

# Exploring Latency-Power Tradeoffs in Deep Nonvolatile Memory Hierarchies



Doe Hyun Yoon, Tobin Gonzalez,  
Parthasarathy Ranganathan, and Robert S. Schreiber

Intelligent Infrastructure Lab (IIL), Hewlett-Packard Labs

# Deeper and Deeper Memory Hierarchy

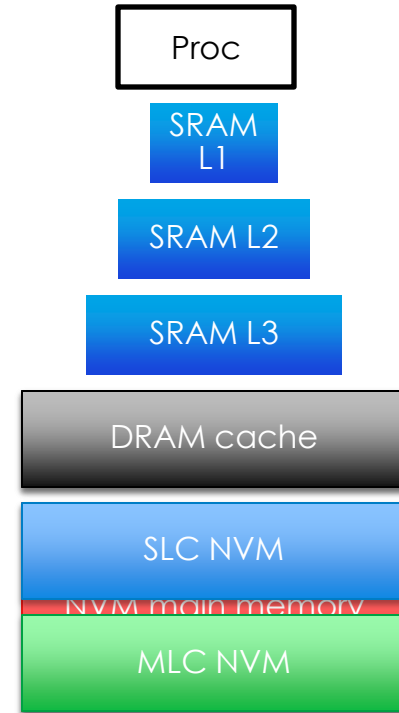
Cache memory for reducing average memory latency

- Initially, a small L1 cache
- Then, L2, L3, ...
- DRAM cache for NVM main memory
- SLC / MLC NVM memory

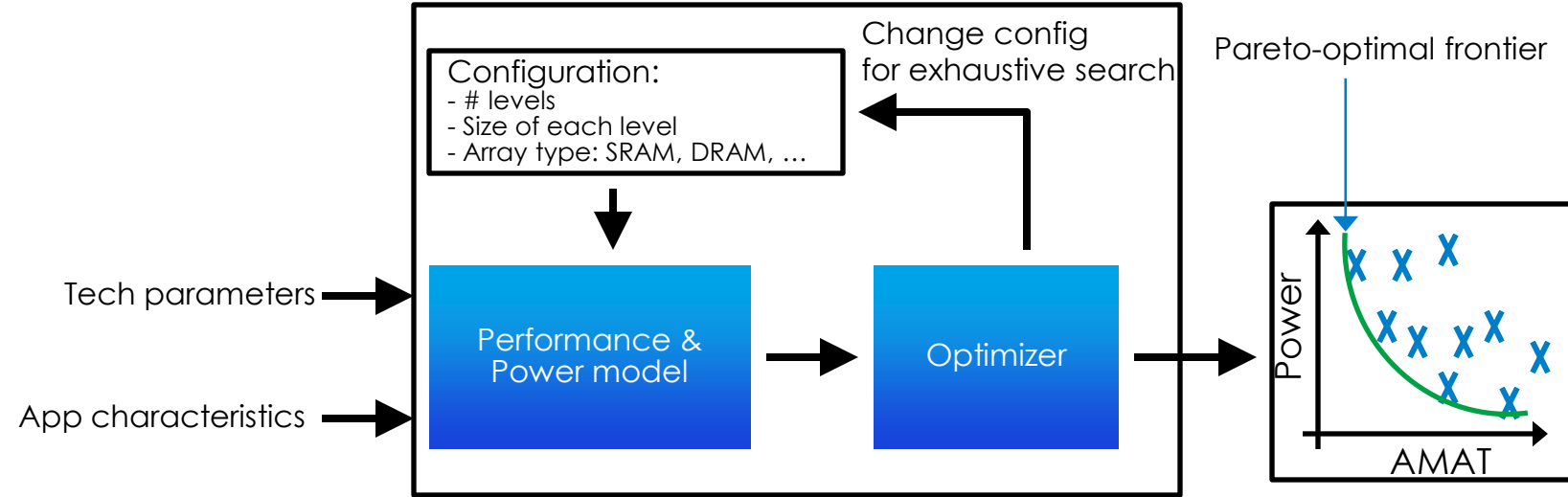
Is a DEEP hierarchy power efficient?

We develop a model to explore latency-power tradeoffs

*Flattened* hierarchies with a large *NVM* cache are power efficient

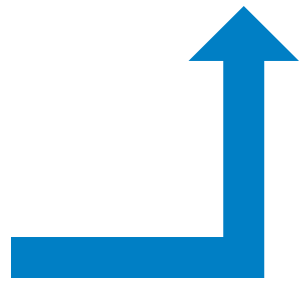
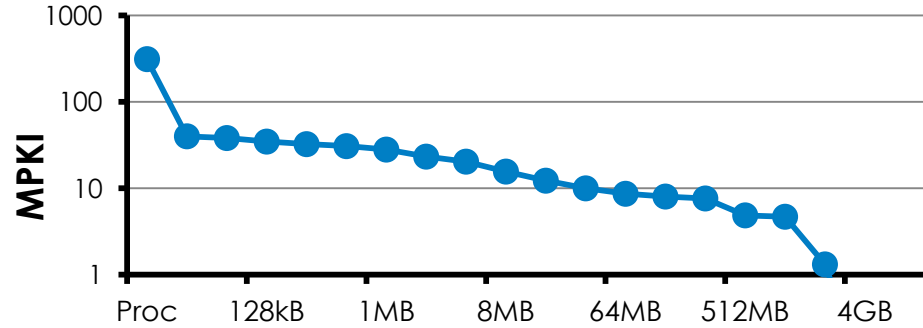
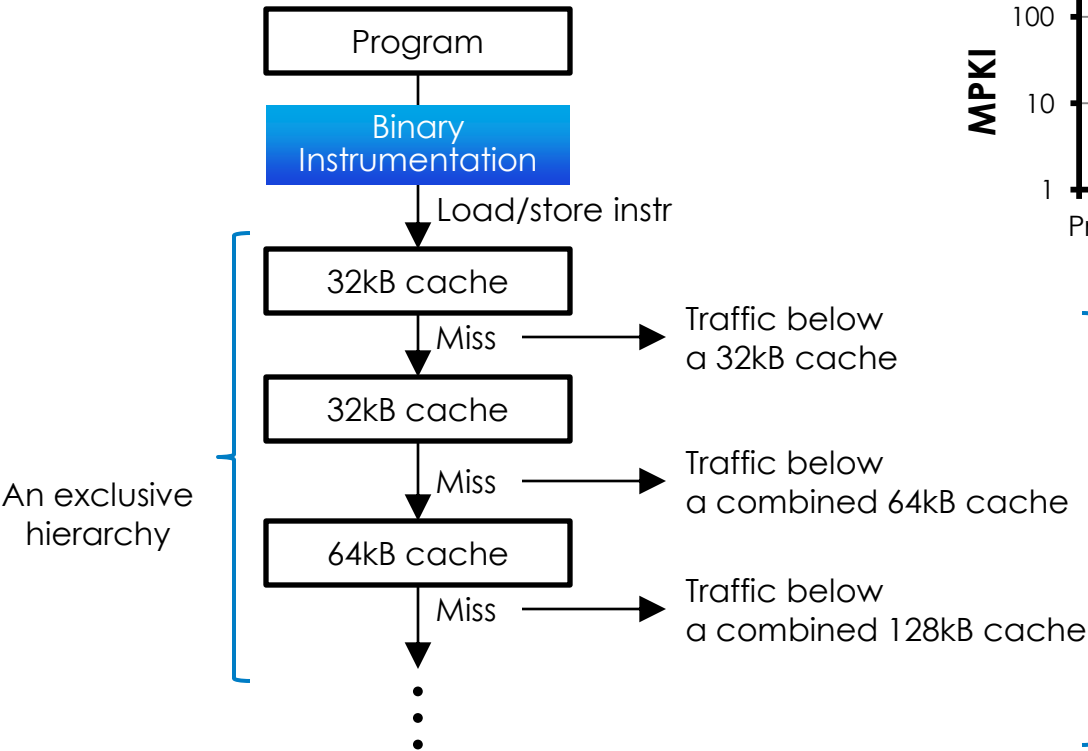


# 1. Latency-Power Model



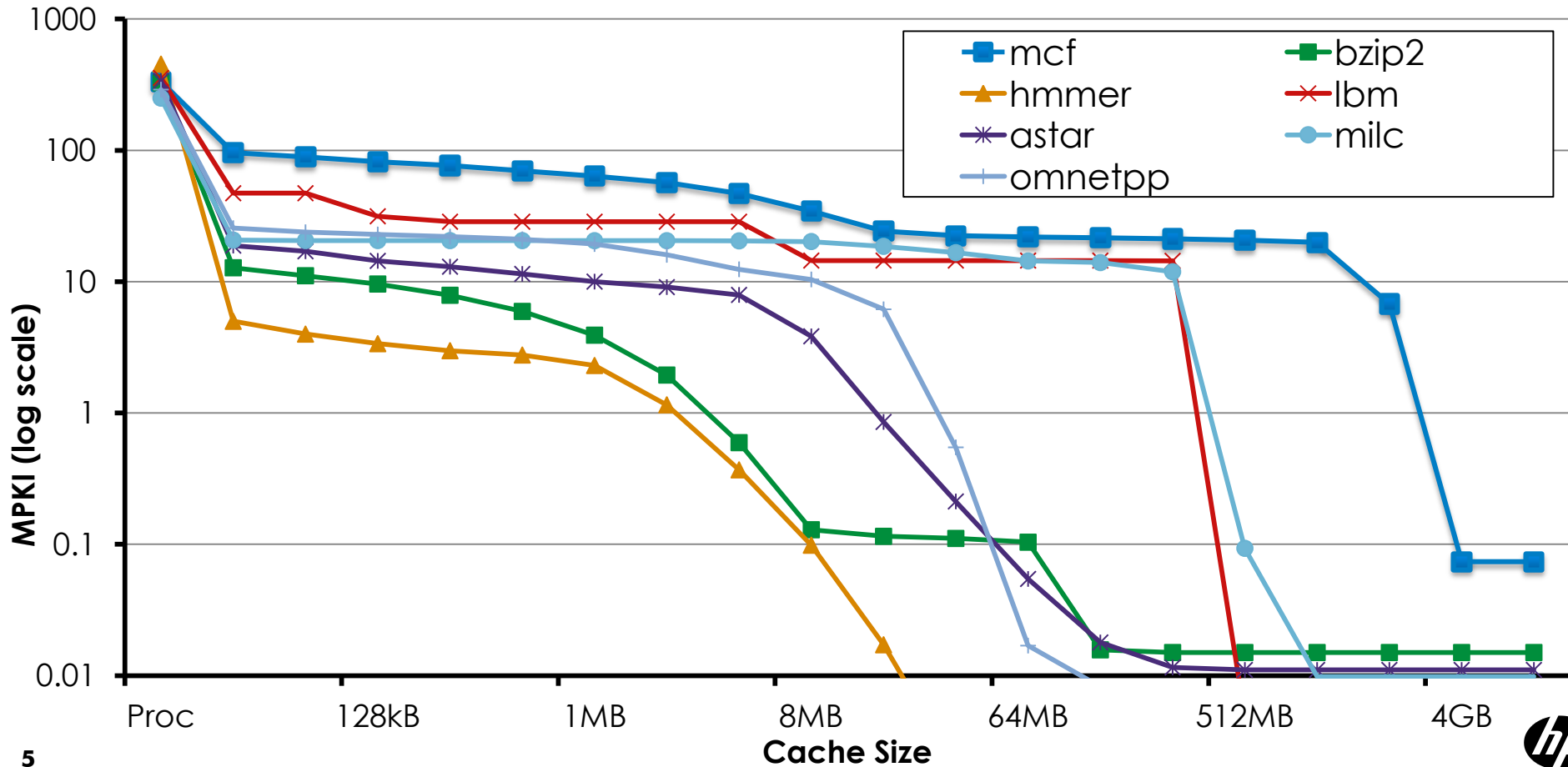
Estimate power and performance of an application  
Exhaustive search to find Pareto-optimal frontier

# Application Characteristics

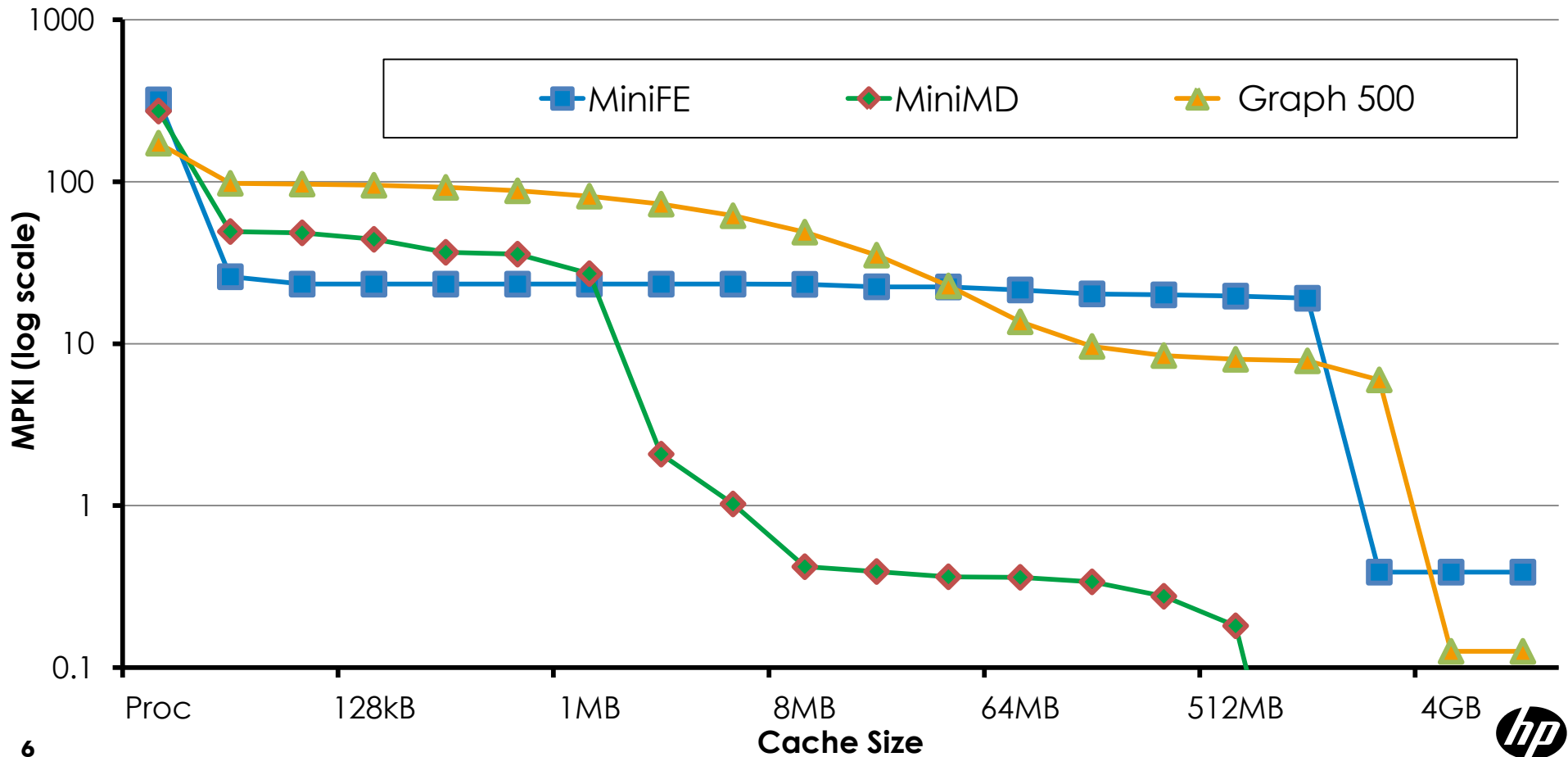


Profile MPKI vs. cache sizes

# Example) SPEC CPU 2006



# Example) MiniFE, MiniMD, and Graph 500



# Technology Parameters

Latency, static power, and energy per access  
of different caches

SRAM and DRAM → CACTI 6


Phase-Change RAM (PCRAM) → NVSim

# Performance Model

Use AMAT (Average Memory Access Time)

$$AMAT = L(1) + \sum [ M(i) / M(0) \times L(i+1) ]$$

# loads / KI



Total N cache levels

L(i): Latency of cache level i (tech parameter)

M(i): MPKI of cache level i (app characteristic)



# Power Model

$$P = P_{\text{static}} + P_{\text{dyn}}$$

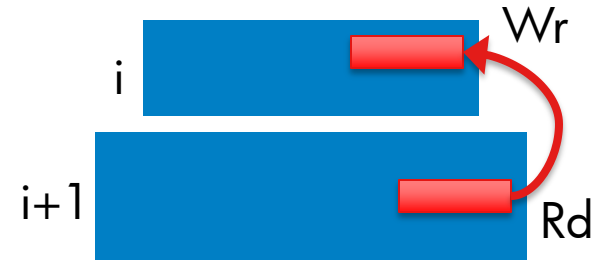
$$P_{\text{static}} = \sum P_s(i)$$

$$E_{\text{dyn}} = M(0) \times E(1) + \sum [ M(i) \times (E_r(i+1) + E_w(i)) ]$$

$$P_{\text{dyn}} = E_{\text{dyn}} / [ (1000 - M(0)) \times T_{\text{cyc}} + \text{AMAT} \times M(0) ]$$

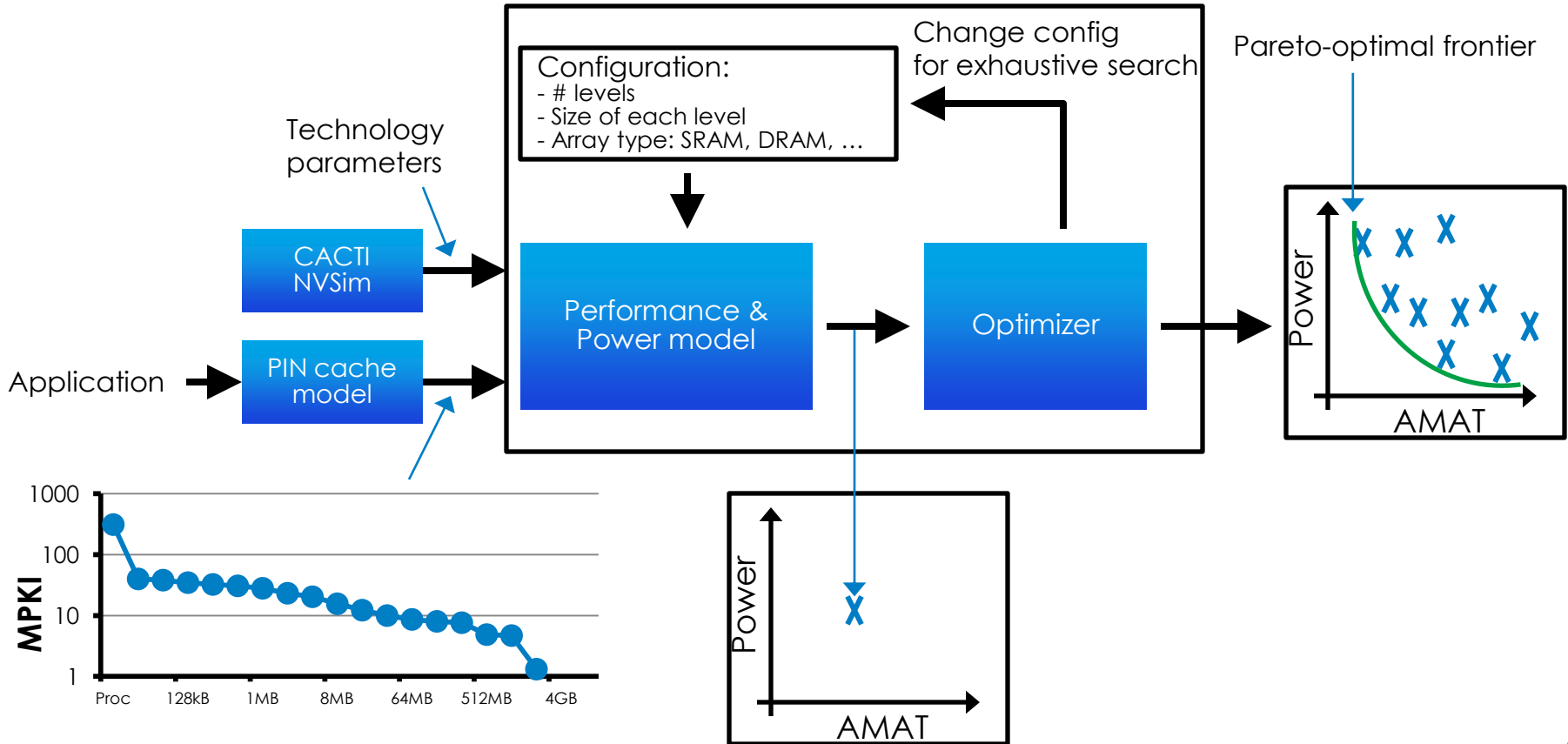
$E(i)$ : Energy per access (tech parameter)

$T_{\text{cyc}}$ : cycle time (1ns in our study)

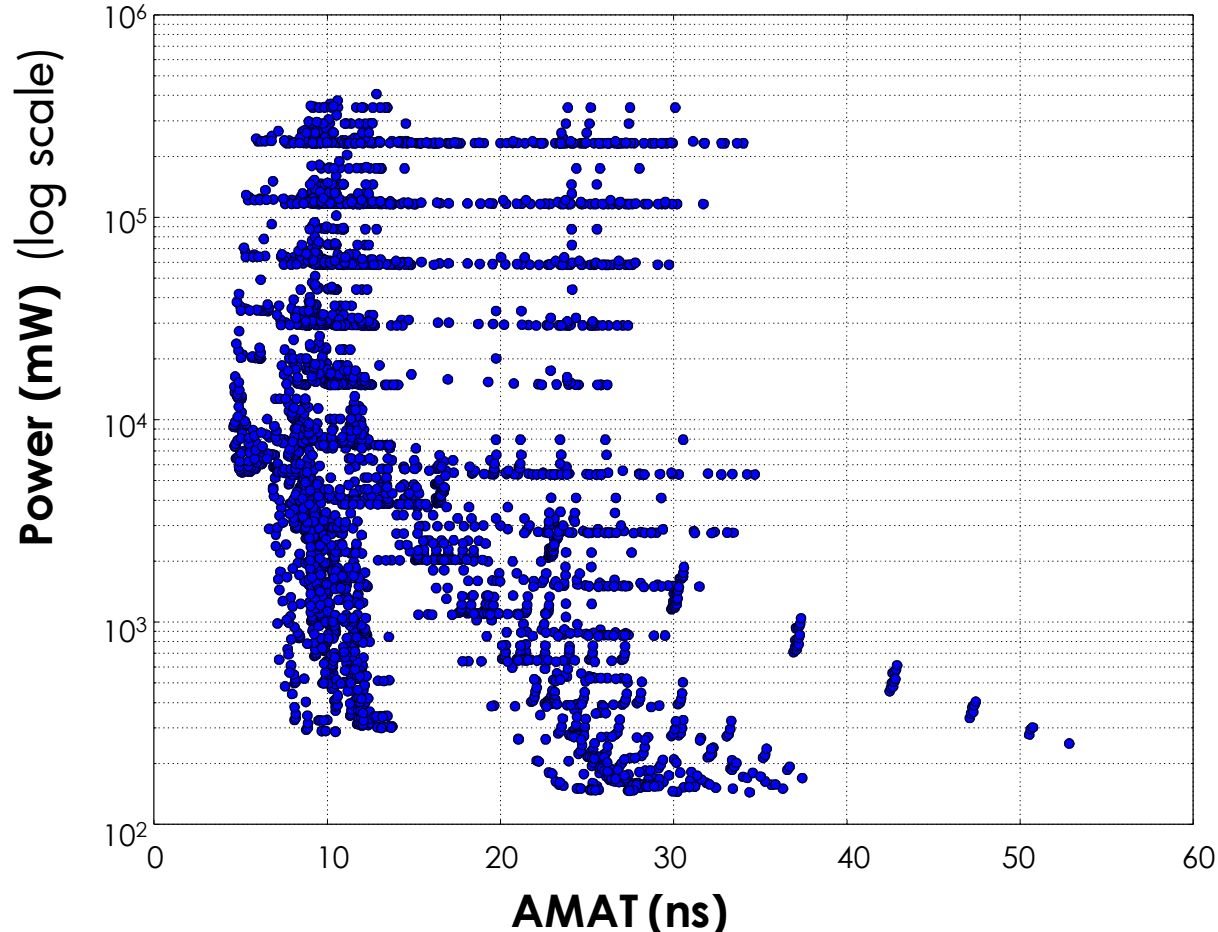


Time to execute  
1000 instrs

# The Model, Again

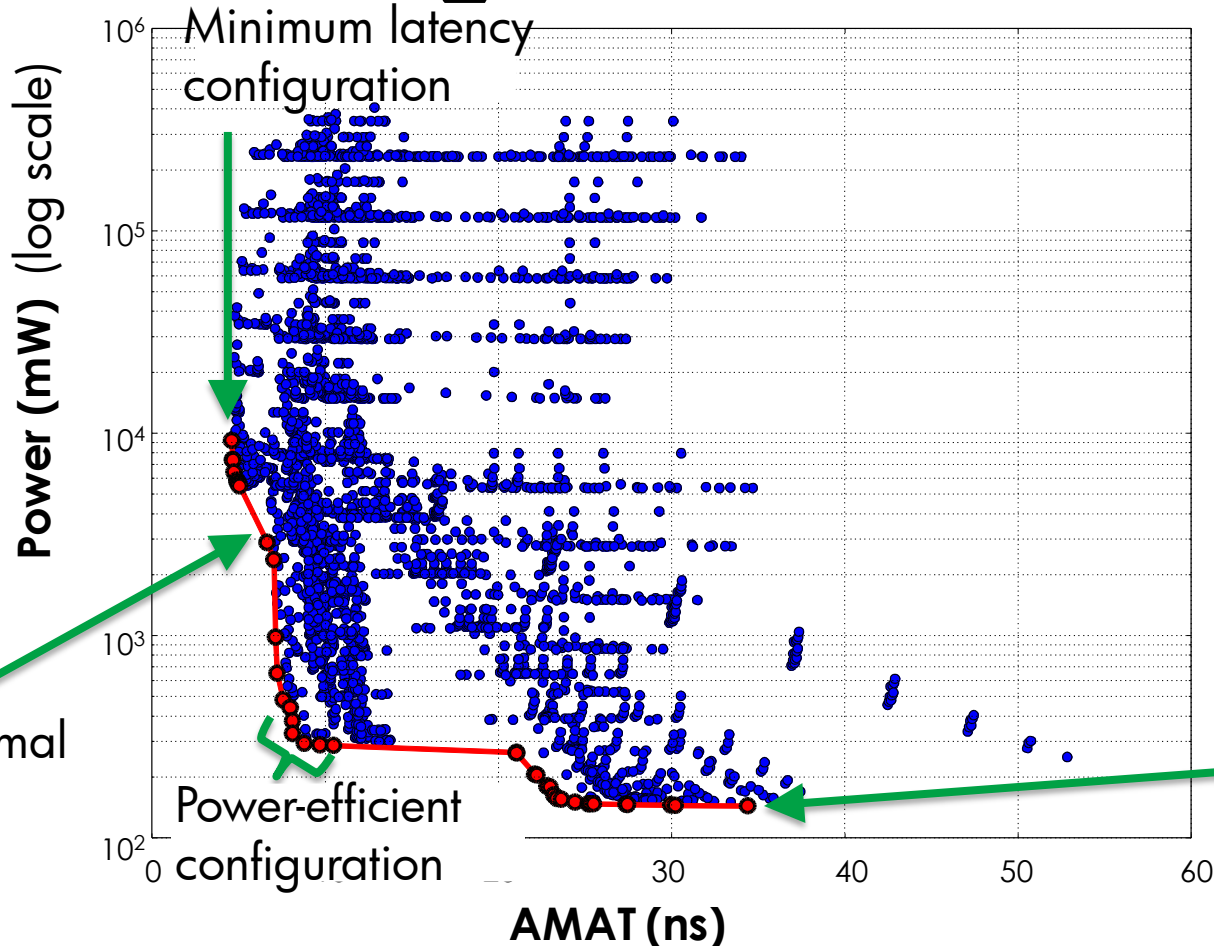


# Exhaustive Search Example





# Optimum Configurations



Minimum power configuration

Pareto-Optimal Frontier

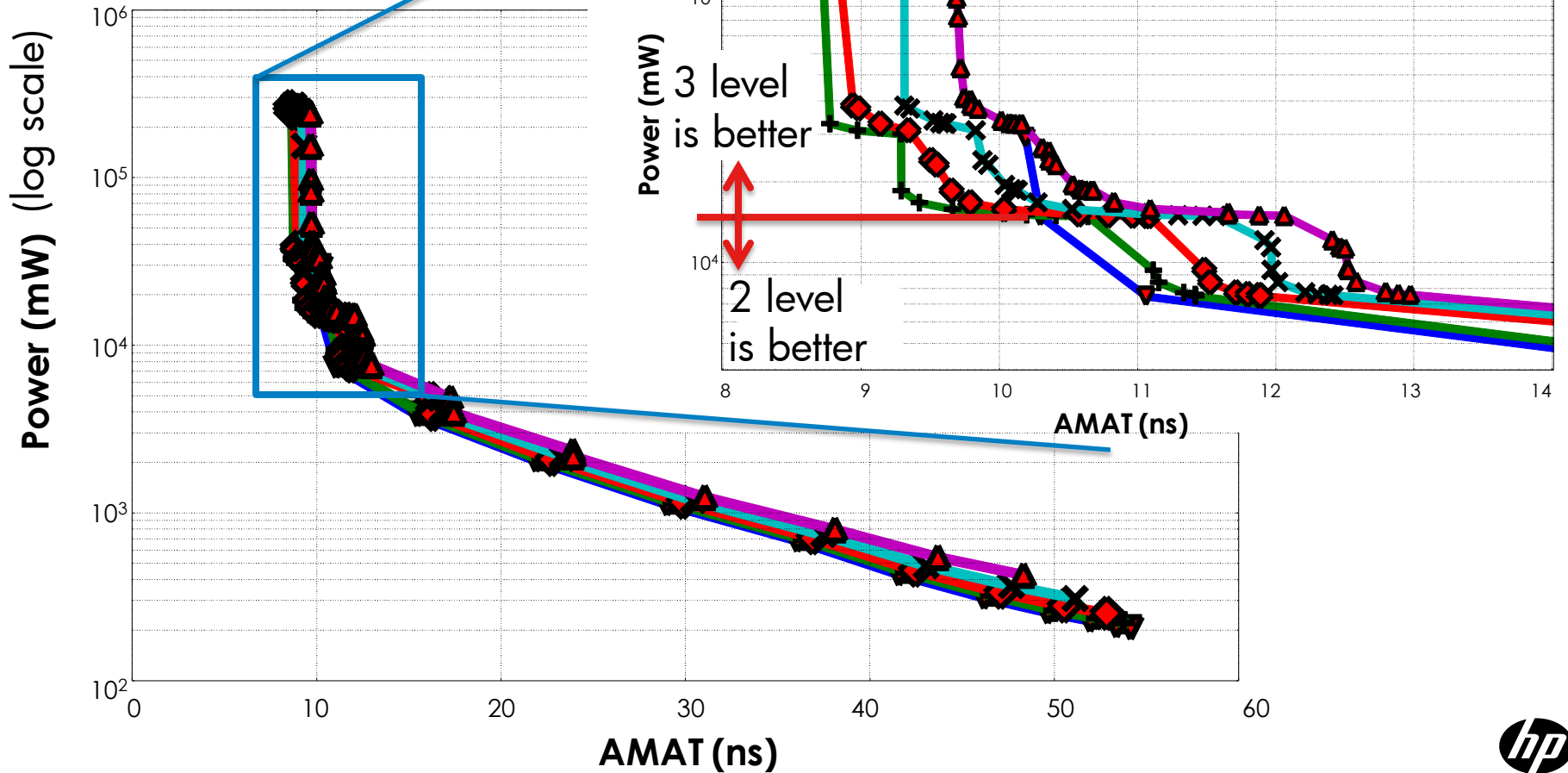
## 2. Depth of a Cache Hierarchy

Use the latency-power model

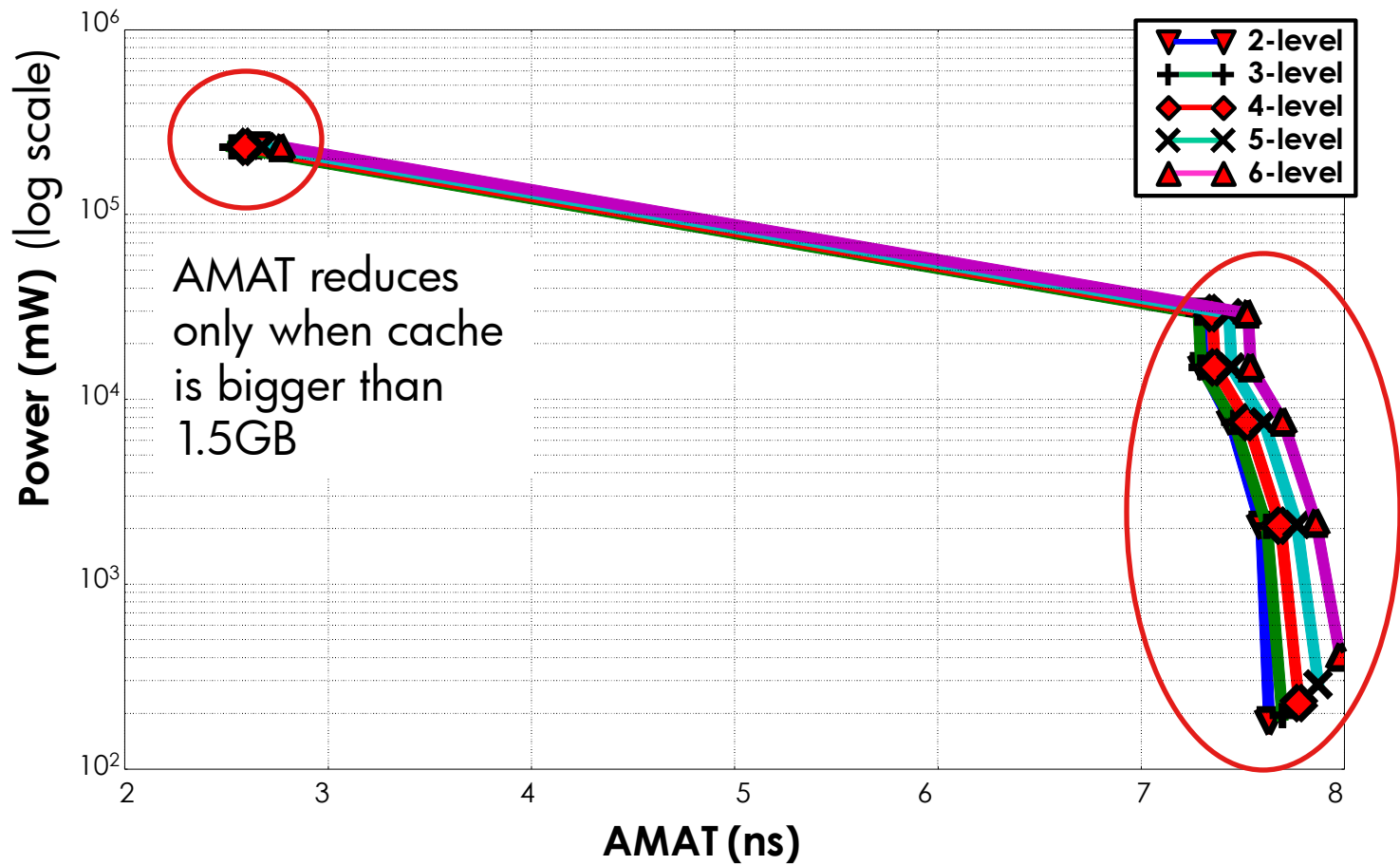
SRAM caches, ranging from 32kB to 2GB

2 to 6-level hierarchies

# Graph 500



# MiniFE





# Lessons Learned

Cache hierarchies deeper than 3 levels

→ Increase AMAT

→ Use more power

Large SRAM caches

→ Increase static power a lot

→ Only a small improvement in AMAT

# 3. Cache Hierarchies with NVM

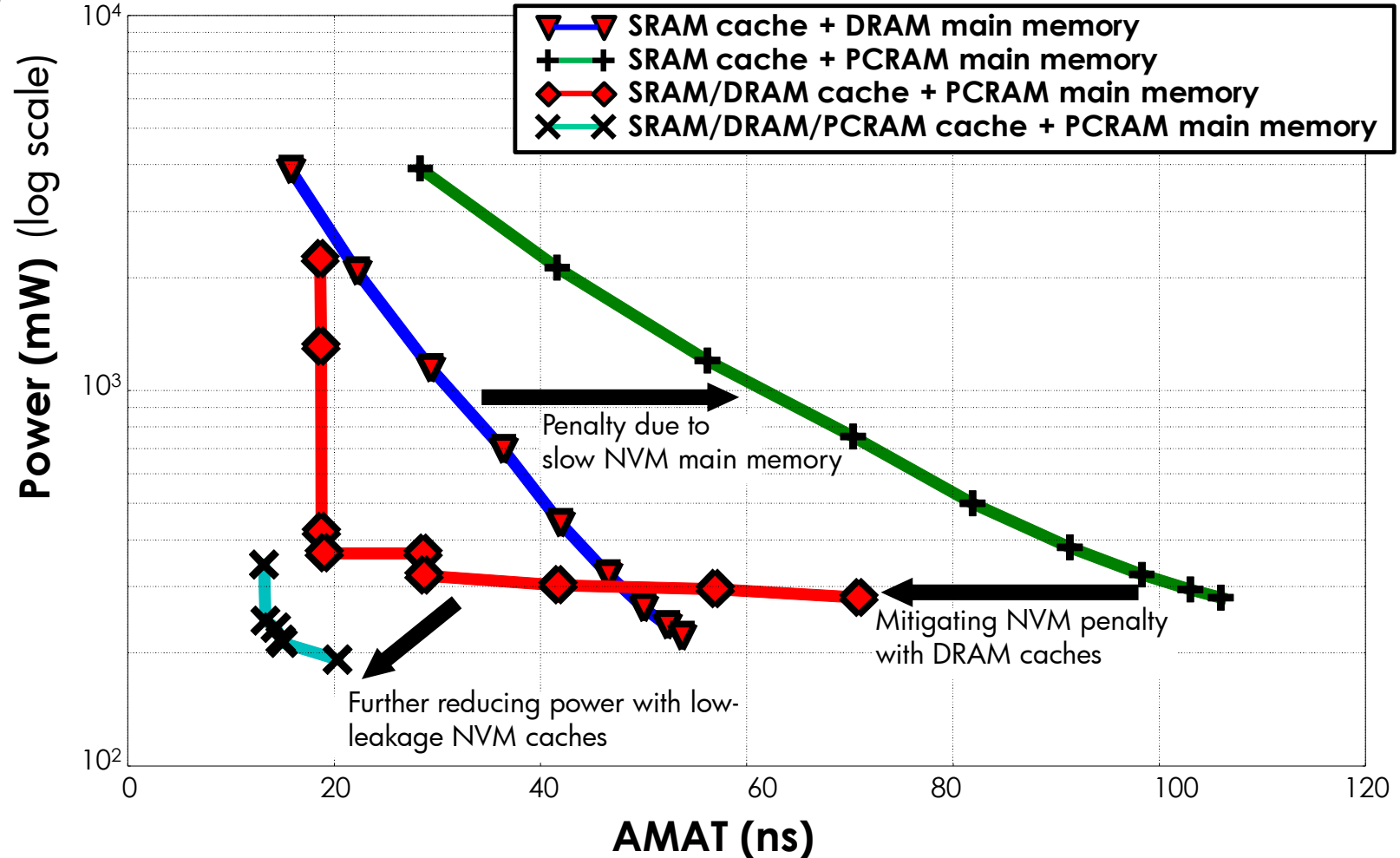
Use the latency-power model

Replace DRAM main memory with NVM main memory

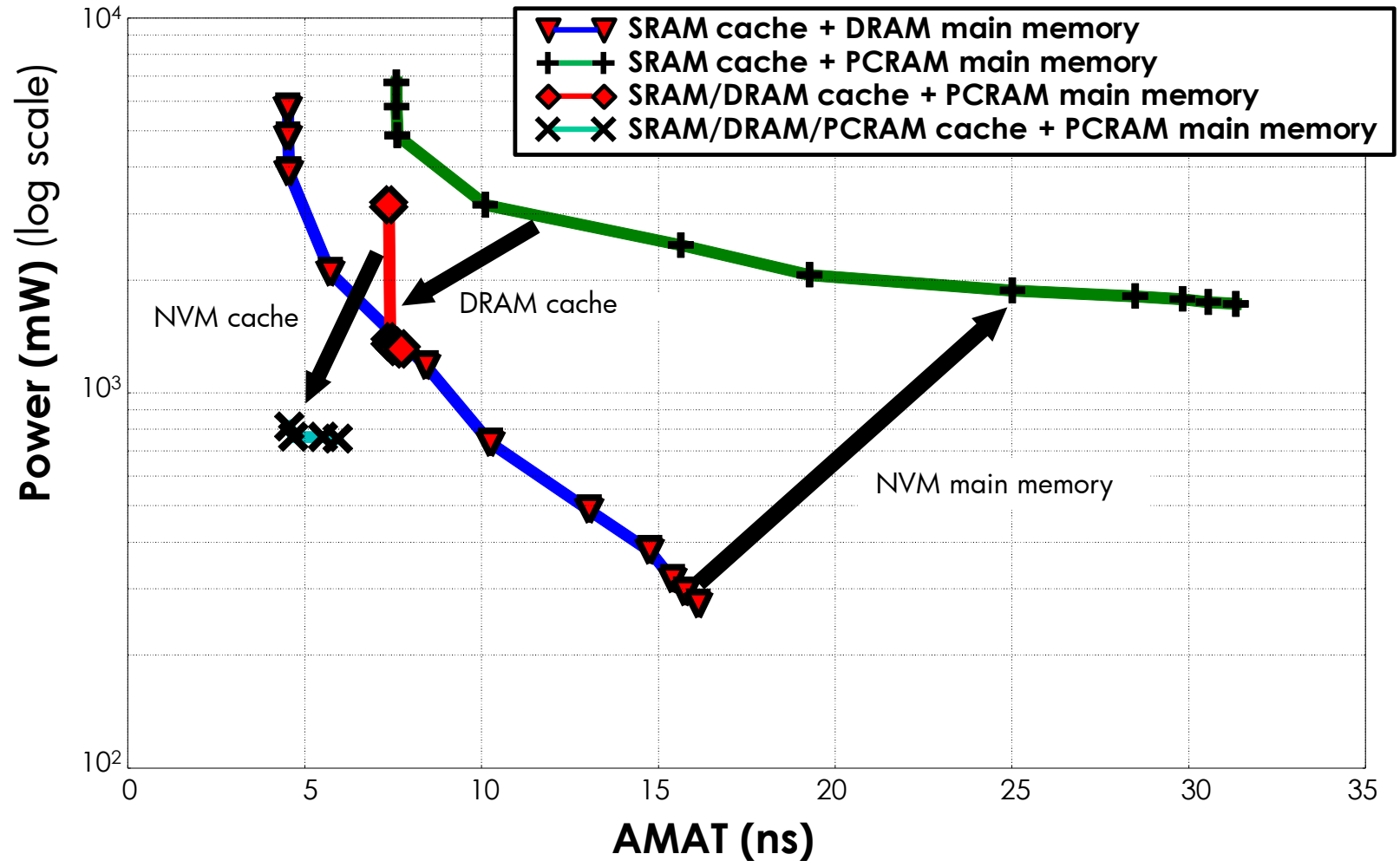
Caches with heterogeneous technologies

- SRAM caches, ranging 32kB to 32MB
- DRAM caches, ranging 4MB to 64MB
- PCRAM caches, ranging 16MB to 1GB

# Graph 500



# MCF



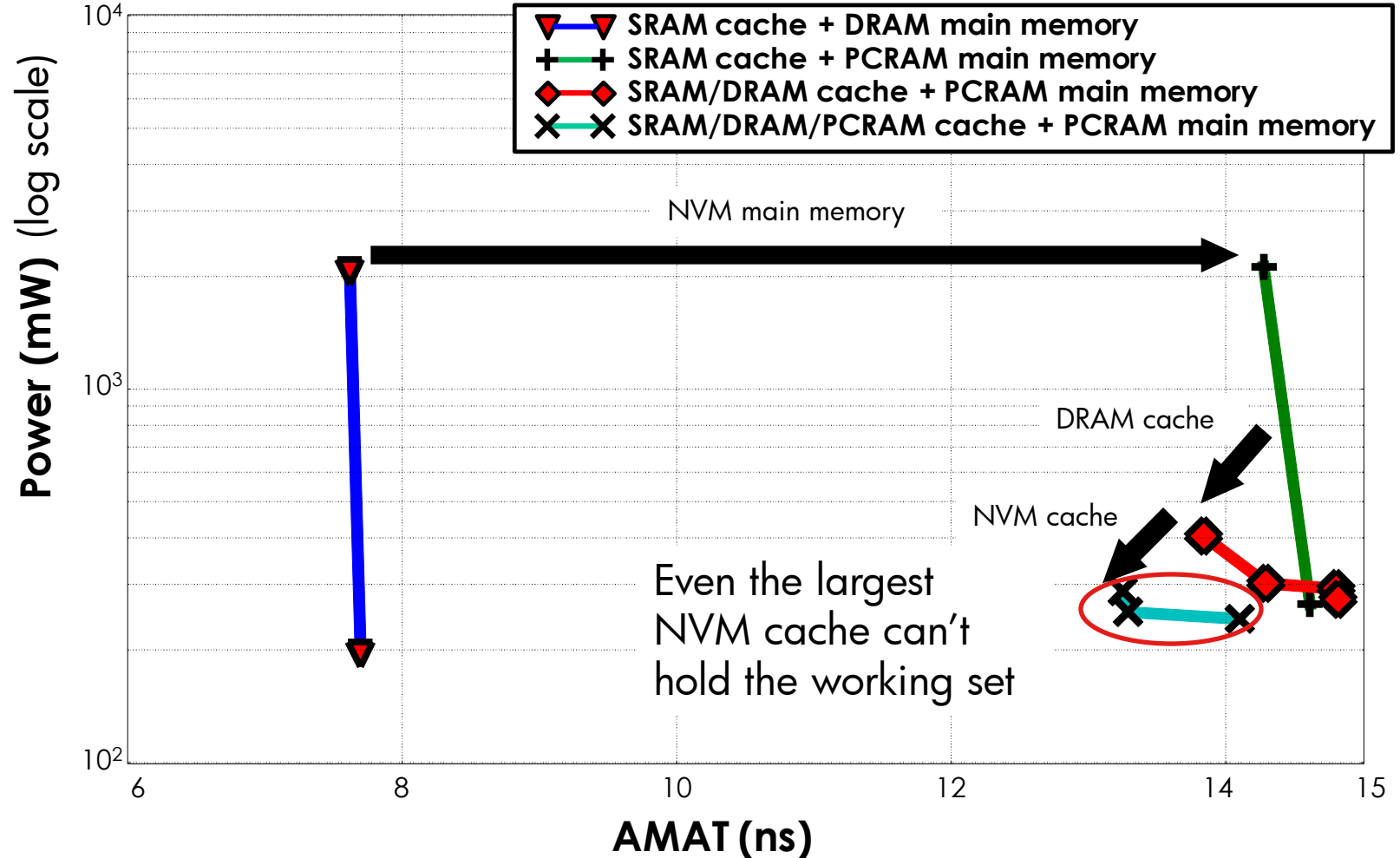
## 4. Streaming Patterns

Working set is simply larger than the cache

LRU policy thrashes data

Large caches can't improve AMAT but waste power unless the cache is larger than the working set

# MiniFE



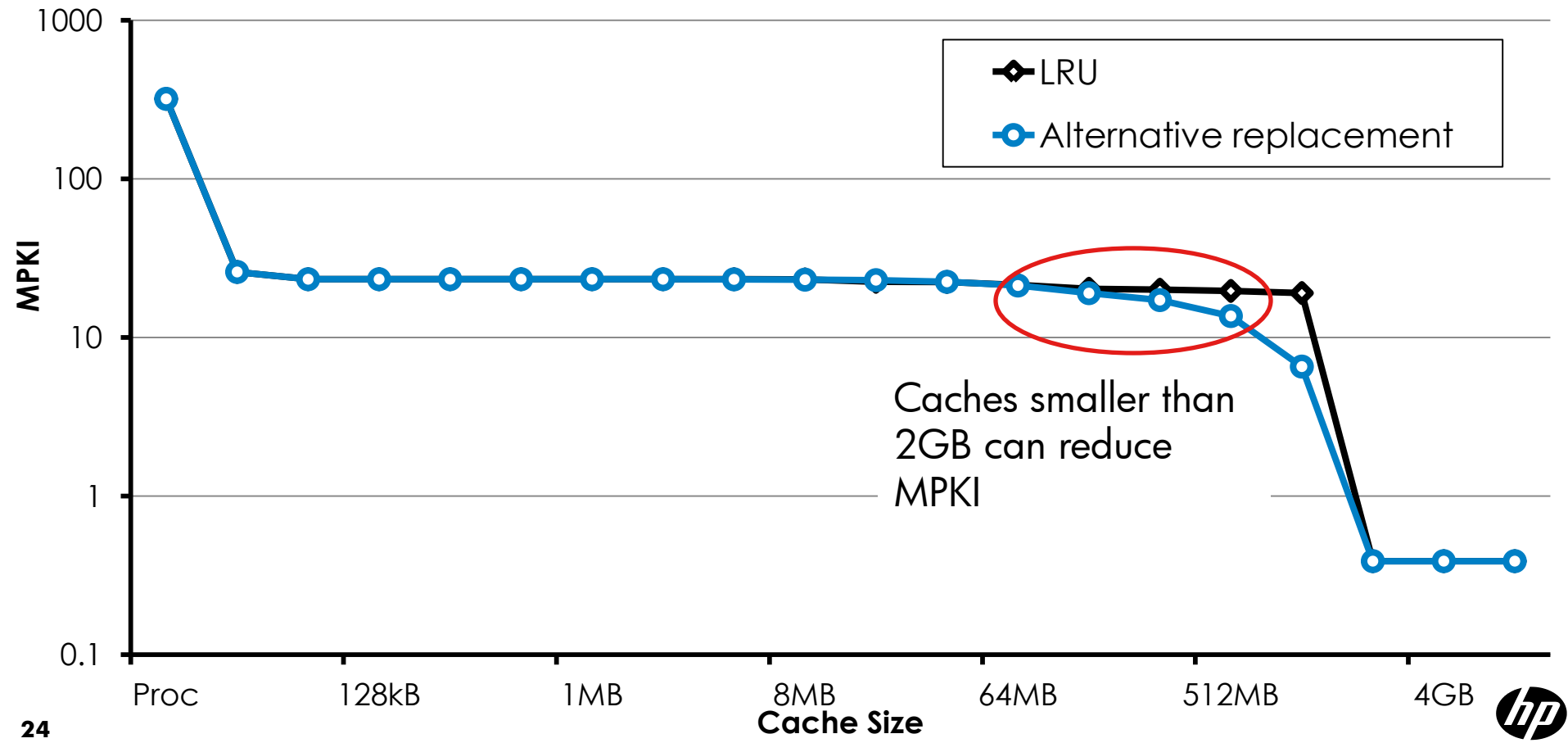
# Alternative Replacement Policy

Inserted lines are never replaced

Mimicking more recent techniques such as  
DIP [Qureshi+ ISCA'07] or RRIP [Jaleel+ ISCA'10]

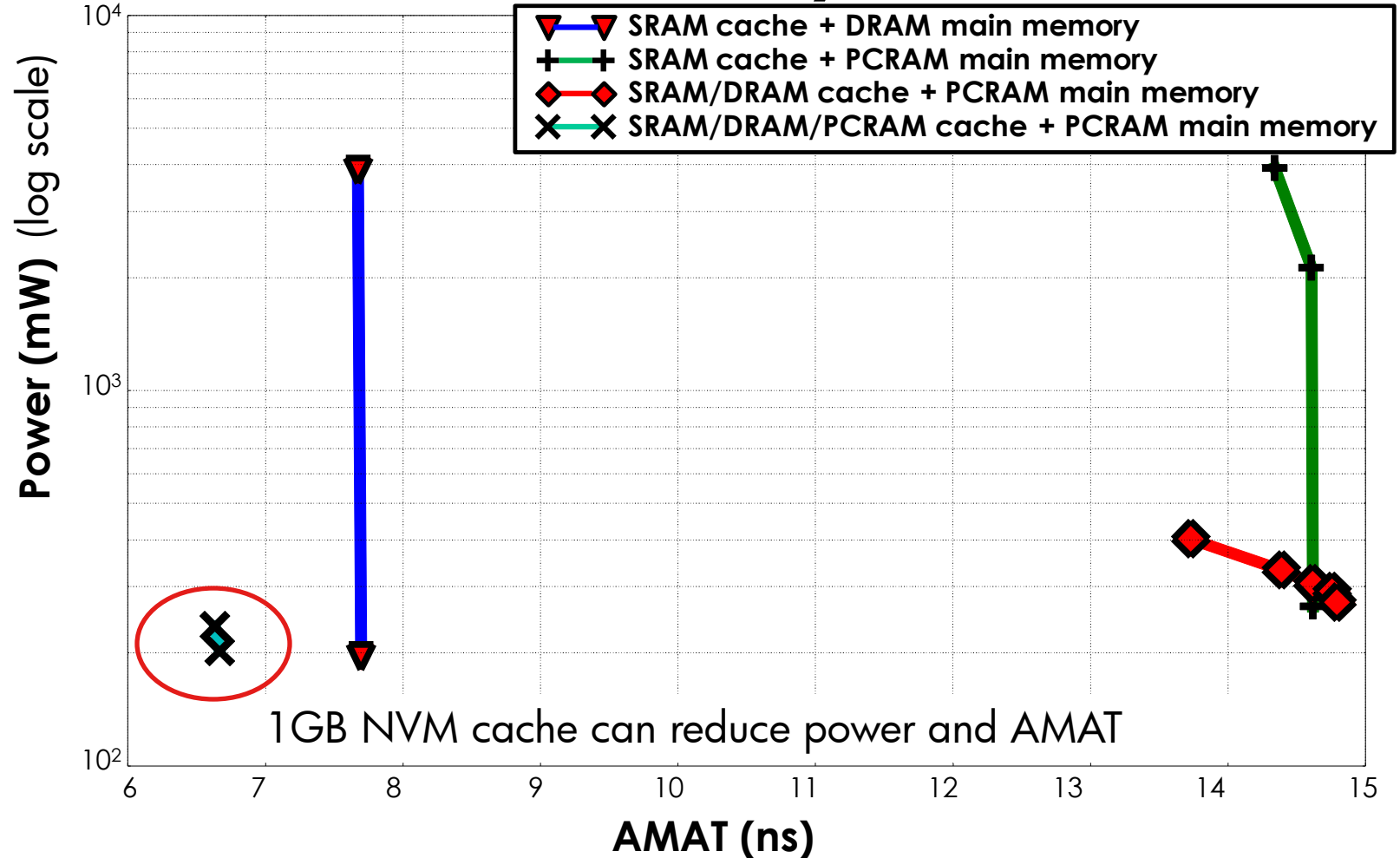
Can preserve a fraction of working set in the cache

# MiniFE: MPKI vs. Cache Size





# MiniFE with the Alternative Policy



# Conclusions

Developed a latency-power model

Deep hierarchies are less power efficient  
than flat hierarchies

Large NVM caches can be power efficient

Advanced insertion/replacement policy should be  
combined to handle streaming patterns.

# Limitations

Assumed a blocking in-order core

Ignored memory level parallelism, prefetching, ...

Write endurance in NVM caches

Aggressive power control in SRAM caches

# Acknowledgement & Disclaimer

This material is based upon work supported by the Department of Energy under Award Number DE-SC0005026.

See <http://www.hpl.hp.com/DoE-Disclaimer.html> for additional information

# Exploring Latency-Power Tradeoffs in Deep Nonvolatile Memory Hierarchies



Doe Hyun Yoon, Tobin Gonzalez,  
Parthasarathy Ranganathan, and Robert S. Schreiber

Intelligent Infrastructure Lab (IIL), Hewlett-Packard Labs