DVFS Space Exploration in Power-Constrained Processing-in-Memory Systems

Marko Scrbak and Krishna M. Kavi

Computer Systems Research Laboratory

Department of Computer Science & Engineering

University of North Texas, USA

Joseph Greathouse and Nuwan Jayasena

AMD Research - Advanced Micro Devices, Inc., USA

Introduction

- End of Dennard scaling
  - Power and thermal challenges for modern processor design
  - Heterogeneous computing and sophisticated DVFS techniques can increase computational efficiency
  - Memory bandwidth becomes a bottleneck
- 3D-stacked memory, e.g. HMC (Micron), HBM (JEDEC standard)
  - Offer high-bandwidth, lower latency, lower energy/access
- Place compute logic within 3D-stack: Processing-in-Memory (PIM)
  - Relax off-chip bandwidth requirements
  - Minimize power consumption by reducing excess data movement

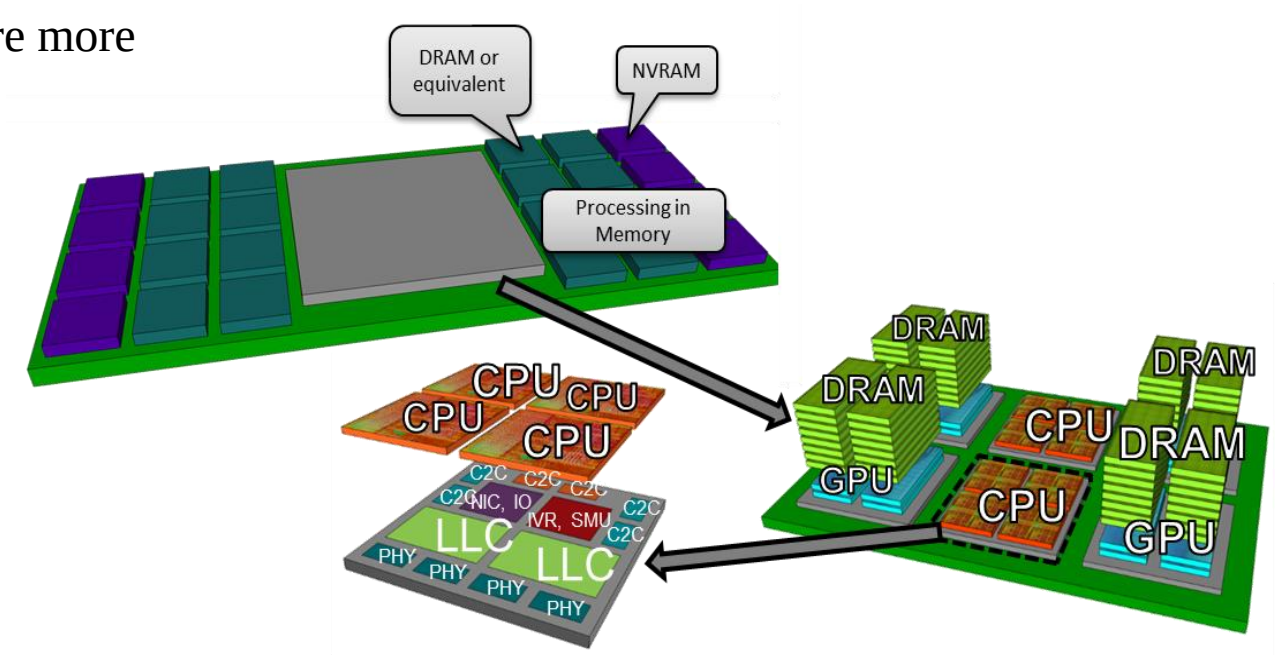
We show that compute intensive kernels should execute on host

bandwidth intensive applications should execute on PIMs.

Even for compute intensive kernels, PIMs are preferred in power constrained environments

Introduction

- Possible HPC Node Architecture
  - On-package stacked memory with GPUs (APUs)
  - Off-package board-level memory with PIM
  - Off-package memory accesses are more expensive in terms of latency and energy



Introduction

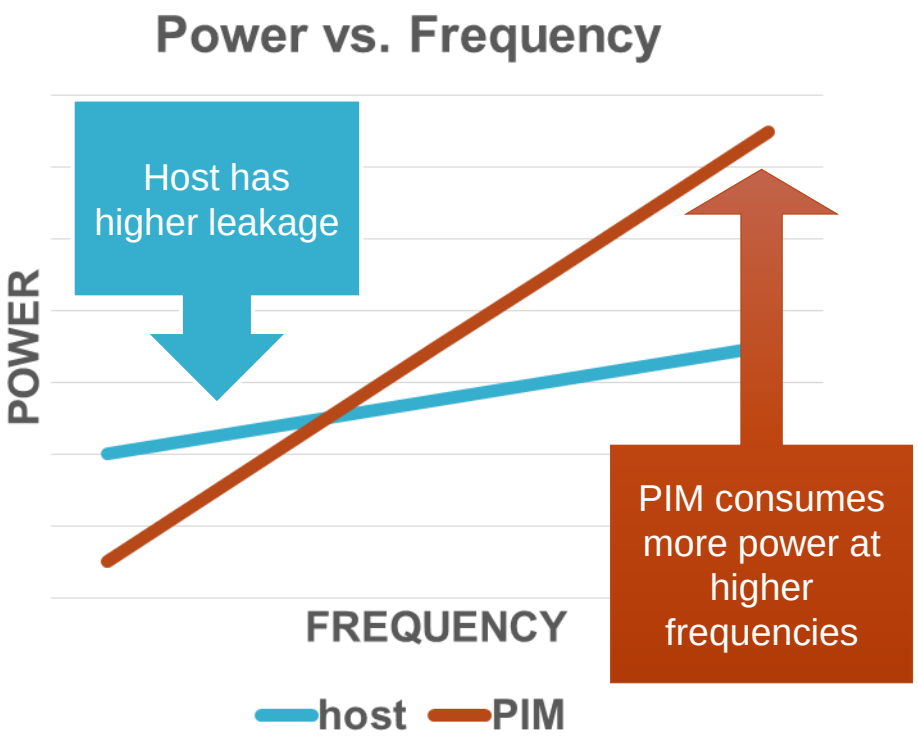
- PIMs can be implemented using a low leakage processes
  - No need for high performance (high frequency) as the performance improvement would be compensated by exploiting high in-stack bandwidth
- What type of architecture to use for PIMs?
- Previously (ARCS-2015) we evaluated 16 ARM cores per stack. Here we evaluate GPUs as PIMs
- GPUs as PIM
  - Energy efficient, high compute and memory throughput, mature programming models, uniform power dissipation
- PIMs target memory intensive applications
  - Locality based computing
  - Bandwidth constrained applications
  - Performance gain from high bandwidth and data locality
- Less compute intensive than the host APU
  - No need for high CU count and high engine frequency
  - More energy efficient than host

In this work we evaluate optimal choice between PIMs and Host APUs-- for application kernels

Motivation

- Different power and performance characteristics for PIM and host
- PIM can compensate for low frequency by exploiting high memory bandwidth
- Host can run at high frequencies, maximizing performance for compute intensive applications

Cartoon Example



DVFS optimization - example

- Optimizing for maximum performance, minimum power and minimum ED^2
- An Example: miniFE

	dotprod	matvec	waxby
HOST		1GHz	1GHz
PIM	600MHz		

MAX PERFORMANCE

	dotprod	matvec	waxby
HOST			
PIM	400MHz	400MHz	400MHz

MIN POWER

	dotprod	matvec	waxby
HOST			
PIM	500MHz	400MHz	500MHz

MIN ED2

DVFS optimization - example

- Optimizing for maximum performance, minimum power and minimum ED^2
- An Example: miniFE

	dotprod	matvec	waxby
HOST		1GHz	1GHz
PIM	600MHz		

MAX PERFORMANCE

	dotprod	matvec	waxby
HOST			
PIM	400MHz	400MHz	400MHz

MIN POWER

	dotprod	matvec	waxby
HOST			
PIM	500MHz	400MHz	500MHz

MIN ED2

We use AMD in-house simulator to gather performance statistics for host and PIM

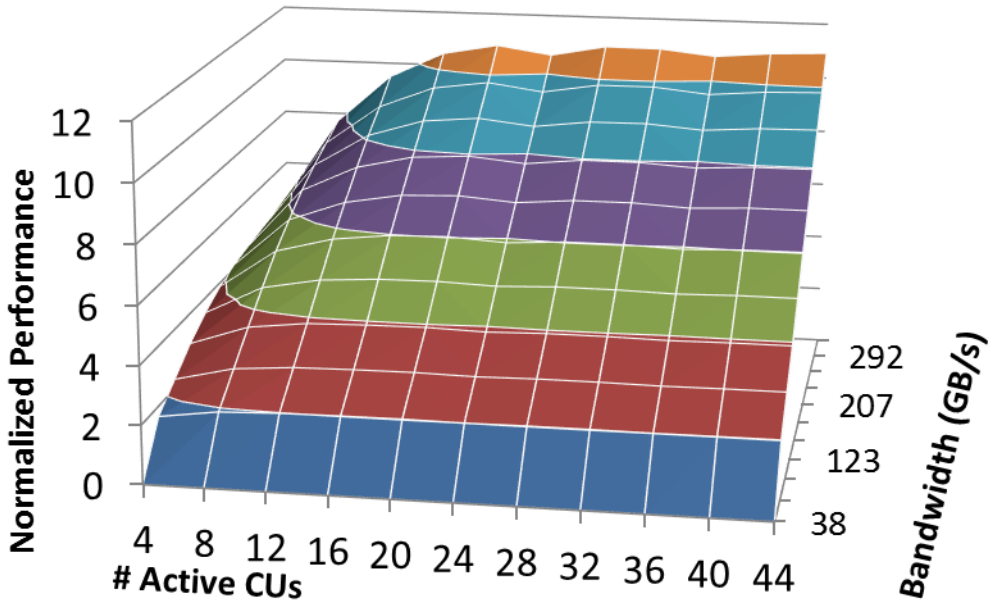
We developed power model for PIM based on host and technology roadmaps

Dynamic power – DVFS characteristics for host and PIM

Leakage power – relative difference in leakage power between host and PIM

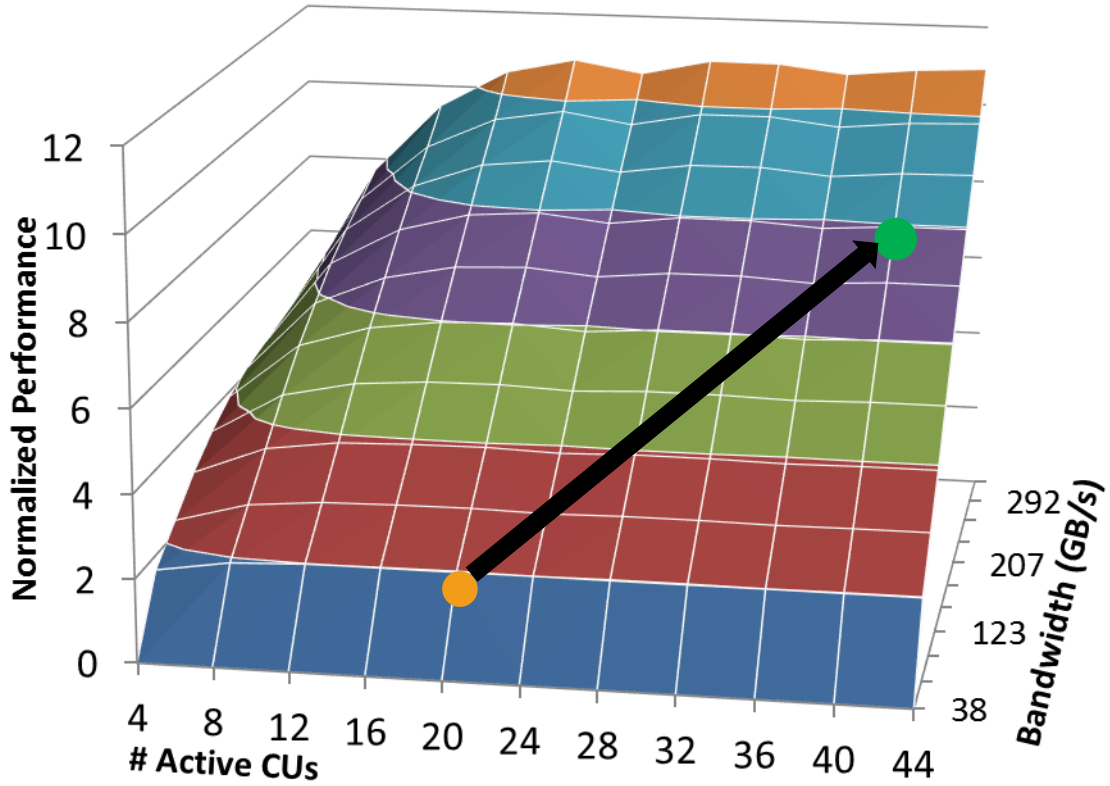
Performance model

- How do we estimate GPU kernel performance for some future hardware configuration?
- If we know how the performance scales with (current) HW resources (CUs, memory bandwidth, frequency) then we can estimate the performance using performance scaling curves for a target HW configuration



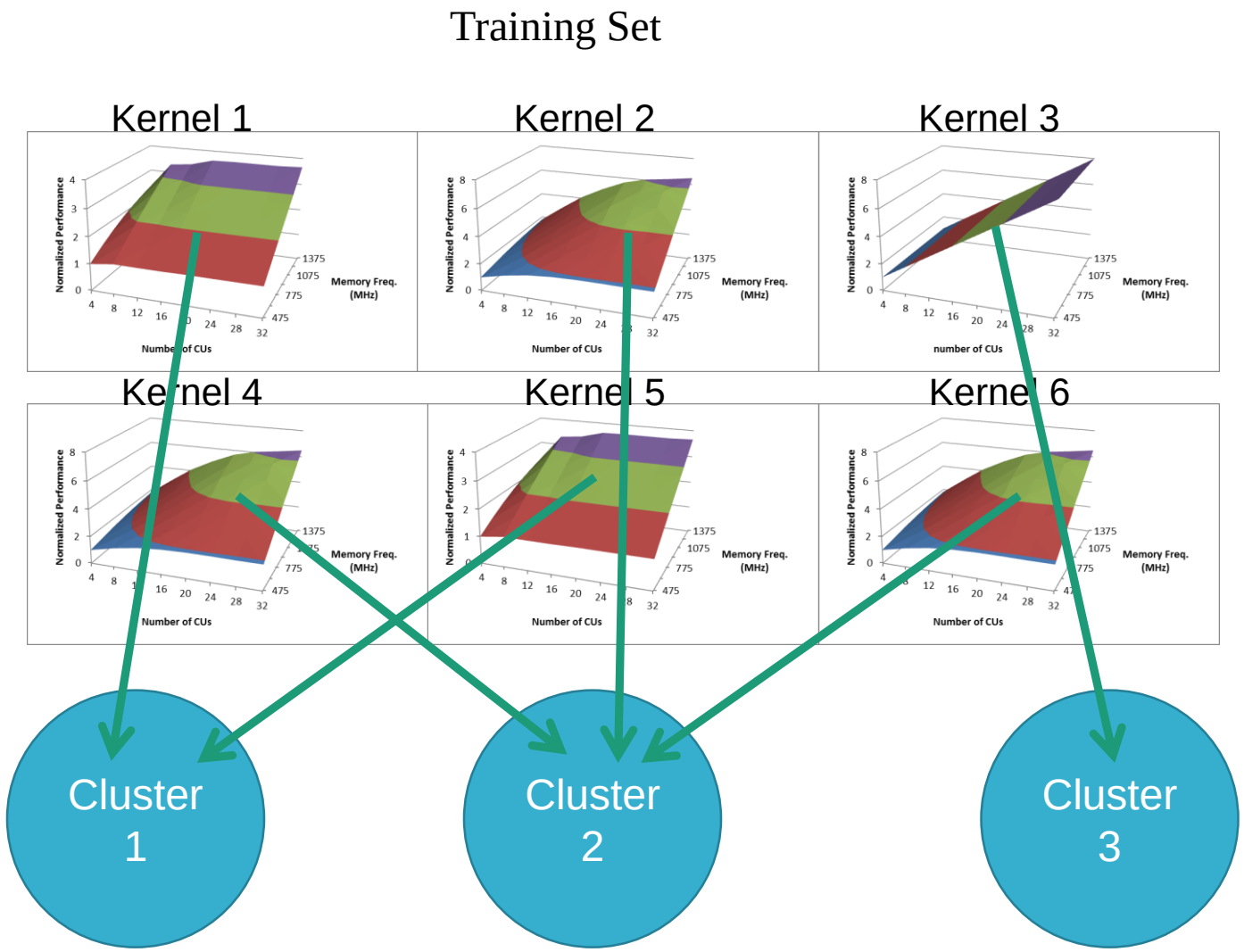
- We can create a performance scaling curve by running the kernel on a GPU and change HW configurations (CUs, Mem. Bandwidth, Frequency).
- The plot is generated by running each kernel on 720 different hardware configurations
- Using these plots we can obtain performance for other hardware configurations

GPU Performance SCALING



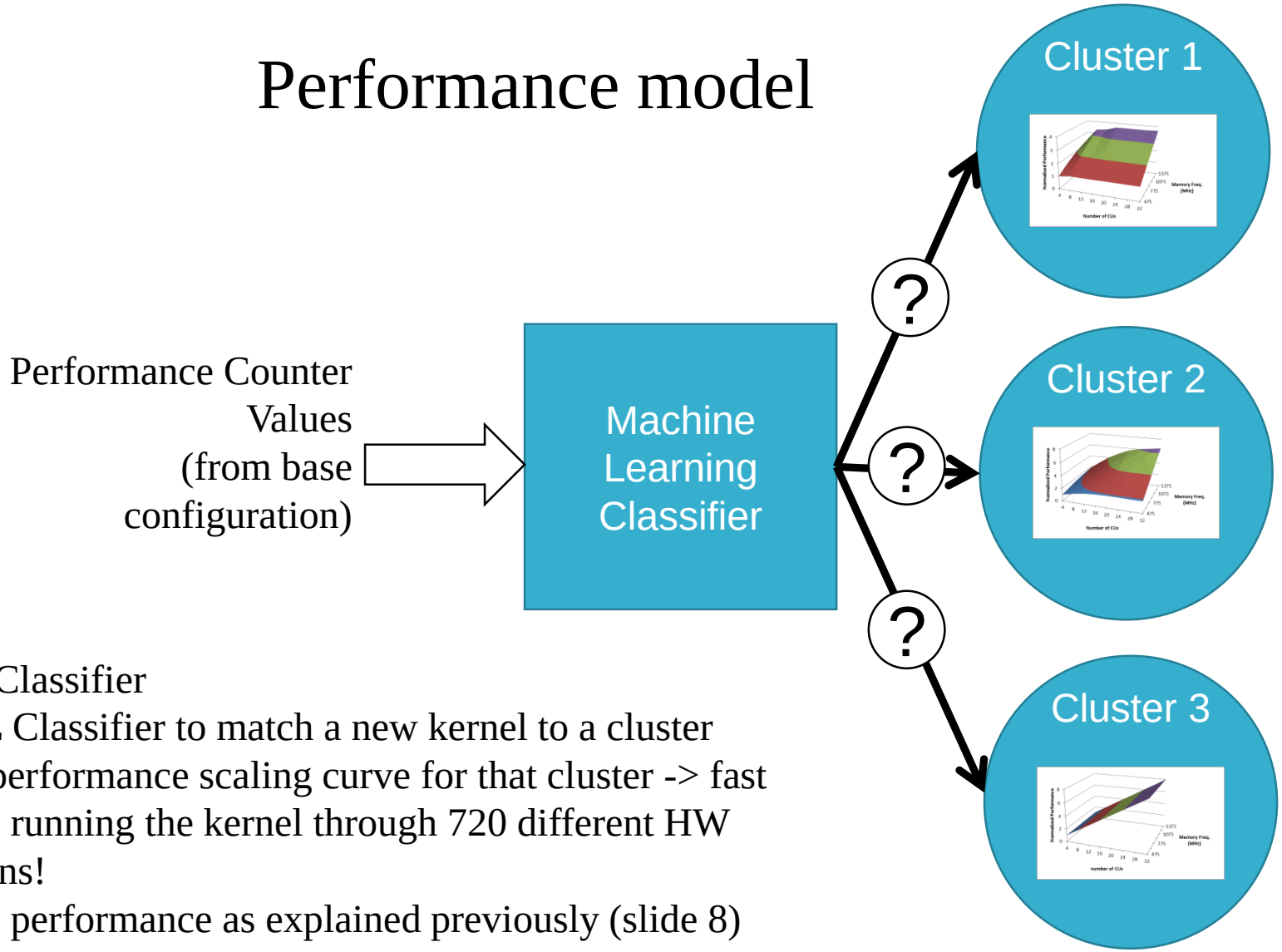
- If we want to know the performance at 40 CUs and 200 GB/s, we can gather performance data on a Base Hardware Configuration (e.g. 20 CUs, 120 GB/s)
- Start from a Base Hardware Configuration and predict performance for a Target Hardware Configuration (e.g. 40 CUs, 200 GB/s) by following the performance scaling curve for that particular kernel
- How do we get a performance scaling curve?
  - Run the kernel at all possible HW configurations -> tedious process
  - **Use a ML model, trained on known kernel scaling curves**

Performance model



- Collect performance scaling curves for many kernels running on an AMD Workstation class GPU
- Group similar kernels into clusters using machine learning techniques
- We can then classify new applications into known clusters
- And predict performance for new applications

Performance model



1. Train a ML Classifier
2. Use that ML Classifier to match a new kernel to a cluster and get the performance scaling curve for that cluster -> fast compared to running the kernel through 720 different HW configurations!
3. Estimate the performance as explained previously (slide 8)

Power model

- Total Power = Dynamic Power + Leakage Power

Predicting power is more complex

Dynamic power depends on switching activity, which in turn depends on *capacitance*, threshold voltage and frequency

The capacitance depends on
V/f scaling factors
Technology scaling (14nm or smaller)
process – high voltage or low voltage threshold devices

We use kernels that produce 100% switching activity and then scale for others

Static (or leakage power) is primarily based on
technology scaling
technology processes (high voltage or low voltage threshold devices)

We use AMD internal simulations and models for this purpose

Power model

- DYNAMIC POWER
  - Start with a known dynamic power
    - Maximum dynamic power at 100% cac on Hawaii (1000MHz/1.2V)
  - Scale by number of CUs
  - Scale by frequency and voltage (technology dependent)
  - Scale capacitance (technology dependent, going from Hawaii 28nm – 14nm = 0.65)
  - Scale by the relative switching activity
    - $\boldsymbol{cac_{rel} = \frac{P_{dynamic}}{Pmax_{dynamic}}}$ at a specific hardware point

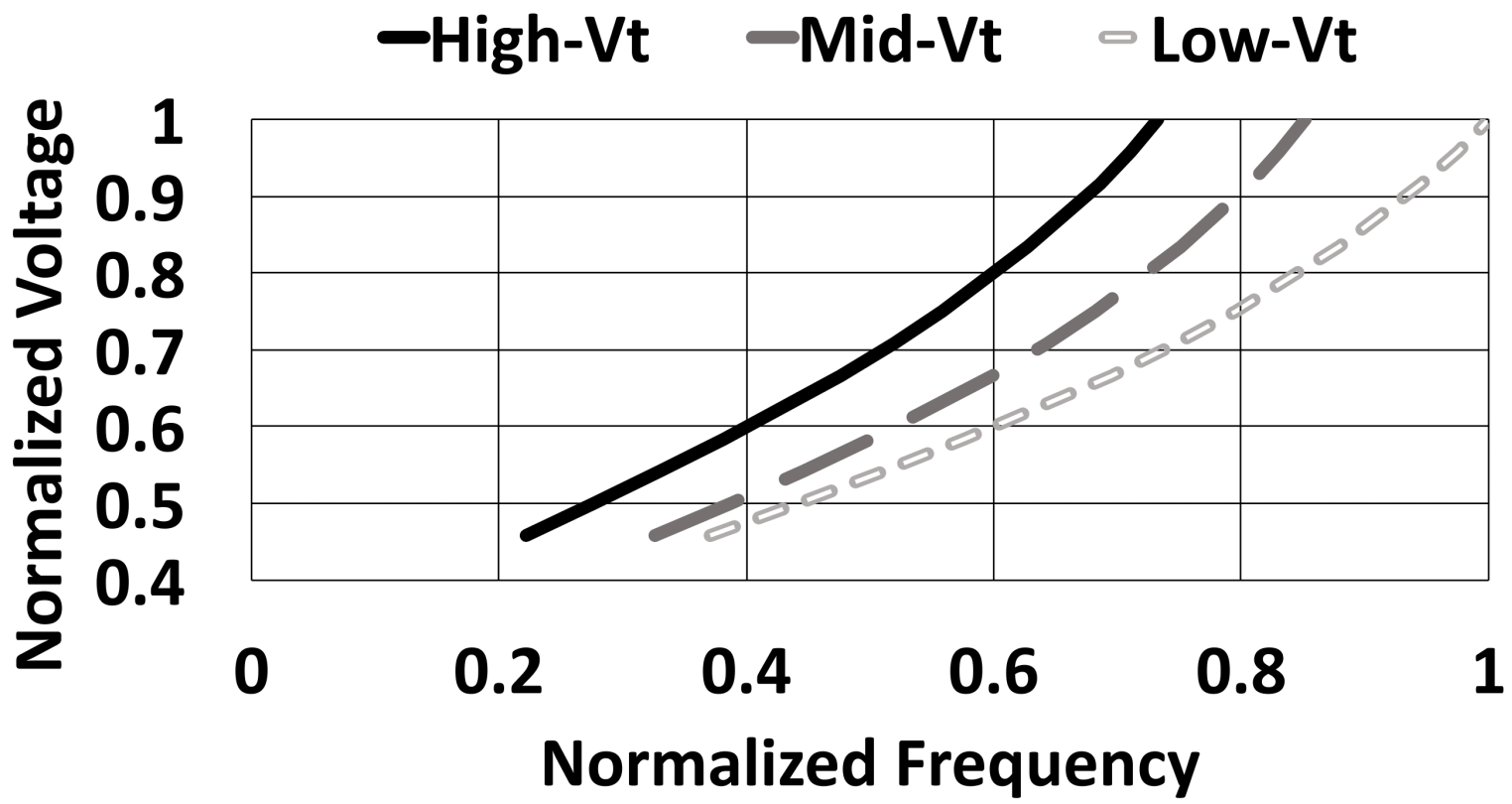
- $P_{dynamic} = Max.Dyn.P_{Hawaii} * \frac{CU_{target}}{CU_{base}} * \frac{f_{target}}{f_{base}} * \left(\frac{V_{target}}{V_{base}}\right)^2 * Cap.Scaling * cac_{rel}$

Power model – V/f characteristics

- The V/f characteristics of PIM and host will depend on the process technology and variation
- A chip design is built using multiple types of transistors to target different tradeoffs (high-performance vs. low power)

  - HVT – High Threshold Voltage causes less power consumption and timing to switch is not optimized. Used to minimize power consumption for power critical functions.
  - LVT – Low Threshold Voltage causes more power consumption and switching timing is optimized. Used on the critical path
  - SVT (MVT) – Standard Threshold Voltage offers trade-off between HVT and LVT i.e., moderate delay and moderate power consumption.

Power model – v/f characteristics



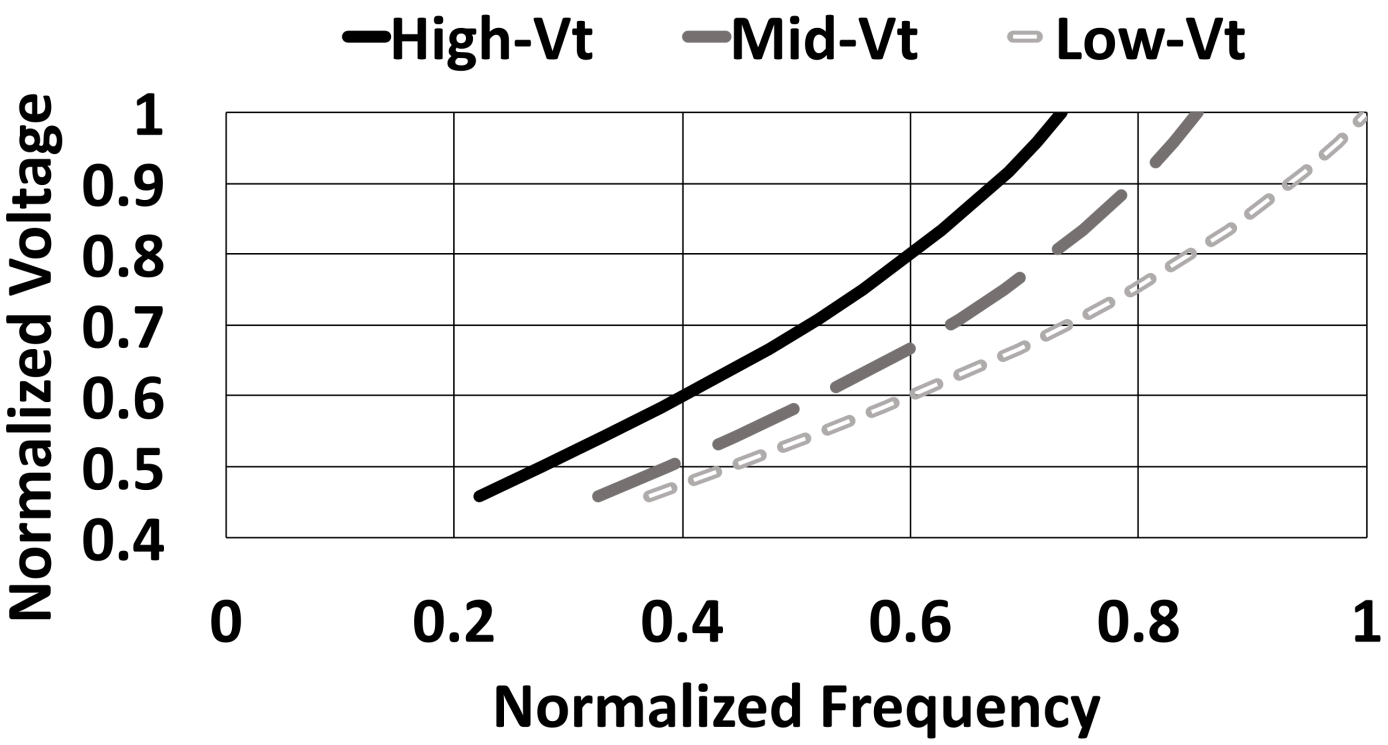
- Typical V/f characteristics of HVT, MVT and LVT transistors

Power model

- We use AMD internal tool/database to get:
  - V/f curves for a 14nm chip similar to Hawaii GPU (high-performance process)
  - Relative difference in leakage power between host and PIM for different VT distributions (HVT/MVT/LVT)

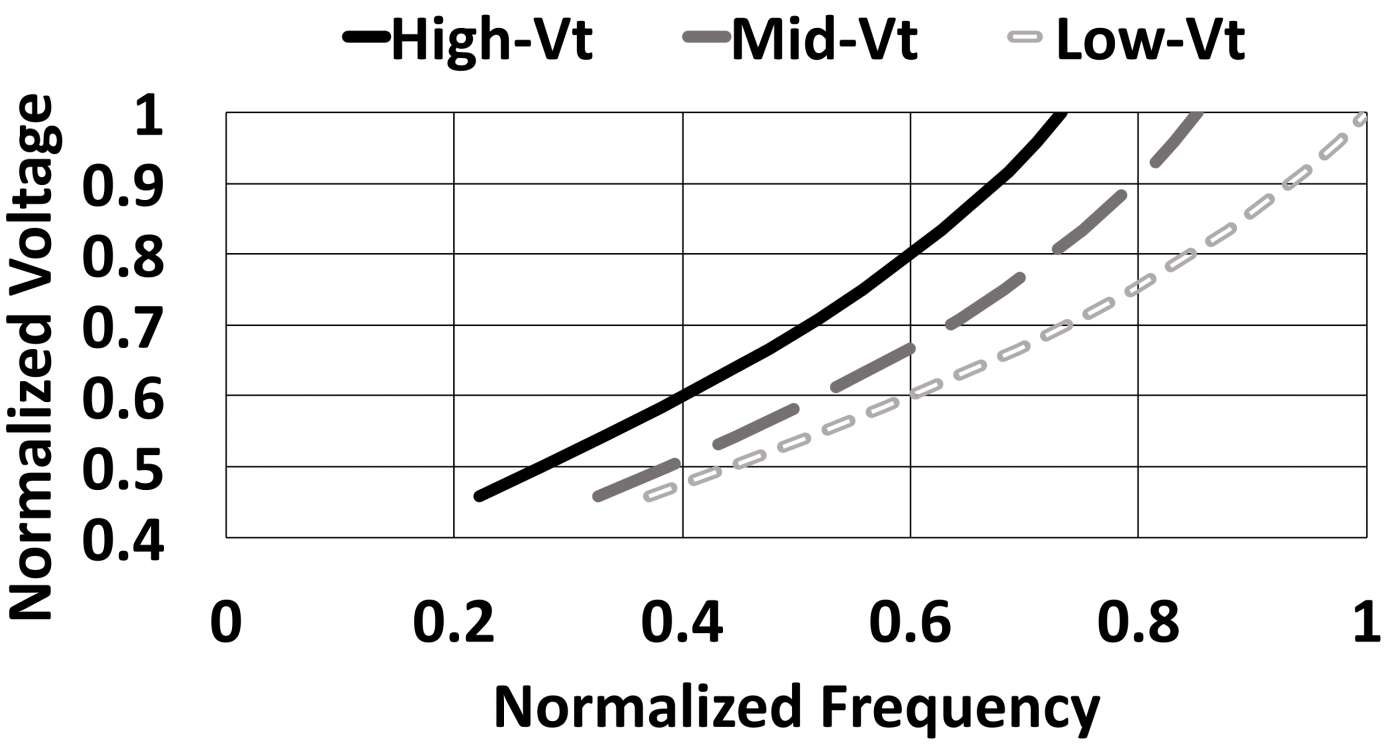
- We assume same V/f curve for PIM and host but limit the operating frequency of PIM to a lower frequency range
  - PIM will deviate from that curve at higher frequencies
  - Since we can't determine what is the "cutoff" frequency for PIM we examine the leakage power for different VT distributions

Power model – v/f characteristics



- We pick a V/f curve for a 14nm chip similar to Hawaii for a specific type of transistors

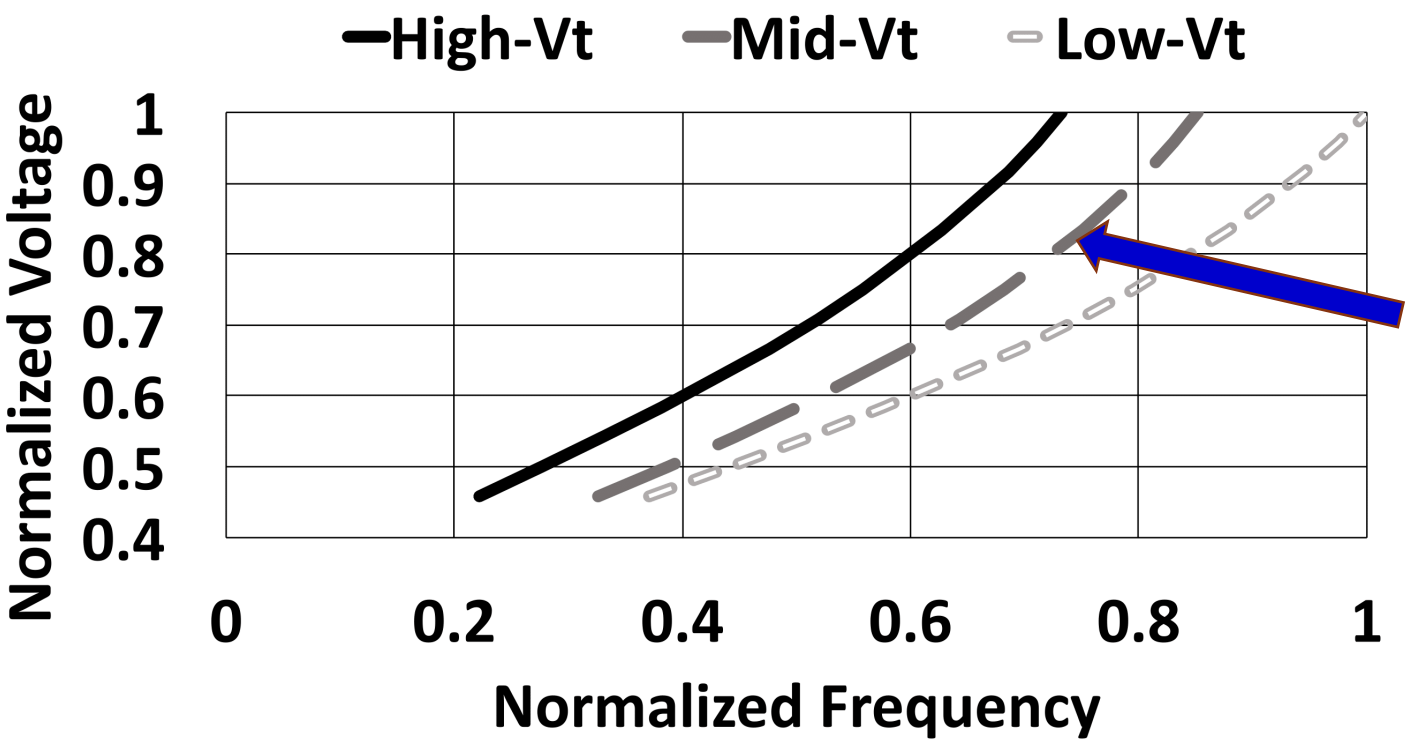
Power model – v/f characteristics



- We pick a V/f curve for a 14nm chip similar to Hawaii for a specific type of transistors

- **And limit the operating frequencies**
- **Host frequency: 600MHz-1000MHz**
- **PIM frequency: 400MHz-600MHz**

Power model – v/f characteristics



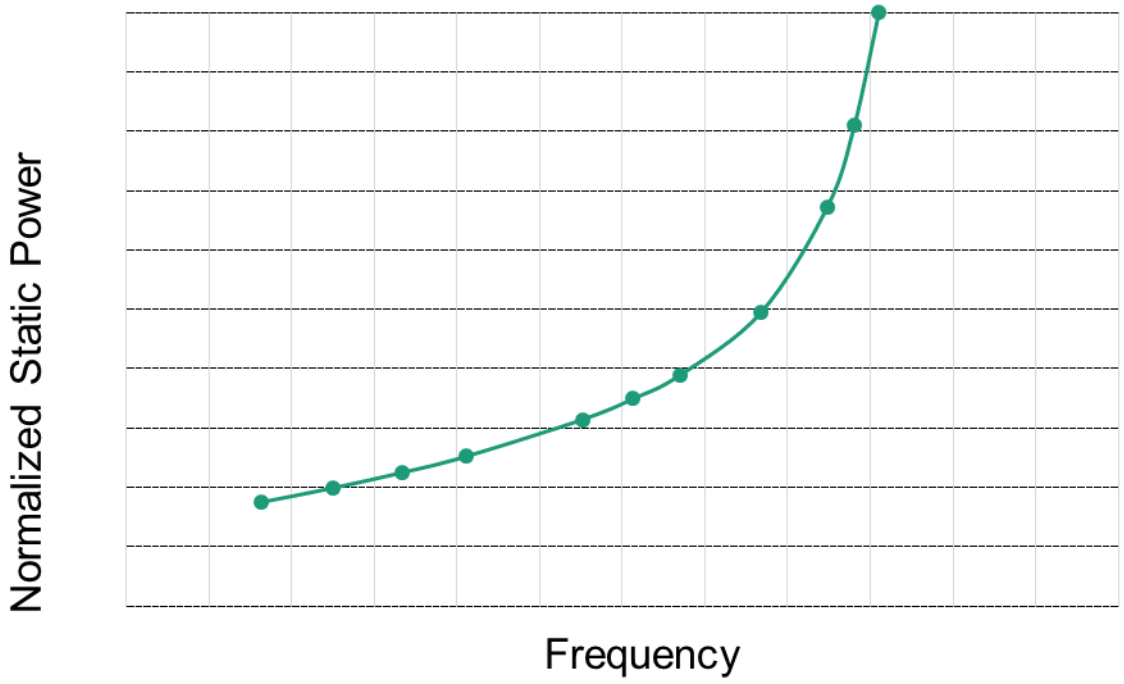
- We pick a V/f curve for a 14nm chip similar to Hawaii for a specific type of transistors

- **And limit the operating frequencies**
- **Host frequency: 600MHz-1000MHz**
- **PIM frequency: 400MHz-600MHz**

Leakage power

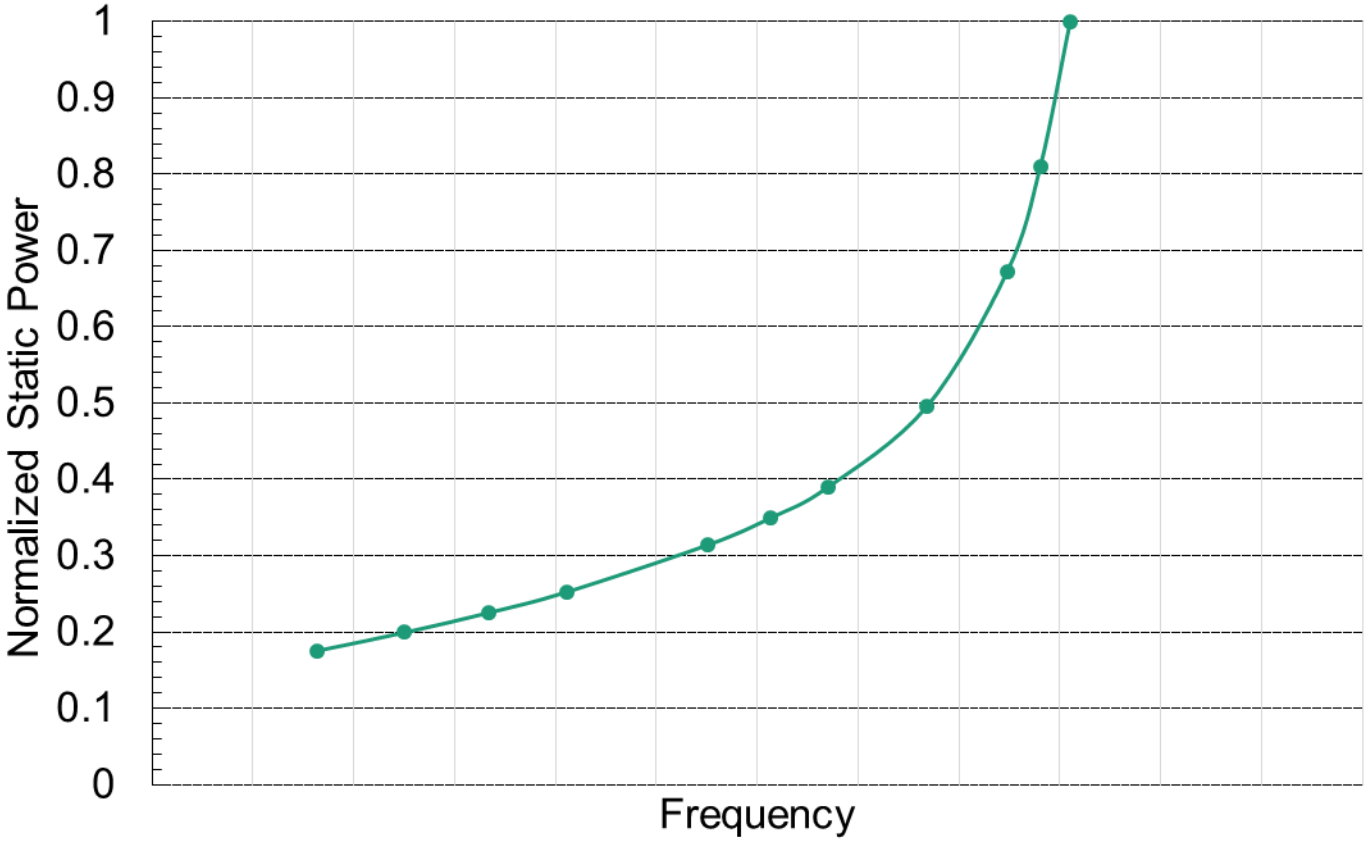
- We use AMD internal tool/database to get:
  - Relative difference in leakage power between host and PIM for different VT distributions
  - We get leakage for a given type of transistors
  - We use this information to model the relative difference in leakage power and use this as leakage scaling factor between PIM and host

Power model – Leakage power



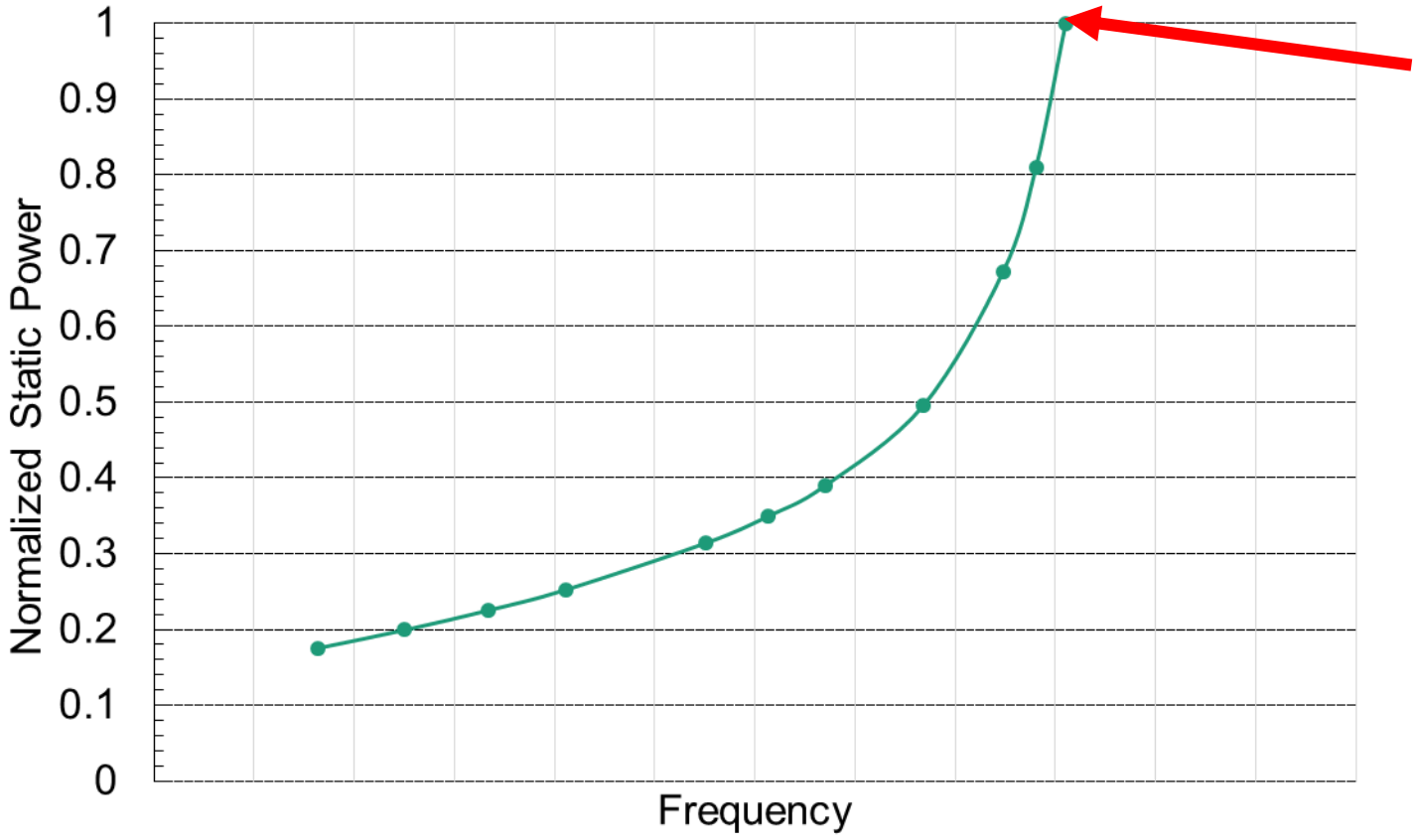
- The curve represents how static power changes with frequency for a circuit built of 50/50 HVT/MVT devices
- Obtained from AMD tools
- All the data points are relative leakage power normalized to the highest leakage power (right-most point)
- So how do we get the leakage?

Power model – Leakage power



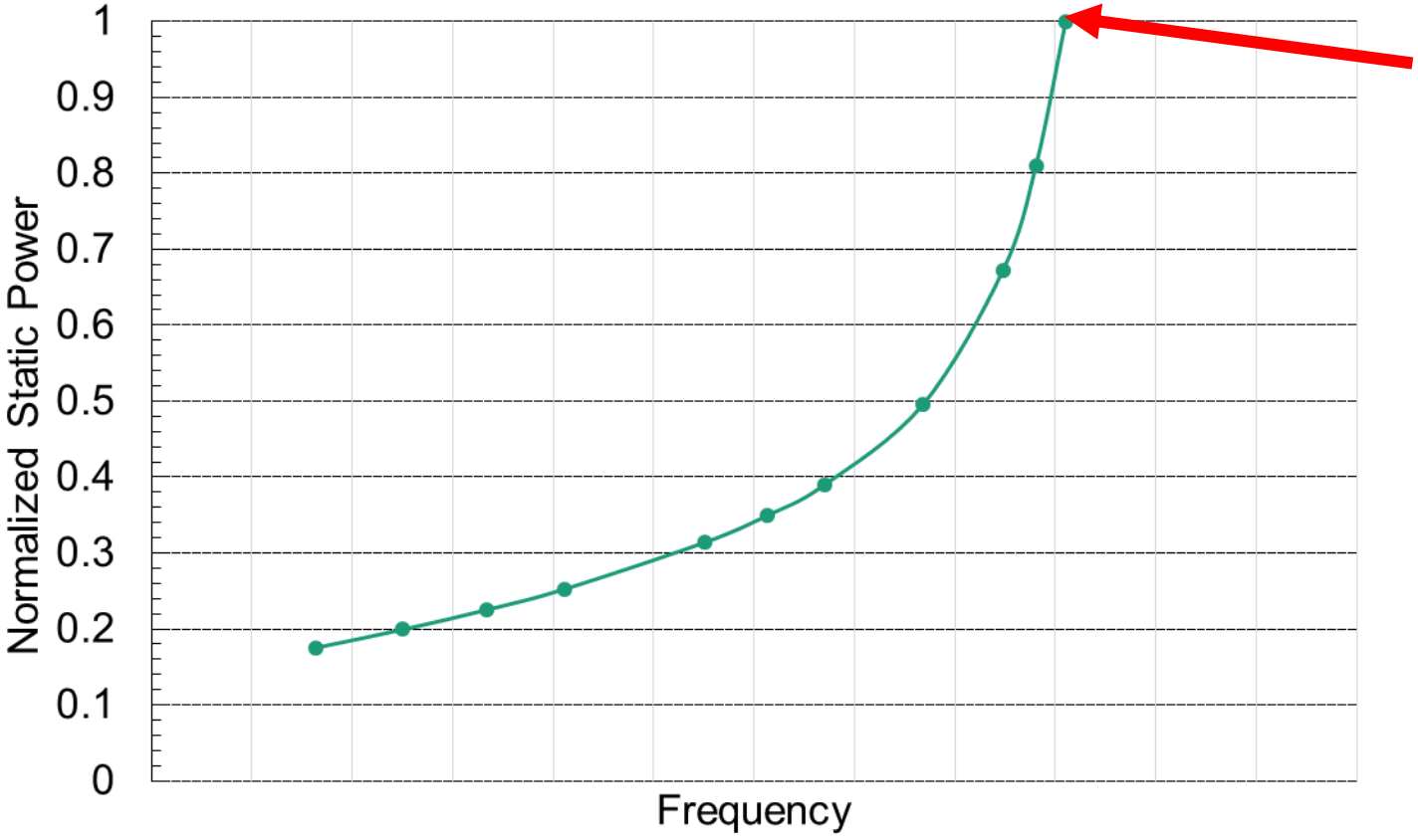
- Pick a baseline point and estimate the leakage power at this particular V/f point
- In our case the baseline point is at 1.2V (at 1200MHz)

Power model – Leakage power



- Pick a baseline point and estimate the leakage power at this particular V/f point
- In our case the baseline point is at 1.2V (at 1200MHz)
- We need to know the actual leakage power for that V/f point[watts]

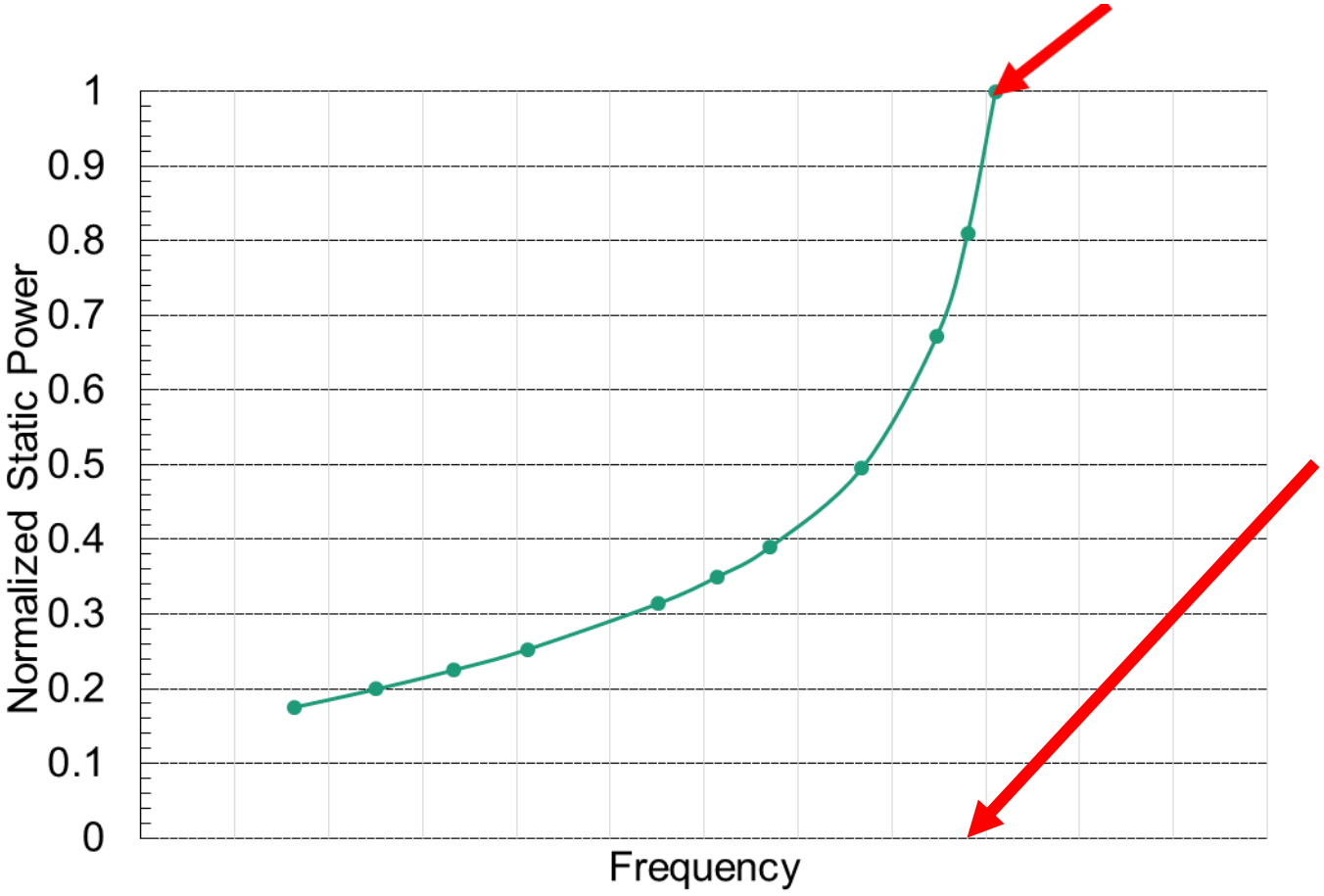
Power model – Leakage power



- Pick a baseline point and estimate the leakage power at this particular V/f point
- In our case the baseline point is at 1.2V (at 1200MHz)
- We need to know the actual leakage power for that V/f point[watts]

We will estimate leakage power using well established estimates – 30% of the TDP
So, we first estimate Dynamic power at maximum switching activity

Power model – Leakage power



- Previous designs (and models) show that 30% of TDP as leakage is a good estimate

- **TDP = Max.Dyn.Power + Leakage**
- Calculate Max.Dyn.Power for host at 1.2V/1200MHz

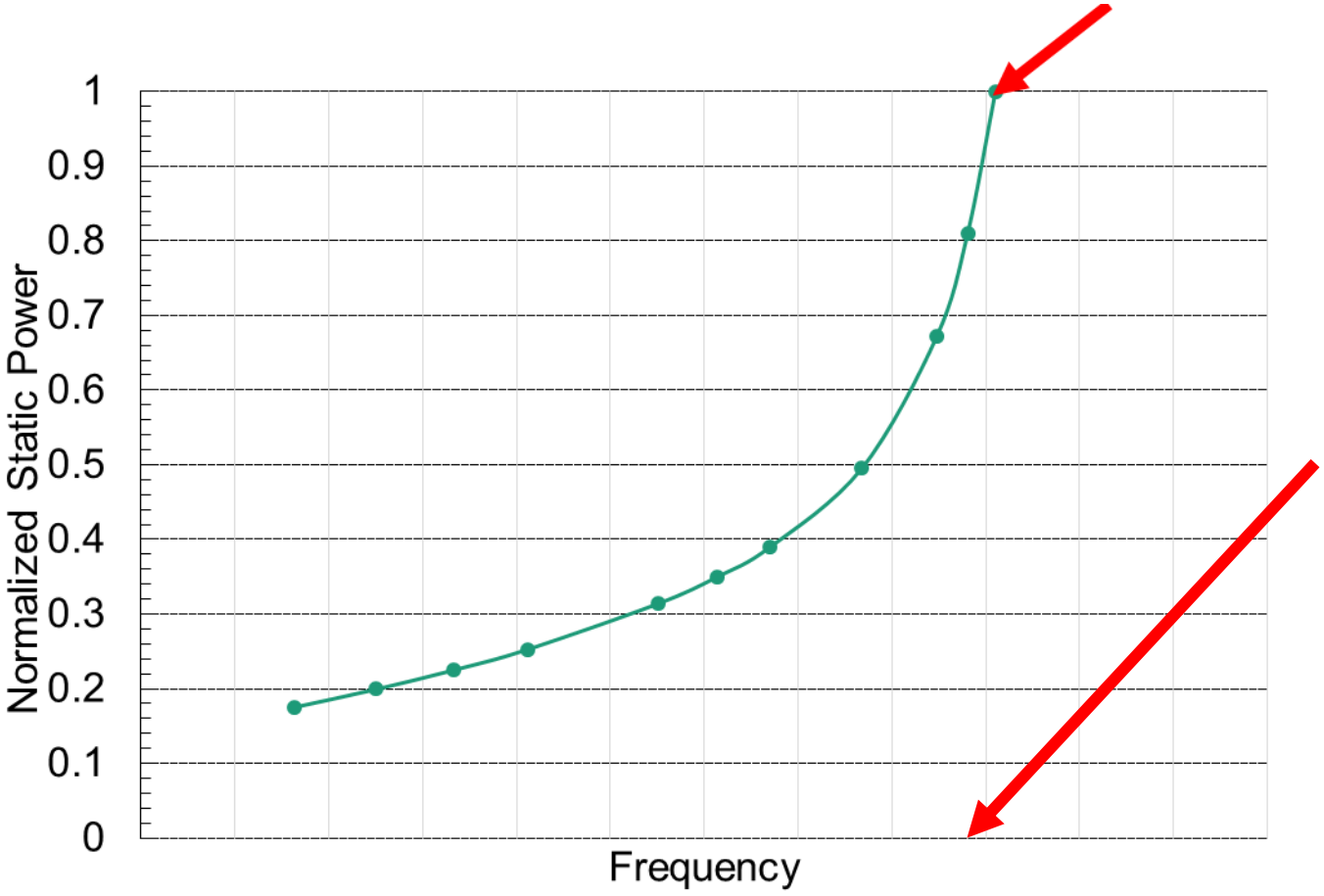
$$P_{leak} = MAX.DYN.POWER * \frac{0.3}{1-0.3}$$

We can calculate the maximum dynamic power by using the formula for 100% switching activity

(Max.Dyn.Power Hawaii is at 1V/1GHz/32nm;

we need 1.2V/1.2GHZ/14nm)

Power model – Leakage power



- Previous designs (and models) show that 30% of TDP as leakage is a good estimate

- **TDP = Max.Dyn.Power + Leakage**
- Calculate Max.Dyn.Power for host at 1.2V/1200MHz

$$P_{leak} = MAX.DYN.POWER * \frac{0.3}{1-0.3}$$

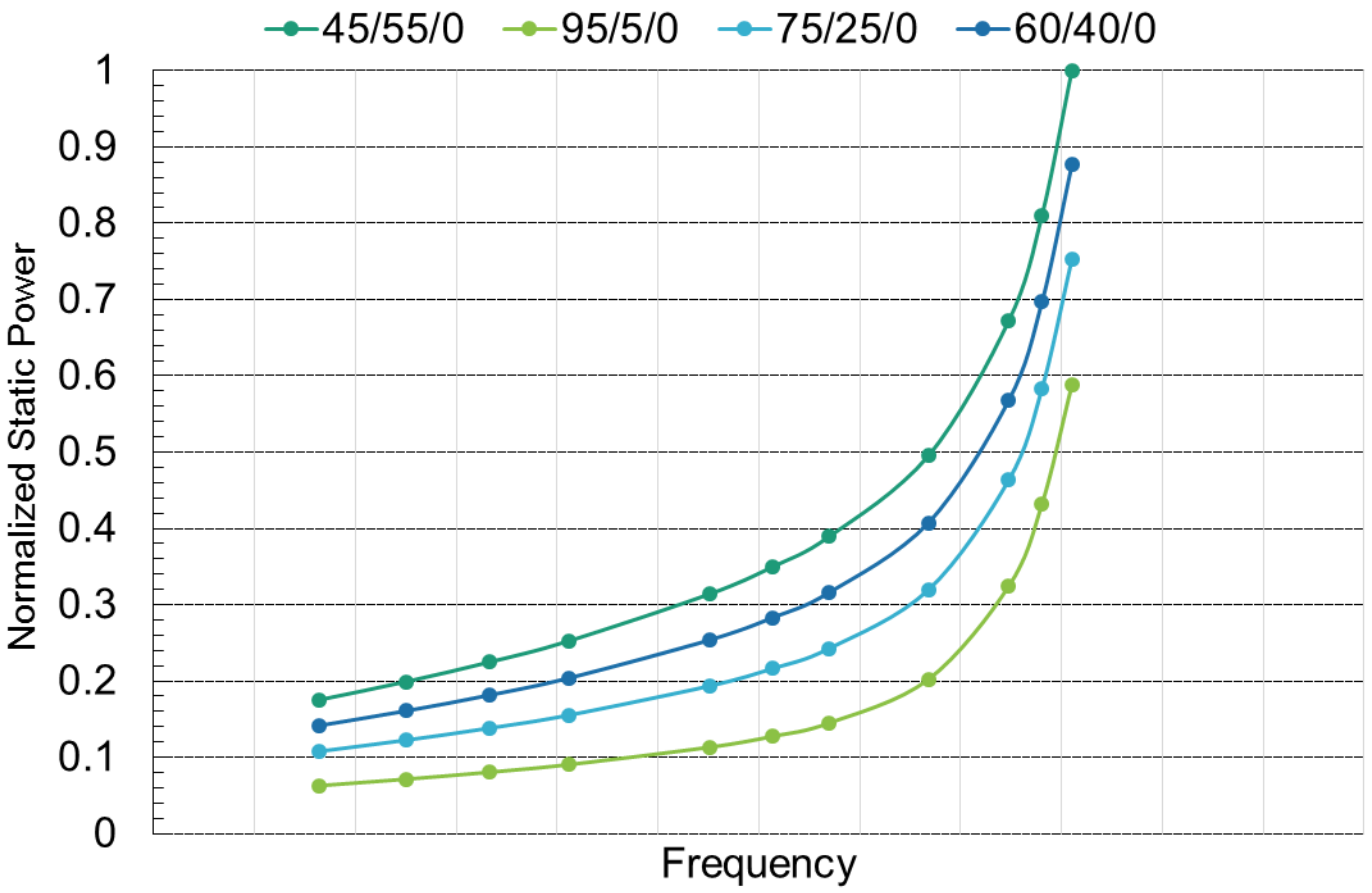
We can calculate the maximum dynamic power by using the formula for 100% switching activity

(Max.Dyn.Power Hawaii is at 1V/1GHz/32nm;

we need 1.2V/1.2GHZ/14nm)

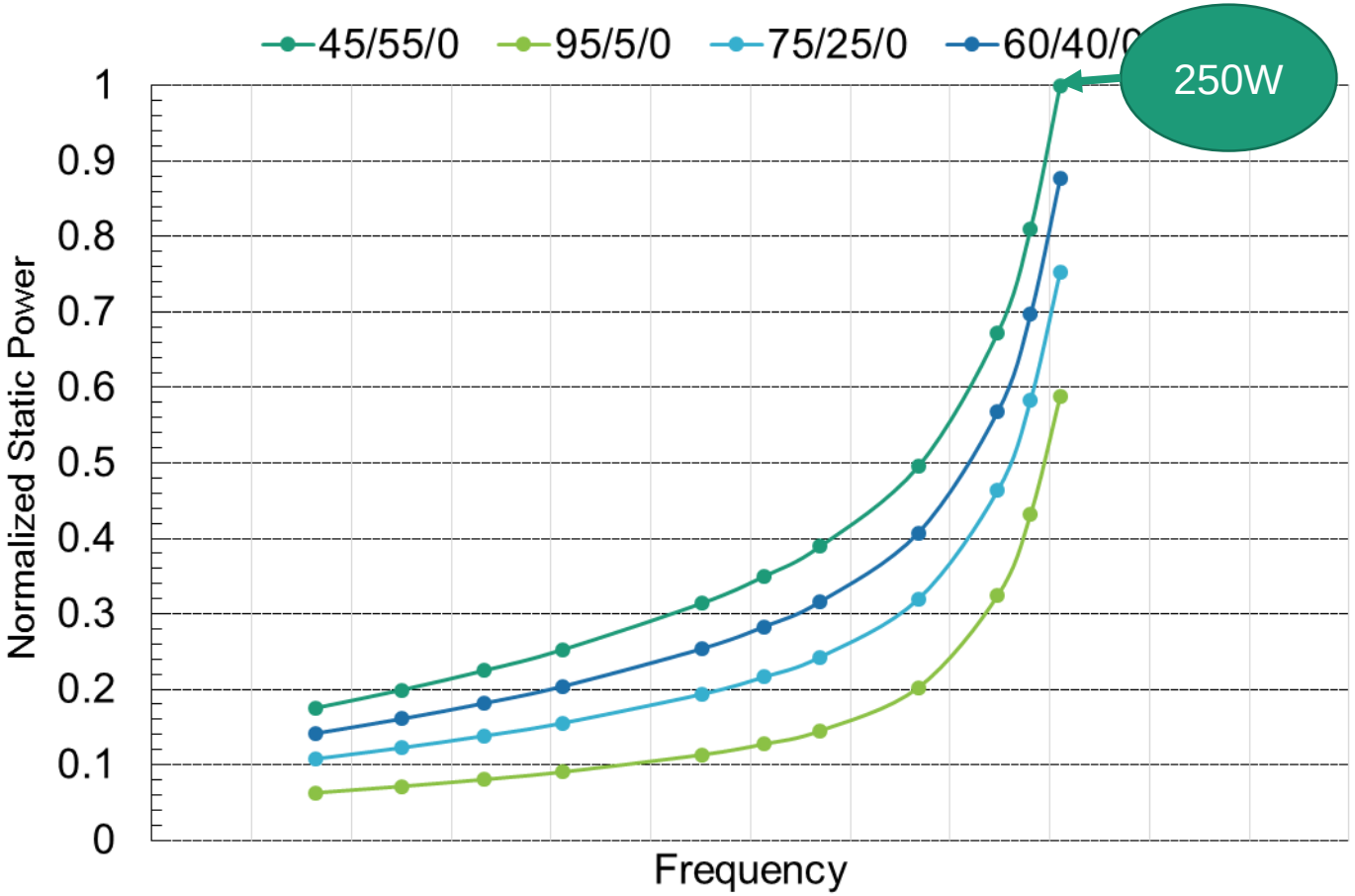
$$P_{dynamic} = MAX.DYN.POWER_{Hawaii} * \frac{CU_{target}}{CU_{base}} * \frac{f_{target}}{f_{base}} * \left(\frac{V_{target}}{V_{base}}\right)^2 * Cap.Scaling * cac_{rel}$$

Power model – Leakage power



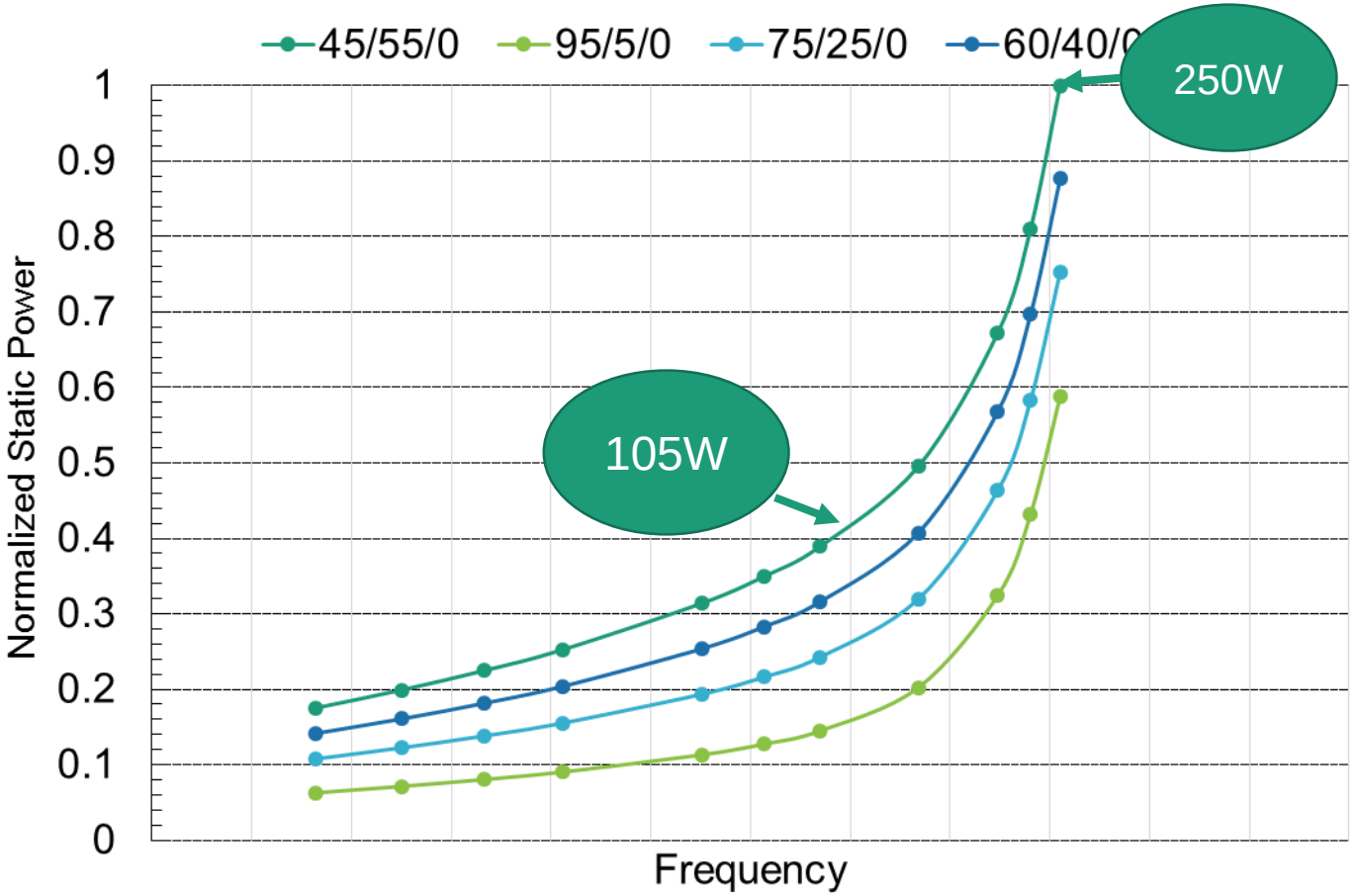
- Baseline point in our case is 250W
- Scale leakage power relative to the base point for different VT breakdowns

Power model – Leakage power



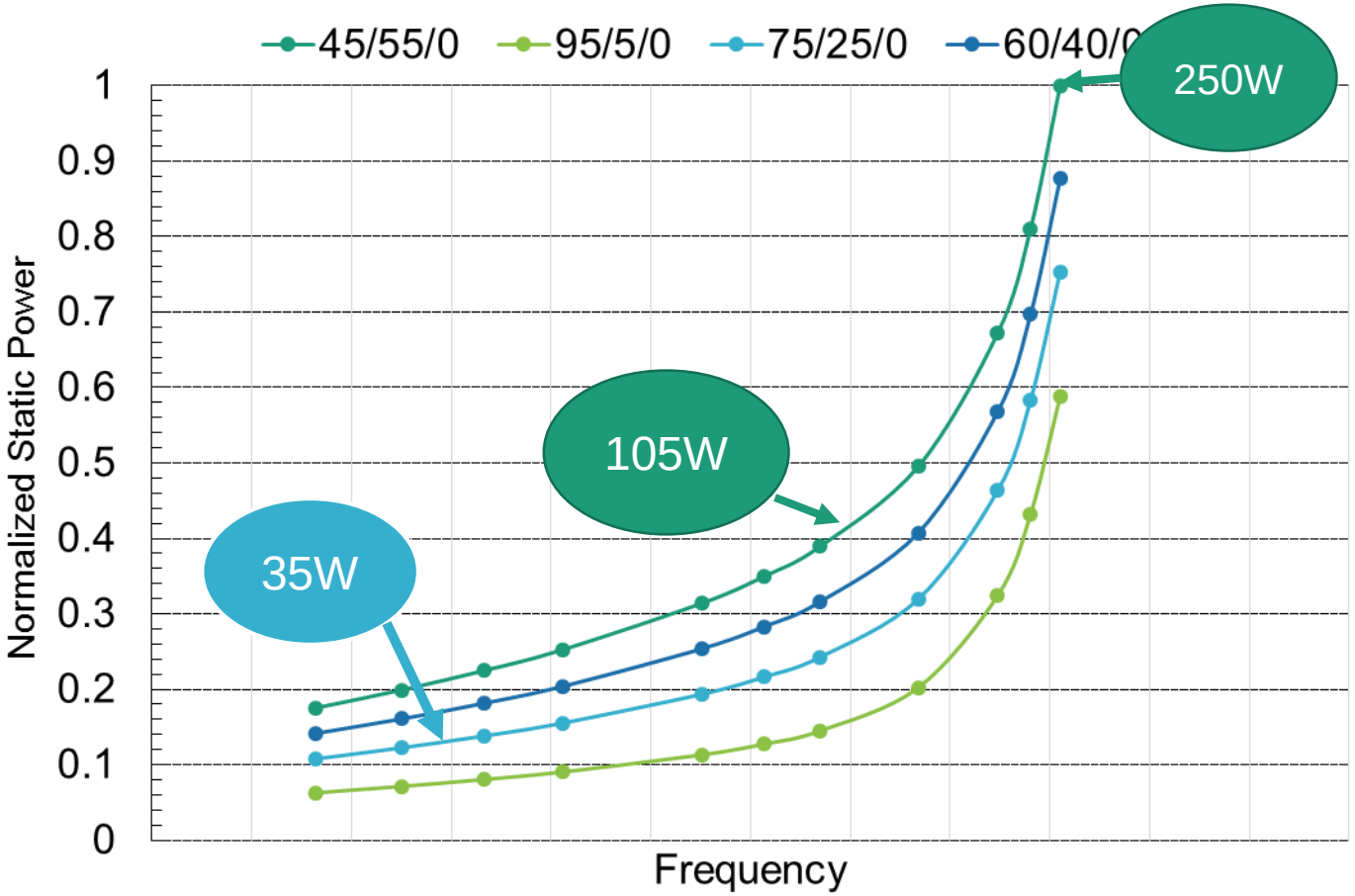
- Baseline point in our case is 250W
- Scale leakage power relative to the base point for different VT breakdowns

Power model – Leakage power



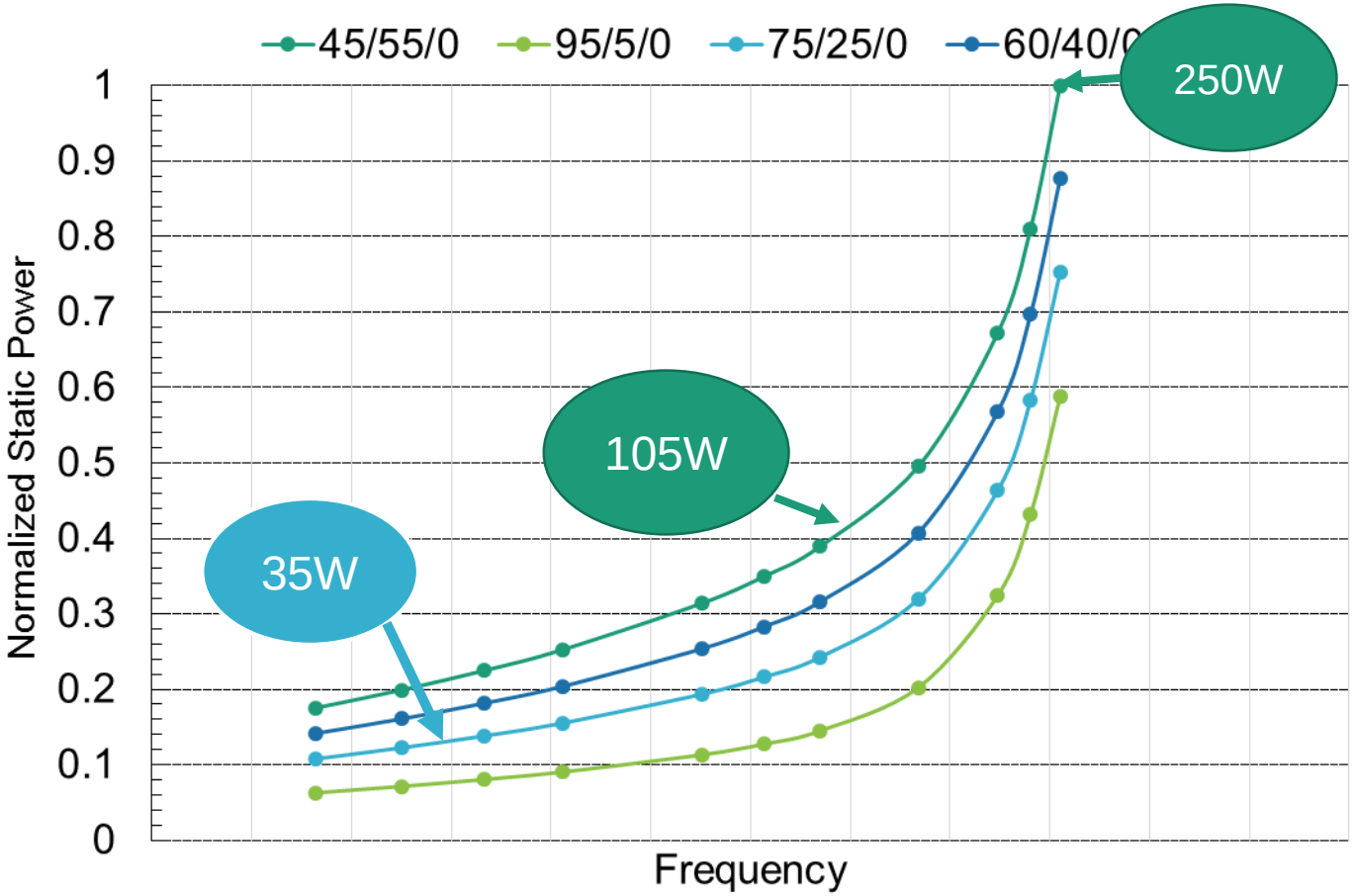
- Baseline point in our case is 250W
- Scale leakage power relative to the base point for different VT breakdowns

Power model – Leakage power



- Baseline point in our case is 250W
- Scale leakage power relative to the base point for different VT breakdowns

Power model – Leakage power



- Baseline point in our case is 250W
- Scale leakage power relative to the base point for different VT breakdowns

Based on the leakage power, we can decide on the processing mix we need to achieve that power goal and the DVFS state to operate

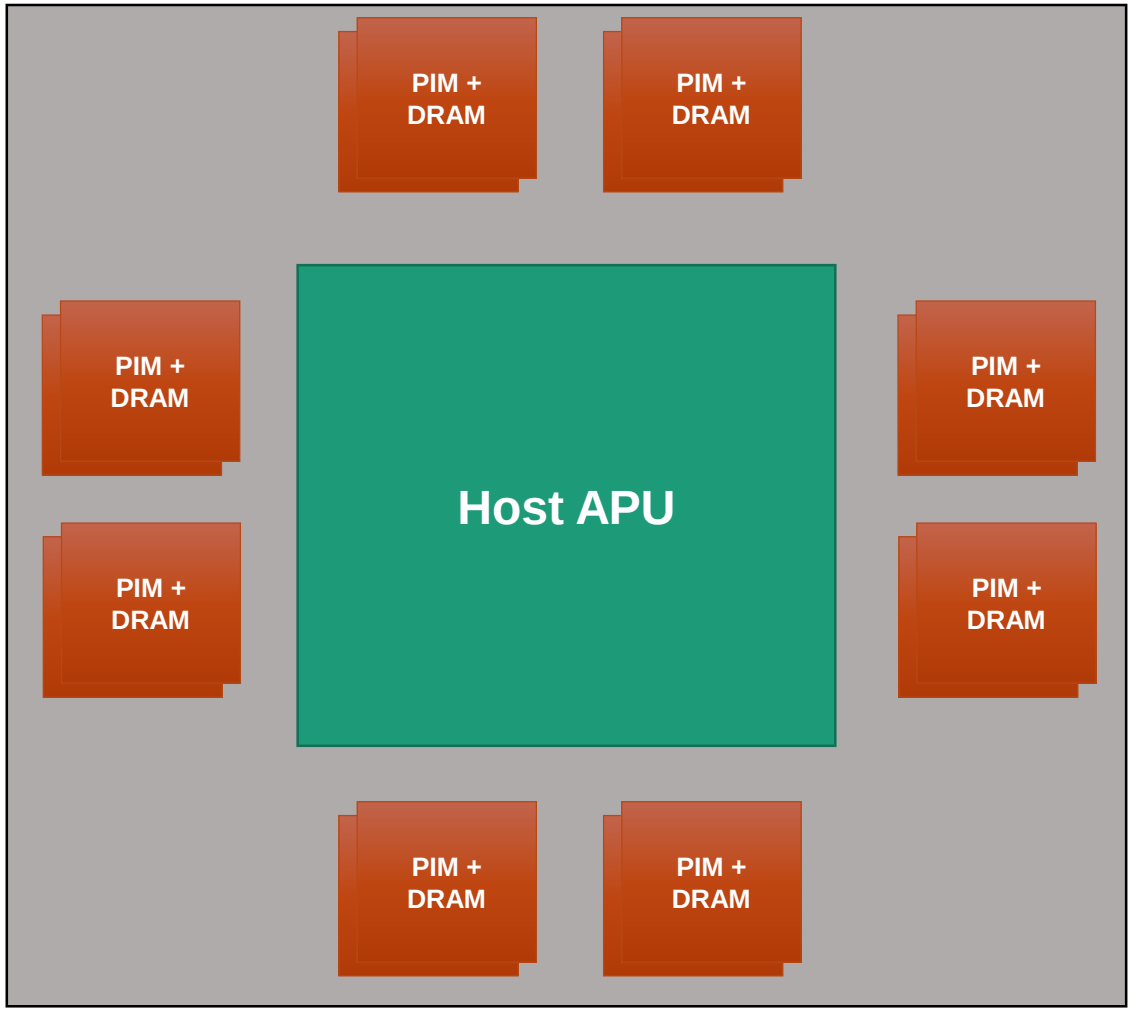
Experiments and Results

Baseline System

- HOST
  - 256 CUs
  - 1 TB/s aggregate bandwidth
  - 600MHz – 1000MHz

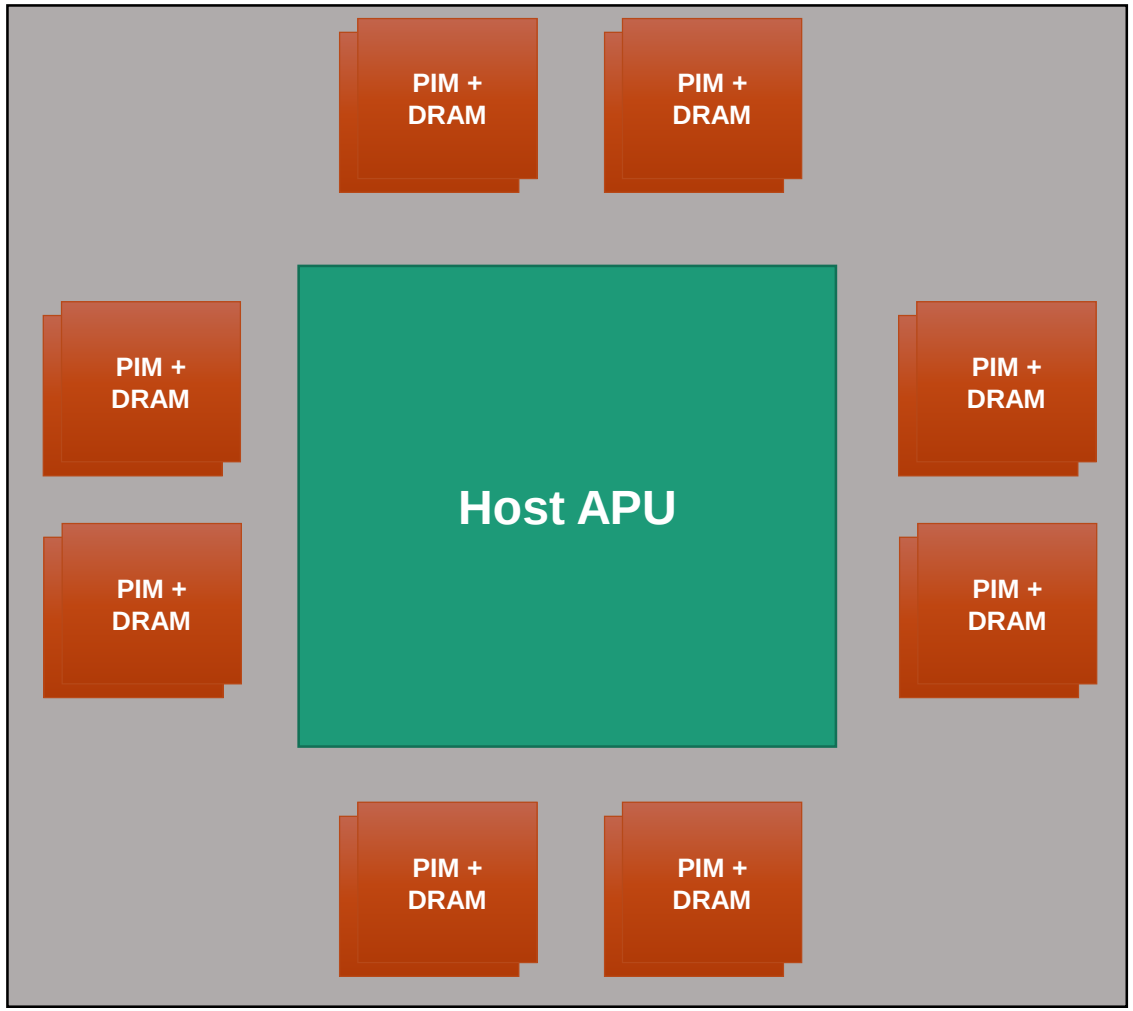
- PIM
  - 8 x 24CUs = 192 CUs
  - 2 TB/s aggregate bandwidth
  - 400MHz – 600MHz

The number of CUs and BW are somewhat constrained by the capabilities of AMD's HLSim

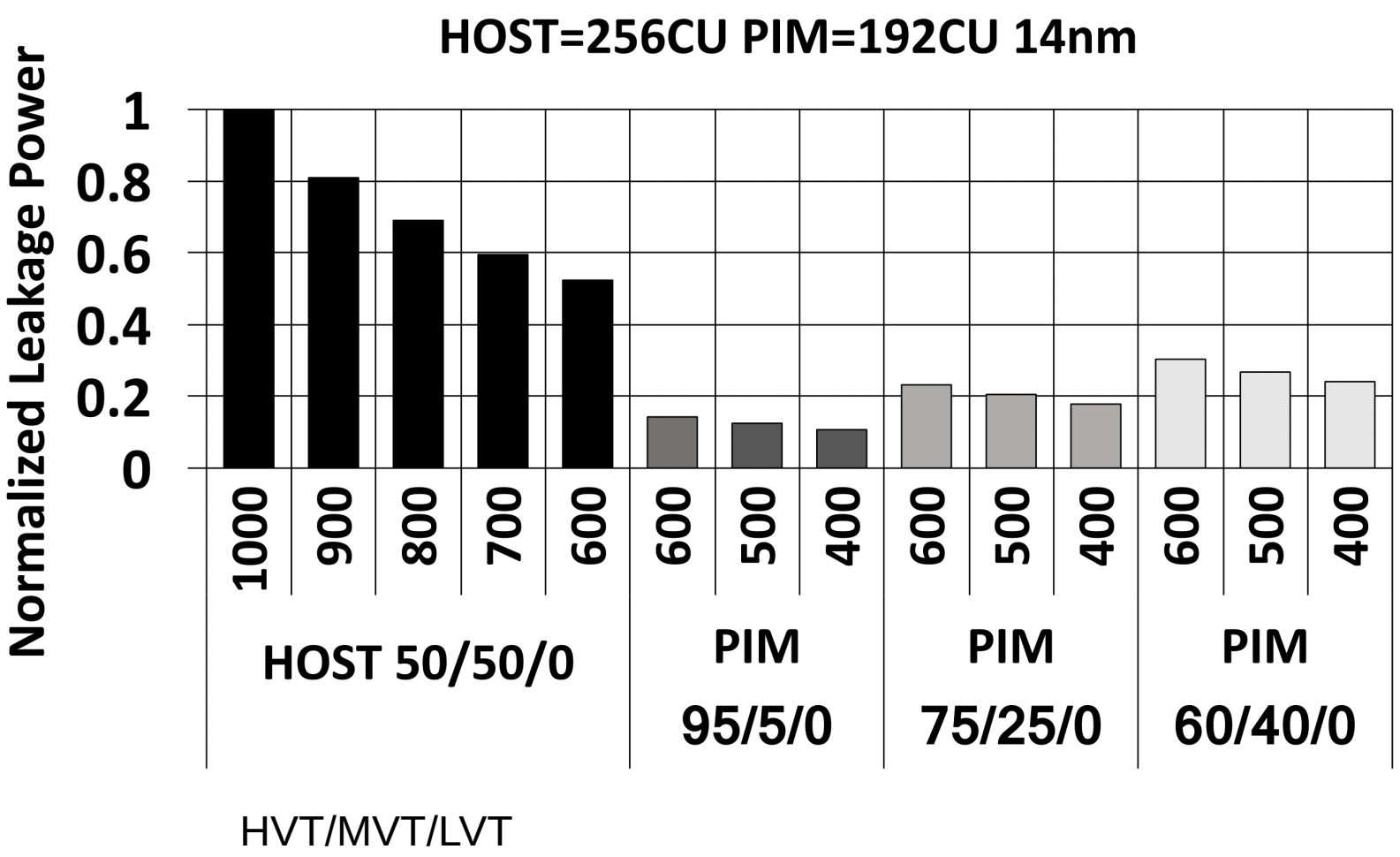


Baseline System

- Host and PIM in 14 nm process
- Host HVT/MVT/LVT – 45/55/0
- PIM: 95/5/0, 75/25/0, 60/40/0
- All of our leakage is assumed to be at some fixed temperature (e.g.100°C)
- Performance counters collected on Hawaii (28nm) 1000MHz, 1.2V, 1250MHz memory frequency

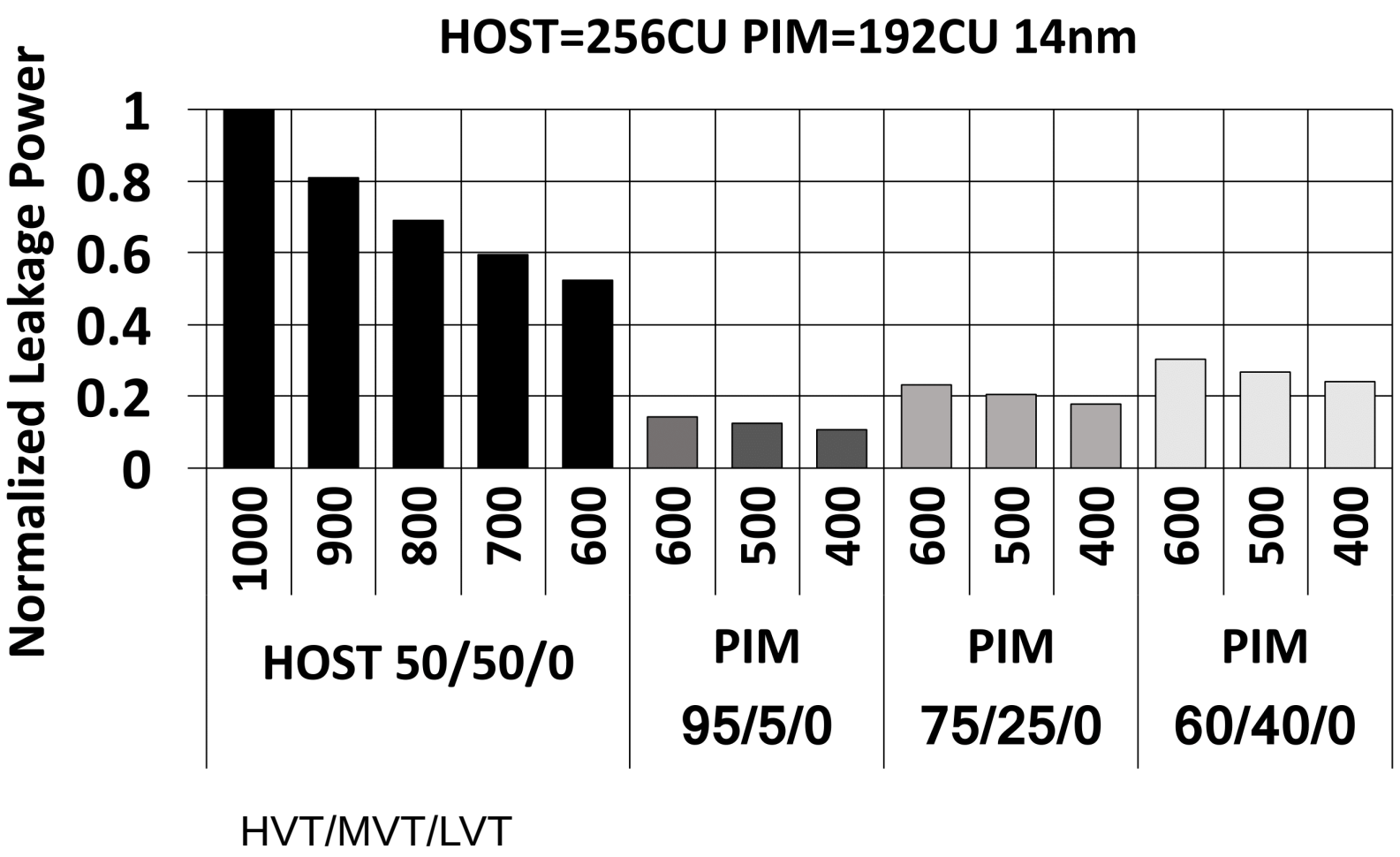


Leakage power comparison



- Minimizing leakage power is important as it is the most significant power contributor for bandwidth intensive applications

Leakage power comparison

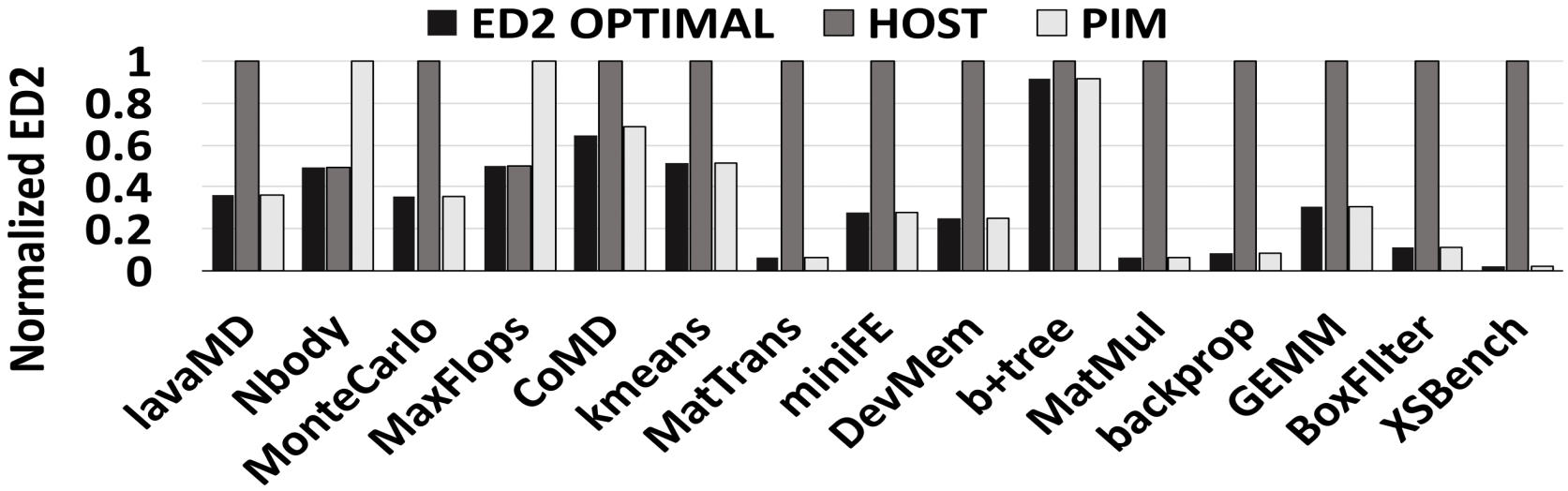


- Minimizing leakage power is important as it is the most significant power contributor for bandwidth intensive applications
- We can rely on HTV implementation of PIM devices as they will compensate any performance losses by exploiting high bandwidth

DVFS optimization

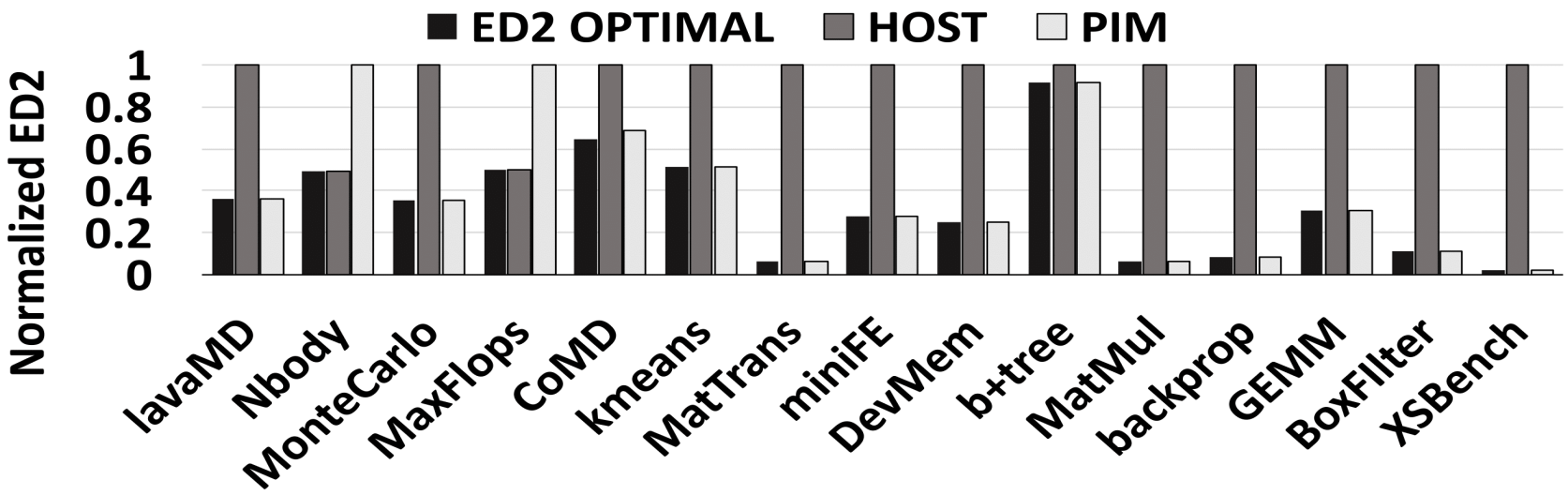
- Adjusting engine frequency (and voltage) to maximize energy efficiency
- Trying to find optimal placement of kernels (PIM/host) such that we maximize energy efficiency
- Comparing execution time with power constraints
- All results are normalized to the best case for each kernel

Target = Minimum ED^2



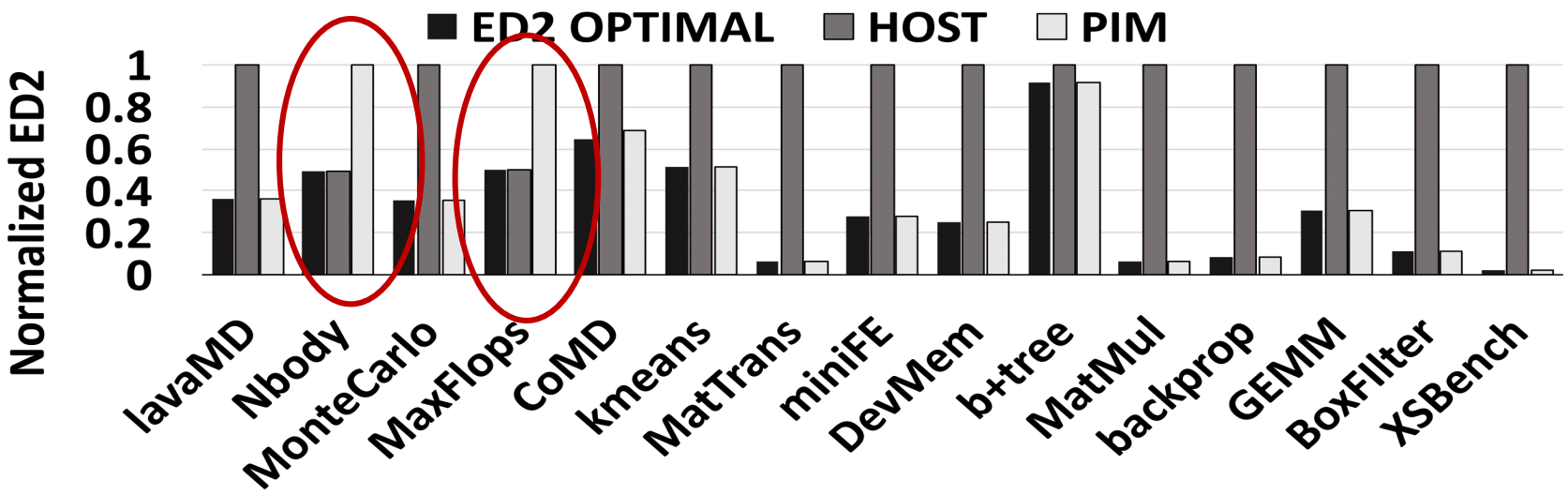
- Addition of PIMs to a heterogeneous node architecture can yield high energy efficiency even compared to applications running on host running at lower DVFS states
- Power will be significantly reduced, at the expense of small performance loss leading to great energy efficiency

Target = Minimum ED^2



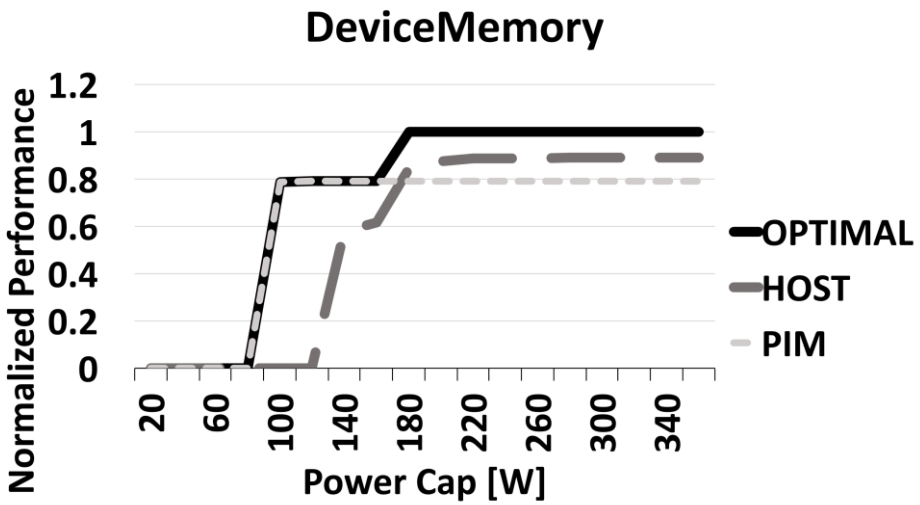
- Addition of PIMs to a heterogeneous node architecture can yield high energy efficiency even compared to applications running on host running at lower DVFS states
- Power will be significantly reduced, at the expense of small performance loss leading to great energy efficiency
- PIMs aren't necessarily the most energy-efficient choice for computation in all cases
- Compute intensive applications like MaxFLops and Nbody favor host (at lower DVFS state) over PIM

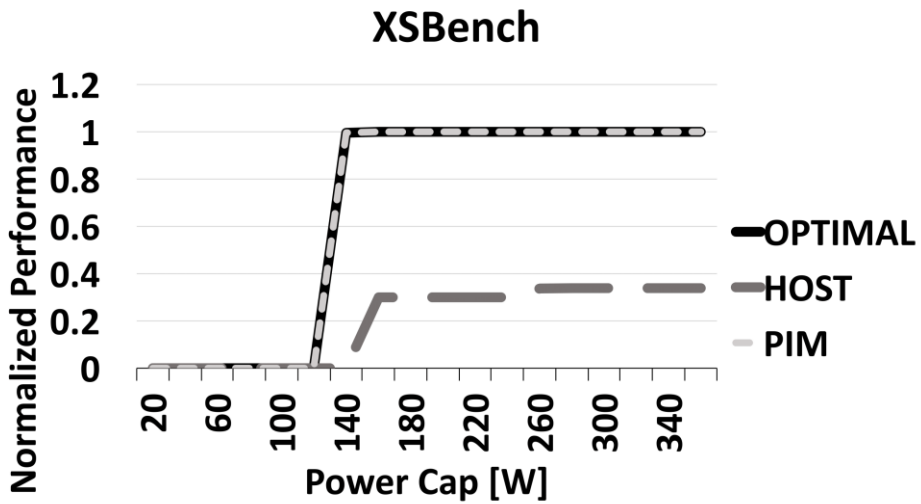
Target = Minimum ED^2

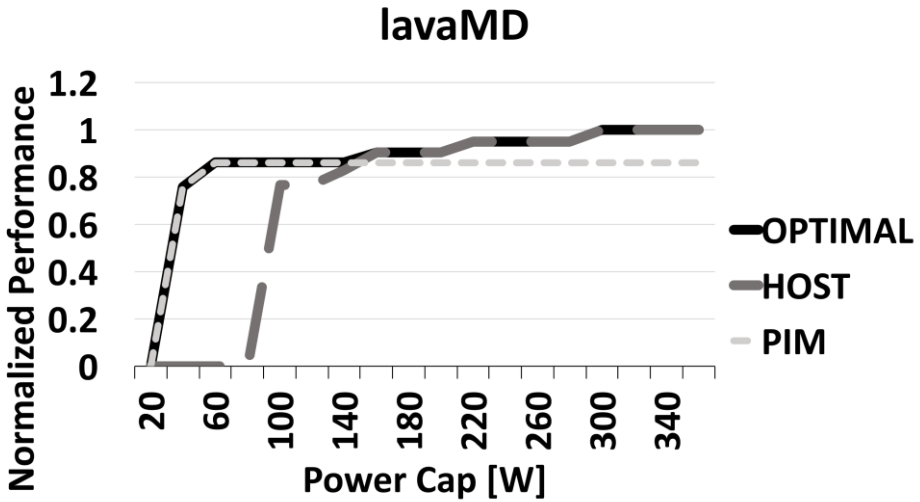


- Addition of PIMs to a heterogeneous node architecture can yield high energy efficiency even compared to applications running on host running at lower DVFS states
- Power will be significantly reduced, at the expense of small performance loss leading to great energy efficiency
- PIMs aren't necessarily the most energy-efficient choice for computation in all cases
- Compute intensive applications like MaxFLops and Nbody favor host (at lower DVFS state) over PIM

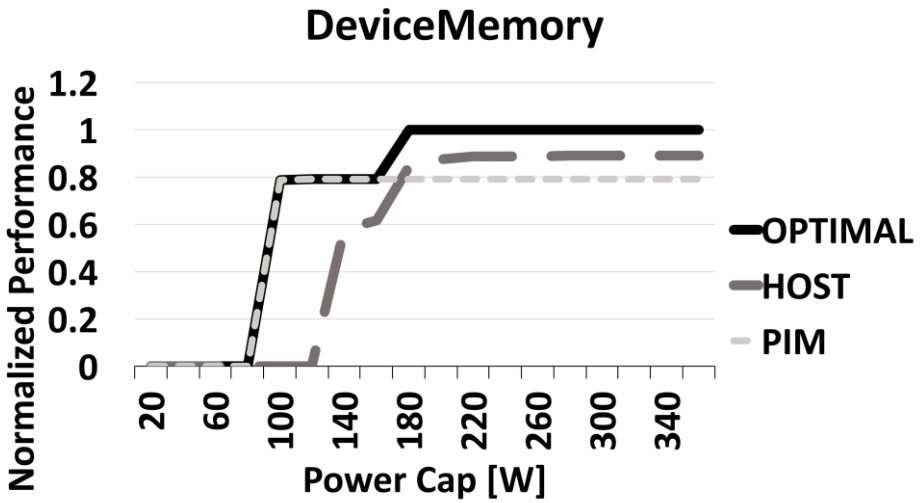
Maximum performance under power constraint

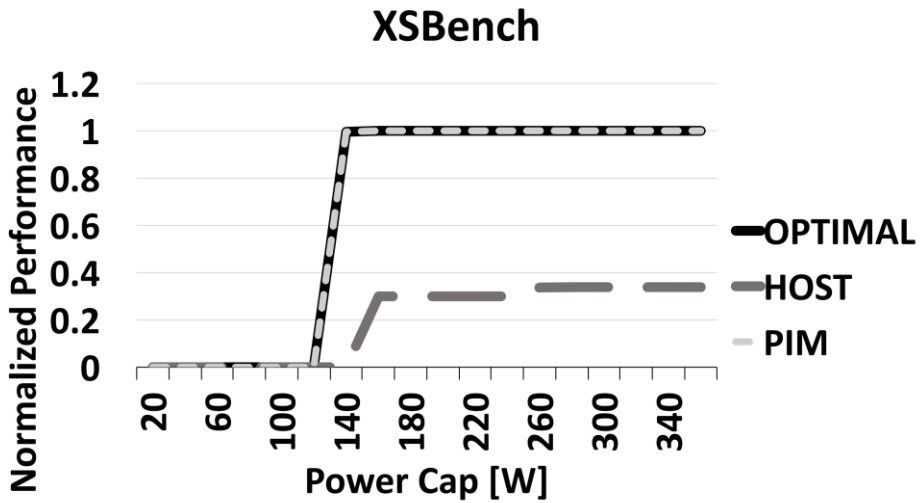


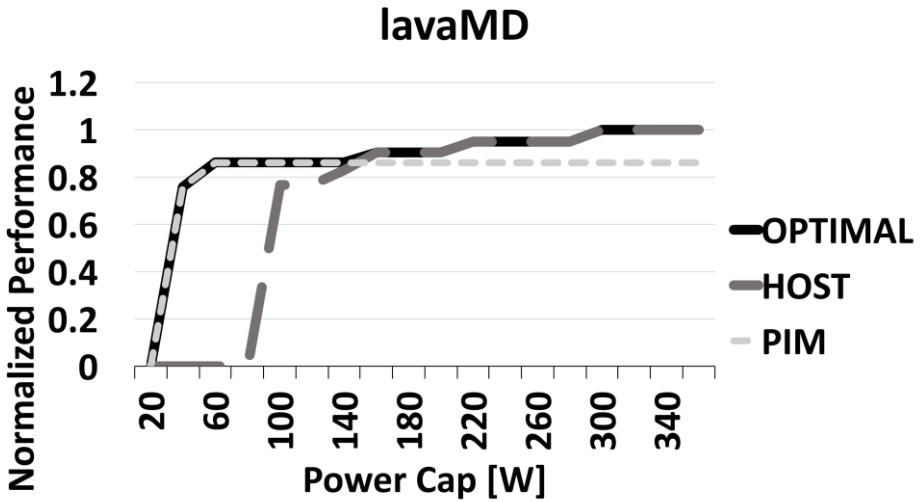




Maximum performance under power constraint

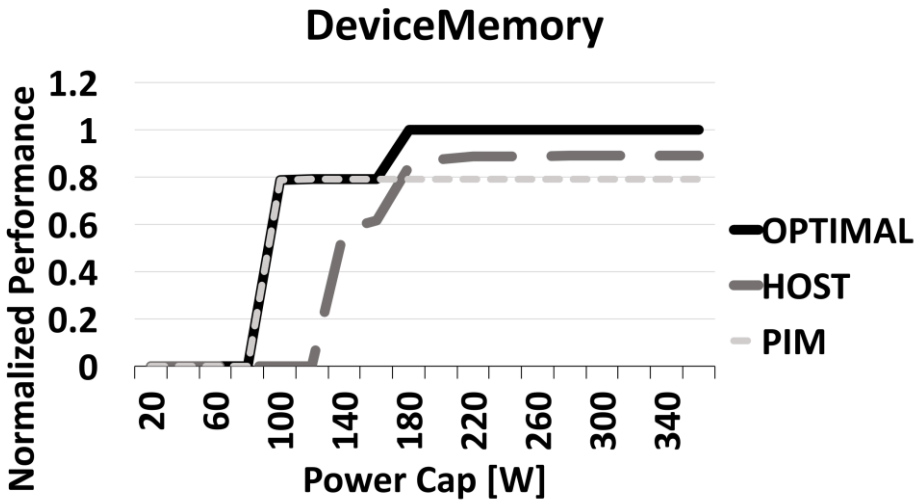


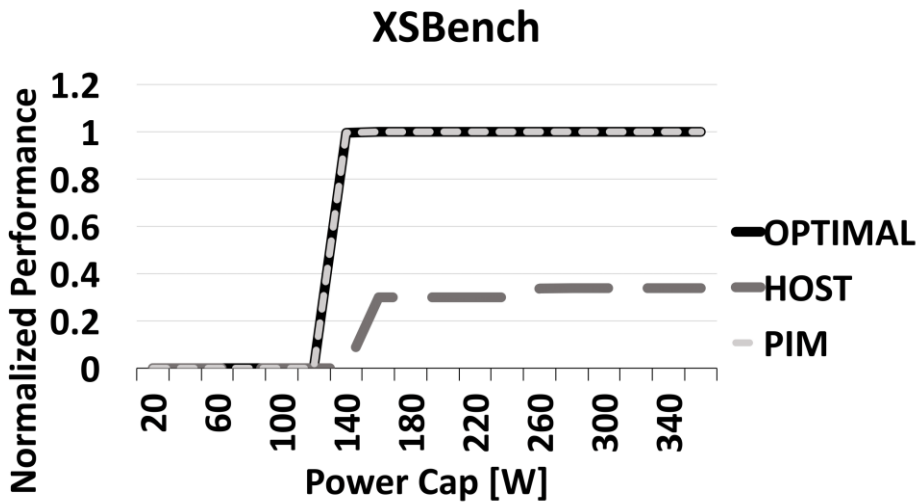


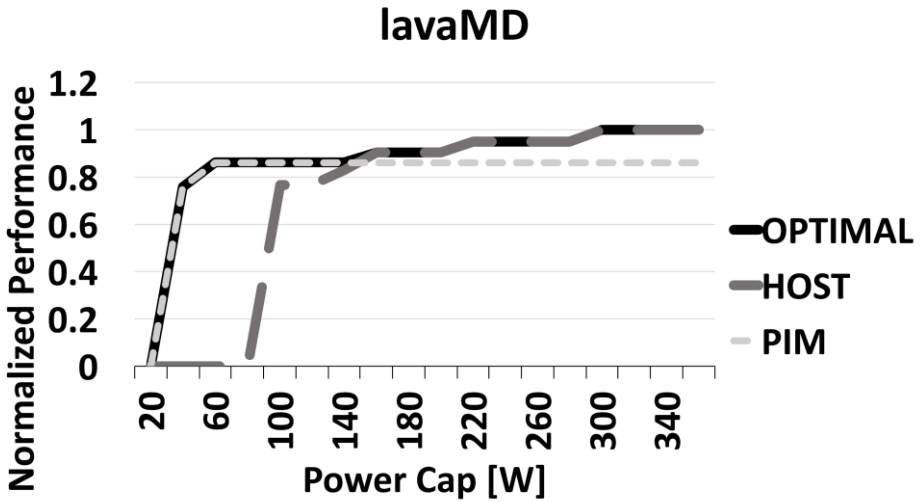


- A high performance host always consumes 100W and for lower power constraints PIMs can deliver good performance

Maximum performance under power constraint

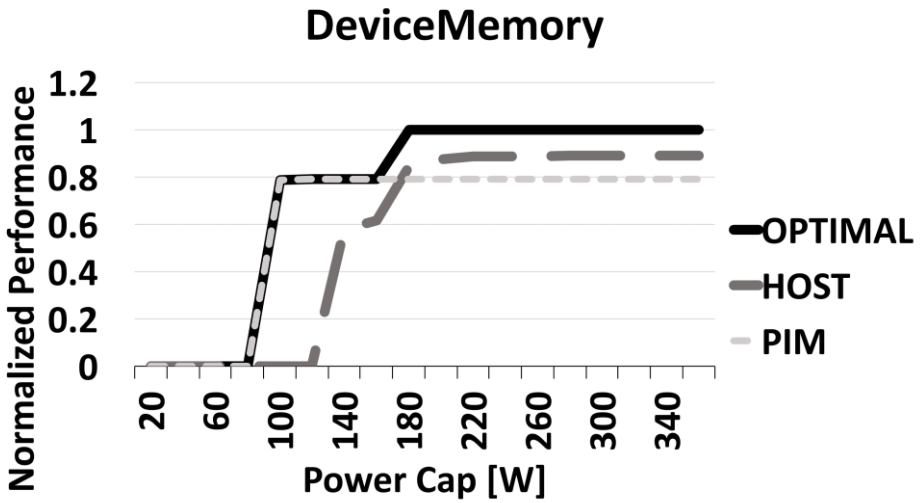


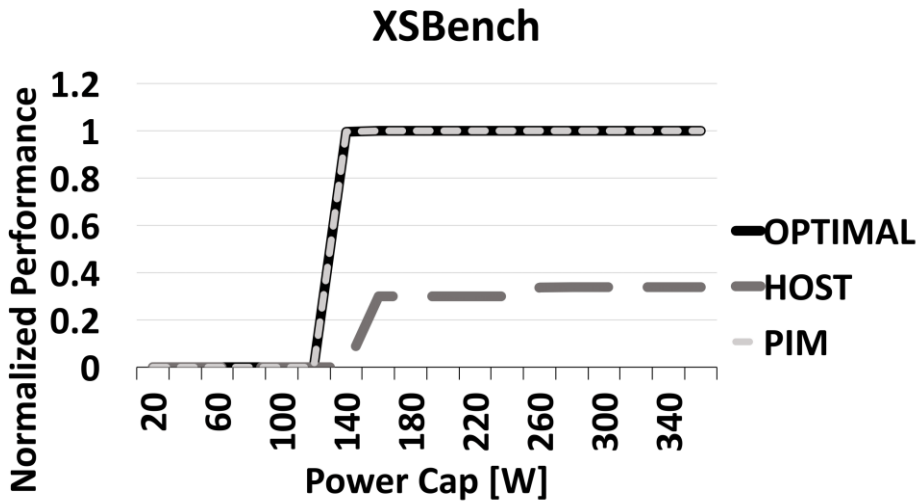


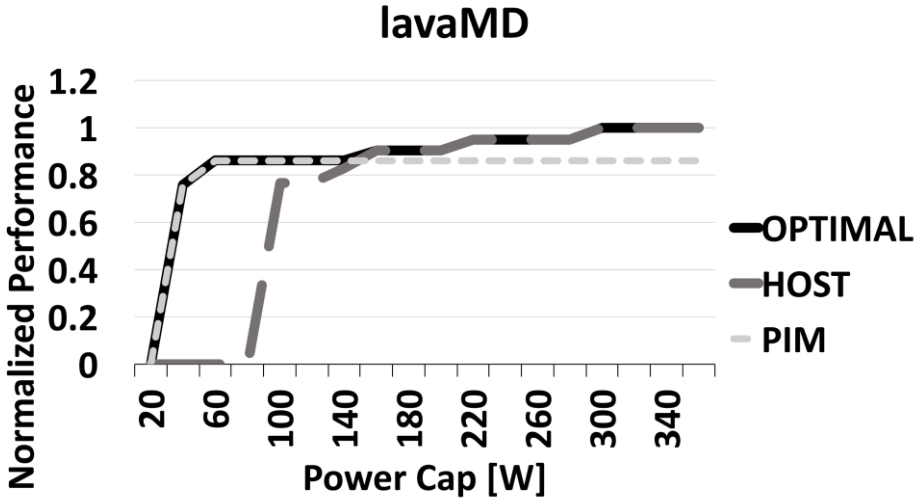


- A high performance host always consumes 100W and for lower power constraints PIMs can deliver good performance
- lavaMD – PIMs outperform host for even at higher power budgets (130W) by running on intermediate DVFS states

Maximum performance under power constraint

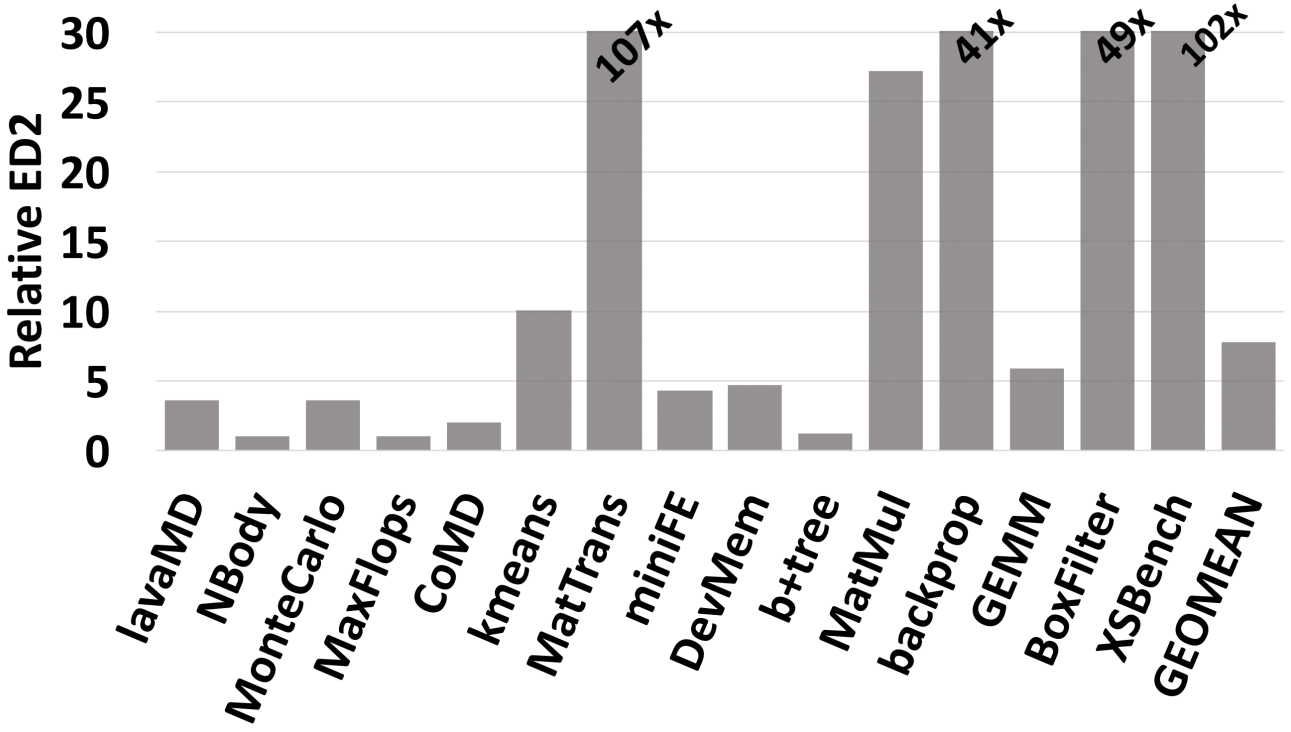






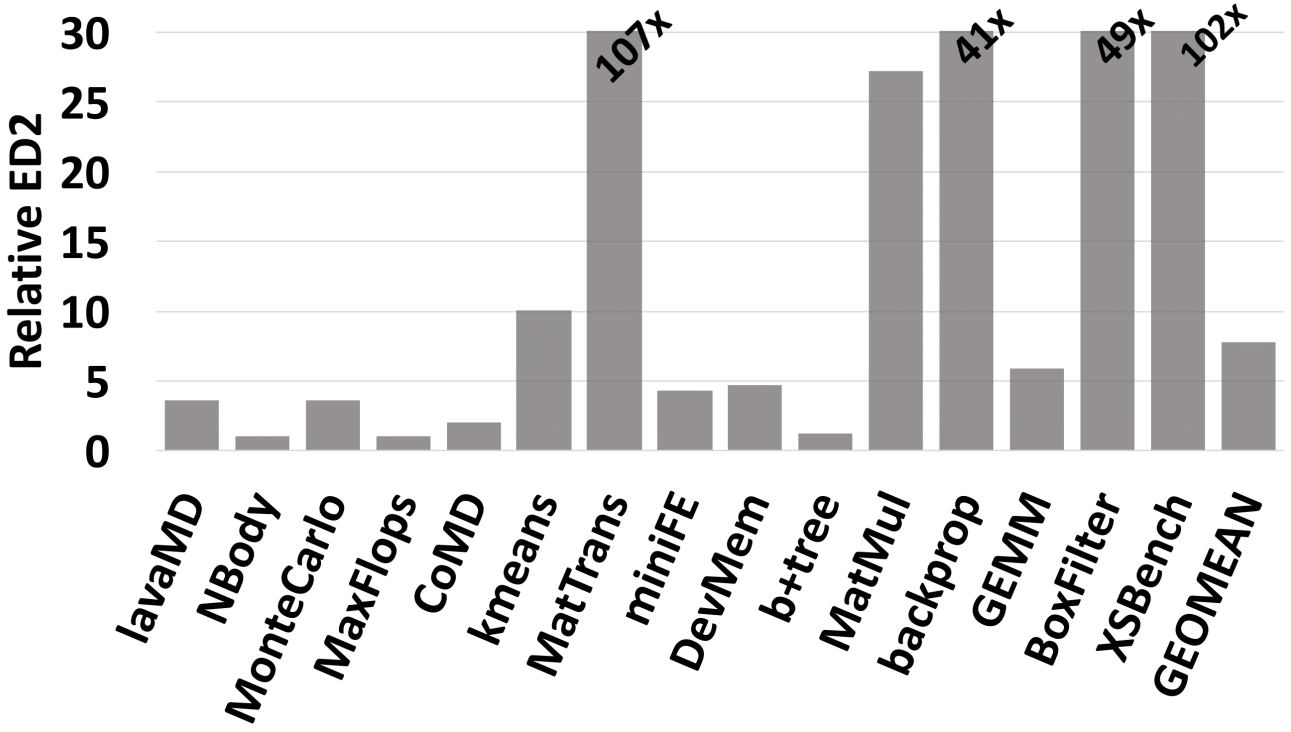
- A high performance host always consumes 100W and for lower power constraints PIMs can deliver good performance
- lavaMD – PIMs outperform host for even at higher power budgets (130W) by running on intermediate DVFS states
- PIMs are often a better choice even when host runs at lower DVFS states
- Application compensate the performance difference by exploiting higher memory bandwidth

Co-optimization and DVFS



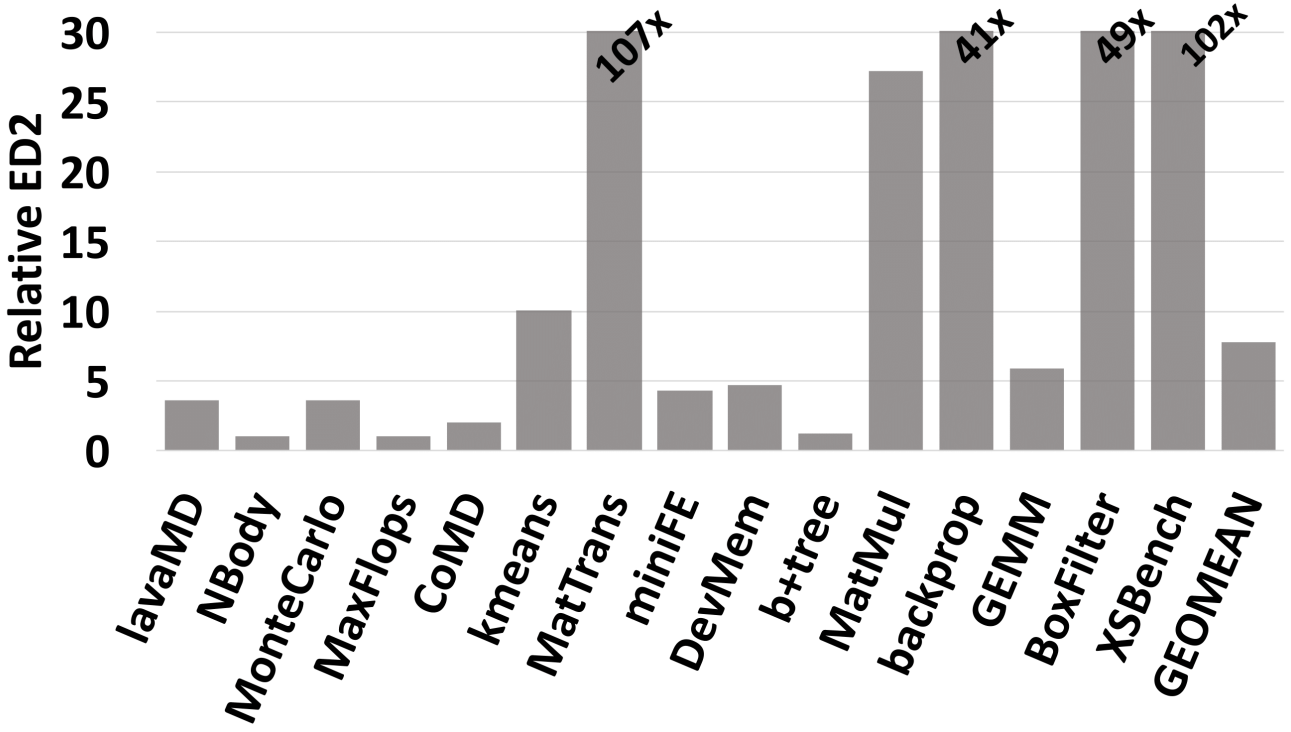
- Minimum achievable ED^2 on a hybrid host/PIMsystem with DVFS to a host/PIM system without DVFS (running on highest DVFS state) and host/PIM co-optimization

Co-optimization and DVFS



- Minimum achievable ED^2 on a hybrid host/PIMsystem with DVFS to a host/PIM system without DVFS (running on highest DVFS state) and host/PIM co-optimization
- By picking the right DVFS state and right hardware to run the kernel we can on average improve energy efficiency by 7x

Co-optimization and DVFS



- Minimum achievable ED^2 on a hybrid host/PIMsystem with DVFS to a host/PIM system without DVFS (running on highest DVFS state) and host/PIM co-optimization
- By picking the right DVFS state and right hardware to run the kernel we can on average improve energy efficiency by 7x
- Our findings strengthen the hypothesis of PIMs being a useful heterogeneous platform and show the importance of DVFS as a mean to maximize performance and energy efficiency in HPC systems with PIM

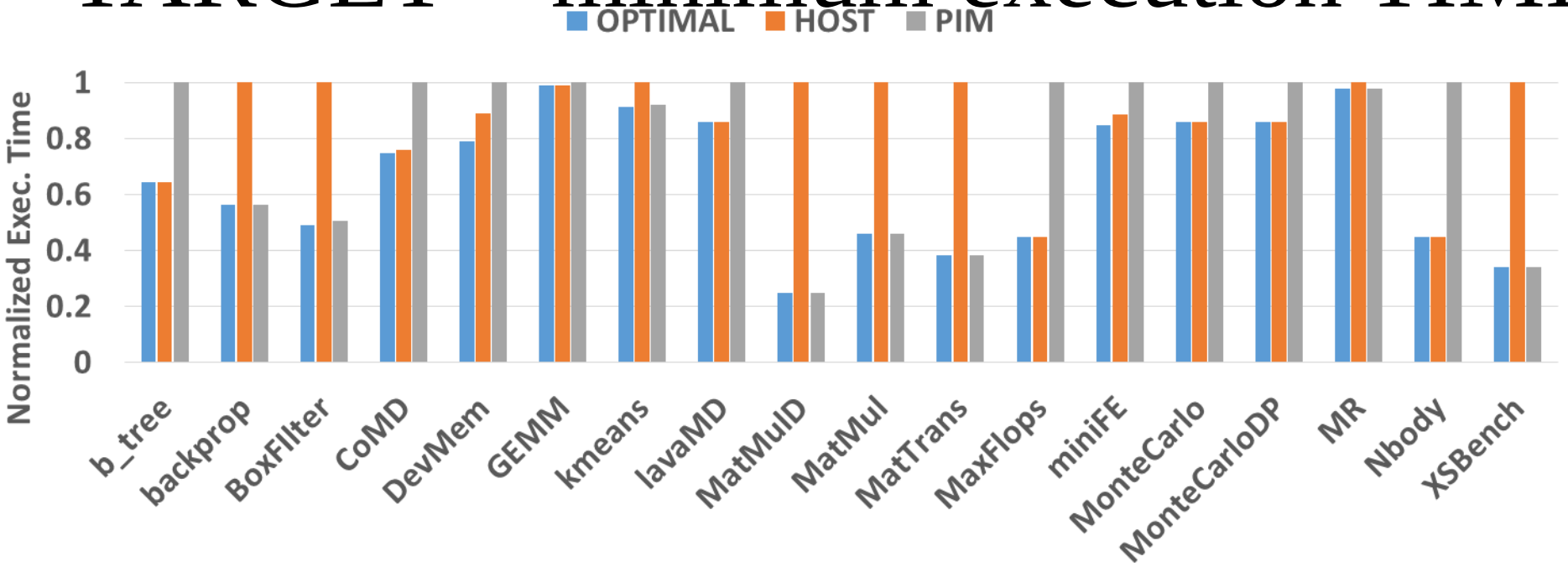
CONCLUSION

- PIM can offer both high performance and low power for memory intensive applications
- Host maximizes performance for compute bound applications but consumes more power than PIMs
- Additional power savings can be obtained with lower DVFS states
- PIMs are more energy efficient compared to host in most cases, even when the host runs at low power DVFS states
- PIMs can deliver significantly better performance under tight power budgets by exploiting high in-stack bandwidth and compensating performance loss from low power DVFS states

Thank you

BACKUP SLIDES

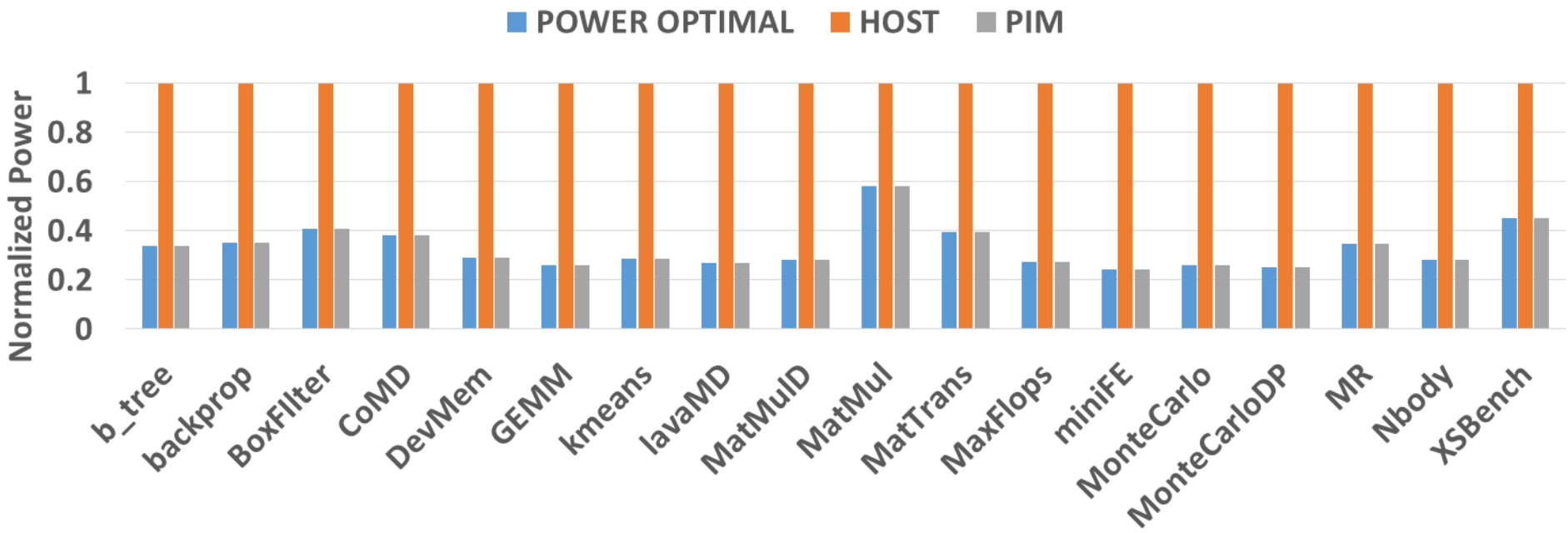
TARGET = minimum execution TIME



- B+tree, MaxFlops, MonteCarlo, Nbody – all compute bound kernels which tend to favor execution only on host
- Backprop, MatrixMultiplication, MatrixTranspose, XSBench – all bandwidth bound kernels which favor PIM

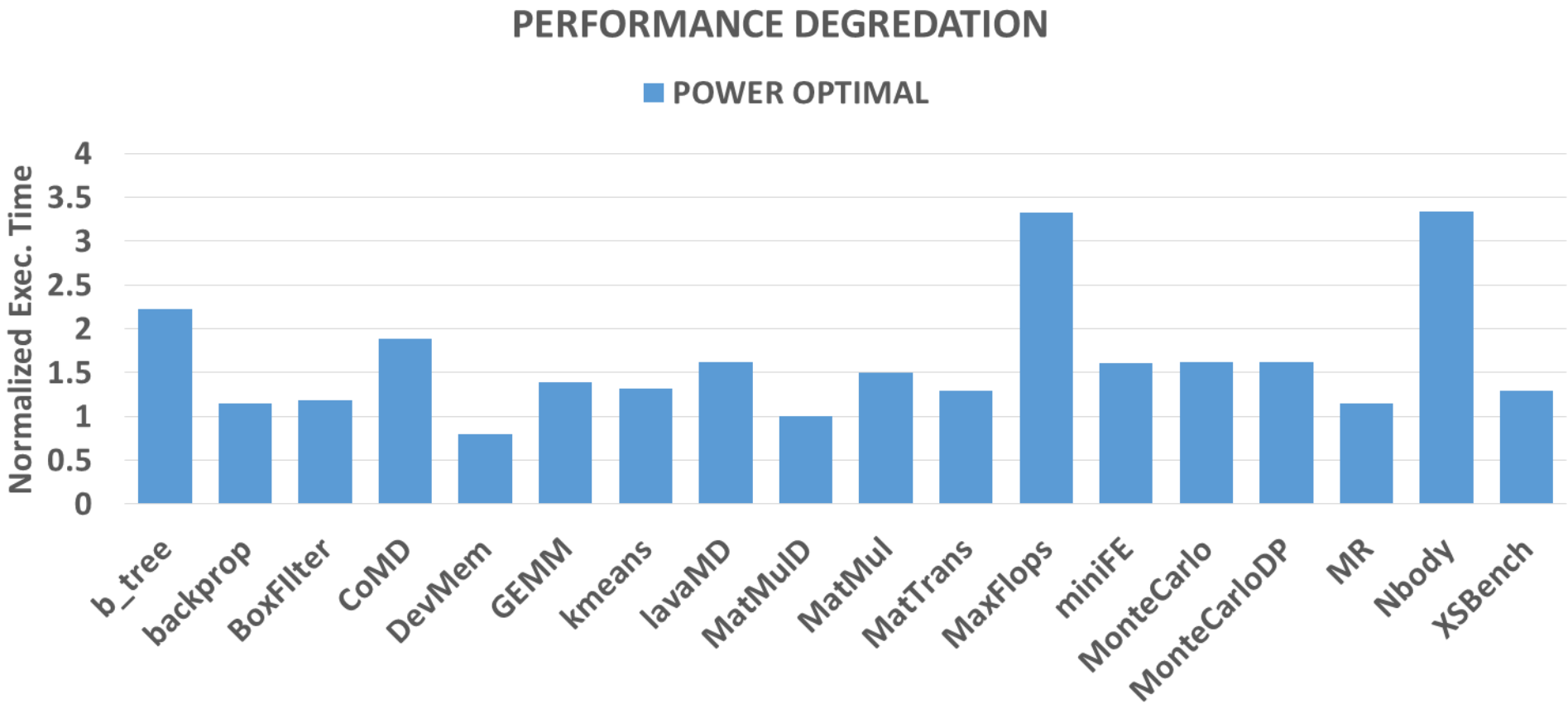
- CoMD, GEMM, miniFE – dominated by one type of kernel, other kernels may prefer PIM over host or vice versa
- Always picking the highest DVFS state – maximizing performance

TARGET: MINIMUM POWER

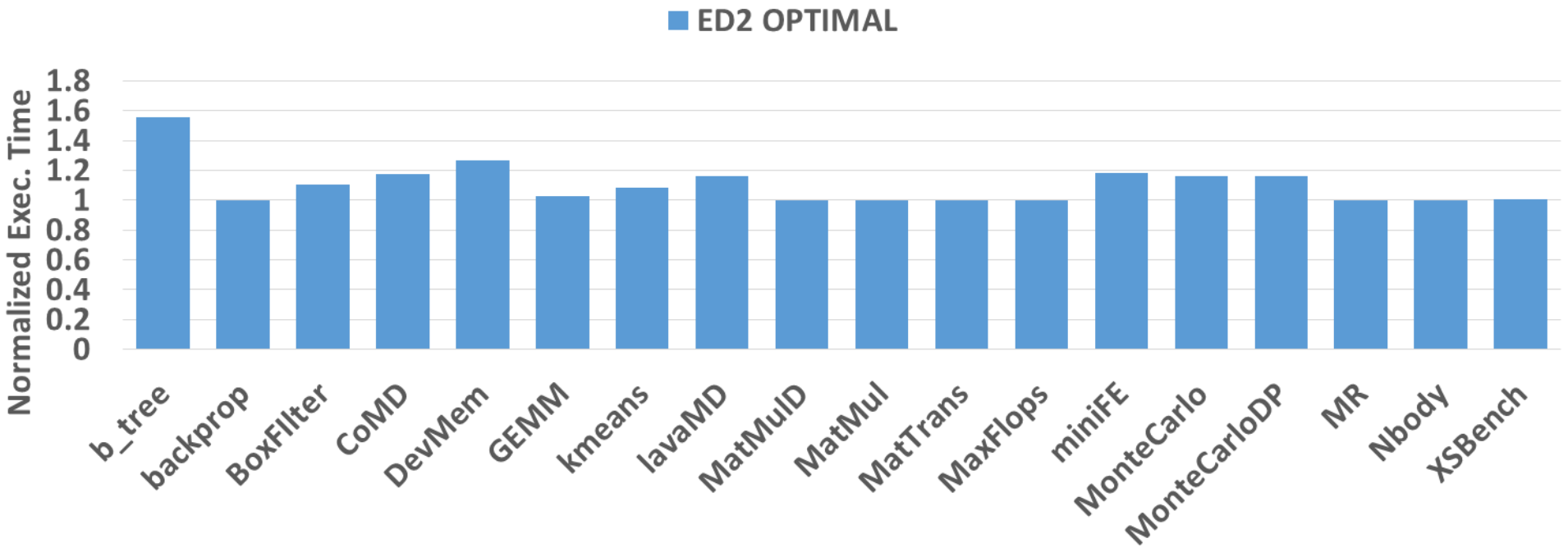


- Always picking PIM and the lowest DVFS states – minimum power
- Even though we get significant power savings when running kernels on PIM, performance will be significantly lower for compute intensive kernels

Performance degradation, Target = Minimum POWER

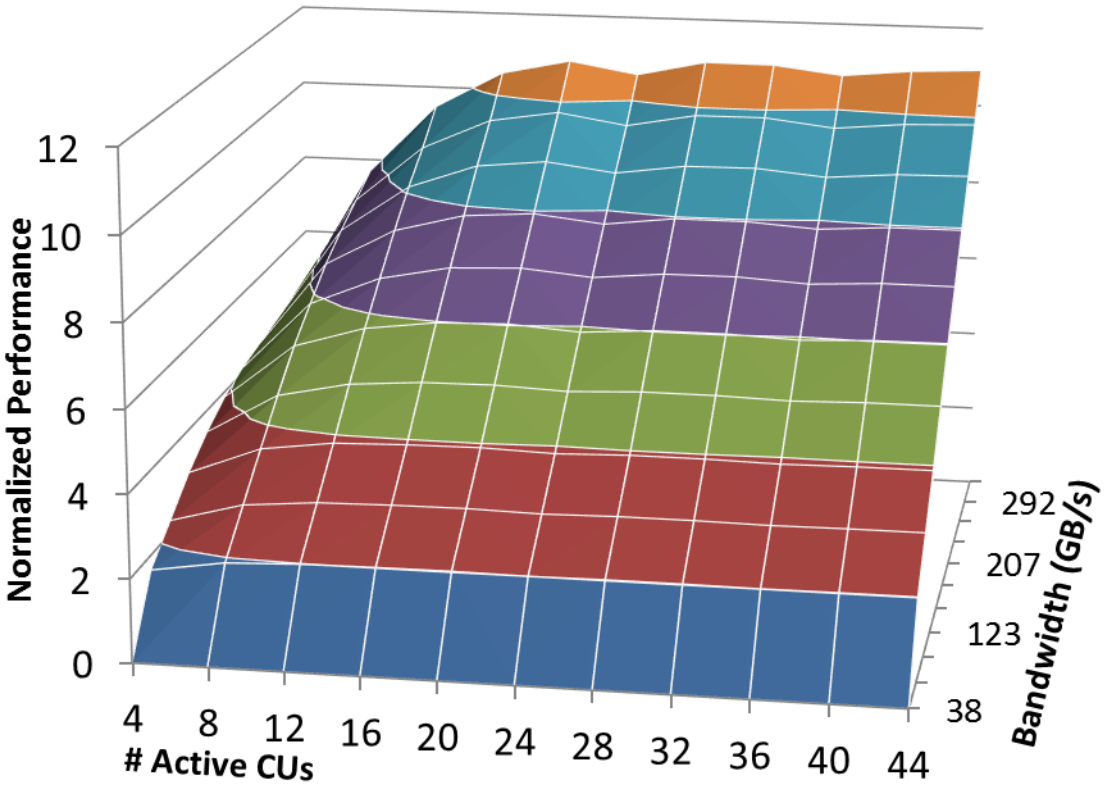


Performance degradation, target=min. ed2



Performance model BACKUP

60 CUs
360GB/s

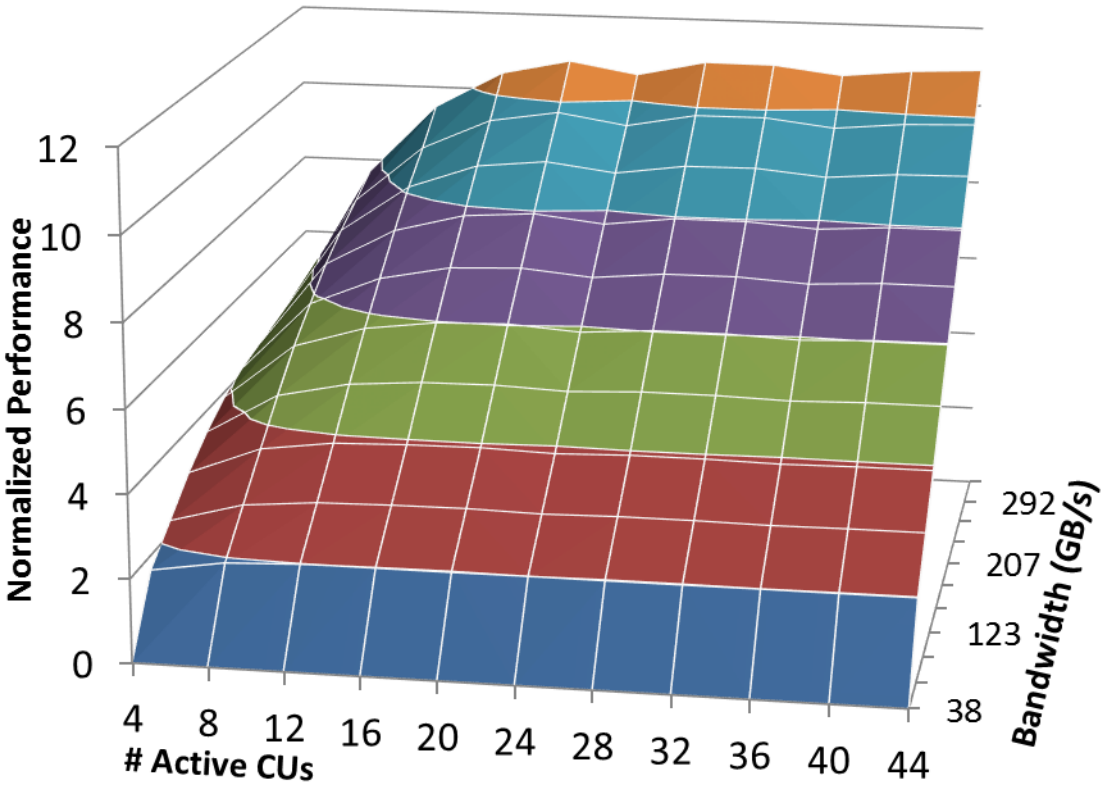


- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?

- Assumption: Performance change across the same CU/BW ratio remains the same

Performance model BACKUP

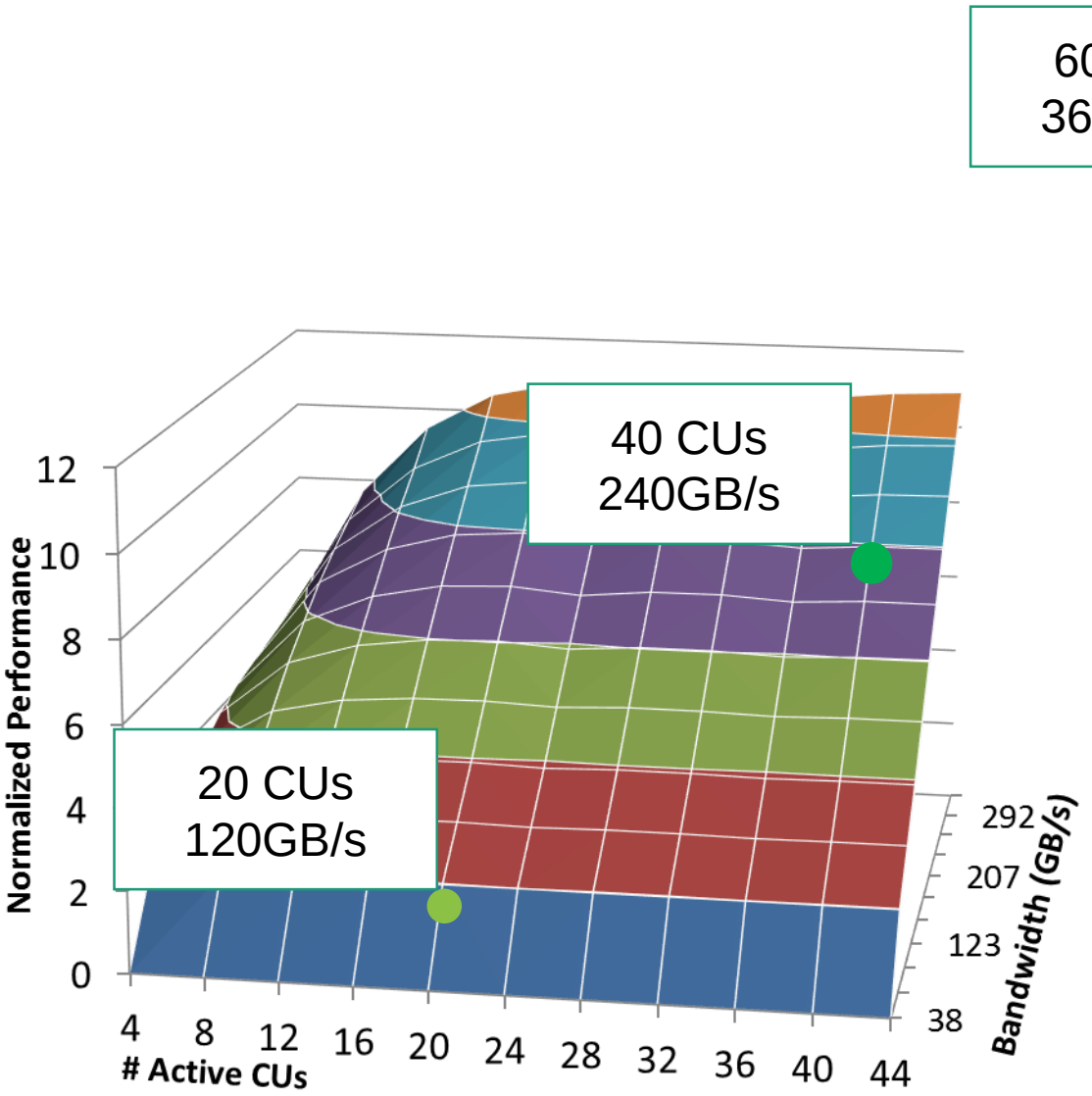
60 CUs
360GB/s



- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW

- Assumption: Performance change across the same CU/BW ratio remains the same

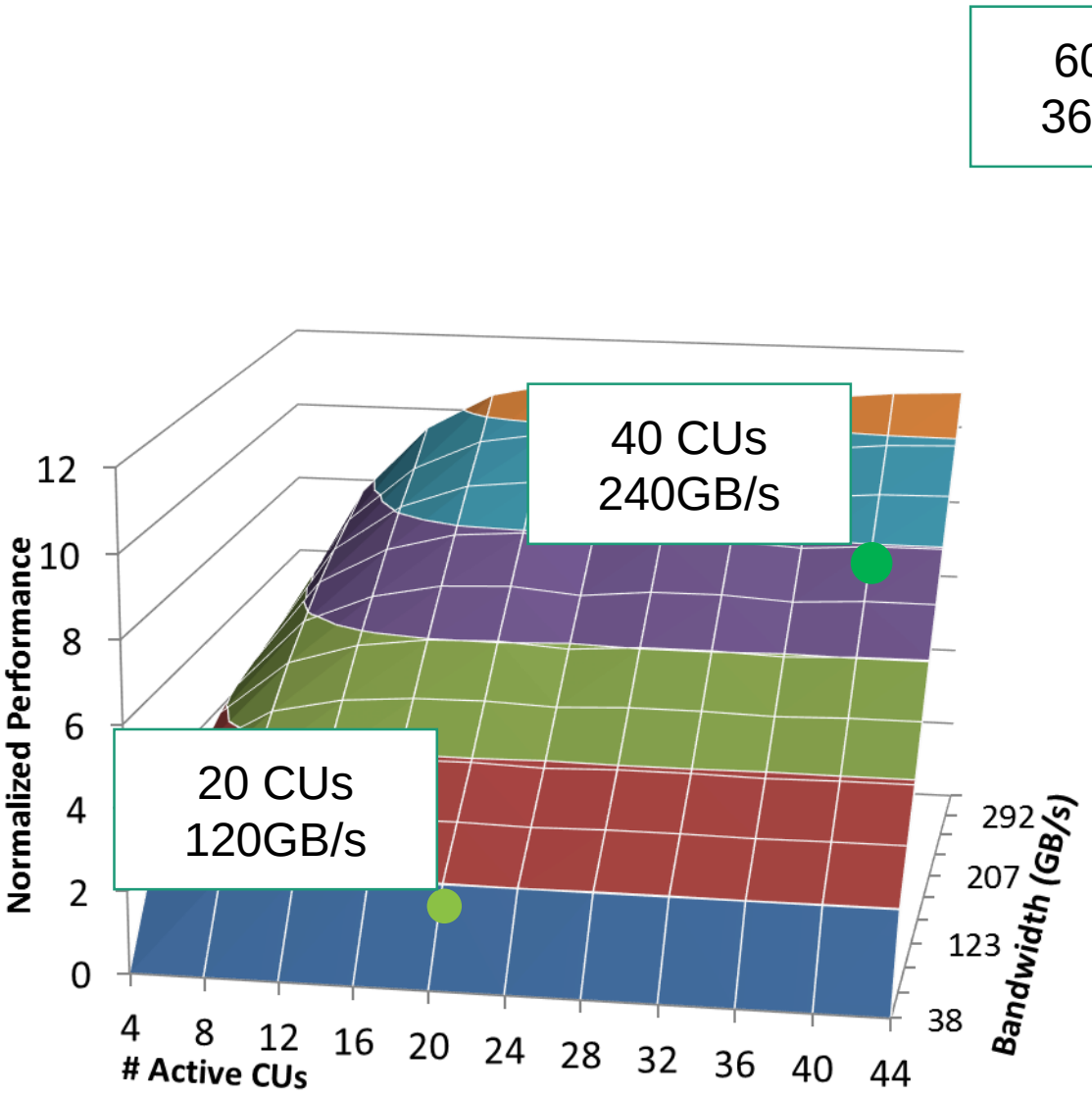
Performance model BACKUP



- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW

- Assumption: Performance change across the same CU/BW ratio remains the same

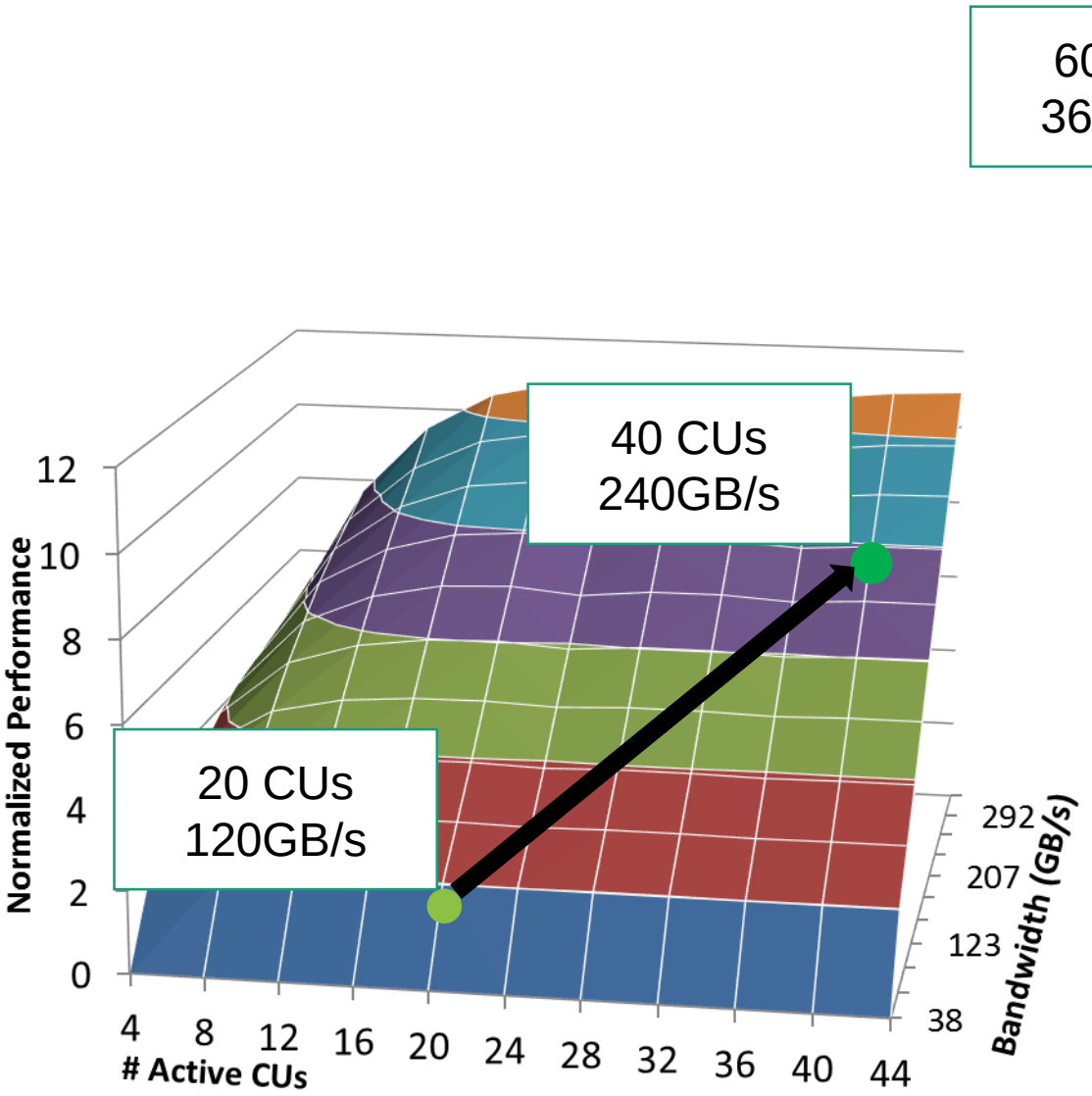
Performance model BACKUP



- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW
- Find the slope on which these 2 points lie

- Assumption: Performance change across the same CU/BW ratio remains the same

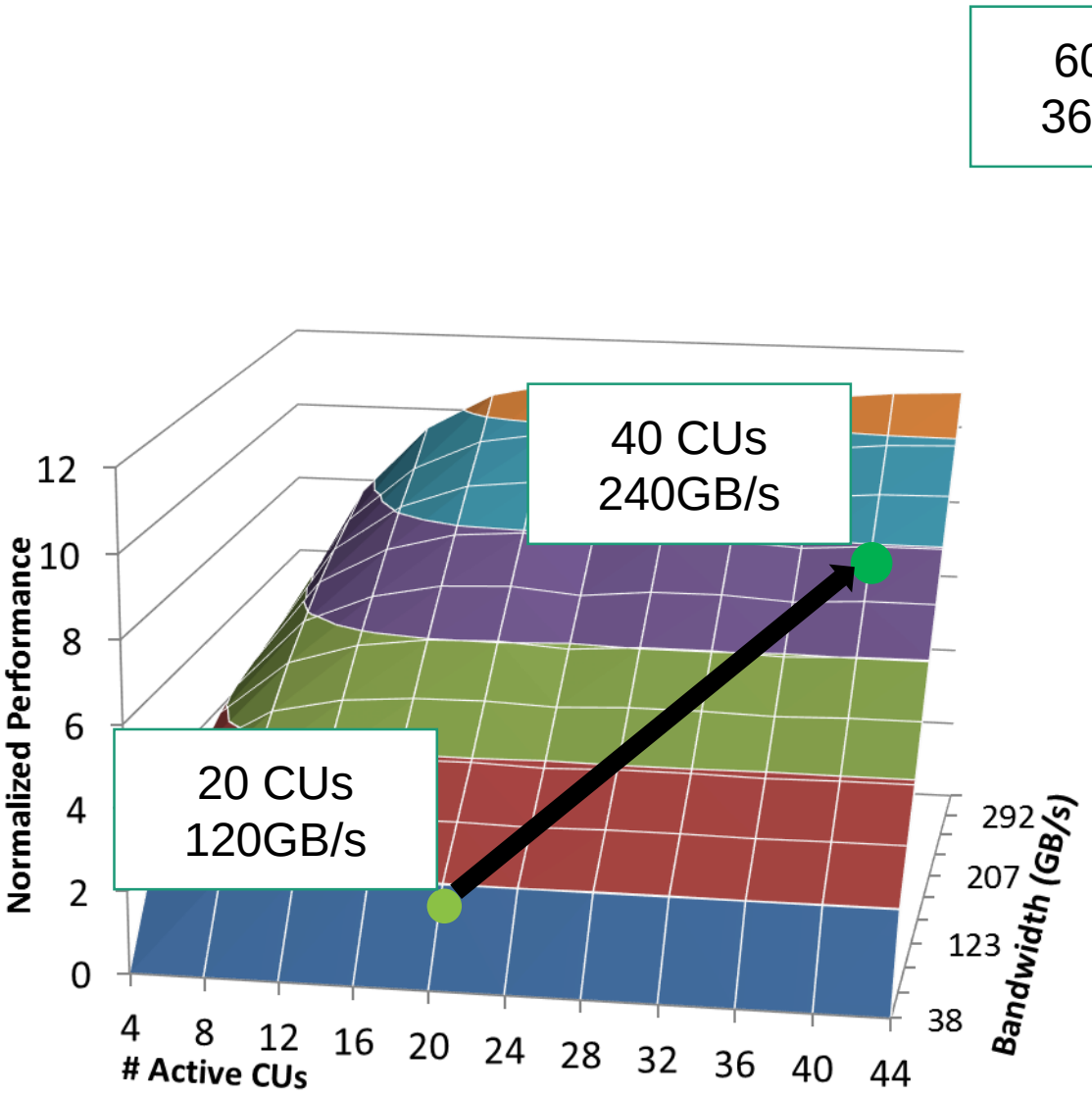
Performance model BACKUP



- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW
- Find the slope on which these 2 points lie

- Assumption: Performance change across the same CU/BW ratio remains the same

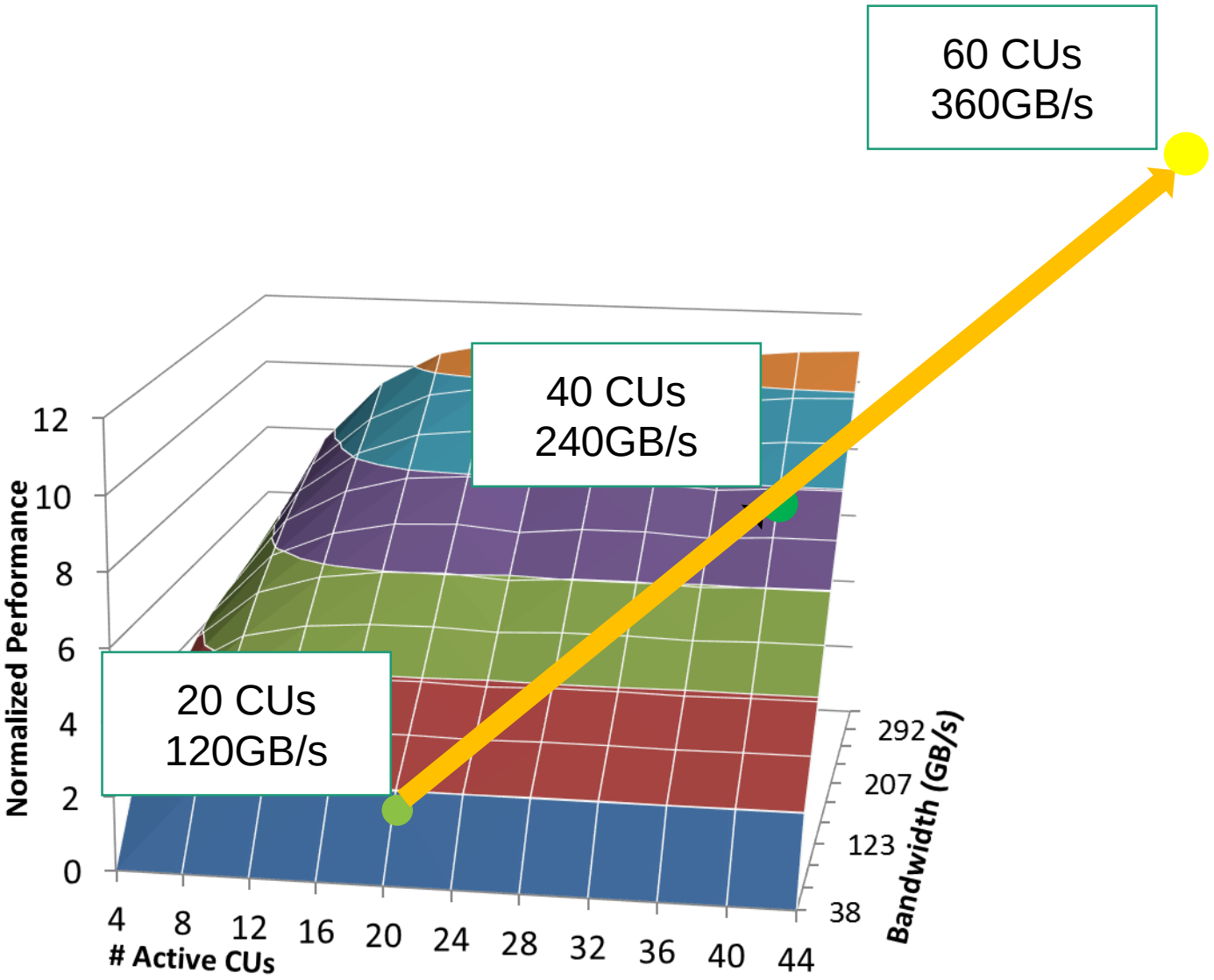
Performance model BACKUP



- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW
- Find the slope on which these 2 points lie
- Extrapolate (follow the slope) to target HW (60 CUs)

- Assumption: Performance change across the same CU/BW ratio remains the same

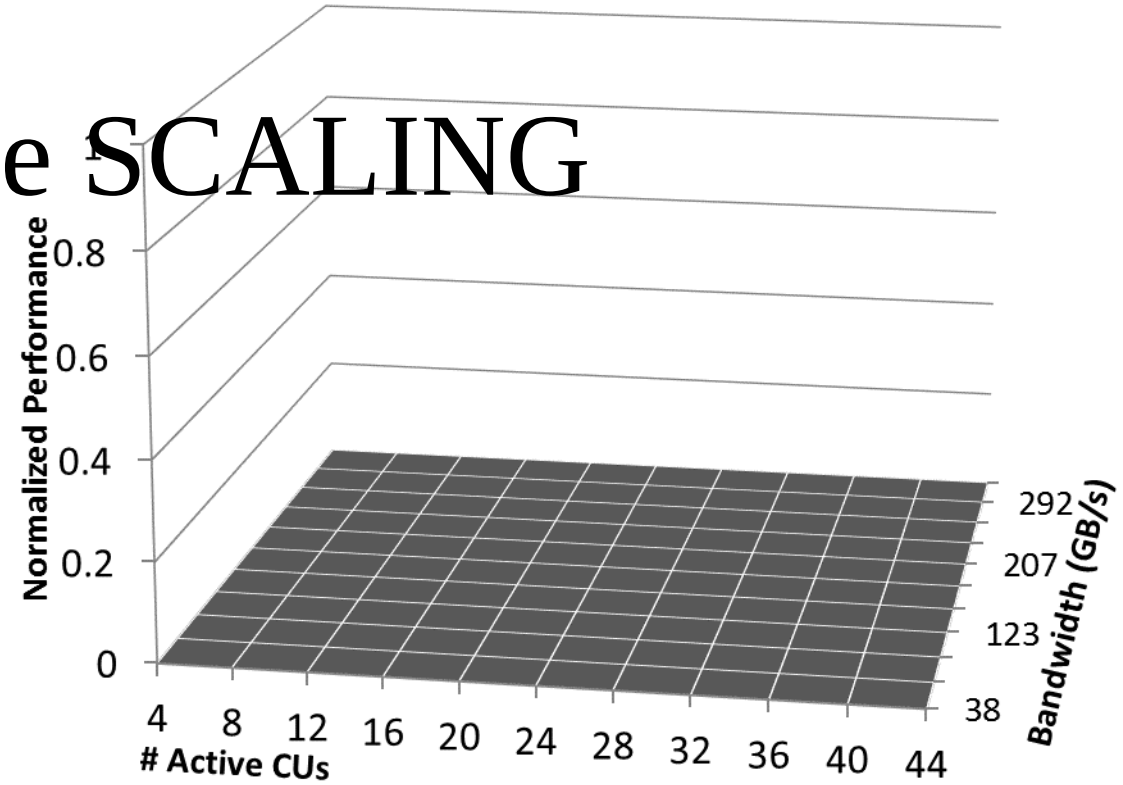
Performance model BACKUP



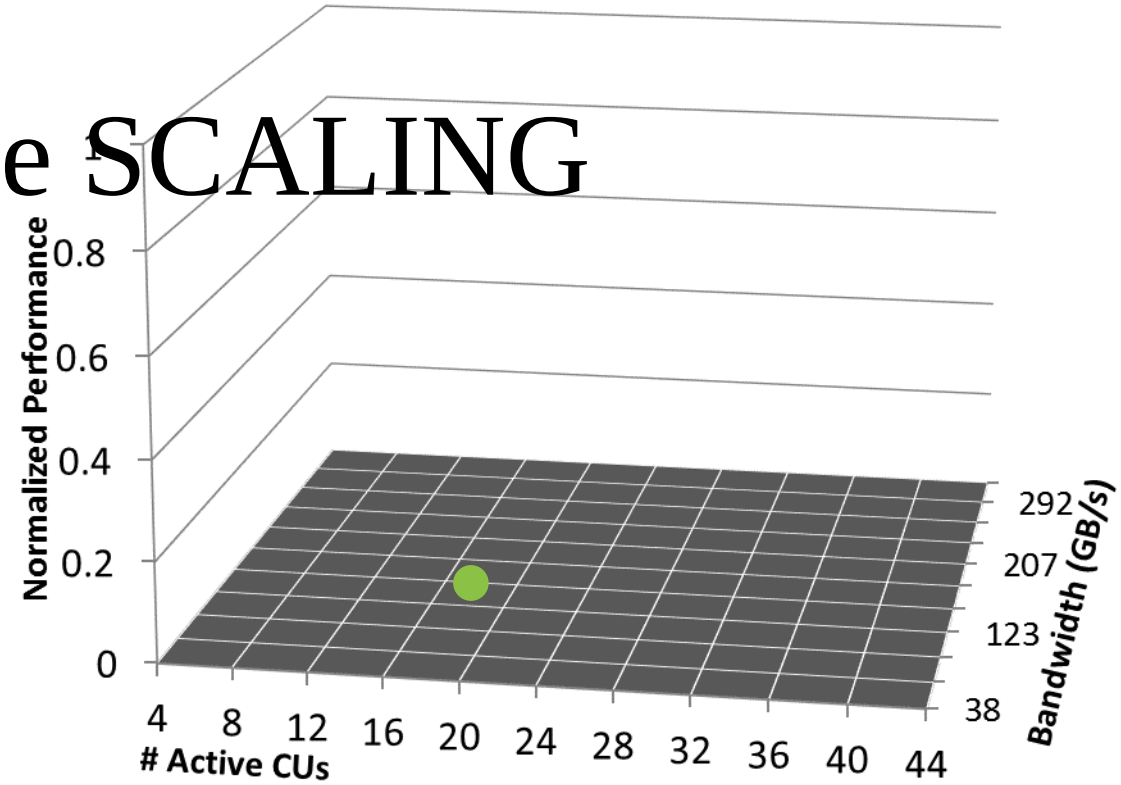
- How do we estimate performance at 164 CUs and 360 GB/s bandwidth?
- Find 2 points which have the same compute/bandwidth ratio as the target HW
- Find the slope on which these 2 points lie
- Extrapolate (follow the slope) to target HW (60 CUs)

- Assumption: Performance change across the same CU/BW ratio remains the same

GPU Performance SCALING

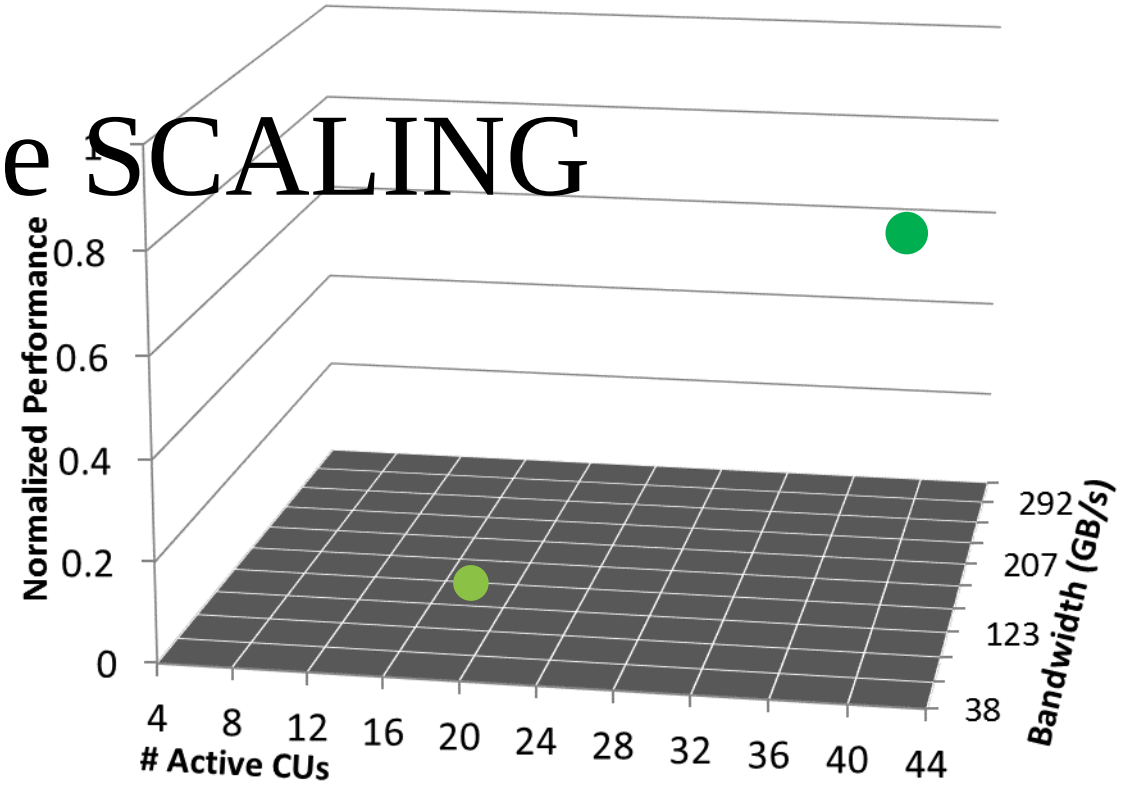


GPU Performance SCALING



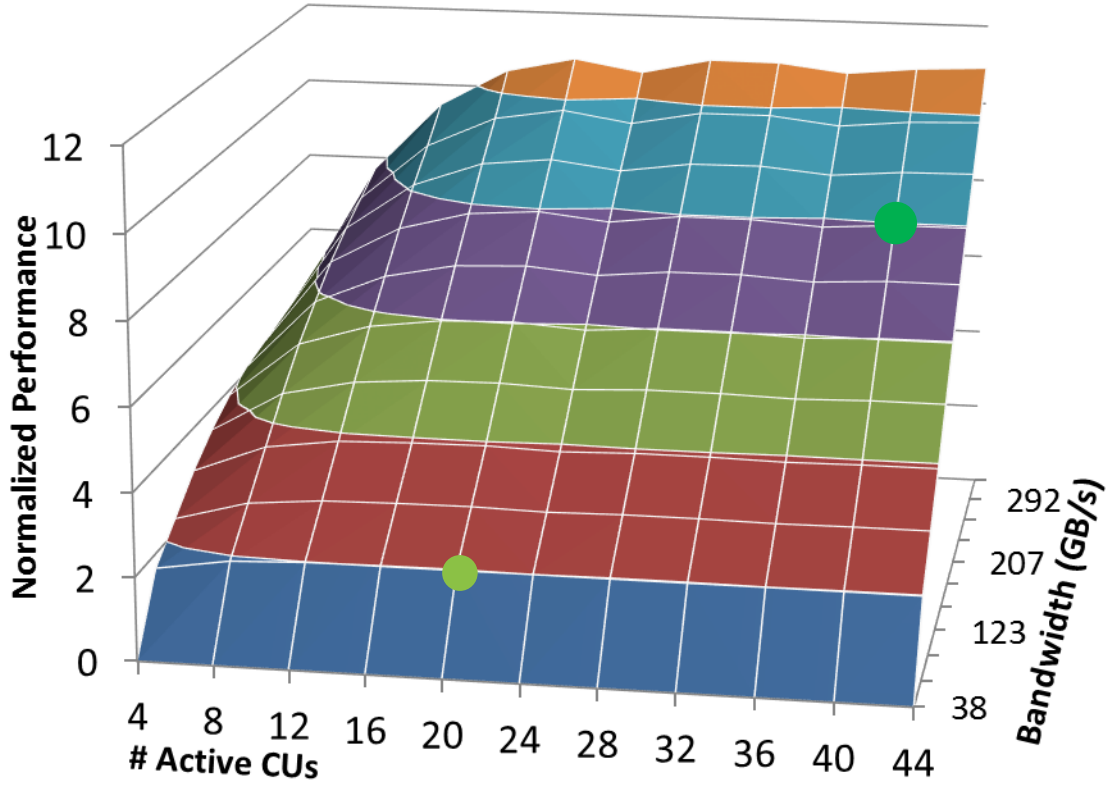
- Gather performance & power data at Base Hardware Configuration

GPU Performance SCALING



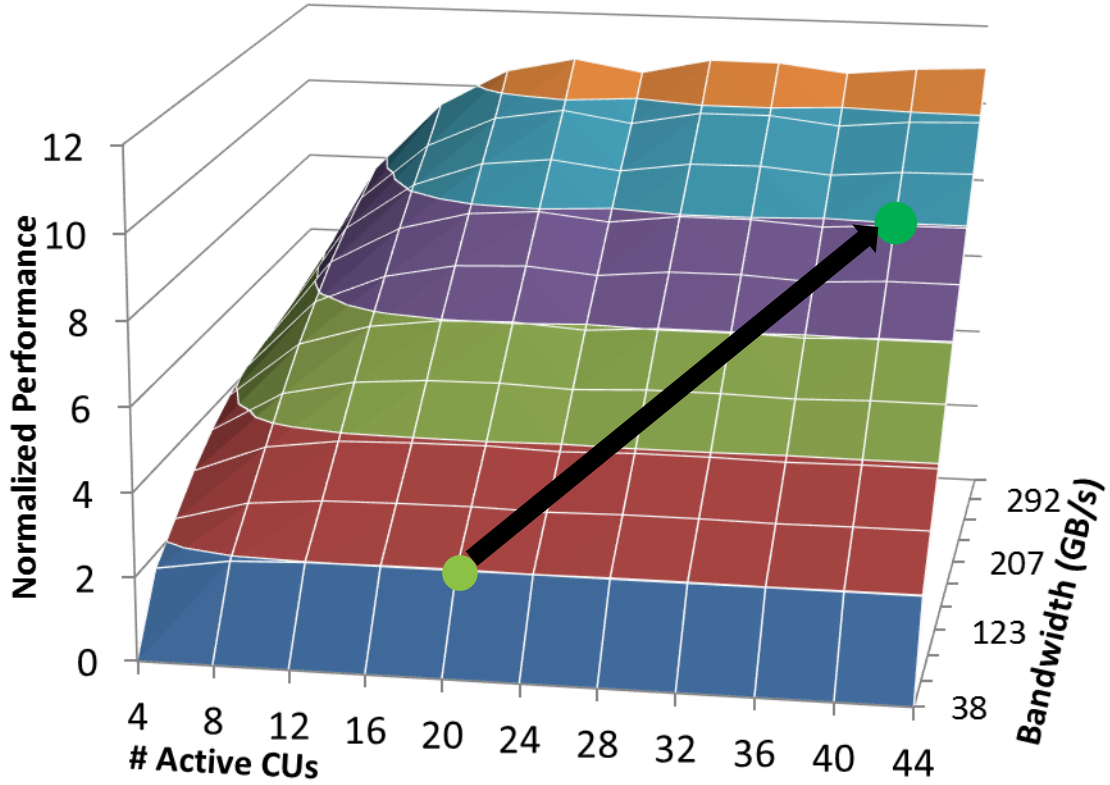
- Gather performance & power data at Base Hardware Configuration
- Start from a Base Hardware Configuration and predict performance for a Target Hardware Configuration

GPU Performanc



- Gather performance & power data at Base Hardware Configuration
- Start from a Base Hardware Configuration and predict performance for a Target Hardware Configuration
- How does the kernel scale? Ask the ML model!

GPU Performanc



- Gather performance & power data at Base Hardware Configuration
- Start from a Base Hardware Configuration and predict performance for a Target Hardware Configuration
- How does the kernel scale? Ask the ML model!

Arithmetic Intensity