The Active Memory Cube:
A Processing-in-Memory System for High Performance Computing

Zehra Sura

IBM T.J. Watson Research Center
Yorktown Heights, New York

IBM

AMC Team Members

Ravi Nair
Samuel Antao
Carlo Bertolli
Pradip Bose
Jose Brunheroto
Tong Chen
Chen-Yong Cher
Carlos Costa
Jun Doi
Constantinos Evangelinos
Bruce Fleischer
Thomas Fox
Diego Gallo
Leopold Grinberg
John Gunnels
Arpith Jacob
Philip Jacob
Hans Jacobson
Tejas Karkhanis
Changhoan Kim
Jaime Moreno
Kevin O'Brien
Martin Ohmacht
Yoonho Park
Daniel Prener
Bryan Rosenburg
Kyung Ryu
Olivier Sallenave
Mauricio Serrano
Patrick Siegl
Krishnan Sugavanam
Zehra Sura

Supported in part by the US Department of Energy

HPC Challenges

- **Power Wall**
  - High power affects:
    - Transistor reliability at circuit level
    - Power delivery/cooling costs at system level

- **Memory Wall**
  - %time for memory ops ↑
  - %time for compute ops ↓

- Many others…

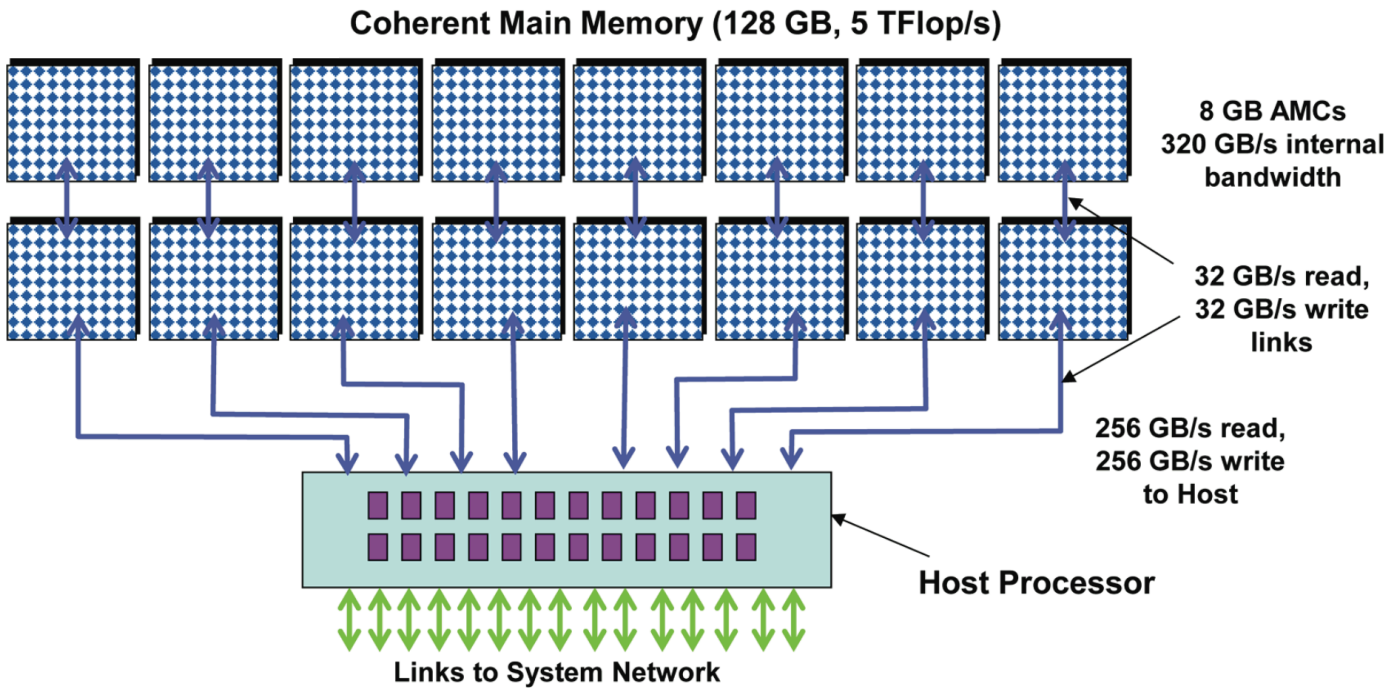
This Talk

- **Experience with the Active Memory Cube (AMC)**
  - Developed microarchitecture, OS, compiler, cycle-accurate simulator, power model
  - Evaluated performance on kernels from HPC benchmarks

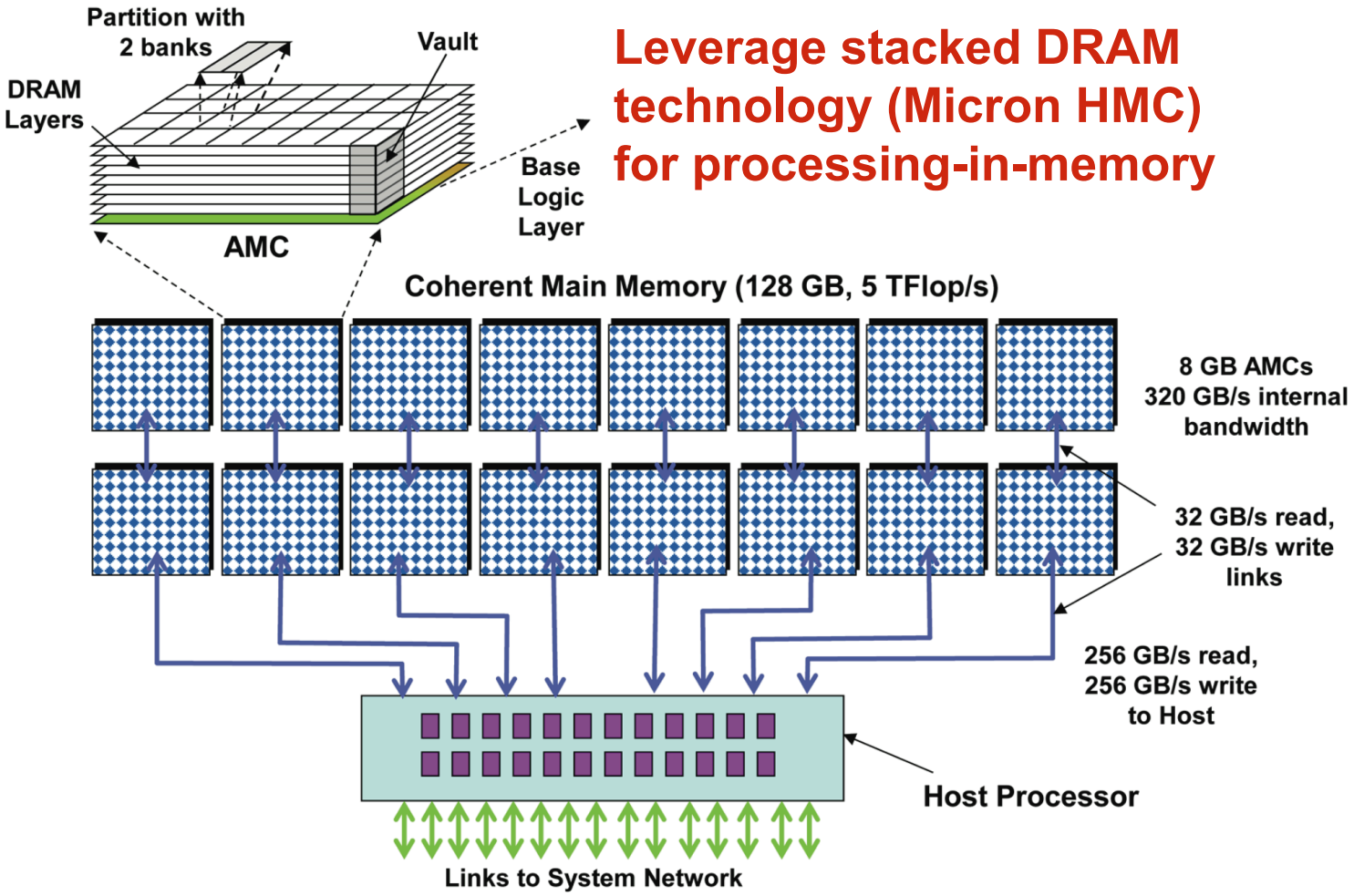
- **Outline**
  - System design and goals
  - Architecture description
  - Power, performance, programmability concerns

AMC System Design

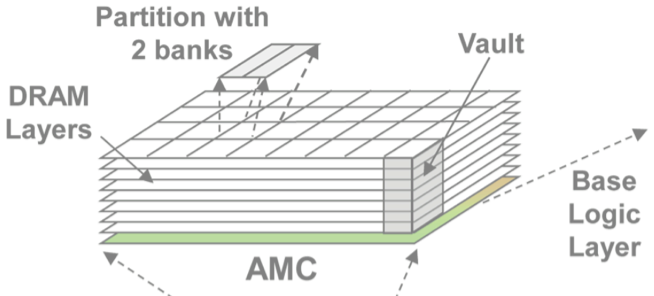
Leverage stacked DRAM technology (Micron HMC) for processing-in-memory



AMC System Design



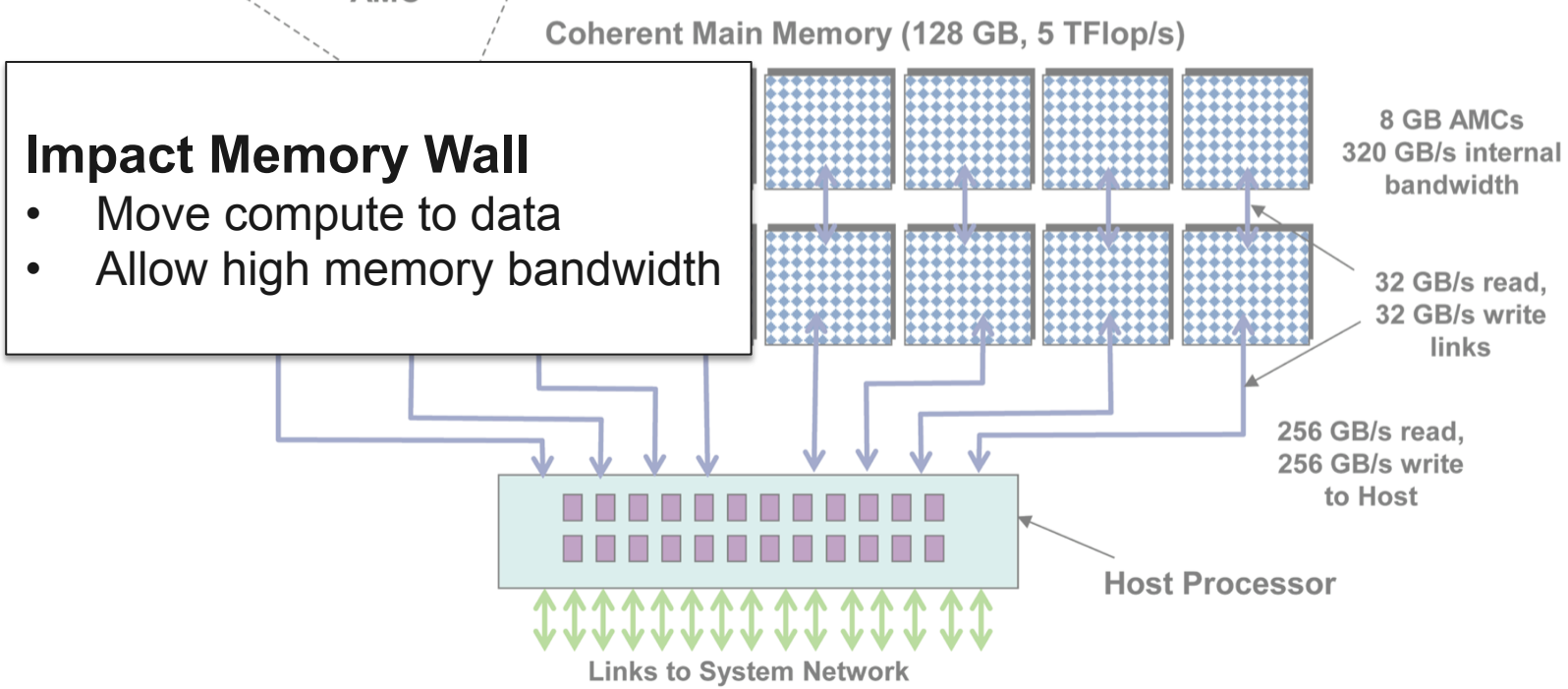
AMC System Design



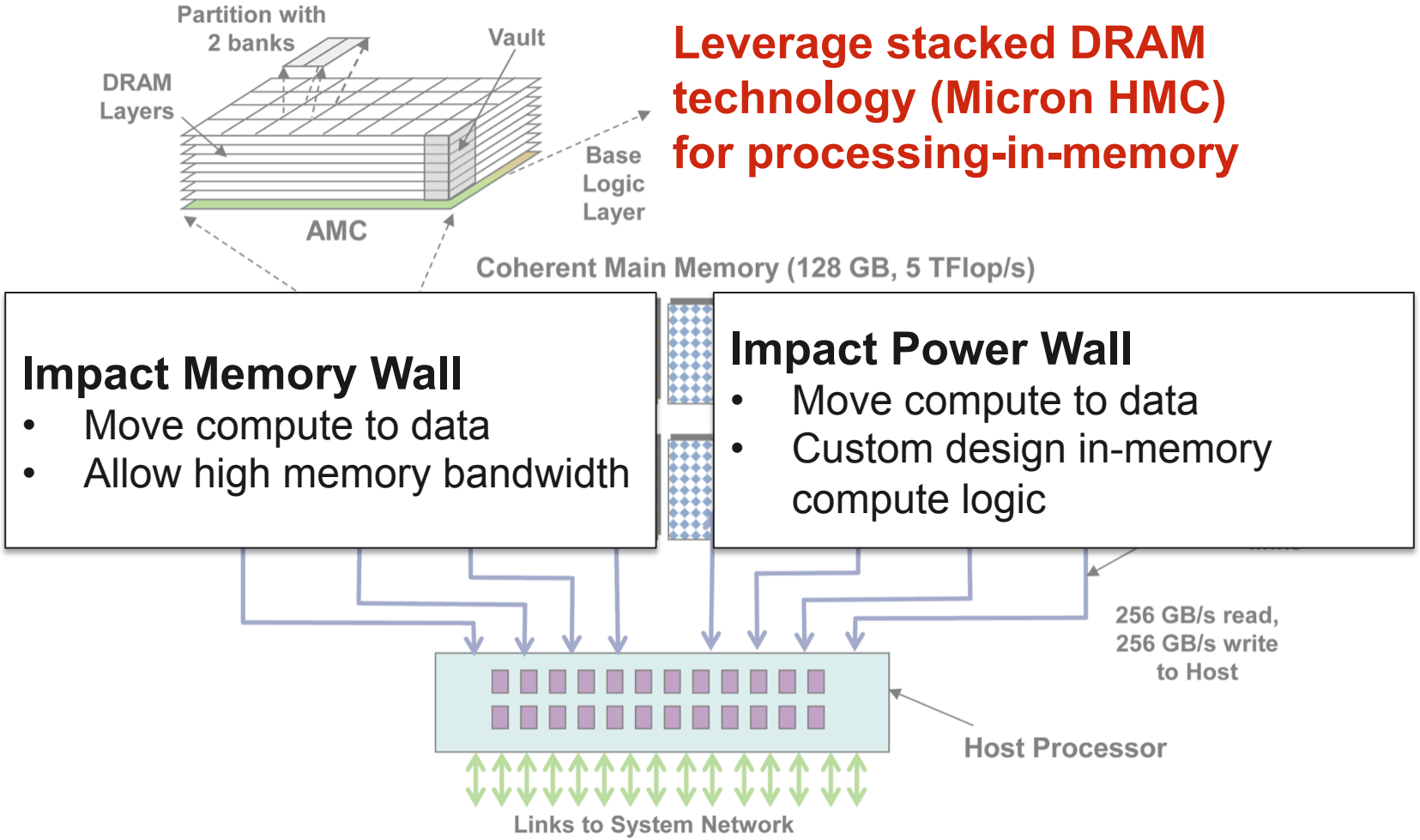
Leverage stacked DRAM technology (Micron HMC) for processing-in-memory

Impact Memory Wall

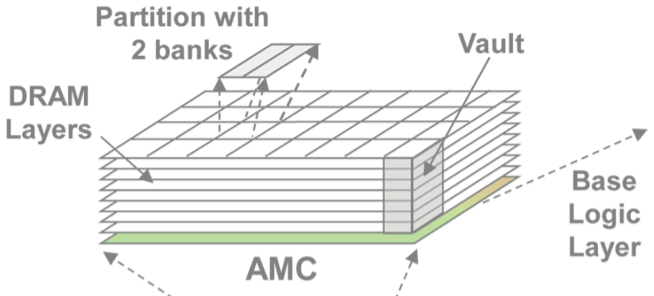
- Move compute to data
- Allow high memory bandwidth



AMC System Design



AMC System Design



Leverage stacked DRAM technology (Micron HMC) for processing-in-memory

Coherent Main Memory (128 GB, 5 TFlop/s)

Impact Memory Wall:
- Move compute to data
- Allow high memory bandwidth

Impact Power Wall:
- Move compute to data
- Custom design in-memory compute logic

Integral Part of Design:
- Help improve performance for a range of applications
- Accessible, i.e. easy to use and program
- Extreme power efficiency

Projected to be 20 GFlops/W for DGEMM in 14nm at 1.25 GHz

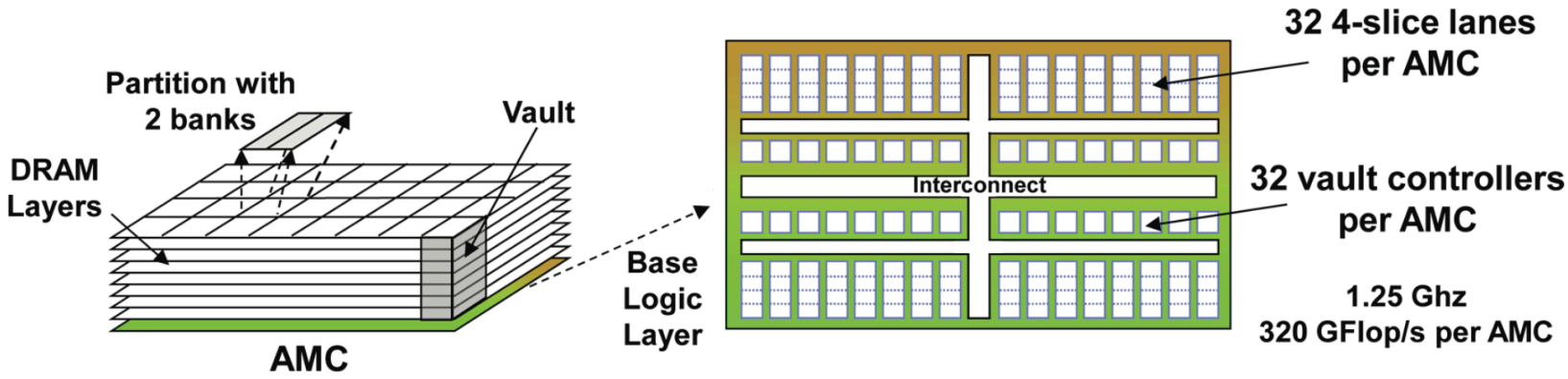
The Green500 List

Source: green500.org

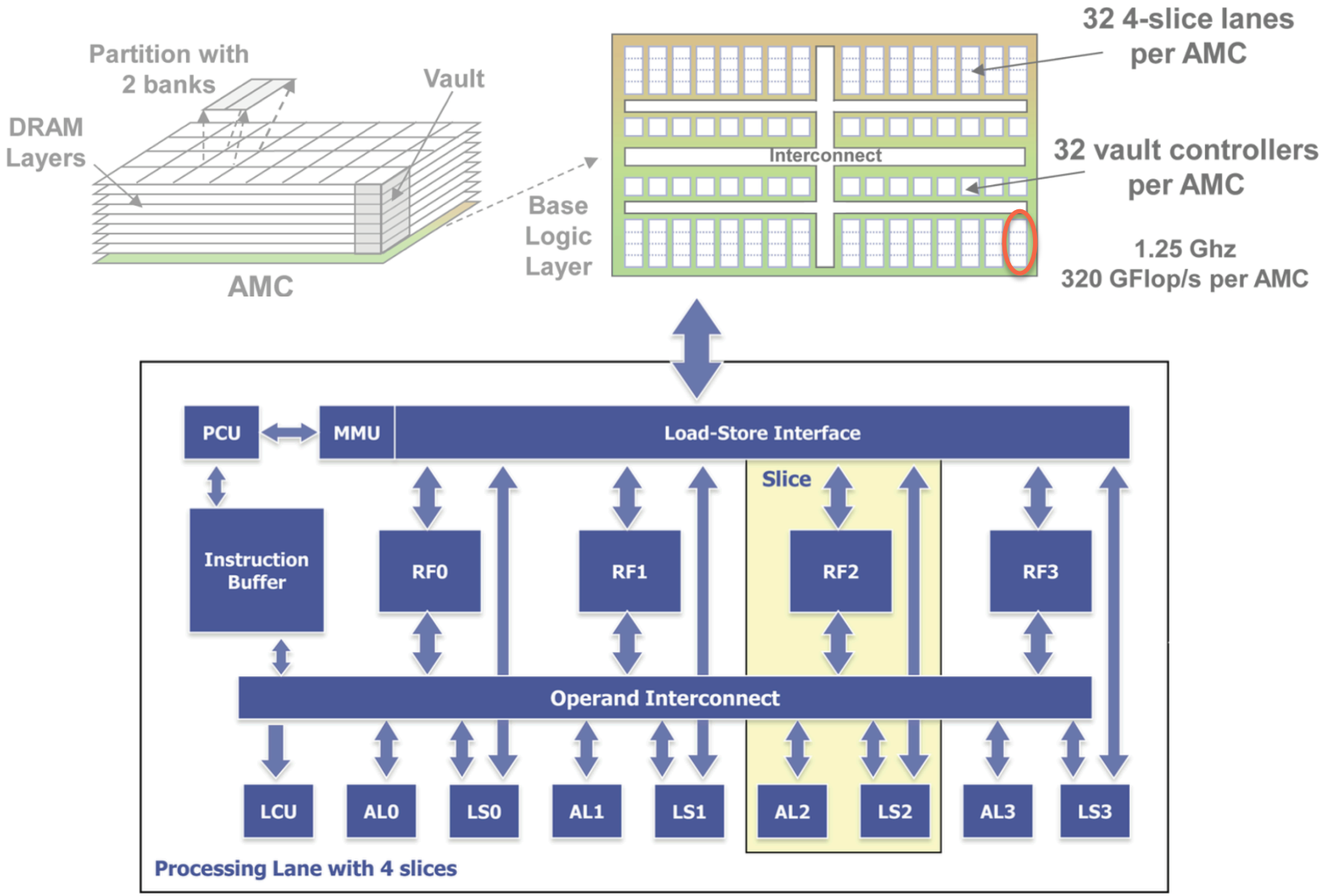
Listed below are the June 2015 The Green500's energy-efficient supercomputers ranked from 1 to 10.

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	7,031.58	RIKEN	Shoubu - ExaScaler-1.4 80Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SC	50.32
2	6,842.31	High Energy Accelerator Research Organization /KEK	Suiren Blue - ExaScaler-1.4 16Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband, PEZY-SC	28.25
3	6,217.04	High Energy Accelerator Research Organization /KEK	Suiren - ExaScaler 32U256SC Cluster, Intel Xeon E5-2660v2 10C 2.2GHz, Infiniband FDR, PEZY-SC	32.59
4	5,271.81	GSI Helmholtz Center	ASUS ESC4000 FDR/G2S, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR, AMD FirePro S9150	57.15
5	4,257.88	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR, NVIDIA K20x	39.83
6	4,112.11	Stanford Research Computing Center	XStream - Cray CS-Storm, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, Nvidia K80	190.00
7	3,962.73	Cray Inc.	Storm1 - Cray CS-Storm, Intel Xeon E5-2660v2 10C 2.2GHz, Infiniband FDR, Nvidia K40m	44.54
8	3,631.70	Cambridge University	Wilkes - Dell T620 Cluster, Intel Xeon E5-2630v2 6C 2.600GHz, Infiniband FDR, NVIDIA K20	52.62
9	3,614.71	TU Dresden, ZIH	Taurus GPUs - Bull bullx R400, Xeon E5-2680v3 12C 2.5GHz, Infiniband FDR, Nvidia K80	58.01
10	3,543.32	Financial Institution	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	54.60

AMC Processor Architecture



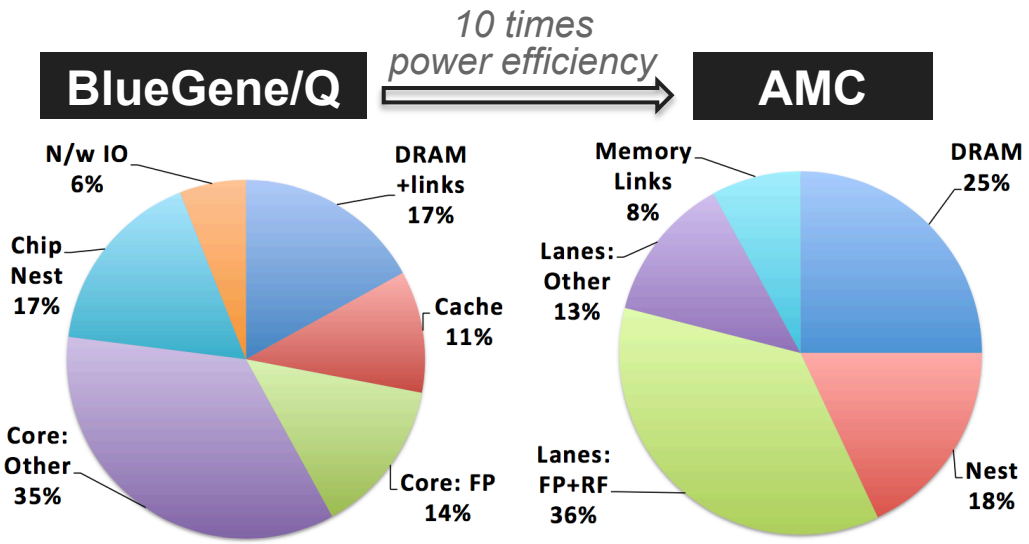
AMC Processor Architecture



Power Consumption Breakdown

Source: green500.org

33	2,304.48	Dassault Aviation	BlueGene/Q, Power BQC 16C 1.6GHz, Custom Interconnect	82.00



20 GFlops/W for DGEMM in 14nm at 1.25 GHz

Enabling Power-Performance Efficiency

I. Exploit near-memory properties
II. Delegate to software
III. Provide lots of parallelism

Balanced architecture design:

★ Save power
★ Improve performance
★ Support programmability

I. Exploit Near-Memory Properties

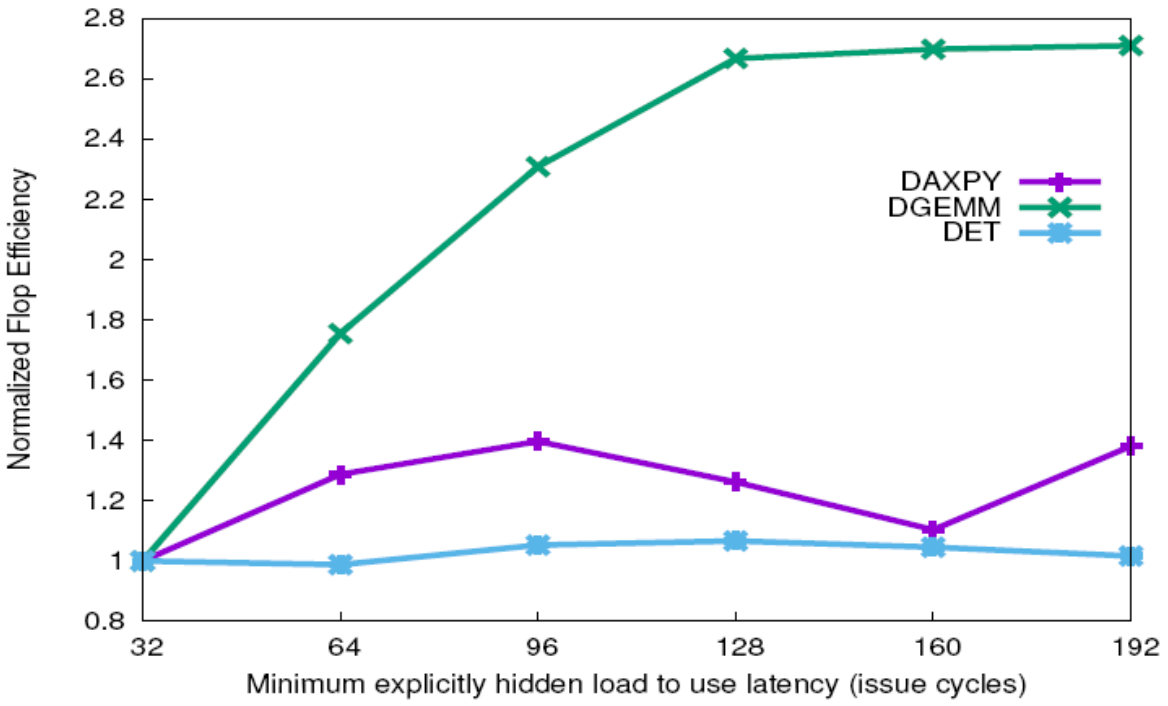
- Latency range 26 cycles to 250+ cycles
  - No caches
  - **Large** register files:
    - 16 vector registers * 32 elements * 8-bytes * 4 slices ➔ 16K per lane
    - 32 scalar registers
    - 4 vector mask registers
  - Buffers in vault controllers
  - Load combining
  - Page policy

I. Exploit Near-Memory Properties

- Latency range 26 cycles to 250+ cycles
  - No caches
  - Large register files
  - Buffers in vault controllers
  - Load combining
  - Page policy

Flop efficiency is the % of peak flop rate utilized in execution.

Theoretical peak for a lane is 8 Flops per cycle.



I. Exploit Near-Memory Properties

- Latency range 26 cycles to 250+ cycles
  - No caches
  - Large register files
  - Buffers in vault controllers
  - Load combining
  - Page policy
- High bandwidth
  - On-chip ★
  - Deep LSQ
  - Multiple load-store units
  - Multiple striping policies

I. Exploit Near-Memory Properties

- Latency range 26 cycles to 250+ cycles
  - No caches
  - Large register files
  - Buffers in vault controllers
  - Load combining
  - Page policy
- High bandwidth
  - On-chip ★
  - Deep LSQ
  - Multiple load-store units
  - Multiple striping policies

DAXPY:

```
for (i=0; i<N; i++)
    B(i) = B(i) + x * A(i);
```

Memory bound

Maximum bandwidth utilization for kernel:
47.8% of peak (153.2 GB/s of 320 GB/s)

Expected bandwidth utilization in apps:
30.9% of peak (99 GB/s of 320 GB/s)

For node with 16 AMCs:
1.58 TF/s (99 GB/s * 16 AMCs)

Peak bandwidth available to host:
256 GB/s

I. Exploit Near-Memory Properties

- Latency range 26 cycles to 250+ cycles
  - No caches
  - Large register files
  - Buffers in vault controllers
  - Load combining
  - Page policy
- High bandwidth
  - On-chip ★
  - Deep LSQ
  - Multiple load-store units
  - Multiple striping policies
- Support programming/heterogeneity:
  - Shared memory
  - Effective address space same as host processors ★
  - Hardware coherence/consistency ★
  - In-memory atomic operations ★

Enabling Power-Performance Efficiency

I. Exploit near-memory properties
II. Delegate to software
III. Provide lots of parallelism

Balanced architecture design:

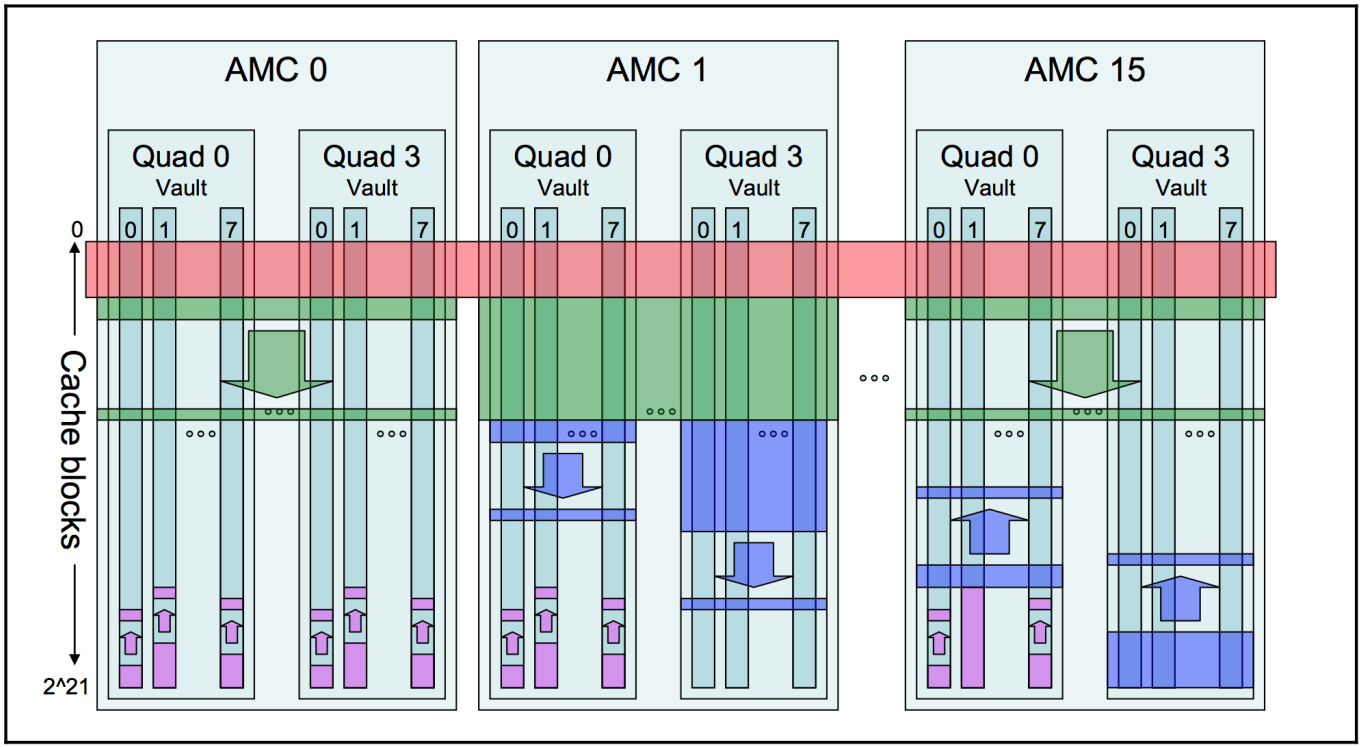
★ Save power
★ Improve performance
★ Support programmability

II. Software Delegation

- Memory
  - ERAT translation : segment-based translation table
  - Striping policy for data placement/affinity

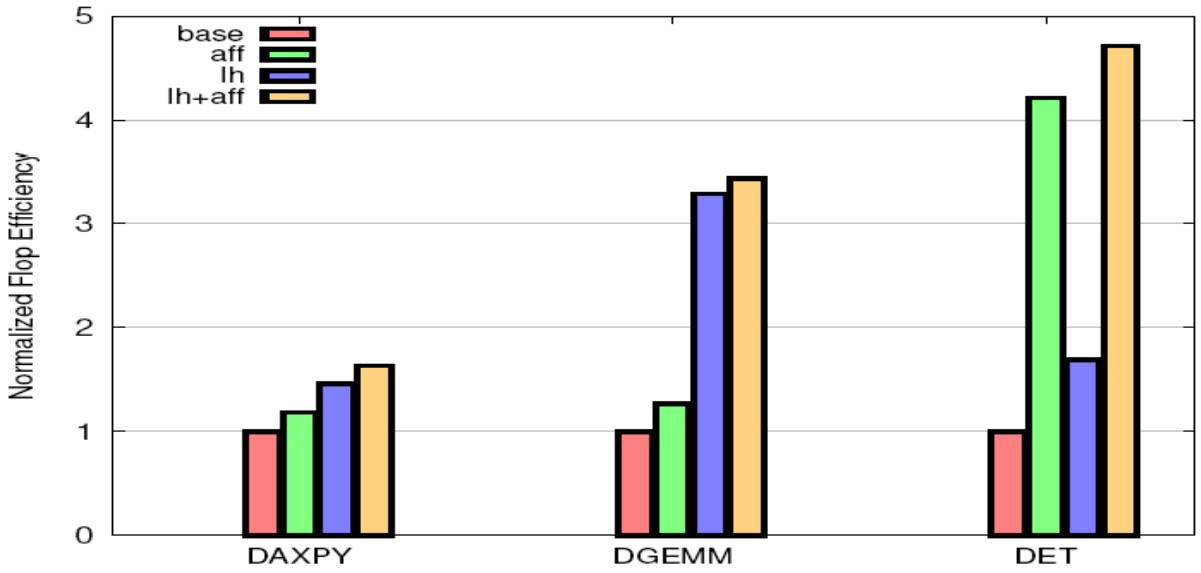
II. Software Delegation

- Memory
  - ERAT translation : segment-based translation table
  - Striping policy for data placement/affinity



II. Software Delegation

- Memory
  - ERAT translation : segment-based translation table
  - Striping policy for data placement/affinity



base
data allocated across AMC default optimizations

lh
base + latency hiding opts

aff
base, but data allocated in specific quadrant

lh+aff
with all opts

II. Software Delegation

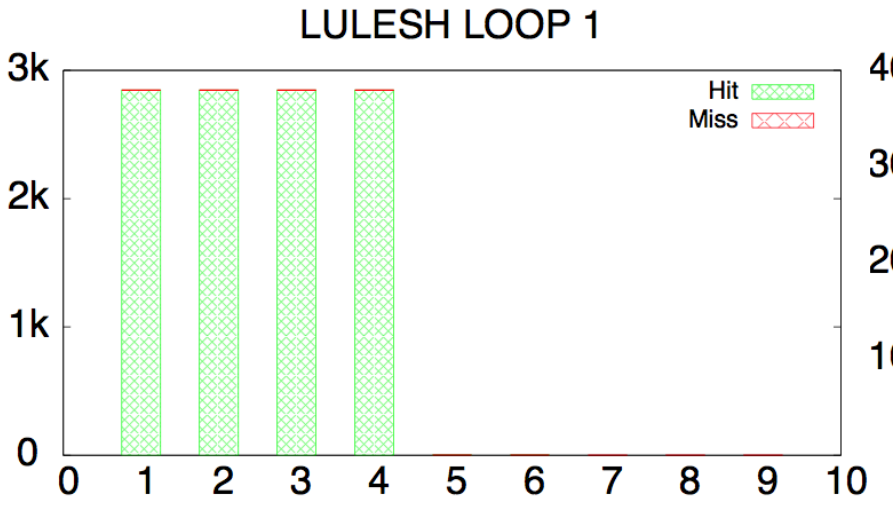
- Memory
  - ERAT translation
  - Striping policy for data placement/affinity

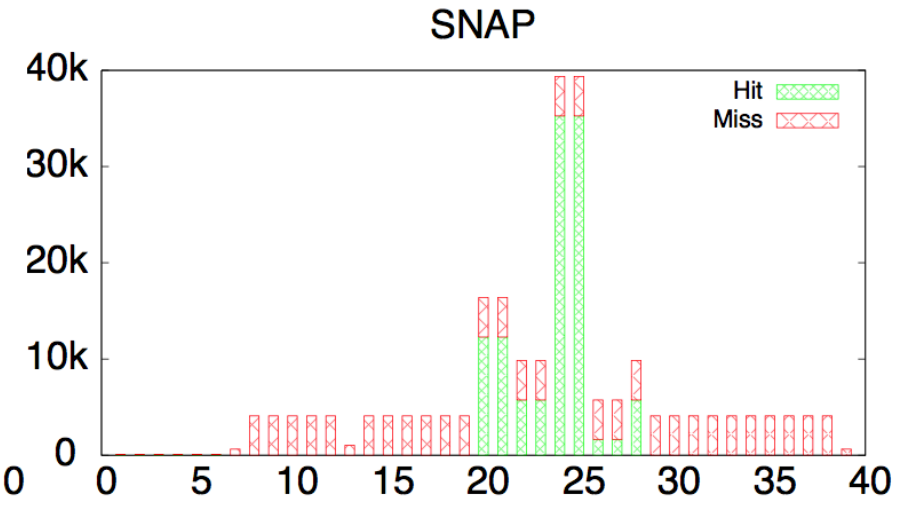
- Computation
  - Pipeline dependence checking
  - ILP detection
  - Instruction cache

II. Software Delegation

- Memory
  - ERAT translation
  - Striping policy for data placement/affinity

- Computation
  - Pipeline dependence checking
  - ILP detection
  - Instruction cache





II. Software Delegation

- Memory
  - ERAT translation
  - Striping policy for data placement/affinity
- Computation
  - Pipeline dependence checking
  - ILP detection
  - Instruction cache
- Parallelization
  - Vectorization and SIMD ★

Enabling Power-Performance Efficiency

I. Exploit near-memory properties
II. Delegate to software
III. Provide lots of parallelism

Balanced architecture design:

★ Save power
★ Improve performance
★ Support programmability

III. Parallelization

Maximize utilization of available resources for power-performance

- Multiple types of parallelism
  - Programmable-length vector processing
  - Spatial SIMD (2-way, 4-way, 8-way)
  - ILP (multiple functional units; horizontal microcoding)
  - Heterogeneous
  - Multithreaded, multicore

- Mixed scalar/vector
- Scatter/gather, strided load/stores with update, packed load/stores
- Predication

Compiler

Supports an MPI+OpenMP4.0 programming model

	MANUAL	COMPILER
DET	71.1 GF/s (22.2% of peak)	121.6 GF/s (38% of peak)
DAXPY (BW)	99 GB/s (30.9% of peak)	99 GB/s (30.9% of peak)
DGEMM*	266 GF/s (83% of peak)	246 GF/s (77% of peak)

DGEMM: Compiler currently needs 2 innermost loops to be manually blocked

Compiler

Supports an MPI+OpenMP4.0 programming model

	MANUAL	COMPILER
DET	71.1 GF/s (22.2% of peak)	121.6 GF/s (38% of peak)
DAXPY (BW)	99 GB/s (30.9% of peak)	99 GB/s (30.9% of peak)
DGEMM*	266 GF/s (83% of peak)	246 GF/s (77% of peak)

DGEMM: Compiler needs 2 innermost loops to be manually blocked

THE GOOD

- Unified loop optimization
  - Blocking
  - Distribution
  - Unrolling
  - Versioning
- Array scalarization
- Scheduling
- Register allocation
- Function calls, SIMD/ predicated functions
- Software instruction caching

THE BAD

- Latency prediction
- Data placement
- Sequence of accesses

THE UGLY

- Alias analysis
- Automatic coarse-grained parallelization

Conclusion

- AMC design demonstrates an aggressive “hardware enablement, software exploitation” model for power-efficient architecture design
  - Judicious division of responsibility between layers of system stack

- Processing-in-memory viable with adoption of 3D stacked memory
  - Save on data movement cost
  - Easier to support programmability for node-level computation