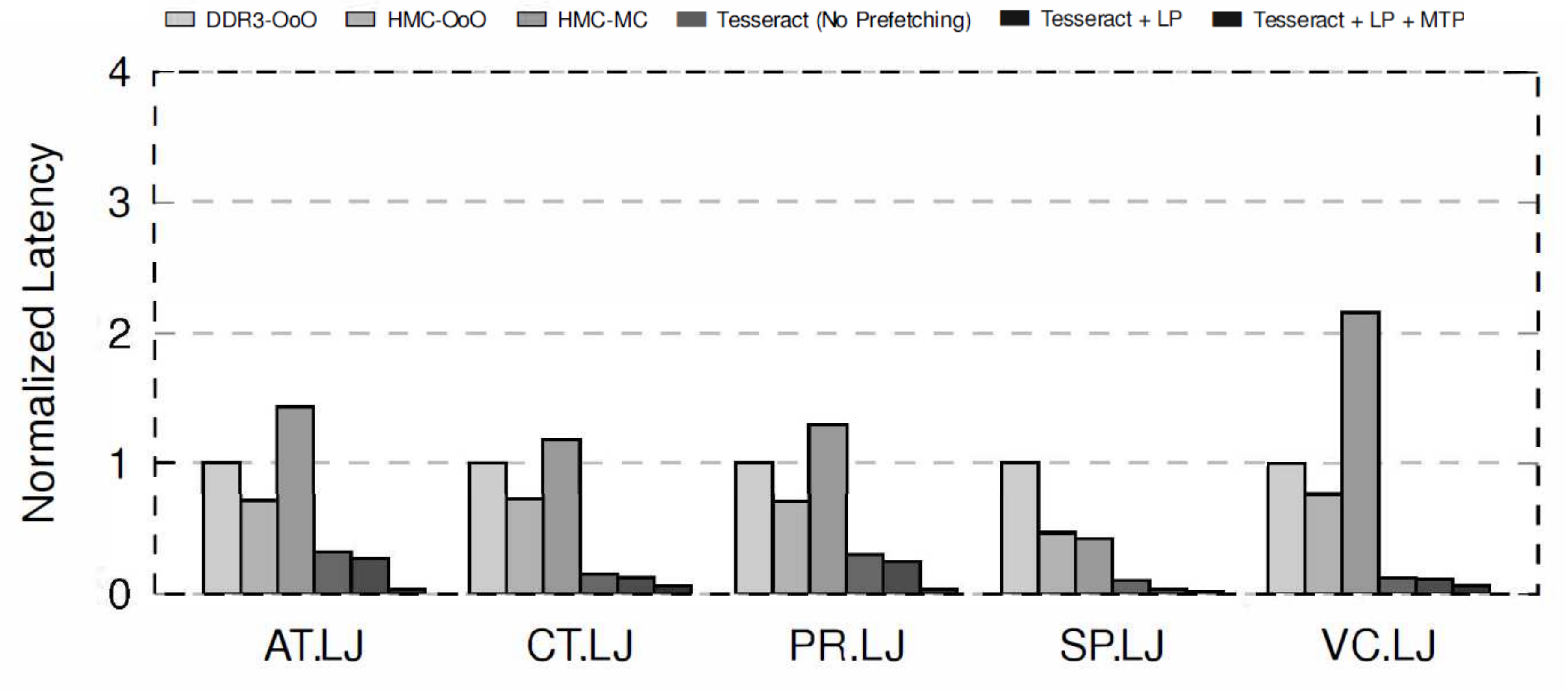
Processing in Memory
Tesseract Core Architecture


DRAM Controller
Prefetch Buffer
⑤ Process multiple **ready** messages at once
Message-triggered Prefetcher
③ Request a prefetch
④ Mark M1 as ready when the prefetch is serviced
Message Queue
NI
② Enqueue M1

- Distributed Memory Architecture
  - No Coherence Traffic
  - Message / Instruction Passing
- Optional List Prefetcher
  - Optimize Locality
- Message Triggered Prefetcher
  - Preload Data before Message handling

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Processing in Memory
Tesseract – Latency



(b) Average memory access latency (normalized to DDR3-OoO)

Ahn, Hong, Yoo, Mutlu, Choi, 2015, A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing