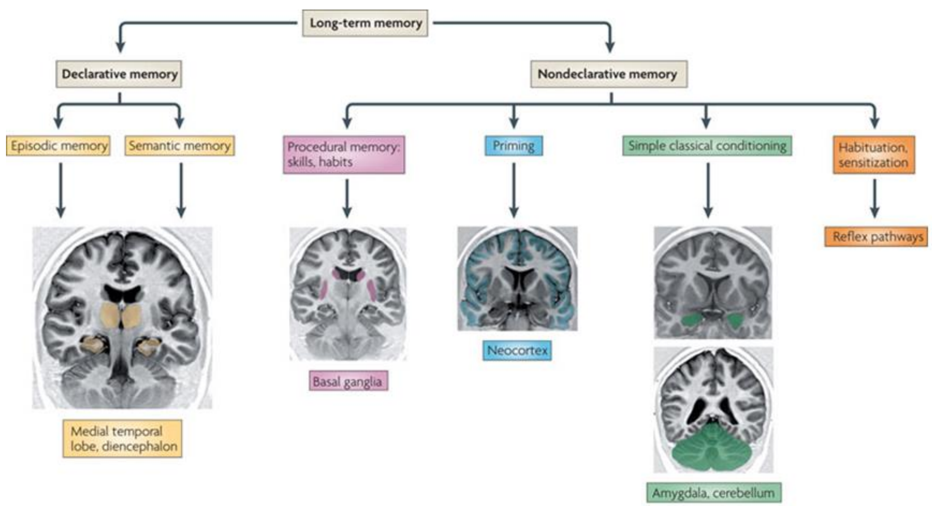
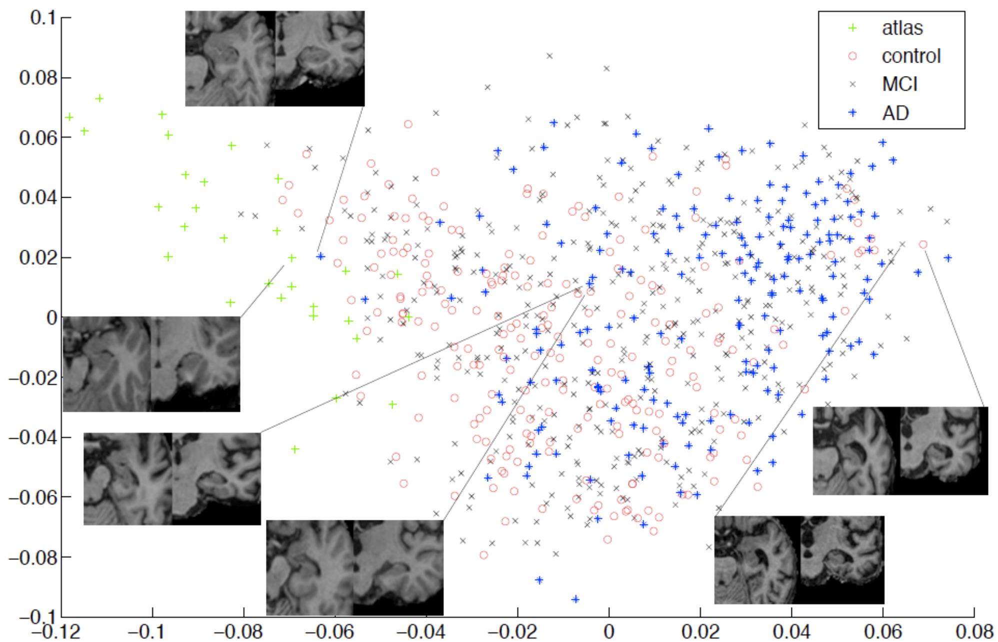
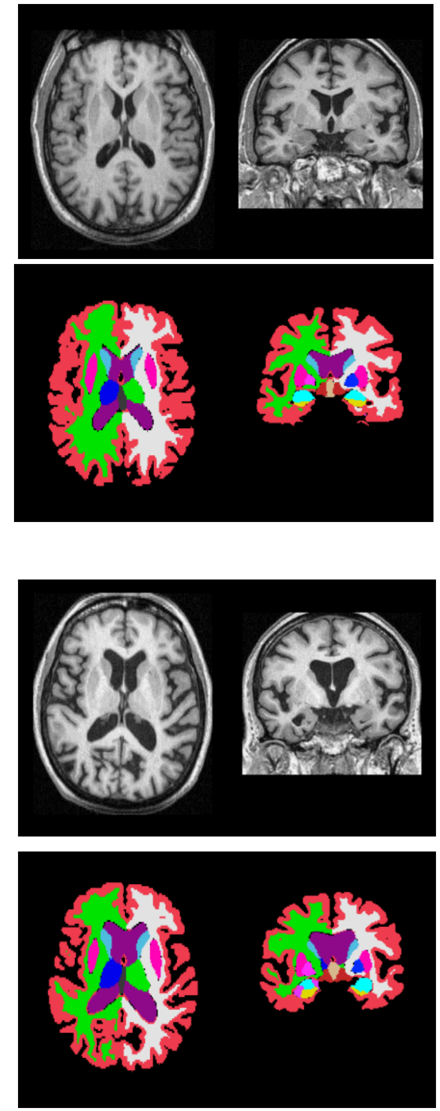
TOP-PIM: Throughput-Oriented Programmable Processing in Memory

Dong Ping Zhang, Nuwan Jayasena, Alex Lyashevsky
Joe Greathouse, Lifan Xu, Mike Ignatowski
AMD Research
25/06/14

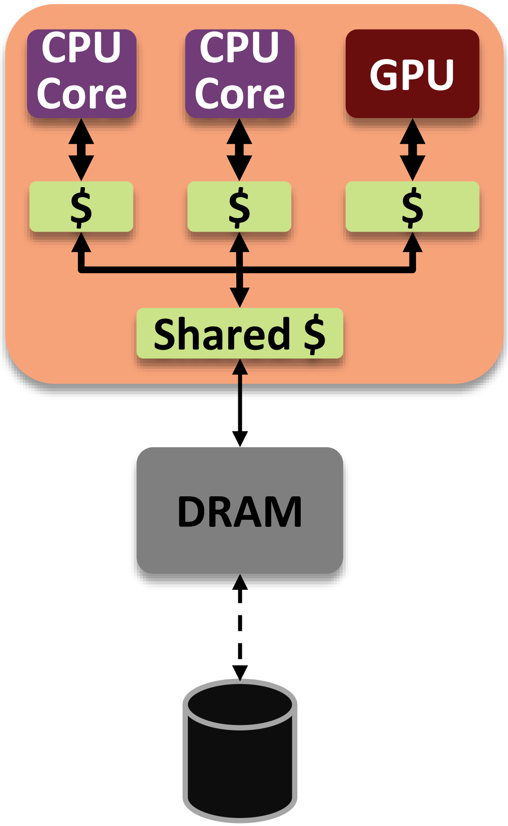
PROCESSING-IN-MEMORY? HBM?



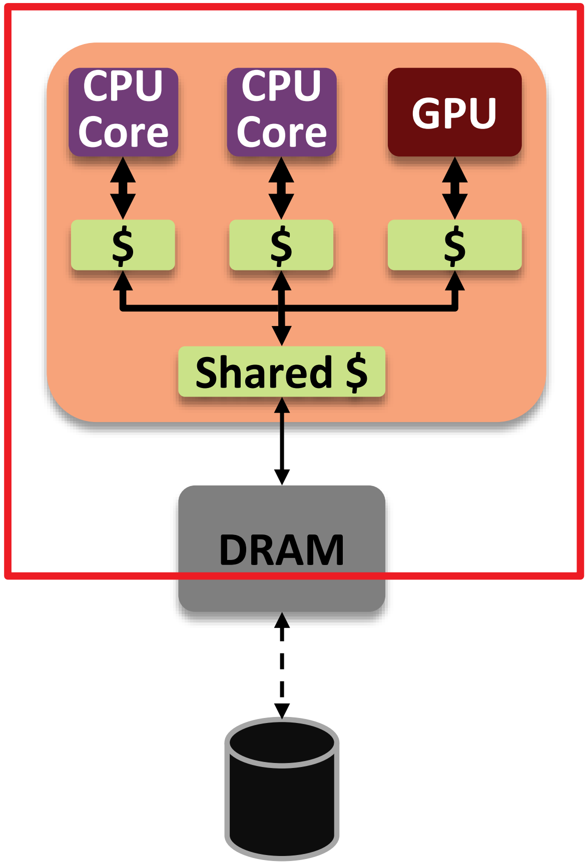




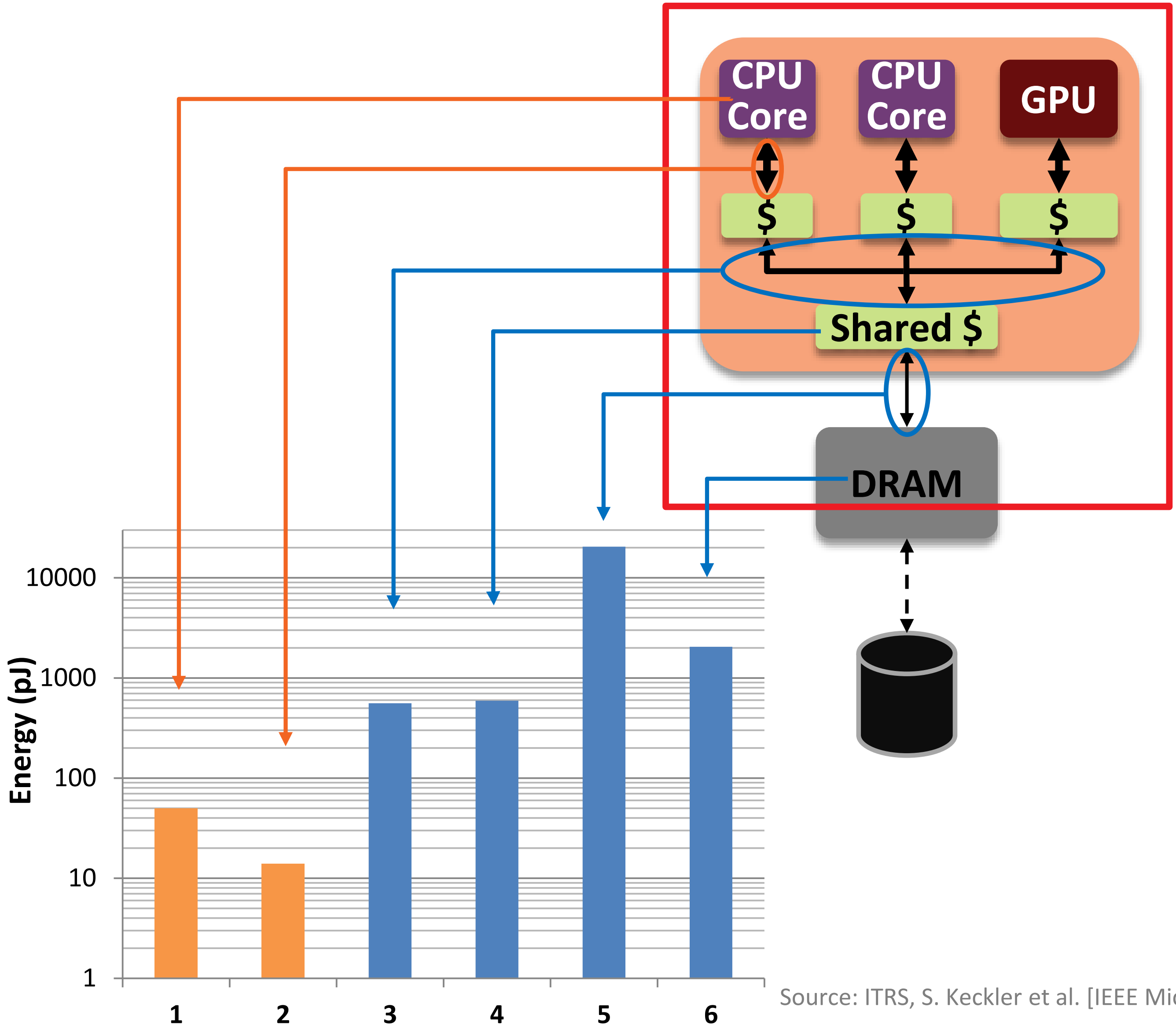
COMPUTE IS CHEAP, DATA MOVEMENT IS NOT



COMPUTE IS CHEAP, DATA MOVEMENT IS NOT



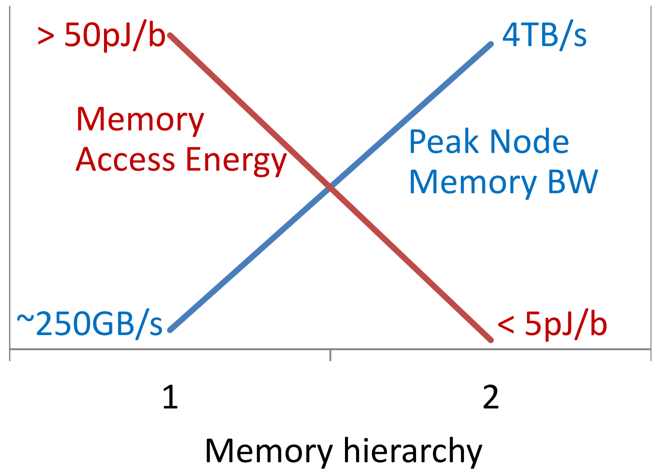
COMPUTE IS CHEAP, DATA MOVEMENT IS NOT

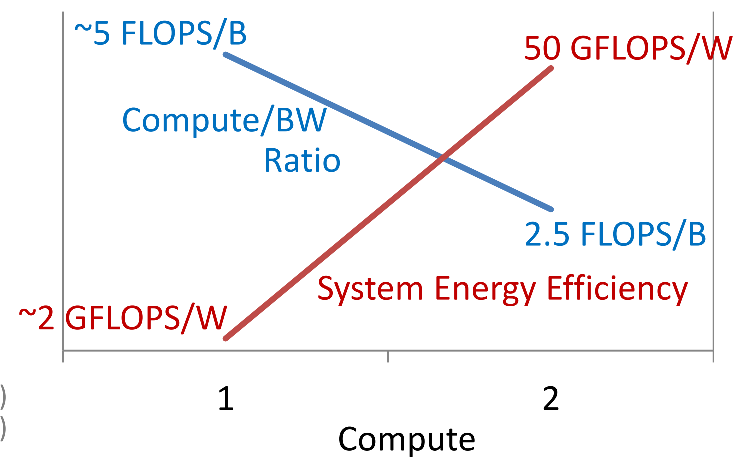


Source: ITRS, S. Keckler et al. [IEEE Micro, Sep-Oct. 2011], T. Vogelsang [MICRO 2010], T. Farrell [Salishan 2014]

EXASCALE COMPUTING CHALLENGES

- Energy is the key limiter
  - Exascale system
    - At 4TB/s, vast majority of node energy could be consumed by the memory system
  - 10x reduction in memory energy
  - 25x improvement in system energy efficiency
  - While improving performance
- Need to rethink compute and memory organization
  - Move computation closer to data
  - Specialized support for bandwidth-intensive applications
- Potential solution: processing-in-memory?





Today: ORNL Titan (node: AMD Opteron+Nvidia Tesla K20X)
2020: DOE FastForward RFP (issued May, 2012)
Source: ORNL, Nvidia, top500.org, LLNL

OUTLINE

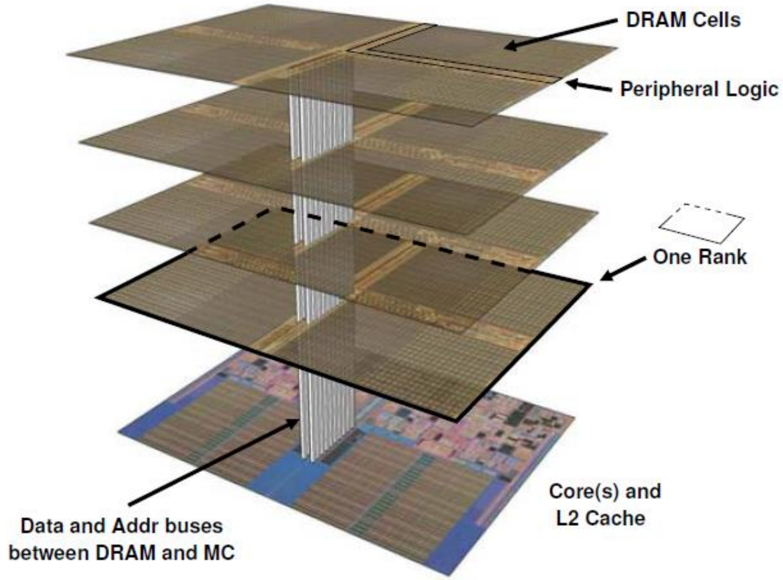
- Background
  - PIM prior work
  - Die-stacking
- PIM architecture and memory organization
- Applications
  - Graph apps, HPC apps, GPGPU benchmark
- PIM performance and energy model
- Evaluation of the PIM design choices
- Conclusion and further research

PIM RESEARCH – IN THE PAST

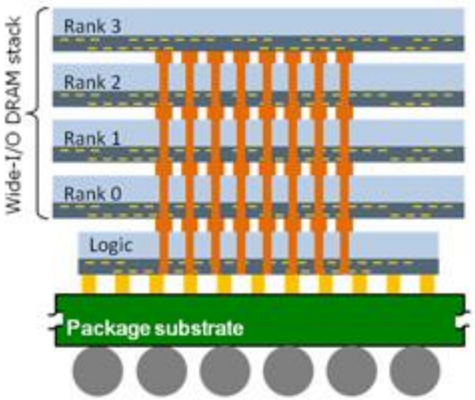
- Prior PIM research constrained by
  - Implementation technology
  - Non-traditional programming models
- Examples of prior work:
  - Integration of caches and computation
    - "A logic-in-memory computer" (1970)
  - Logic in DRAM processes
    - In-memory processors with reduced performance or highly specialized
    - Reduced DRAM due to presence of logic unit
  - Embedded DRAM in logic processes
    - Not cost-effective to have sufficient memory capacity, reduced DRAM density
- Recent work:
  - Micron's Automata Processor
  - 3D stacked processor for accelerating 3D ultrasound beamformation
  - Specialized in-stack processor to accelerate MapReduce workloads

3D INTEGRATION

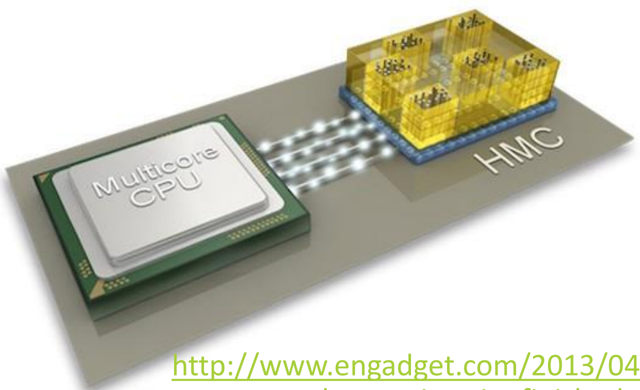
- Logic die under DRAM using TSVs
  - Higher bandwidth, lower access power
- Significant industry momentum
  - Recent JEDEC standards (HBM, Wide I/O 2)
  - Hybrid Memory Cube (HMC) consortjum
    - Micron, Samsung, IBM, ARM, Xilinx, Altera etc.



Gabe Loh, 3D-Stacked Memory Architectures for Multi-Core Processors, ISCA 2008



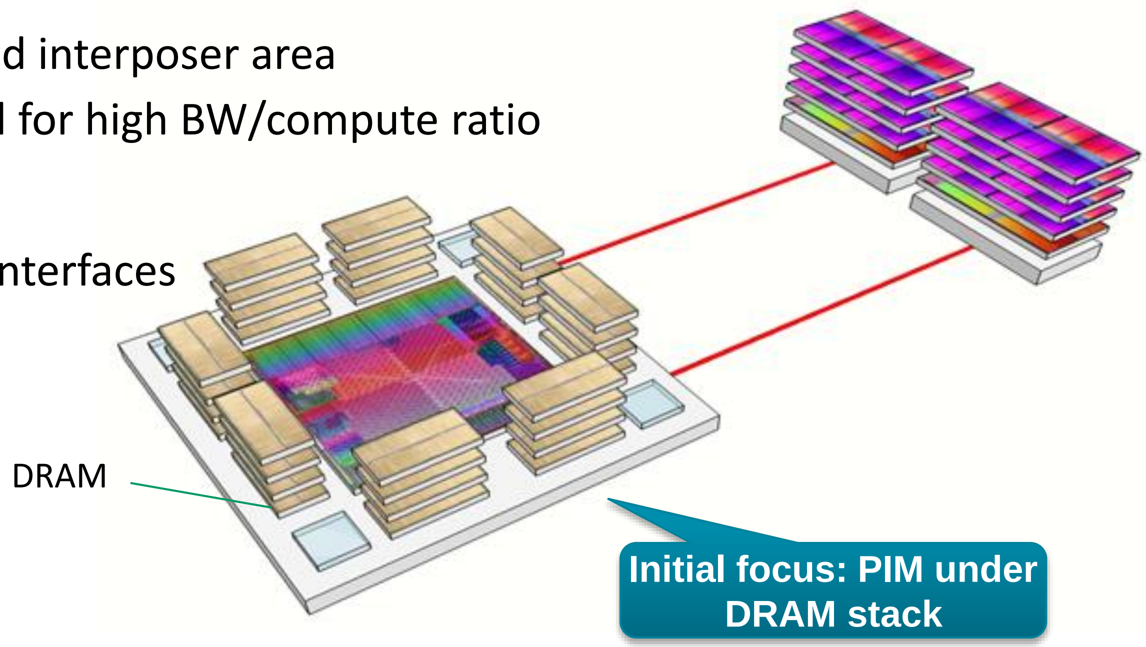
www.cadence.com/Community/blogs/ii/archive/2013/01/22/cadence-imec-test-methodology-enables-3d-ic-memory-on-logic.aspx



http://www.engadget.com/2013/04/03/hybrid-memory-cube-receives-its-finished-spec/

PIM RESEARCH – NEW PERSPECTIVE

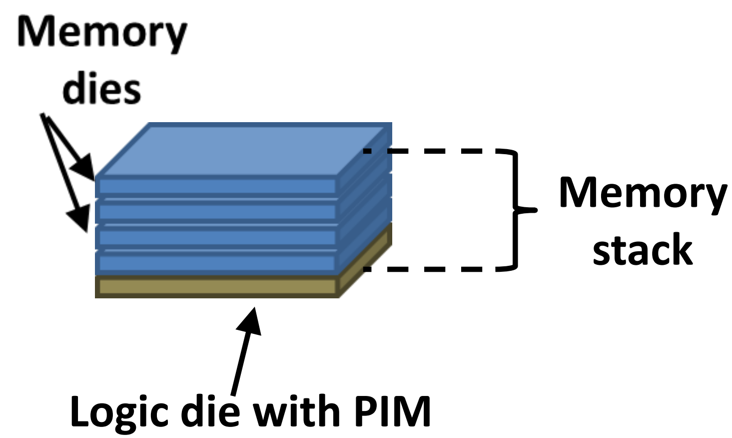
- New opportunity: logic die stacked with memory
  - Logic die needed anyway for signal redistribution and integrity
  - Potential for non-trivial compute
- Key benefits:
  - Reduce bandwidth bottlenecks
  - Improve energy efficiency
  - Increase compute for a fixed interposer area
  - Processor can be optimized for high BW/compute ratio
- Challenges:
  - Programming models and interfaces
  - Architectural tradeoffs
  - Application refactoring



GUIDING PRINCIPLES OF AMD'S PIM RESEARCH

AMD

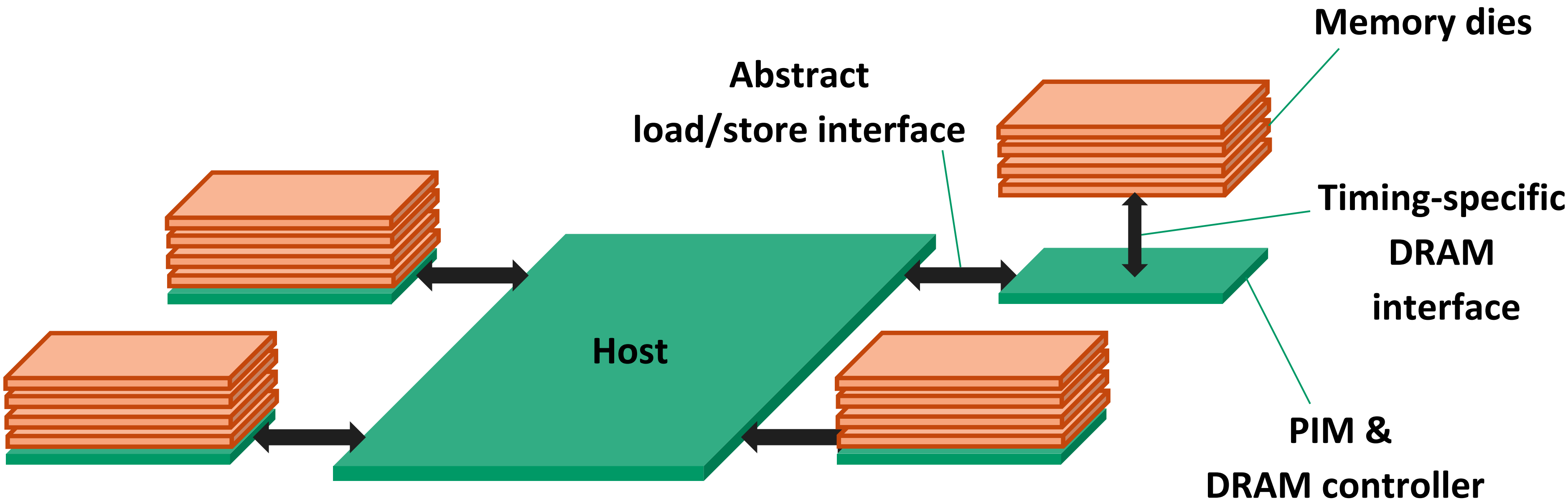
- Our focus
  - 3D die stacking
  - Use base logic die(s) in memory stack
    - General-purpose processors
    - Support familiar programming models



- Ease of use
  - Support familiar programming models
  - Build on HSA fundamentals
  - Any processor (host or PIM) can access all memory on node
  - No significant application change for host and PIM.
- Broad applicability
  - Across a broad range of applications
  - Viable across multiple market segments

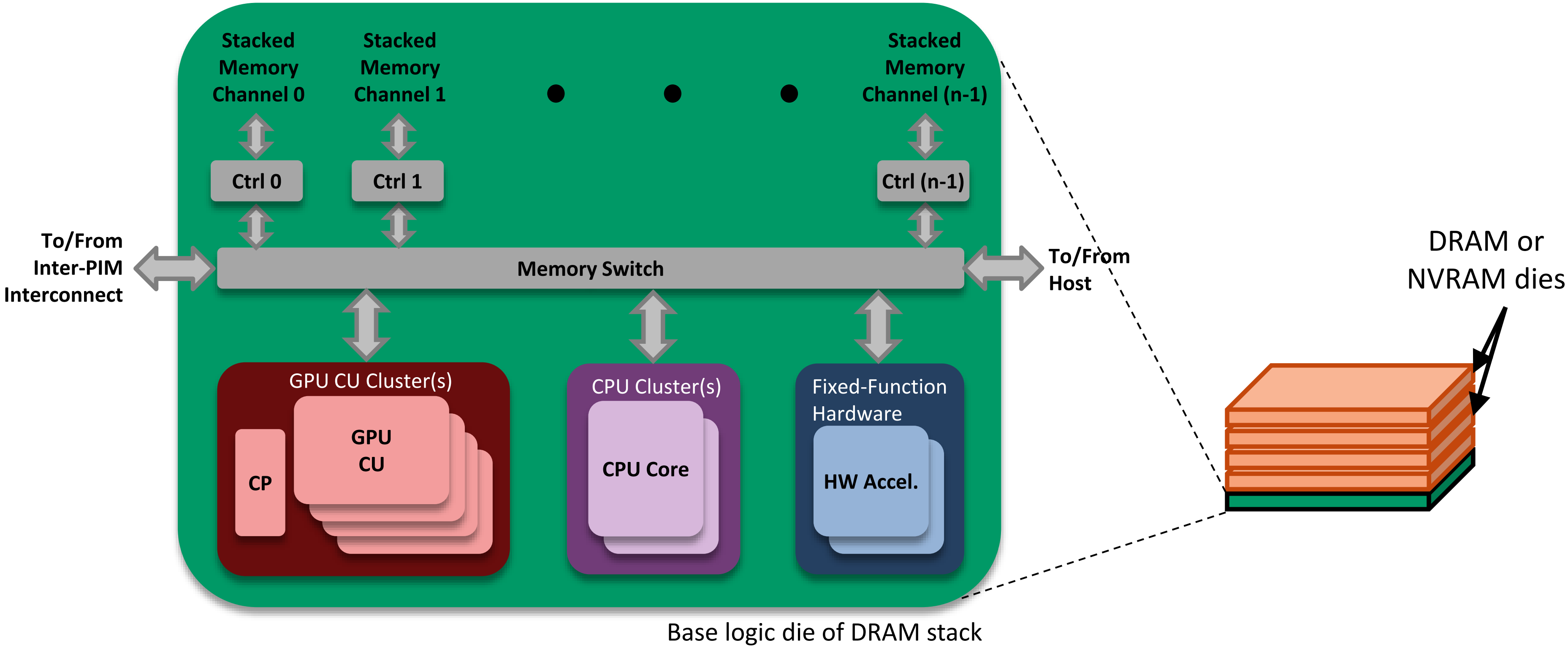
BASELINE PIM ARCHITECTURE
AN OVERVIEW

- An in-memory processor incorporated on the base die of each memory stack
- No DRAM die stacked on host processor



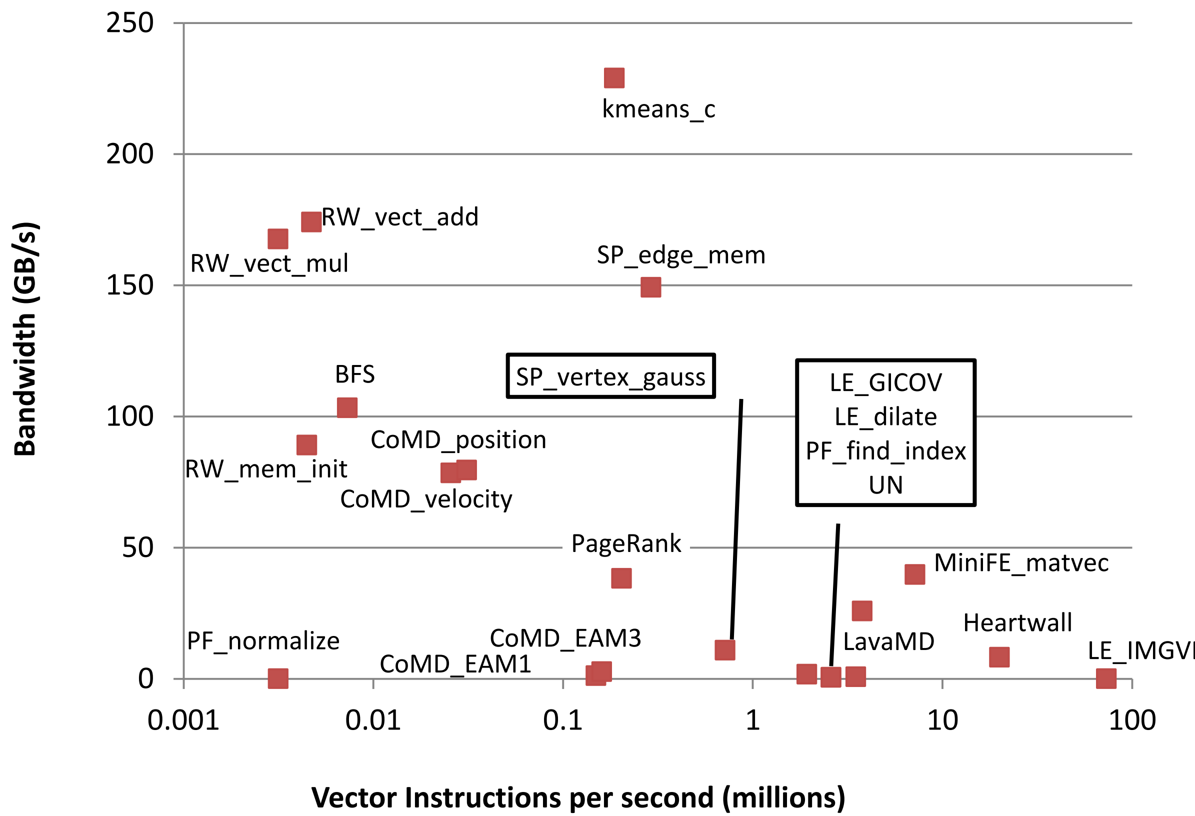
BASELINE PIM ARCHITECTURE

- GPU CUs provide compute throughput
- CPU cores provide control and flexibility
- Optional fixed-function accelerators



EXPLORE BREADTH OF APPLICABILITY OF PIM

- Broad set of kernels from HPC apps, graph algorithms, GPGPU benchmarks etc.
- Analyzed using PIM GPU performance and energy models



PF = ParticleFilter

SP = ShortestPath

RW = RandomWalk

LE = Leukocyte

WHY A NEW SIMULATOR?

- Why spend time building a new simulator when we could have used:
  - SimNow, TSIM, gem5, Multi2Sim, MARSSx86, PTLsim, Zesto, FeS2, RSIM, ZSIM, Graphite, Flexus, SESC, SST, GPGPUSim, MacSim, Simics+GEMS, SimpleScalar……
- Because they don't answer the question we want to ask
  - Runtime overhead too high
  - Changes take too long to implement
  - Memory overheads preclude large working sets
- As a result: they can't test the PIM design space on applications that matter
- PIM Simulator trades off some accuracy for major performance improvements

CHALLENGES OF MODELING FUTURE SYSTEMS

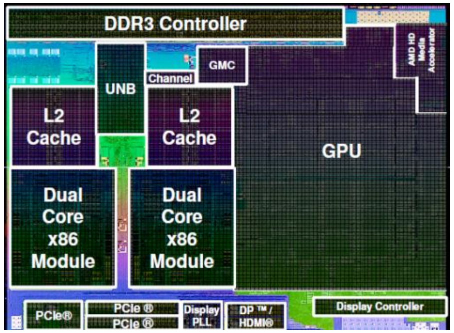
LARGE DESIGN SPACE TO EXPLORE

Ref: J. Greathouse et al. Simulation of Exascale Nodes through Runtime Hardware Monitoring, ModSim, 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

LARGE DESIGN SPACE TO EXPLORE

Heterogeneous Cores



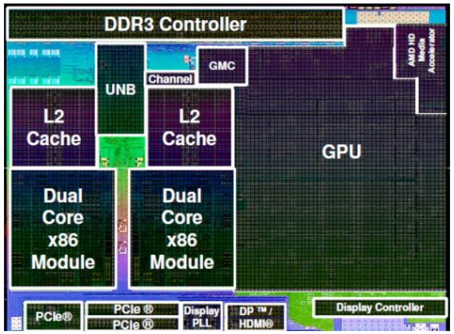
Composition? Size? Speed?

Ref: J. Greathouse et al. Simulation of Exascale Nodes through Runtime Hardware Monitoring, ModSim, 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

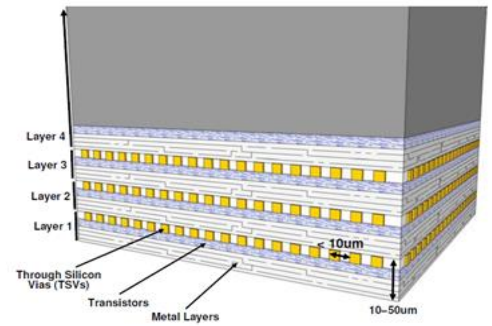
LARGE DESIGN SPACE TO EXPLORE

Heterogeneous Cores



Composition? Size? Speed?

Stacked Memories



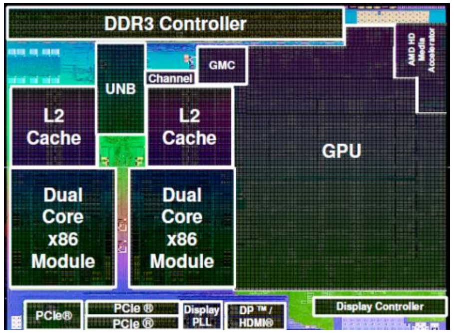
Useful? Compute/BW Ratio? Latency? Capacity? Non-Volatile?

Ref: J. Greathouse et al. Simulation of Exascale Nodes through Runtime Hardware Monitoring, ModSim, 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

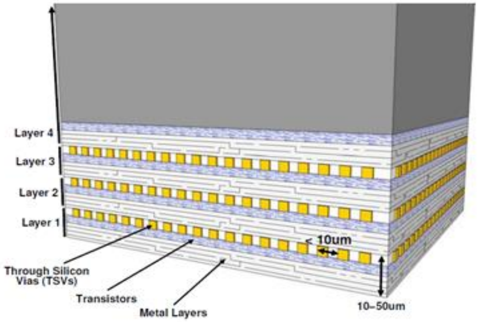
LARGE DESIGN SPACE TO EXPLORE

<u>Heterogeneous Cores</u>



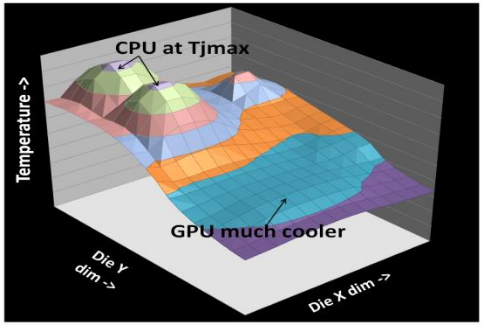
Composition? Size? Speed?

<u>Stacked Memories</u>



Useful? Compute/BW Ratio? Latency? Capacity? Non-Volatile?

<u>Thermal Constraints</u>



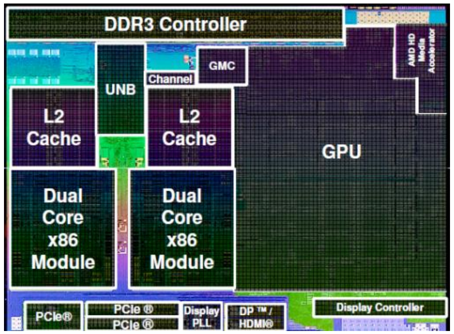
Power Sharing? Heat dissipation? Sprinting?

Ref: J. Greathouse et al. Simulation of Exascale Nodes through Runtime Hardware Monitoring, ModSim, 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

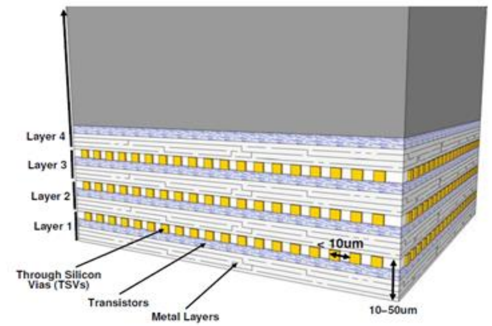
LARGE DESIGN SPACE TO EXPLORE

Heterogeneous Cores



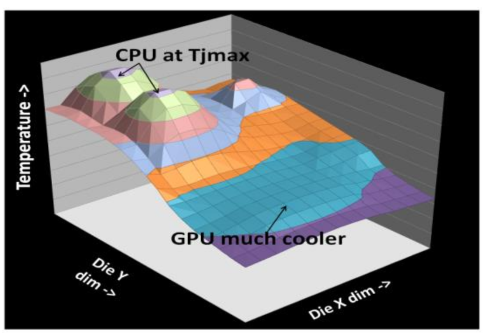
Composition? Size? Speed?

Stacked Memories



Useful? Compute/BW Ratio? Latency?
Capacity? Non-Volatile?

Thermal Constraints



Power Sharing? Heat dissipation?
Sprinting?

Software Co-Design

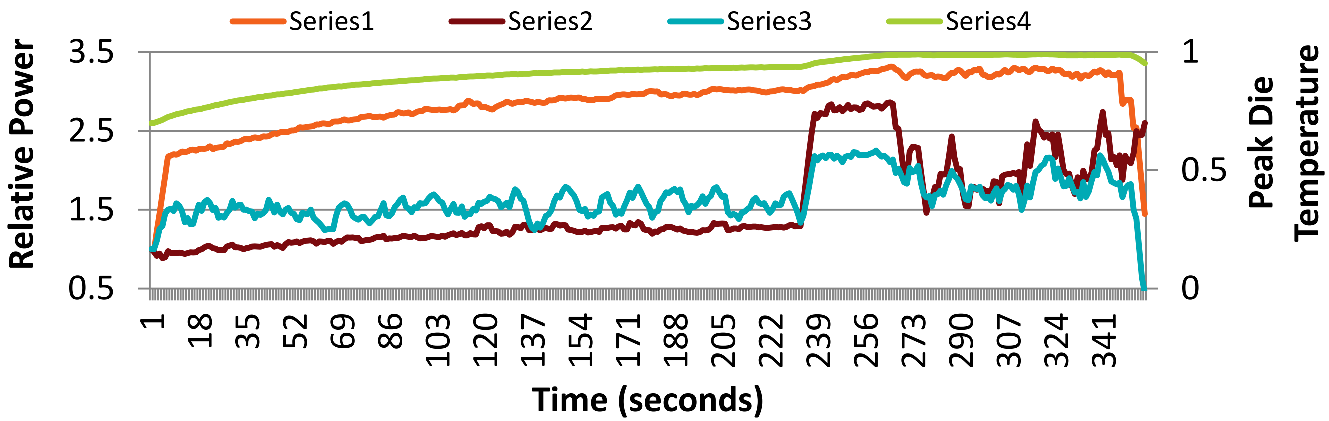
```
Real_t vol = volo[i]*vnew[i] ;
Real_t norm = (Real_t)(1.0) / ( vol + ptiny ) ;

Real_t dxj = (Real_t)(-0.25)*(SUM4(x0,x1,x5,x4) - SUM4(x3,x2,x6,x7)) ;
Real_t dyj = (Real_t)(-0.25)*(SUM4(y0,y1,y5,y4) - SUM4(y3,y2,y6,y7)) ;
Real_t dzj = (Real_t)(-0.25)*(SUM4(z0,z1,z5,z4) - SUM4(z3,z2,z6,z7)) ;

Real_t dxi = (Real_t)( 0.25)*(SUM4(x1,x2,x6,x5) - SUM4(x0,x3,x7,x4)) ;
Real_t dyi = (Real_t)( 0.25)*(SUM4(y1,y2,y6,y5) - SUM4(y0,y3,y7,y4)) ;
Real_t dzi = (Real_t)( 0.25)*(SUM4(z1,z2,z6,z5) - SUM4(z0,z3,z7,z4)) ;

Real_t dxk = (Real_t)( 0.25)*(SUM4(x4,x5,x6,x7) - SUM4(x0,x1,x2,x3)) ;
Real_t dyk = (Real_t)( 0.25)*(SUM4(y4,y5,y6,y7) - SUM4(y0,y1,y2,y3)) ;
Real_t dzk = (Real_t)( 0.25)*(SUM4(z4,z5,z6,z7) - SUM4(z0,z1,z2,z3)) ;
```

New algorithms? Data placement?
Programming models?

Ref: J. Greathouse et al. Simulation of Exascale Nodes through Runtime Hardware Monitoring, ModSim, 2013

CHALLENGES OF MODELING FUTURE SYSTEMS
WHY DOES SIMULATOR PERFORMANCE MATTER?

- Need to run long enough to trigger interesting memory phenomena
  - Working sets >> stacked memories of 100s of MB to multiple GB
- Run long enough to observe power and thermal effects
  - Example measured on a real heterogeneous processor

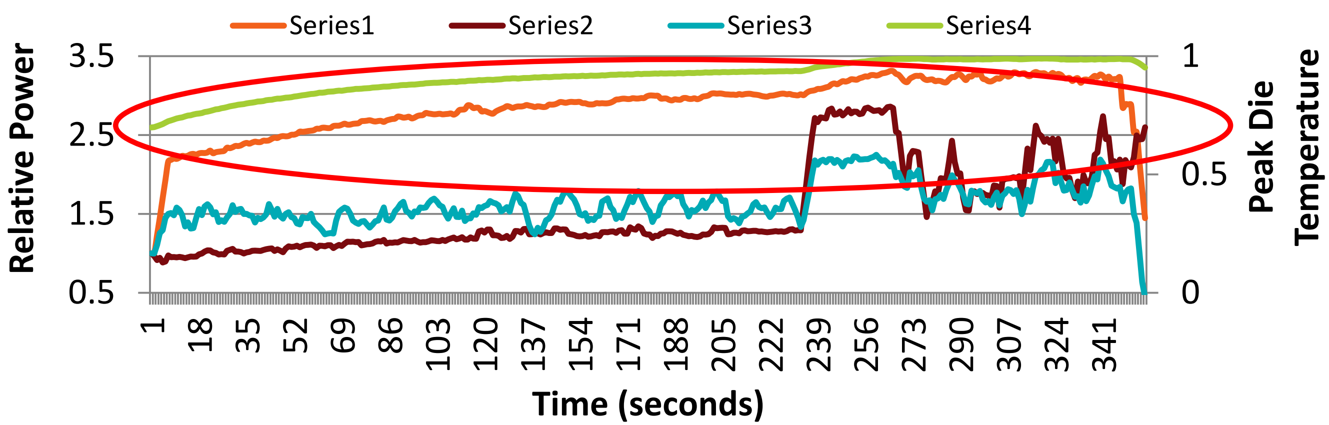


Ref: I. Paul et al., "Cooperative Boosting: Needy Versus Greedy Power Management", ISCA 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

WHY DOES SIMULATOR PERFORMANCE MATTER?

- Need to run long enough to trigger interesting memory phenomena
  - Working sets >> stacked memories of 100s of MB to multiple GB
- Run long enough to observe power and thermal effects
  - Example measured on a real heterogeneous processor



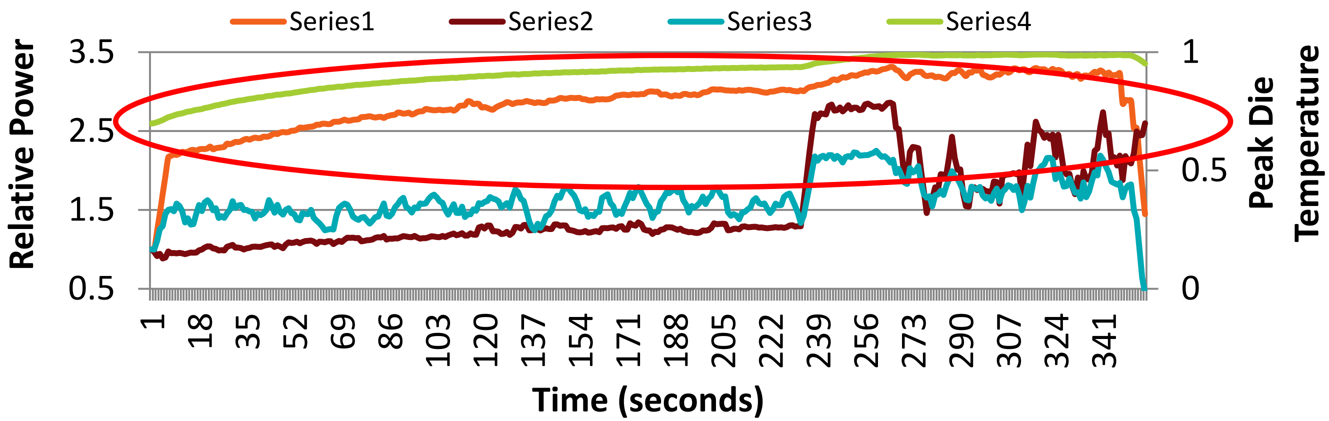
  - ~2.5 trillion CPU instructions, ~60 trillion GPU operations

Ref: I. Paul et al., "Cooperative Boosting: Needy Versus Greedy Power Management", ISCA 2013

CHALLENGES OF MODELING FUTURE SYSTEMS

WHY DOES SIMULATOR PERFORMANCE MATTER?

- Need to run long enough to trigger interesting memory phenomena
  - Working sets >> stacked memories of 100s of MB to multiple GB
- Run long enough to observe power and thermal effects
  - Example measured on a real heterogeneous processor



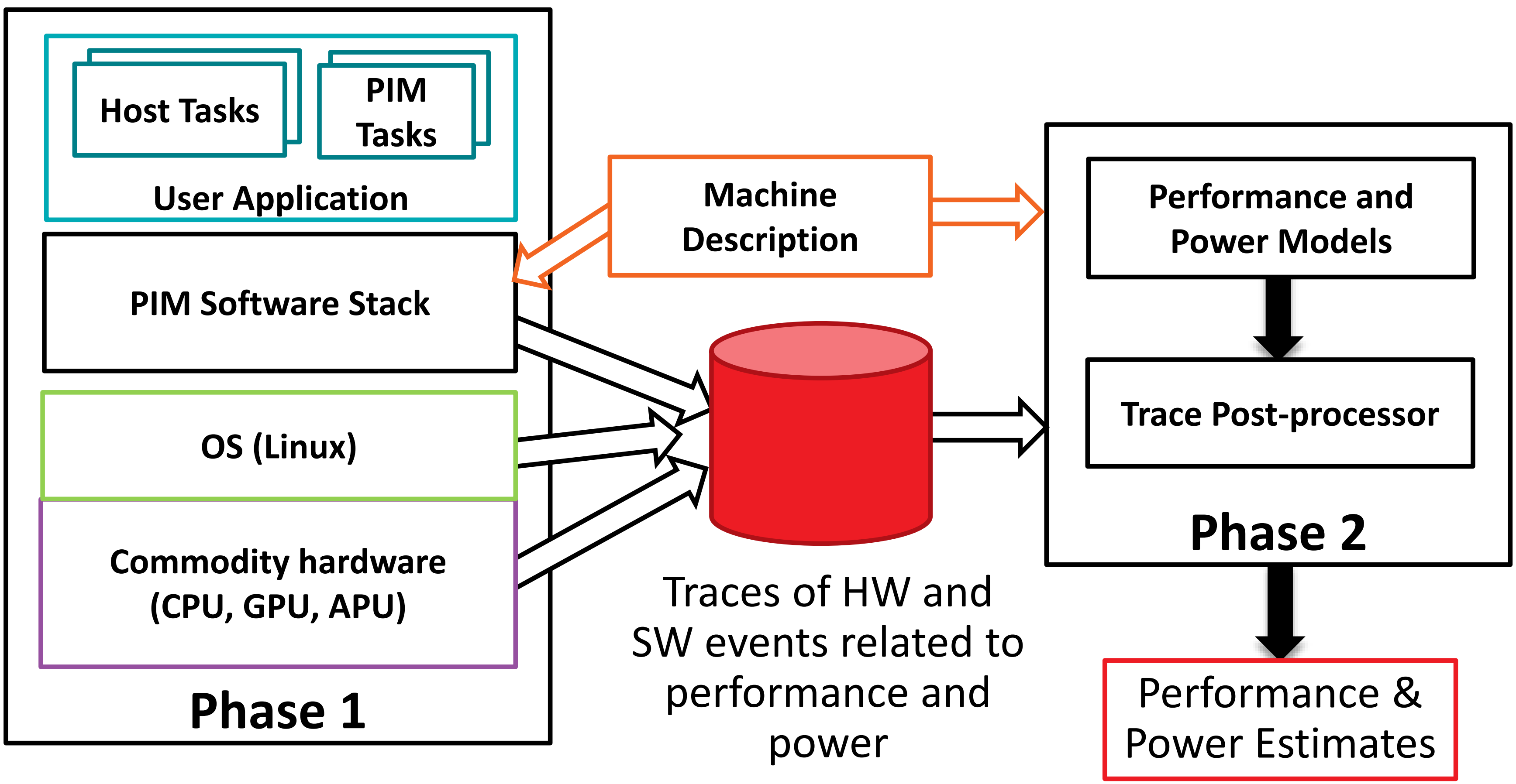
  - ~2.5 trillion CPU instructions, ~60 trillion GPU operations
- Applications of interest can be large
  - Scaled studies can be challenging and misleading for complex applications

Ref: I. Paul et al., "Cooperative Boosting: Needy Versus Greedy Power Management", ISCA 2013

PIM SIMULATOR OVERVIEW

MULTI-STAGE PERFORMANCE ESTIMATION PROCESS

AMD



ML-BASED PERFORMANCE MODEL

- Execution time <--- *F* (architecture, application)
  - Kernel time (and power) depends on:
    - Underlying HW configuration
    - Algorithms and data structures of the application

- PIM GPU Architecture is represented by:
  - Number of CUs (8, 16, 32)
  - Processor frequency (500 – 100 – 1000 MHz)
  - Memory Bandwidth (500 – 100 – 1300 MHz)

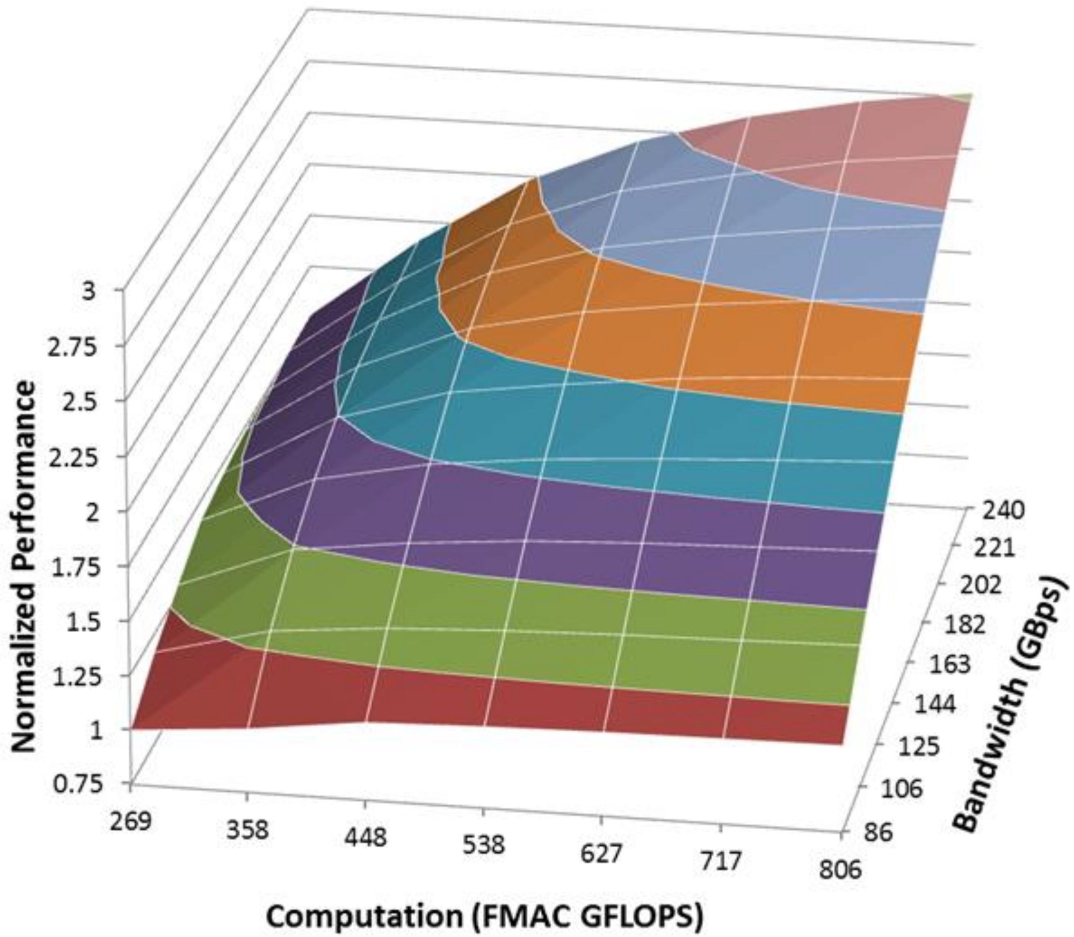
162 design points

- Application kernel is represented by feature vectors
  - Dynamic CodeXL/Sprofile data, derived from HW counters.

- Goals:
  - Learn scaling pattern in offline training
  - Estimate runtime and power for online prediction

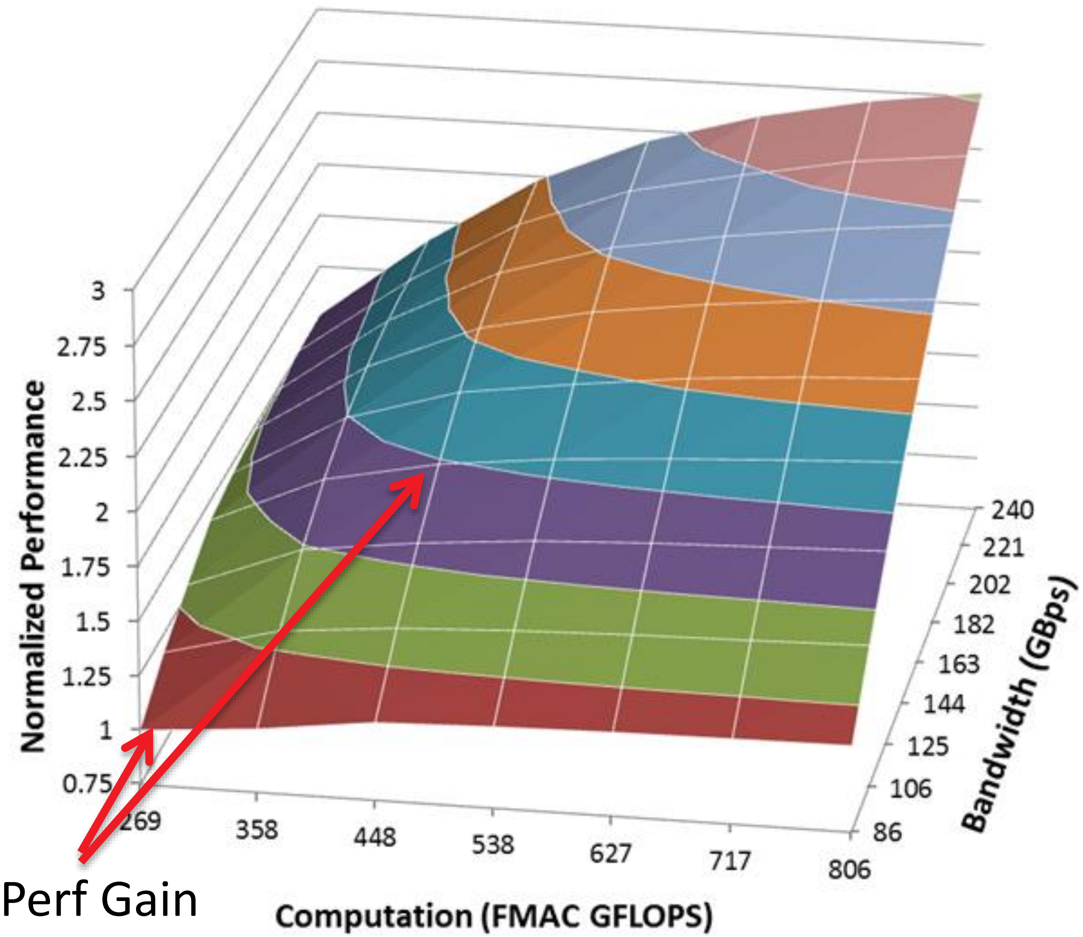
PERFORMANCE MODEL – OFFLINE LEARNING

- Gathering data
  - 70 OpenCL kernels
  - Each kernel: 162 hw configurations → 162 pairs of execution time & performance counter feature vector



PERFORMANCE MODEL – OFFLINE LEARNING

- Gathering data
  - 70 OpenCL kernels
  - Each kernel: 162 hw configurations → 162 pairs of execution time & performance counter feature vector

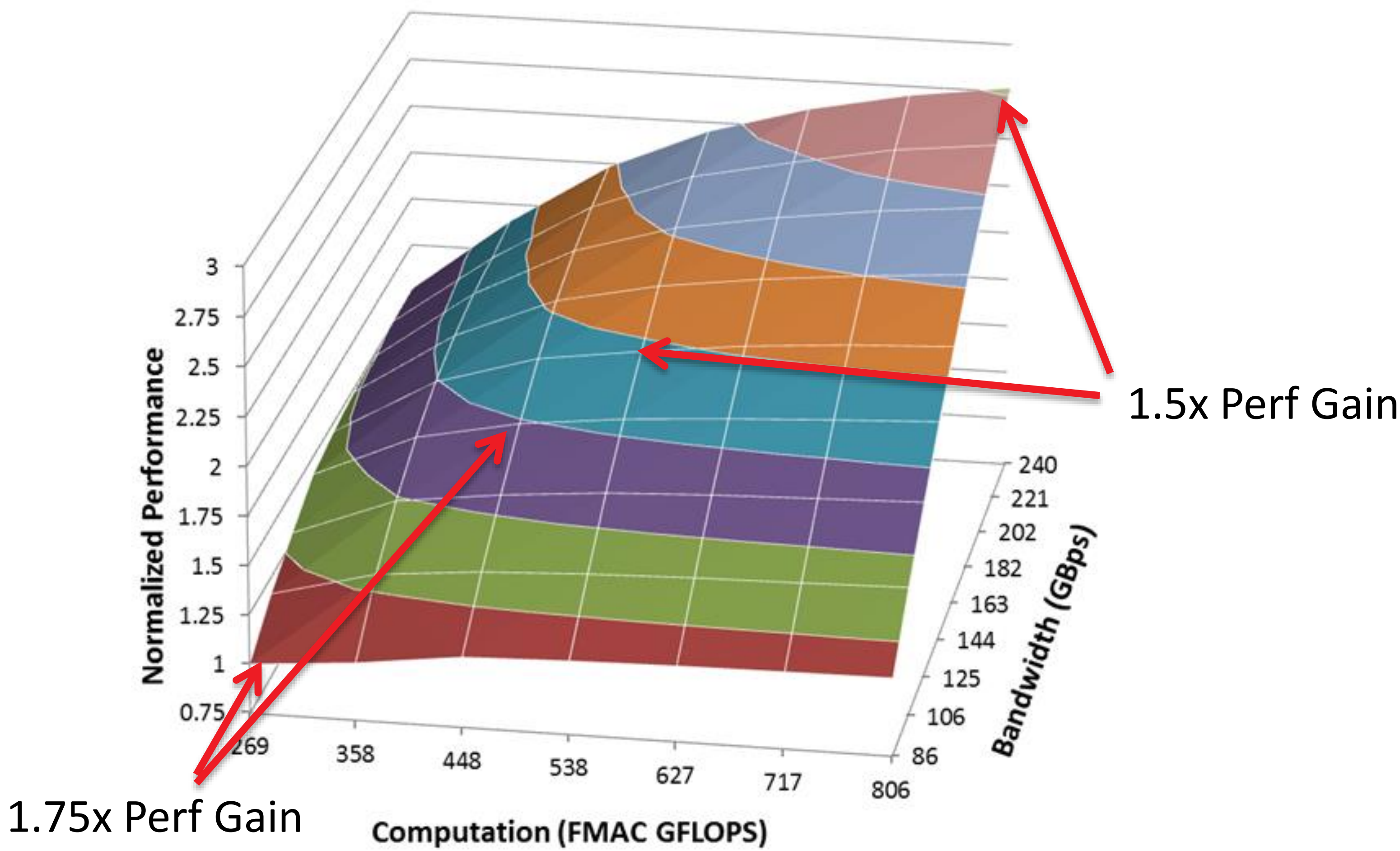


1.75x Perf Gain

PERFORMANCE MODEL – OFFLINE LEARNING

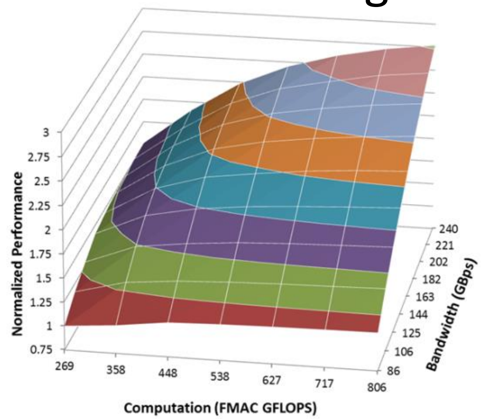
AMD

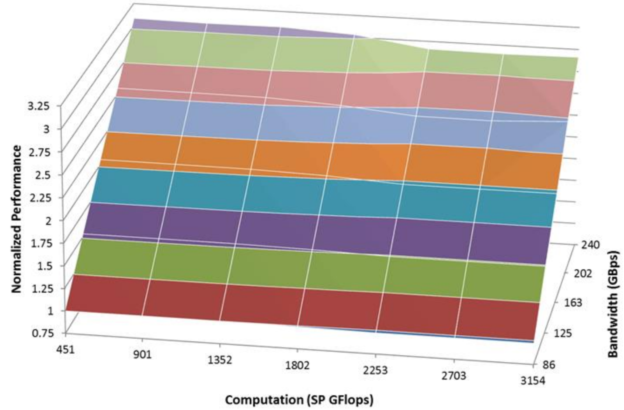
- Gathering data
  - 70 OpenCL kernels
  - Each kernel: 162 hw configurations → 162 pairs of execution time & performance counter feature vector

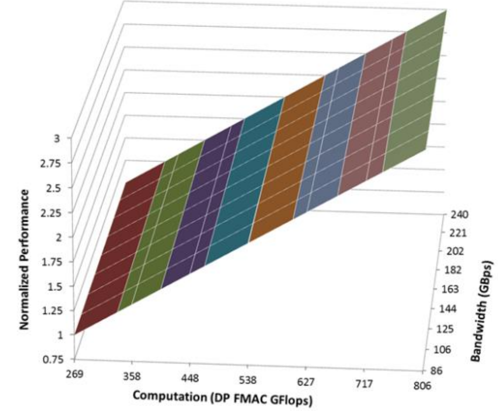


PERFORMANCE MODEL

- Offline learning – clustering of the scaling pattern

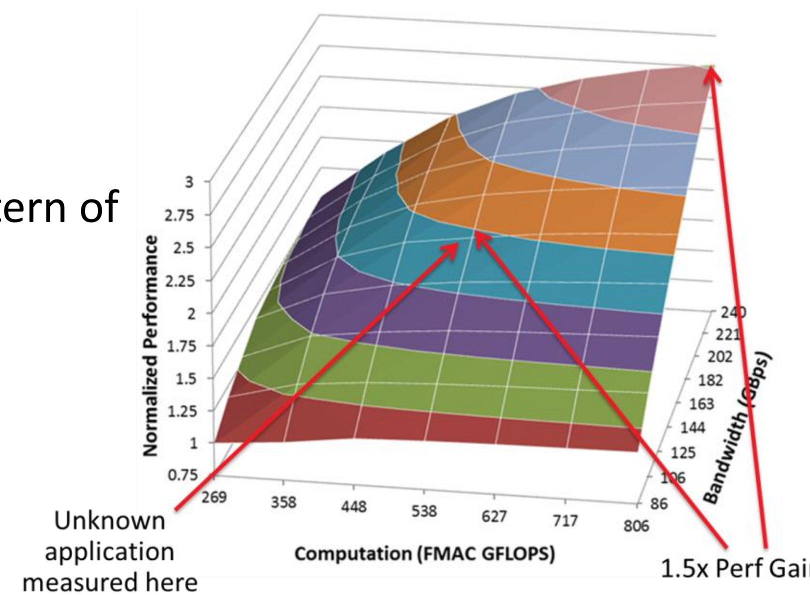






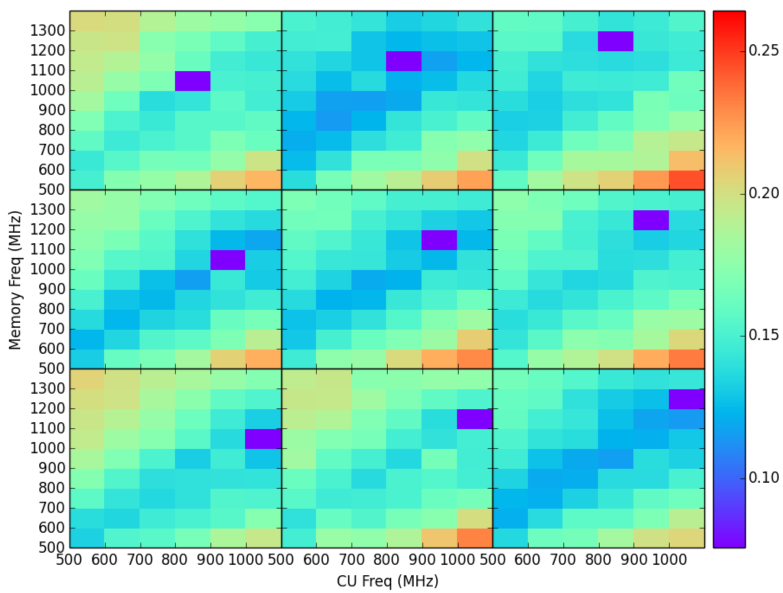
Feature vector: VALUUtilization, VALUBusy, SALUBusy, MemUnitBusy, MemUnitStalled, CacheHit, …

- Online classification and prediction
  – Classify the feature vector of the new kernel
  – Performance projection with the scaling pattern of this cluster

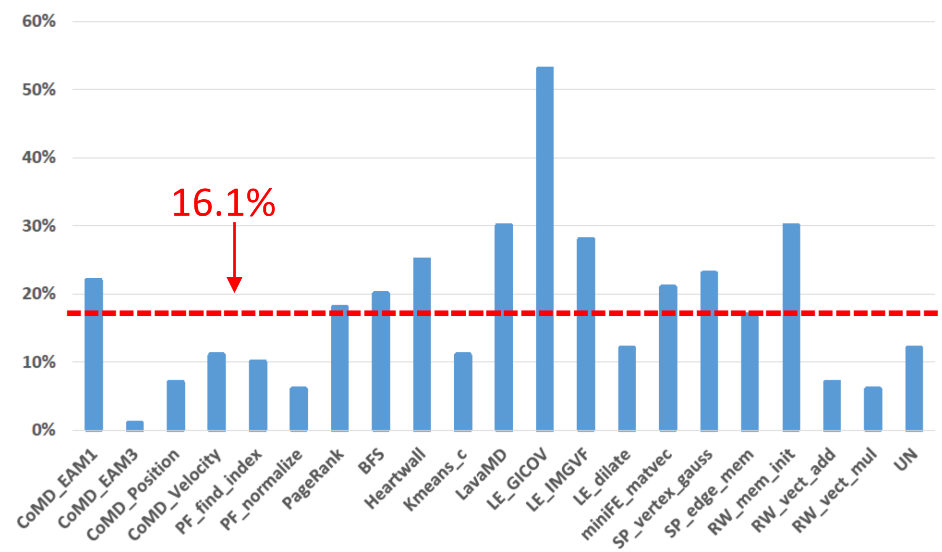


PERFORMANCE MODEL VALIDATION

- 69 training kernels; leave one kernel out → prediction → validation
  - relative error between the prediction and real processing: accuracy verification
- Make predictions for all other HW design points from each design point
  - Variation of #CUs, bandwidth, engine frequency -> 162 operating points -> 162*161 (26K) data points on the 3D grid for each kernel!



Average prediction error - subset



Individual benchmark prediction errors

TECHNOLOGY AND CONFIGURATIONS

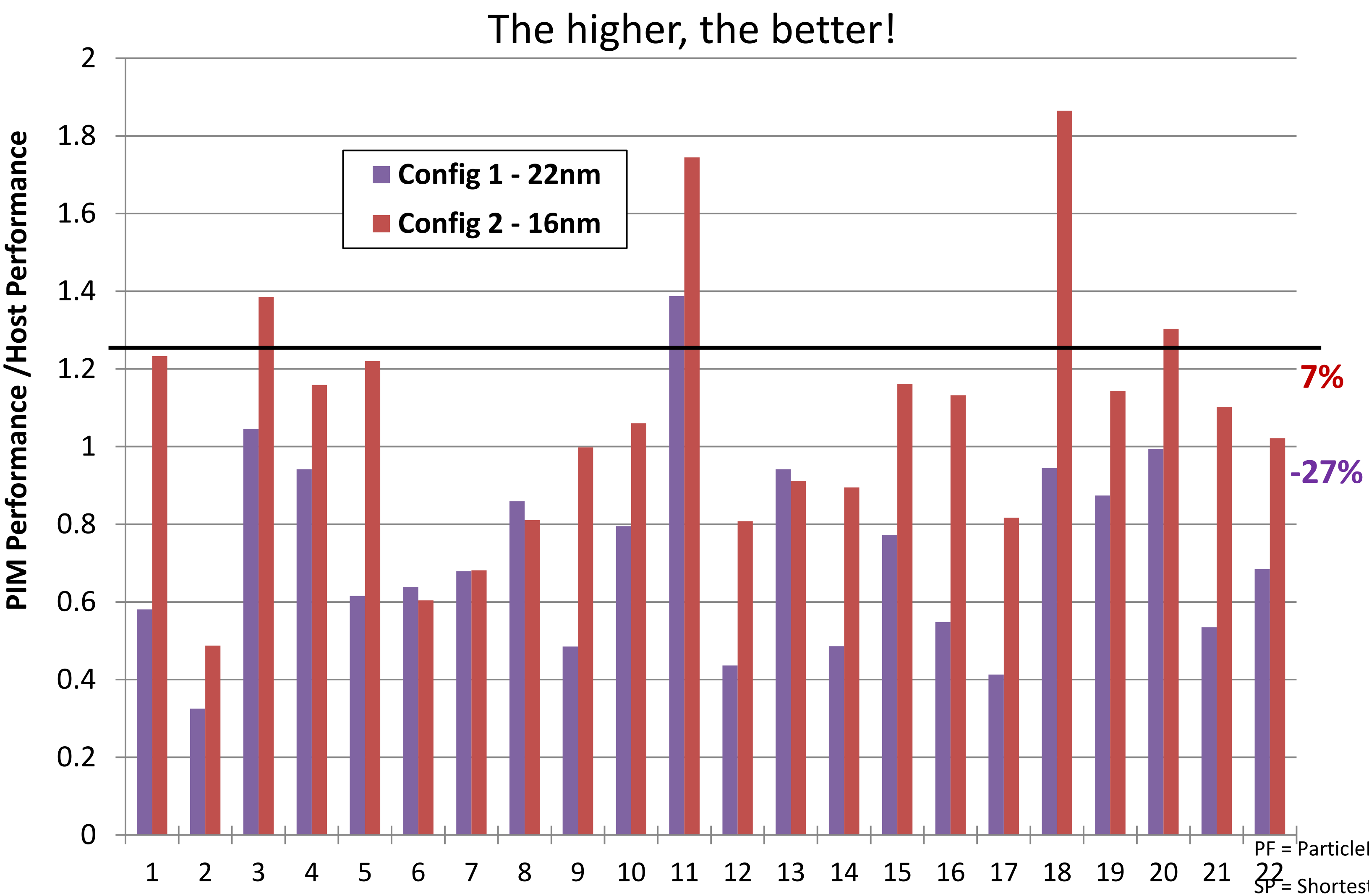
- Evaluated for 22nm and 16nm
  - Explore viability prior to Exascale timeframe
  - Identify tech transfer opportunities
- Design points and technology scaling
  - PIM: limited by DRAM footprint and 10W/PIM
  - Host: extrapolate current trends (assumes HMC-like DRAM interface)

TECHNOLOGY AND CONFIGURATIONS

- Evaluated for 22nm and 16nm
  - Explore viability prior to Exascale timeframe
  - Identify tech transfer opportunities
- Design points and technology scaling
  - PIM: limited by DRAM footprint and 10W/PIM
  - Host: extrapolate current trends (assumes HMC-like DRAM interface)

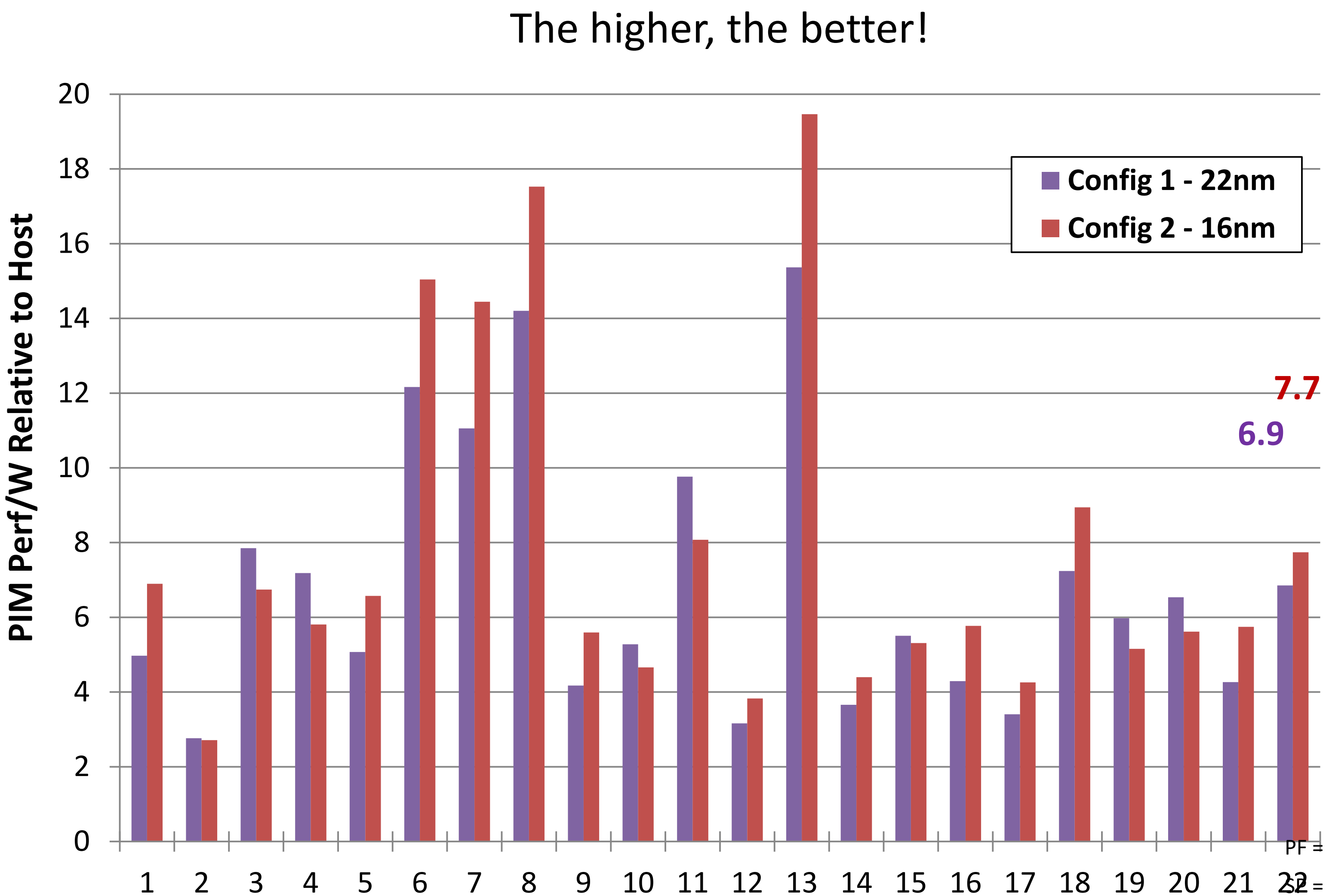
	Baseline	22nm		16nm	
	dGPU	Host	PIM	Host	PIM
Freq	1GHz	1GHz	650MHz	1GHz	650MHz
Number of CUs	32	32	8	64	12
Number of memory stacks		2		4	
DRAM BW (GB/s)		160	640	160	640
Dynamic power scaling	1.00	0.61	0.25	0.41	0.17
Memory Energy (pJ/64b)		522	159	520	155

PIM CAN BE PERFORMANCE-COMPETITIVE WITH HOST



PF = ParticleFilter
SP = ShortestPath
RW = RandomWalk

SIGNIFICANT PERF/W IMPROVEMENTS



PF = ParticleFilter

SP = ShortestPath

RW = RandomWalk

CONCLUSIONS AND FURTHER RESEARCH

- "Computing" is increasingly about data and data movement
  - Exploit locality to reduce wasteful data movement
  - Specialization to improve efficiency
  - PIM potentially provides significant reductions in off-chip traffic.
- TOP-PIM implemented using 3D die-stacking feasible in near future.
  - Efficiently utilize the high bandwidth available in local stack
  - Programmability -> Support a broad range of applications
  - Performance and energy efficiency of PIM vs Host
    - At 22nm, 27% performance degradation, 76% reduction in EDP
    - At 16nm, 7% performance gain, 85% reduction in EDP.
- Future Work:
  - High level programming models to express data-compute affinity.
  - Data movement management and task scheduling for host and PIMs.
  - Evaluation of alternative PIM organizations and design options.

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION