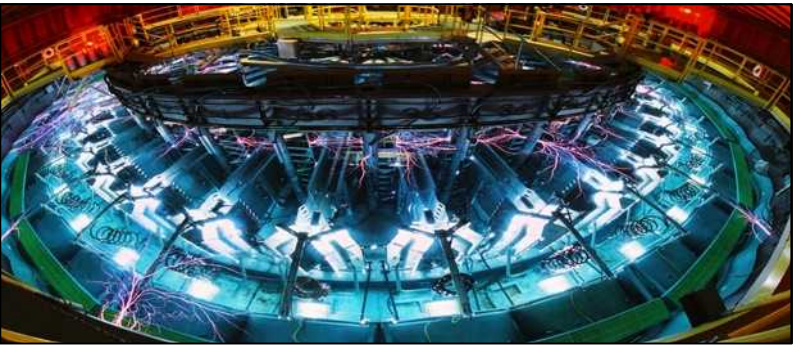
SAND2016-3984C

Exceptional service in the national interest

Sandia National Laboratories



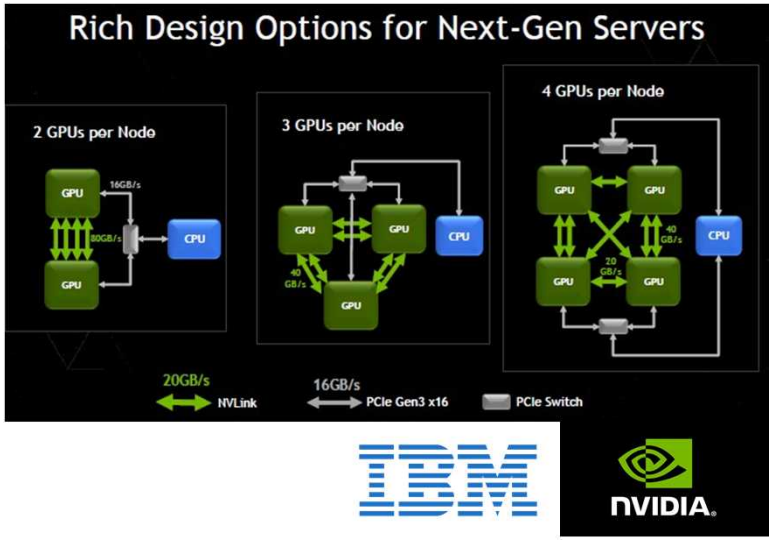
Multi-Level Memory – The Next Opportunity for Performance?

S.D. Hammond (and lots of help from people at SNL/NM)
Center for Computing Research,
Scalable Computer Architectures
Sandia National Laboratories, NM
sdhammo@sandia.gov

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Motivations

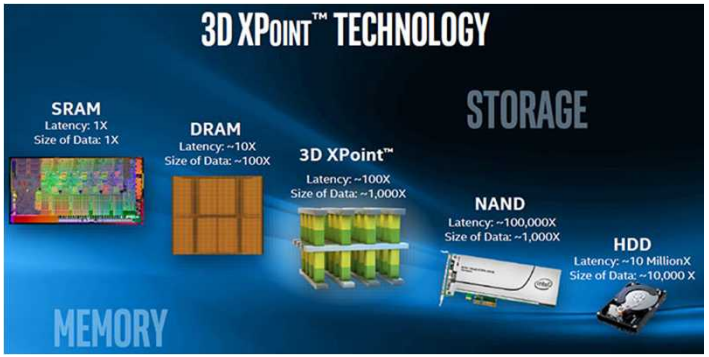


- **Knights Landing (Trinity)***
  - "Two Memories"
  - On-Package developed with Micron
  - Over 400GB/s (HBM)
  - >90GB/s (DDR)
  - Complex NUMA modes
  - Complex cache options
- **CORAL (POWER + Volta GPU)**
  - Lots under NDA
  - NVLINK-Based GPU connections
  - Fast on-package memory
  - Slower (but still fast) memory on socket
  - Complex NUMA

* = See: Intel Disclosure (https://software.intel.com/en-us/articles/what-disclosures-has-intel-made-about-knights-landing)

Into the Future ..

- Situation may get much more complex…
- Potential (transformative) use of Non-Volatile Memory:
  - Storing data
  - Object Key/Value Store
  - Fine-grained checkpoints
  - Experimental data
  - …
- Data Analytics pushing towards network/fabric attached storage/memory

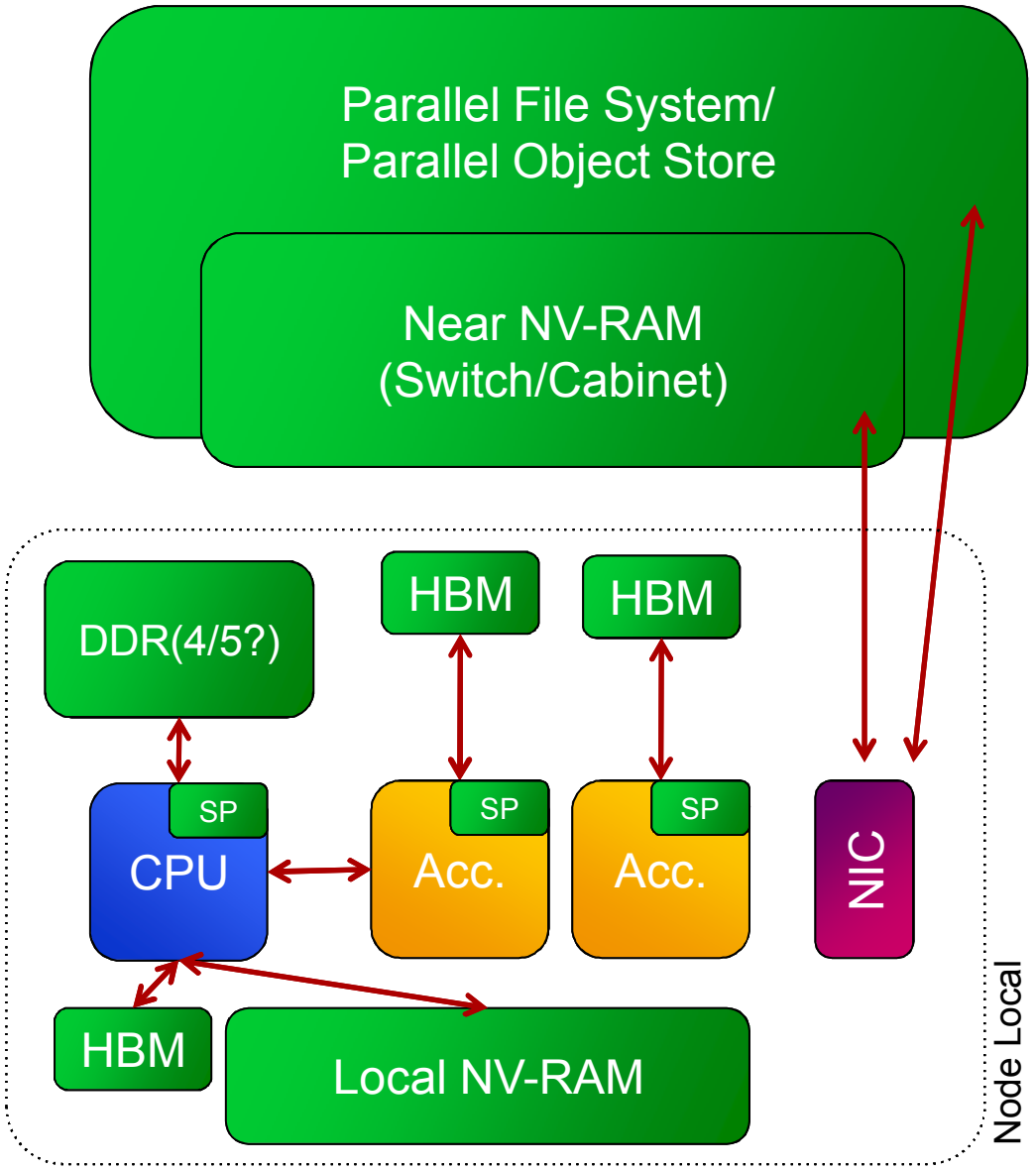


So what does this look like…

- Lots of “Levels” in the memory hierarchy

- But ..
  - These aren’t really “levels”
  - They are memory pools
  - **Not always a clearly strict ordering of performance**

- Huge opportunities when using the “right” memory for the task at hand
- Caches – temporary fix?



Longer lines = Higher Latency (not drawn to scale)

What it Really Looks Like …

Sandia National Laboratories

- **Is a total headache for application developers**

- Where does any individual data structure go?
  - May vary by physics package
  - May vary by input deck
  - May vary by time step
  - Probably will vary by libraries used
  - Probably will vary by machine/hardware being used (algorithm/performance)

- Convergence between storage and memory seen as a good thing
  - Most won't miss POSIX

An additional $0.02

Sandia National Laboratories

- Adding threading in many ways hasn't been the hardest challenge
  - But we have told ourselves that this is hard
  - Probably because the programming models haven't helped us

- But the *really* hard challenge isn't the threading, it's the migration of data structures to something which is more flexible

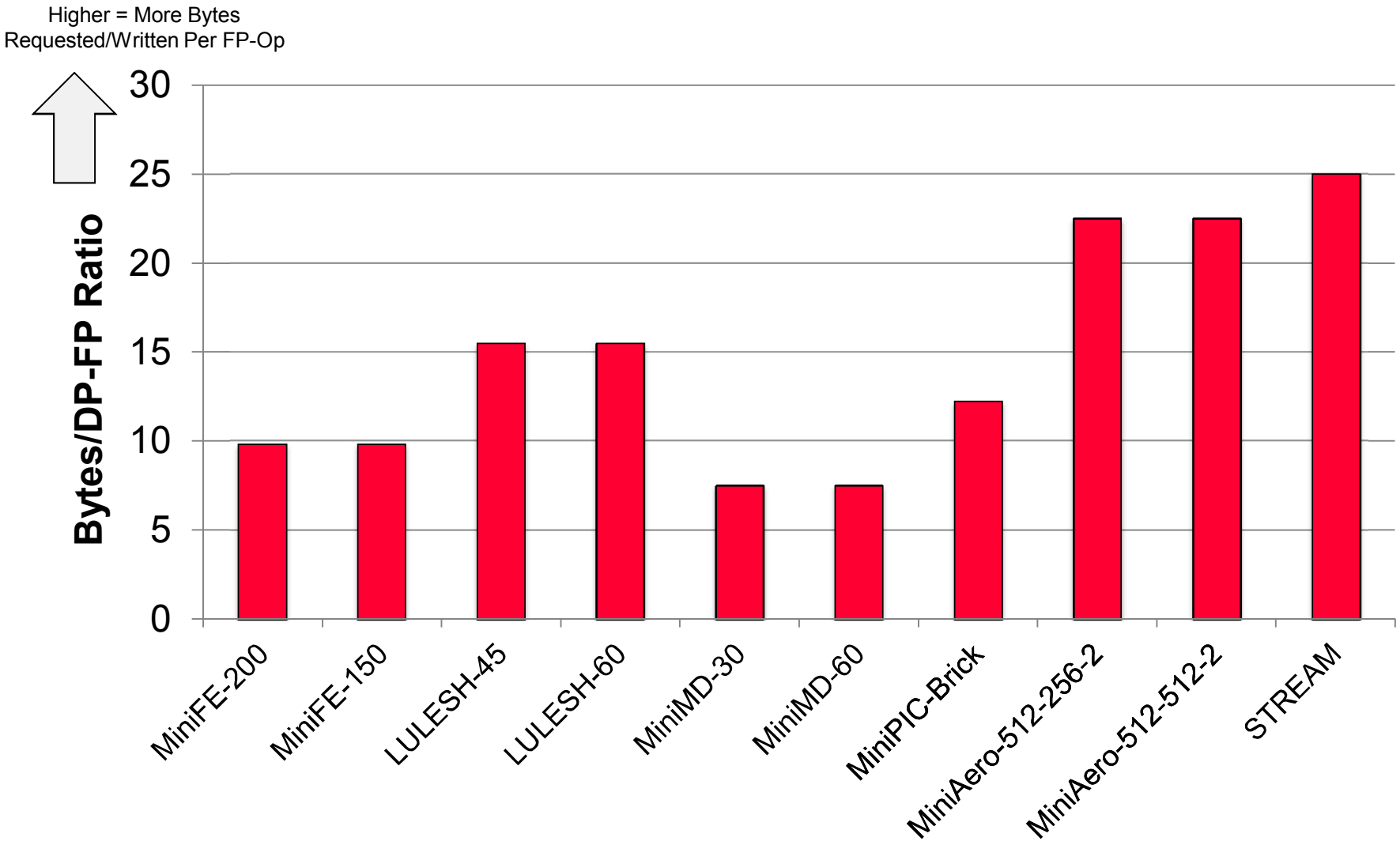
- I really think that performance is coming down to how best we can optimize our data structure design more than our compute kernels
  - When we get this wrong it *really* hurts (like 35X hurts)

- Going to be a much longer, harder and more intrusive change to our applications than modify-as-you-go parallelism

Sandia National Laboratories

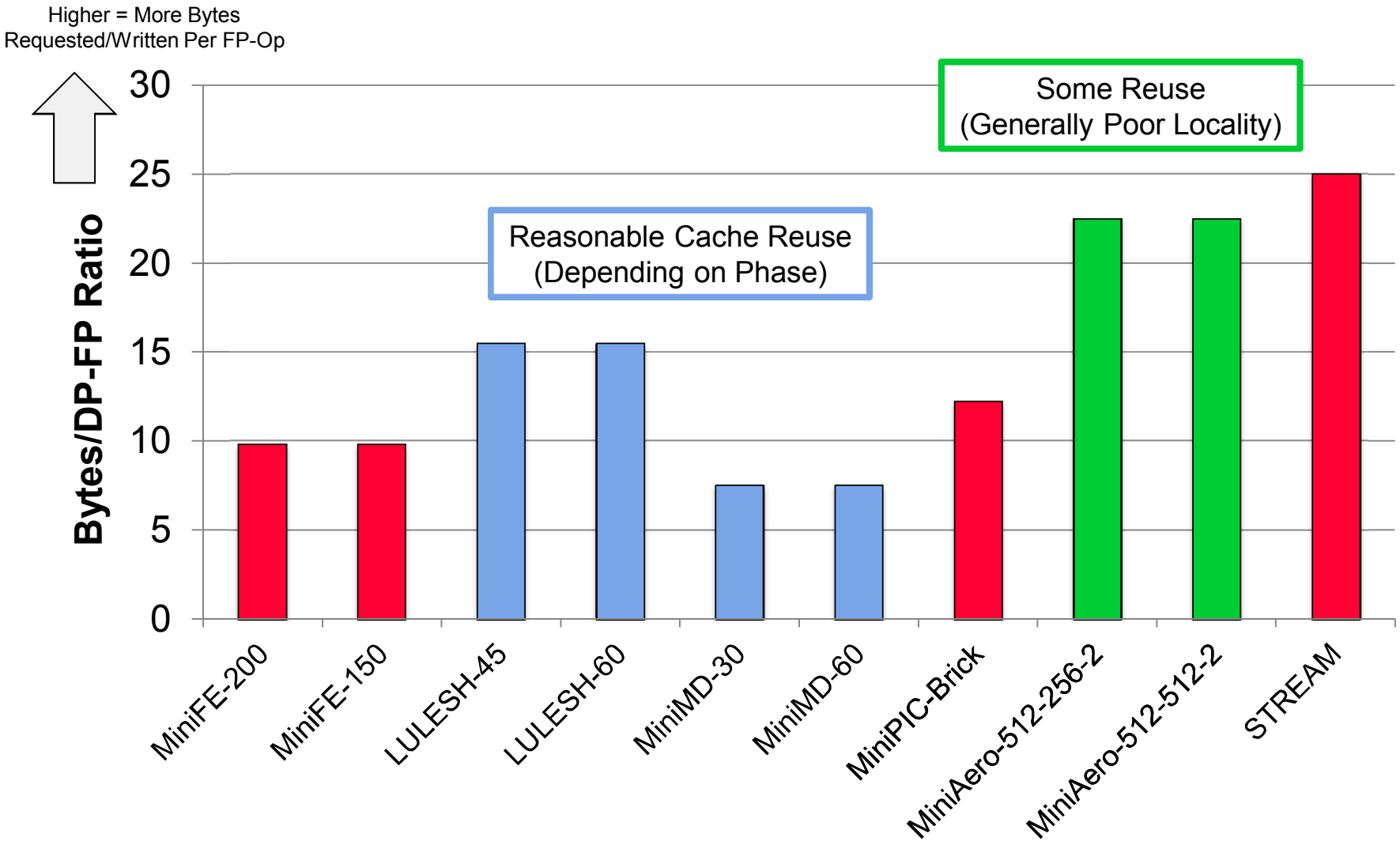
So Why is This a Problem?

Bytes and FLOP/s



APEX Application Characterization Tools (SNL Research, Compiled for Haswell, Intel Compiler)

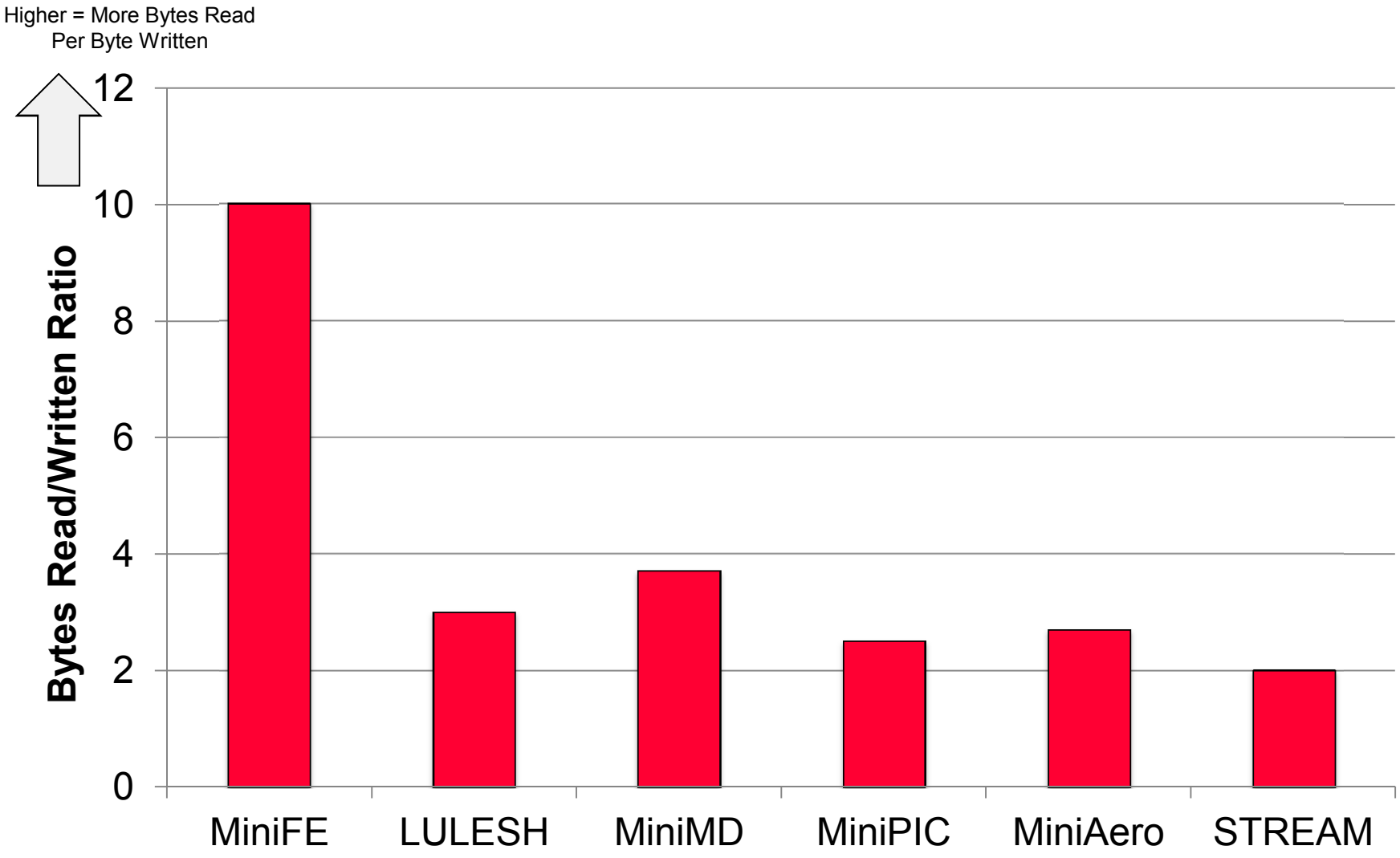
Bytes and FLOP/s



APEX Application Characterization Tools (SNL Research, Compiled for Haswell, Intel Compiler)

Read/Write Ratio

Sandia National Laboratories



Simple Survey

Sandia National Laboratories

- When we get machines capable of huge FLOP/s rates, we also need strong memory systems
  - Surprisingly, still the weak point of architectures nearly 20 years after McKee's famous "Memory Wall" paper
- Significant variation in how applications use the memory system
- All developers (application and libraries) will want to put *everything* into high bandwidth memory
  - Who is going to be the one to take on the pain?
  - Huge downsides if you are the slowest point in the critical path (trust me, I've worked with code teams who are in that)

- **Needs us to recognize codes may go slower while we work this out**

Sandia National Laboratories

Analyzing Application Behavior

Using the Structural Simulation Toolkit and APEX to characterize behavior

These guys did all the hard work

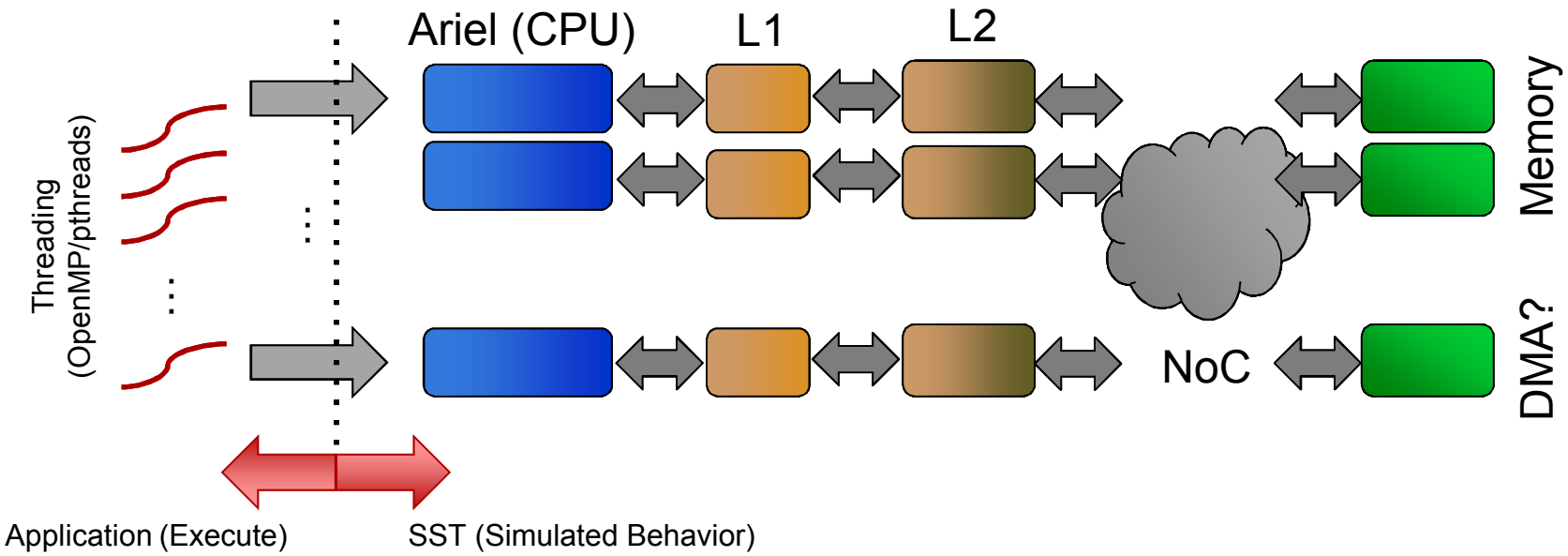
Work with Arun Rodrigues, Gwen Voskuilen and Mike Frank (SNL/NM)

Initial Studies into Memory Systems

Sandia National Laboratories

- One of the biggest problems we have today are the metrics associated with mixed-performance memory pools are not well understood
- This isn't just a more is better thing, lots of trade offs including the very limited budget the DOE has for machines and application modification
- Two studies:
  - **Application Allocation/Accesses** – what can we learn from looking at the way our kernels operate today?
  - **Page-Based Memory Management Policies** – can we do better by having simple state machines in memory where we decide what they do?

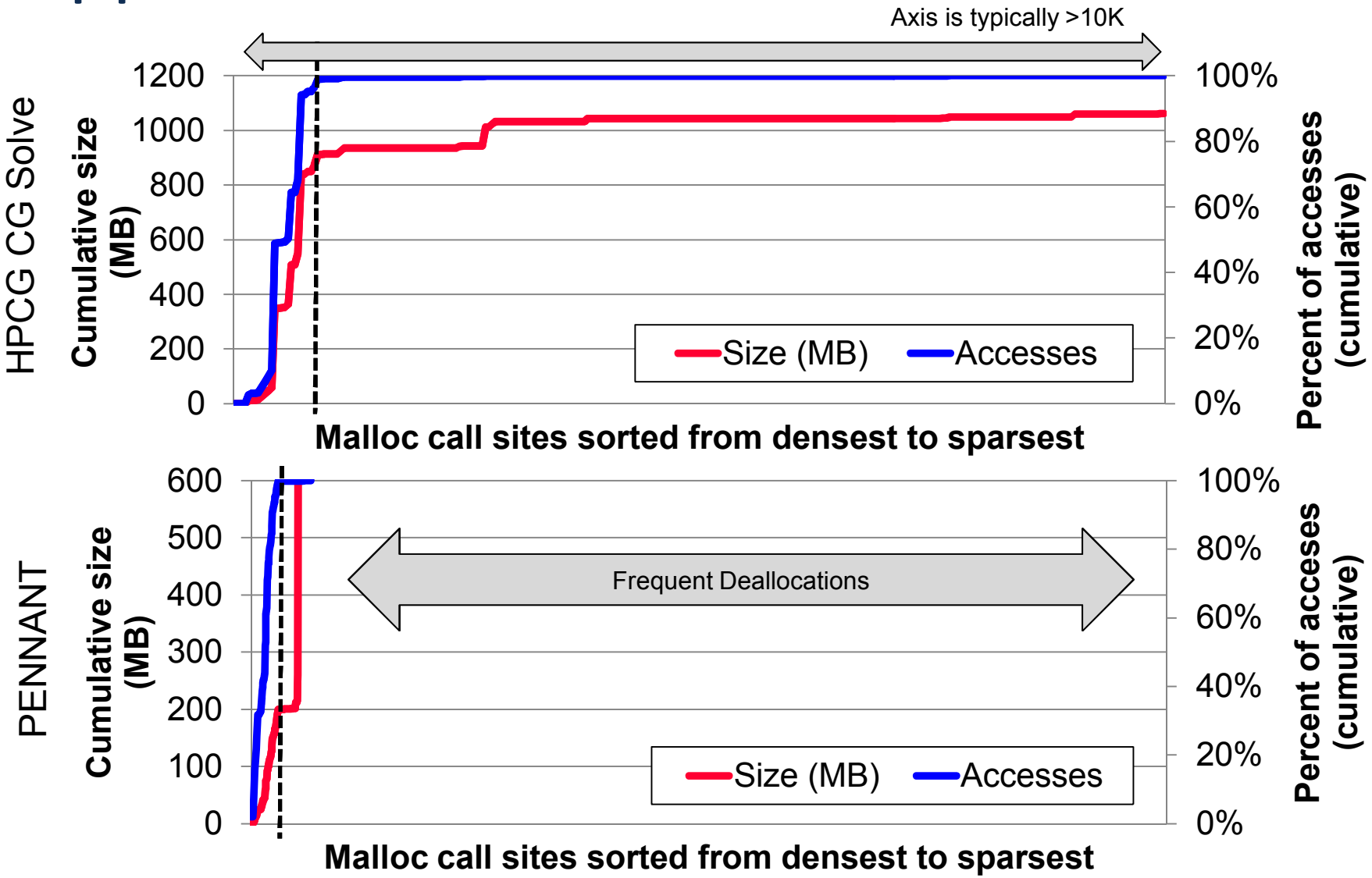
SST Models of the System



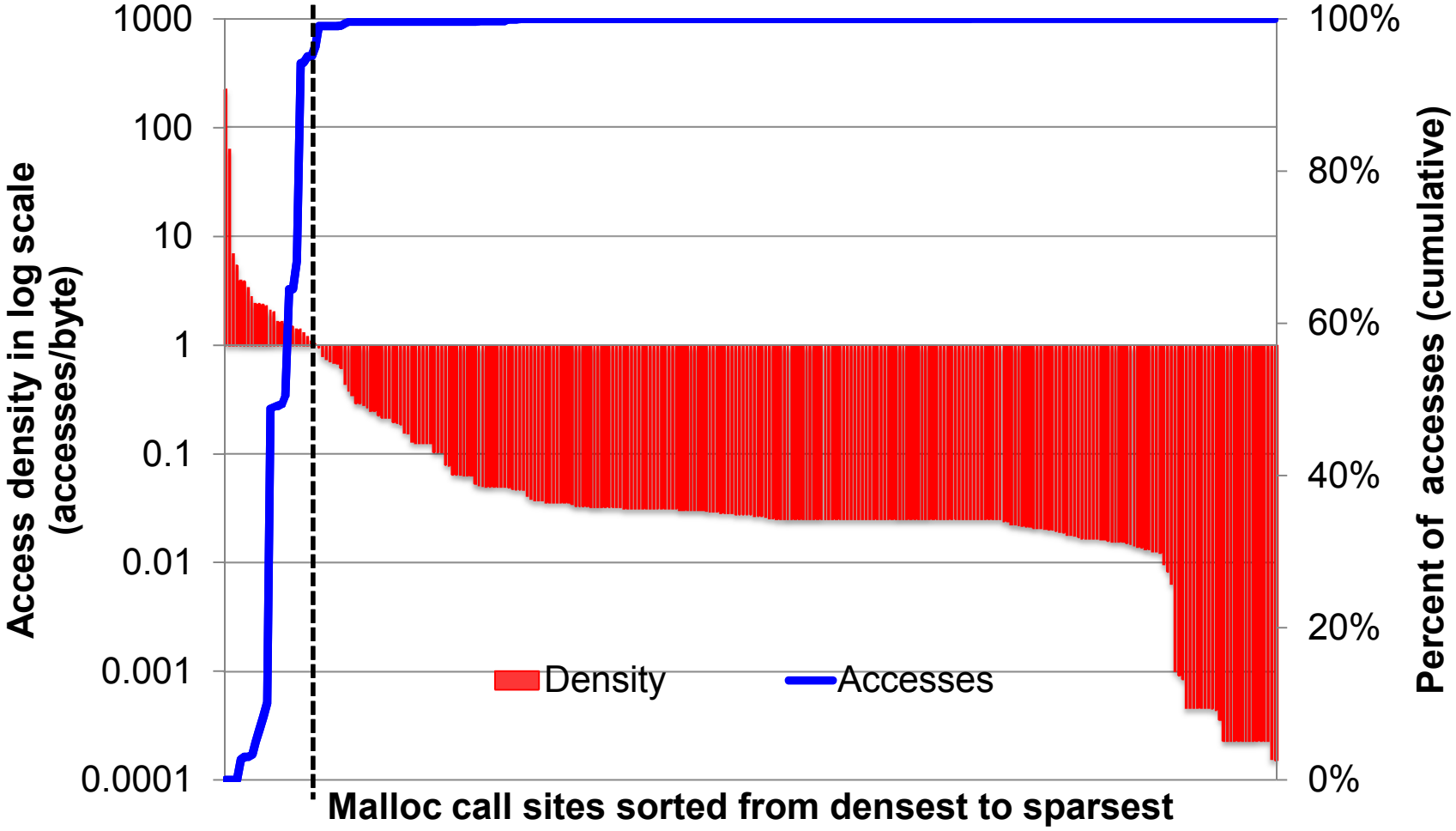
- Complex models – coherency, timing, back-pressure, prefetching *etc*
- Execution driven (but captured) application behavior for speed
- Configurable statistics and behavior analysis (during or post-simulation)
- Very extensible – HBM, HMC, DDR, NV-RAM, simple memory timing *etc*

http://www.github.com/sstsimulator

Application Behavior



Access Density (HPCG)



Application Behavior

Sandia National Laboratories

- Typically, small number O(5-10%) capture vast majority of accesses and the footprint

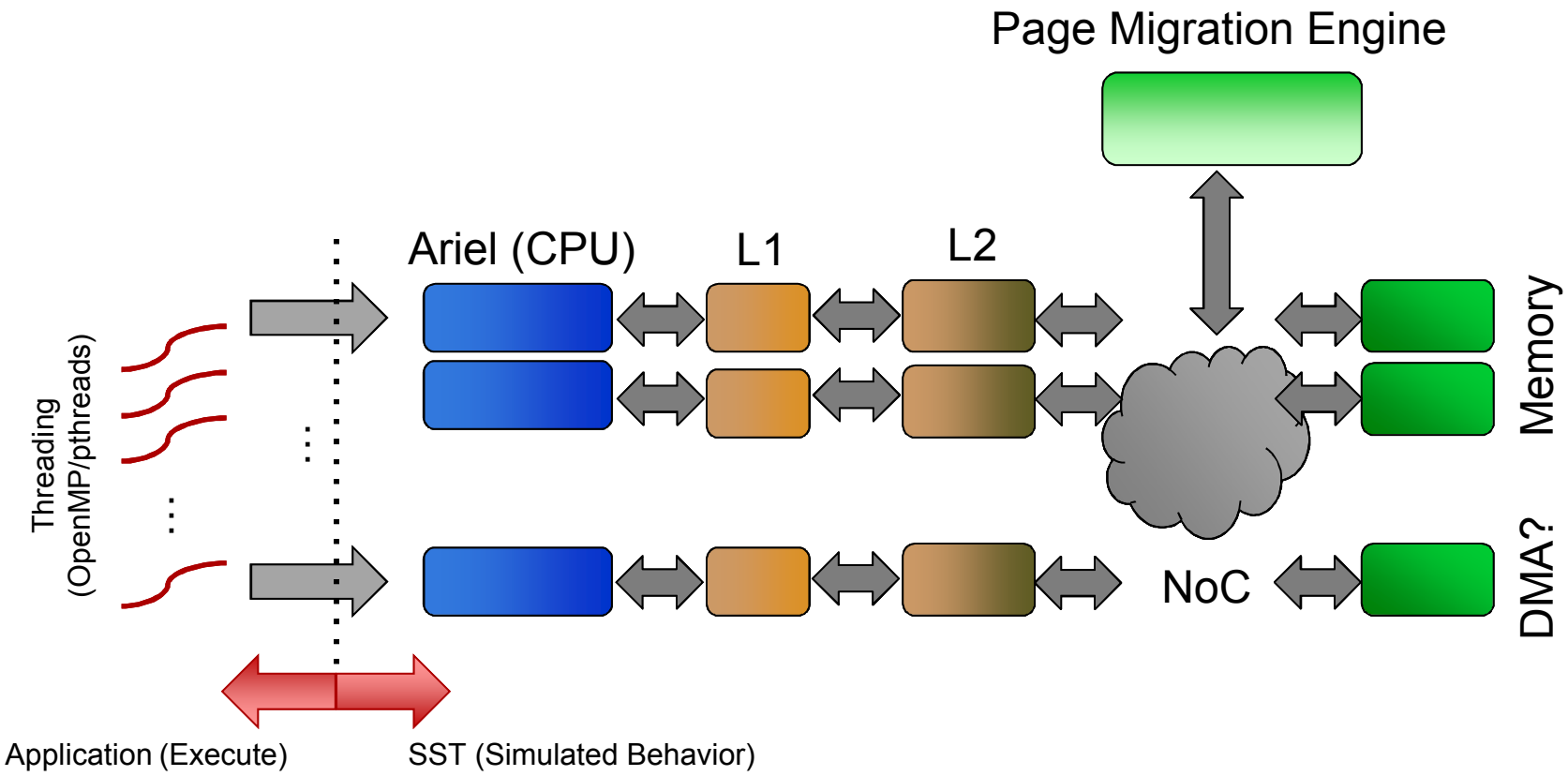
- Possibly implies we can focus our efforts on the most important and leave automated mechanisms or defaults for the rest

- Reminder – that O(5-10%) of allocations is still a huge amount of work

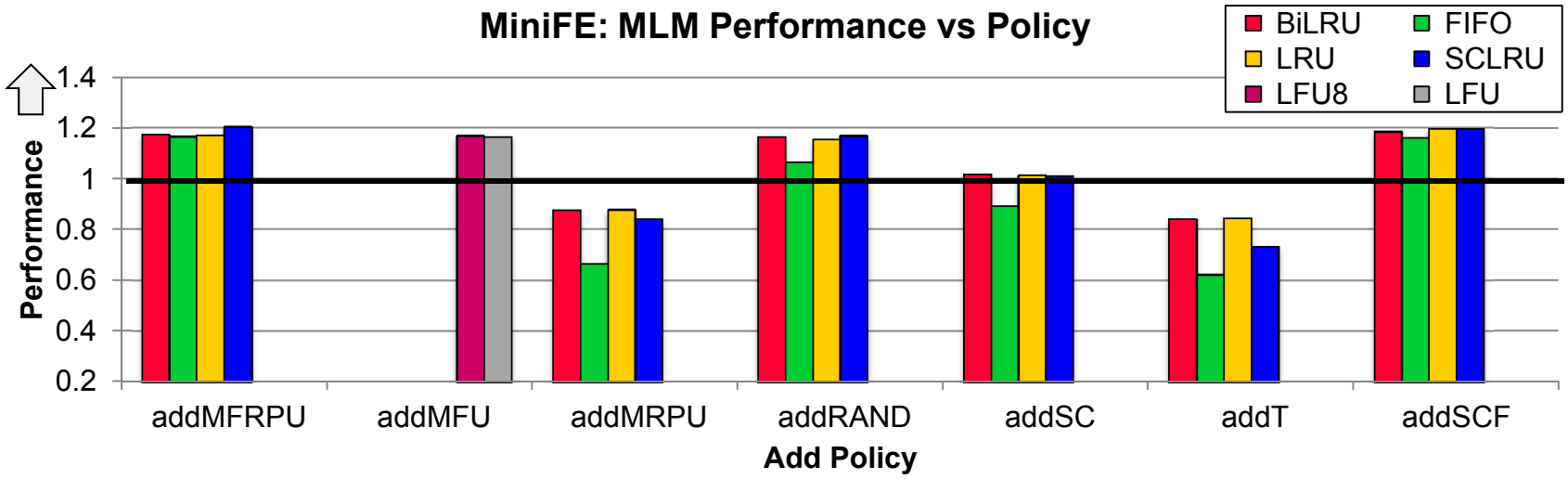
- Our focus is the density of use in the allocations (i.e. put higher density allocations into HBM first)

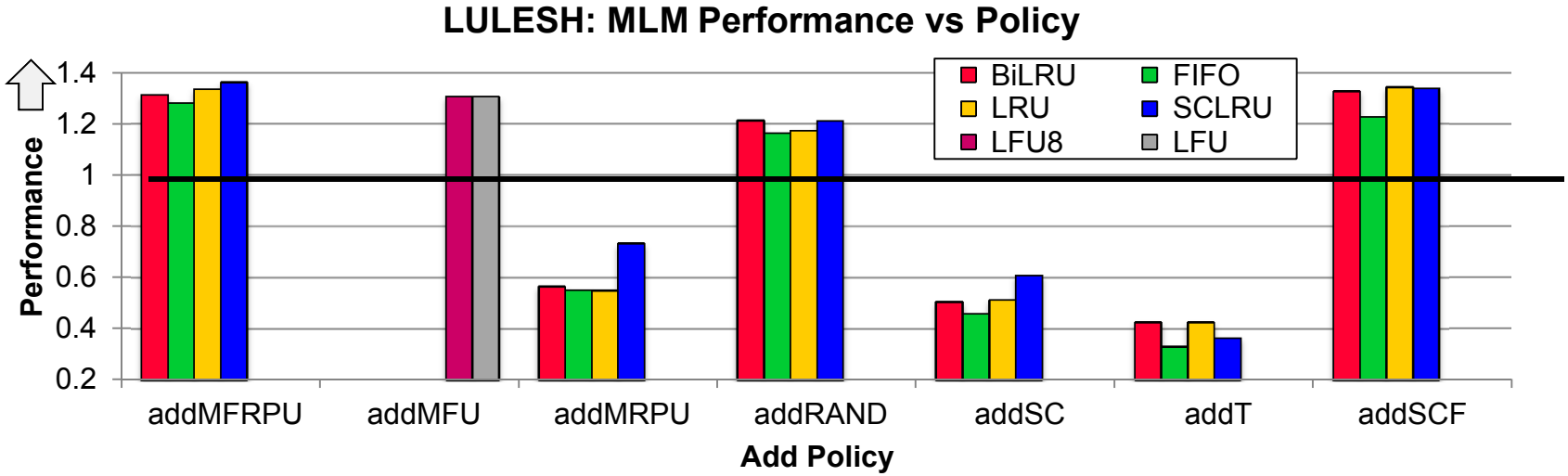
- But – we have some excellent tools to capture this behavior now within SST and our support performance analysis suite

(Smart?) Page Movement Engine



Page Replacement Schemes





Performance is Relative to Standard LRU

Page Replacement Schemes

Sandia National Laboratories

- Shows that a *one-size-fits-all* scheme may not be the best approach

- Do we want to be able to programmatically control page migration from fast-to-slow memories?
  - Probably something a runtime or O/S can handle?

- Today this requires a CPU core but we want specialized movement engines
  - By-pass caches (but still be coherent?)
  - Optimized load/store queues for migration?
  - Greater energy efficiency?

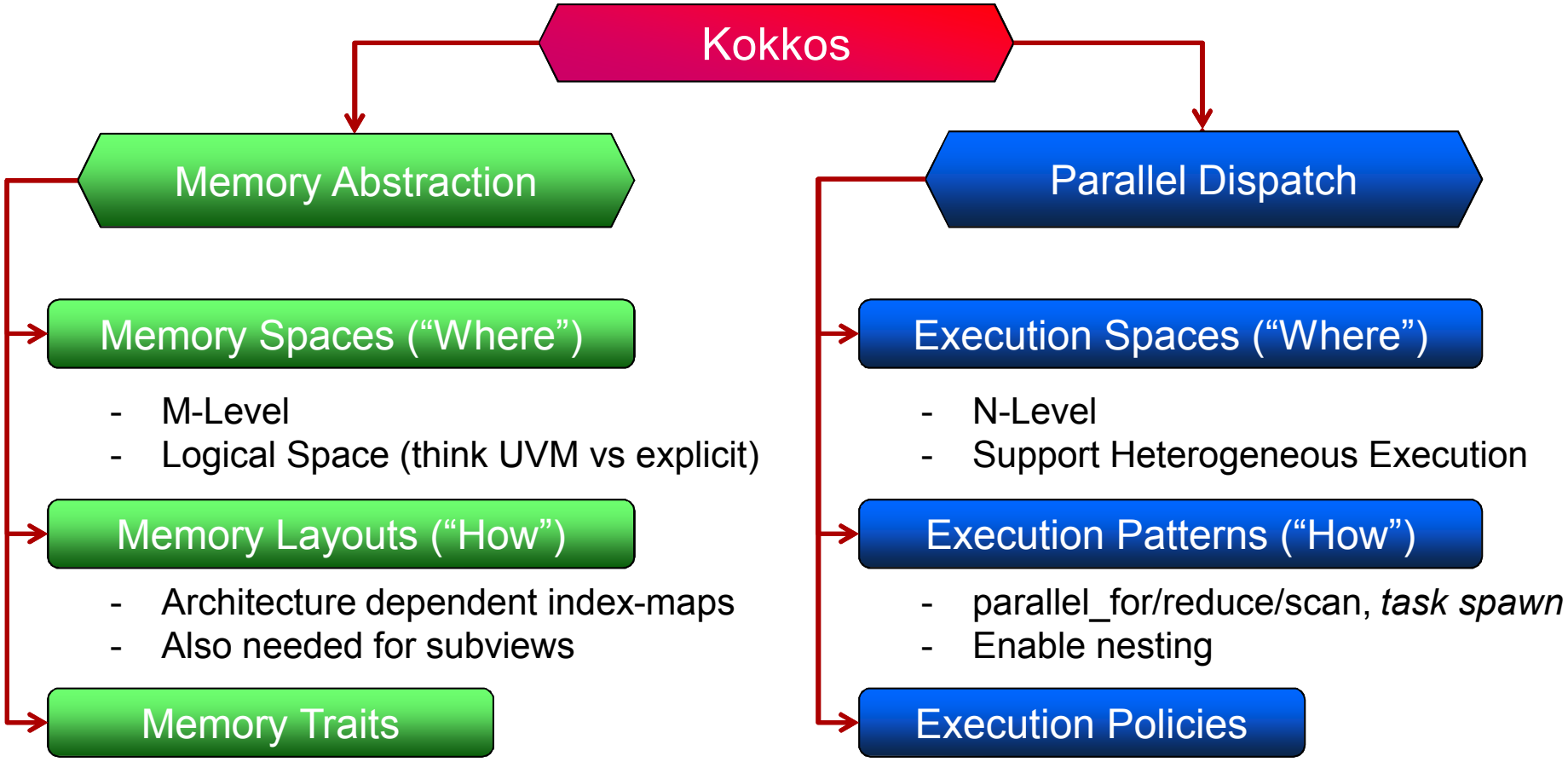
Sandia National Laboratories

Programming to the Memories

Performance Portability through Abstraction

Sandia National Laboratories

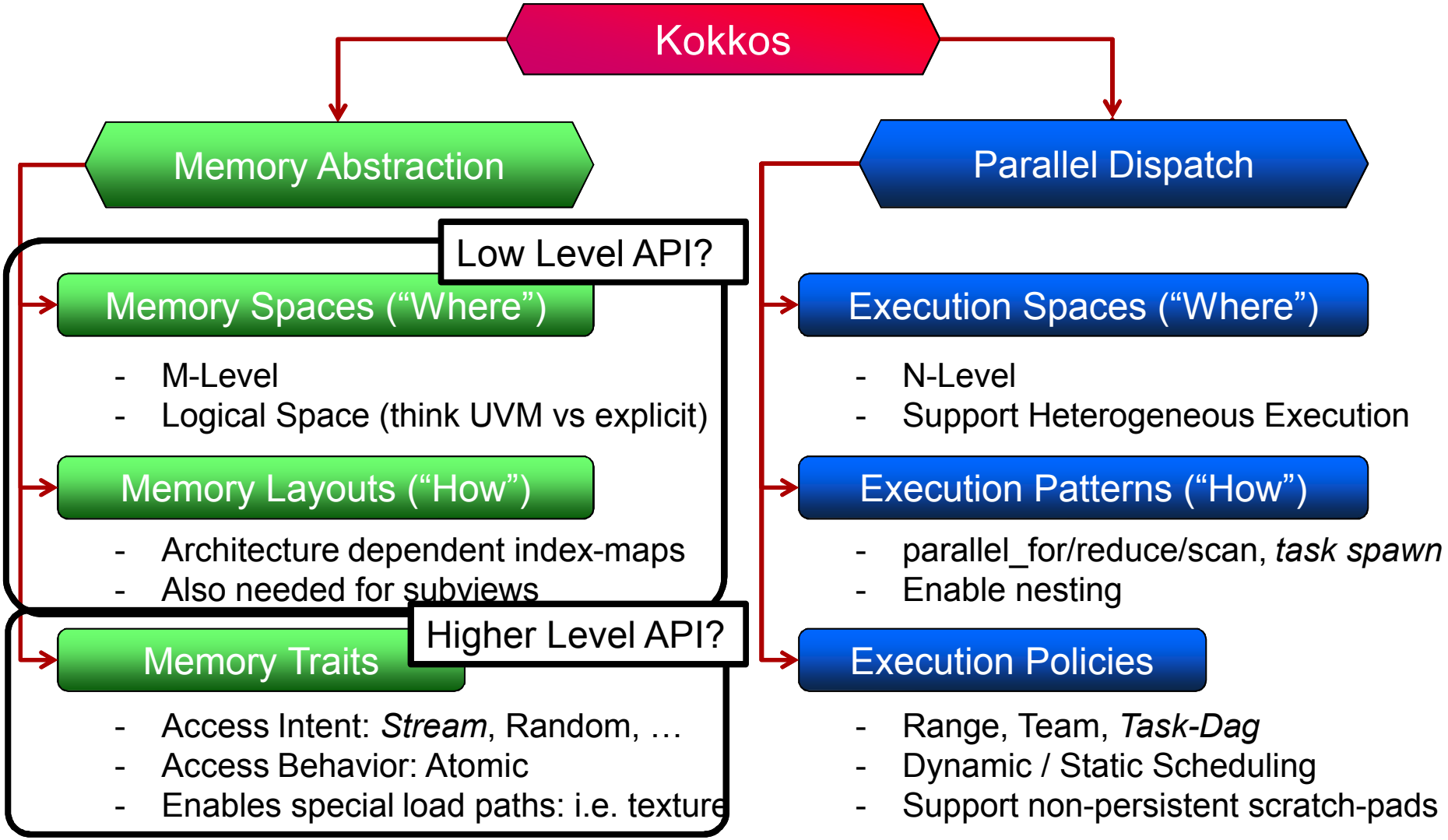
Separating Concerns for Next Generation Applications..



Performance Portability through Abstraction

Sandia National Laboratories

Separating Concerns for Next Generation Applications..



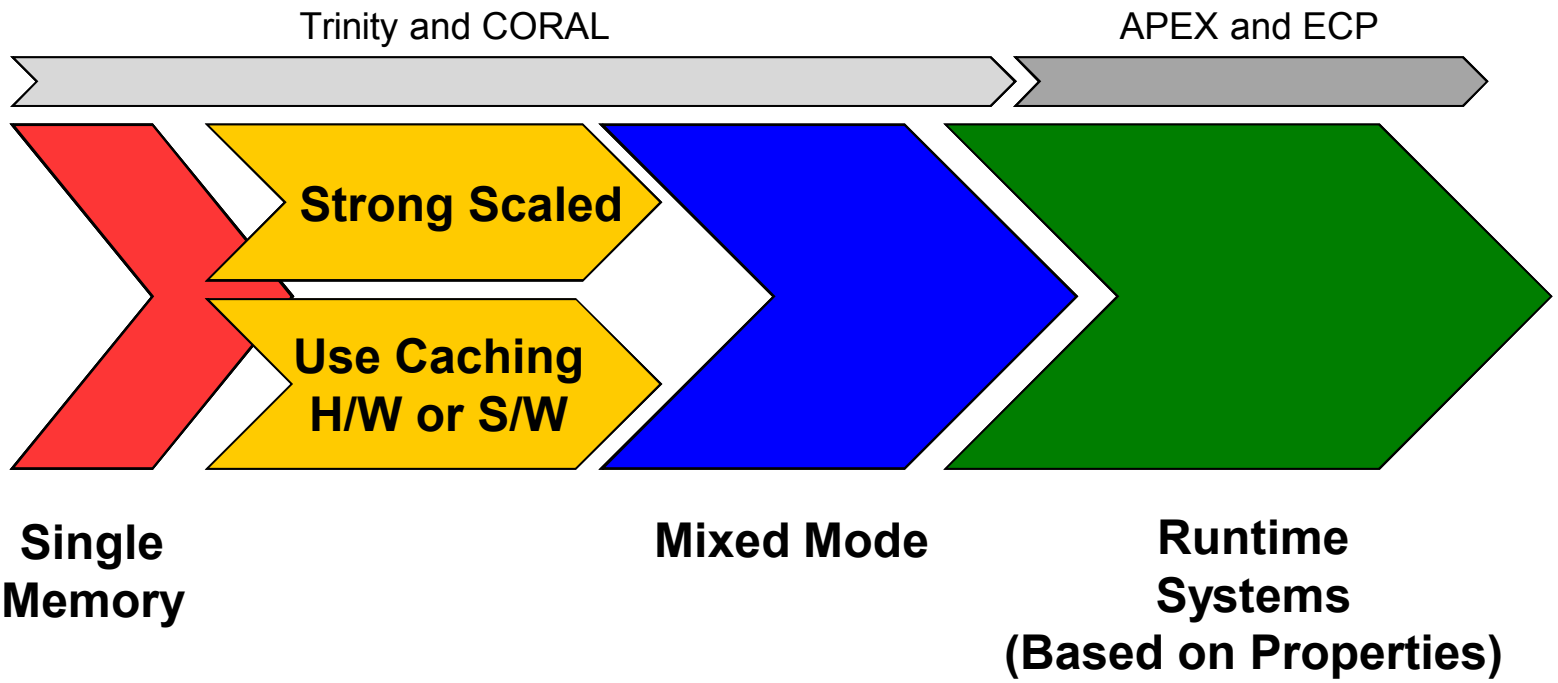
Towards Portable Memory Allocation

Sandia National Laboratories

- Want to be able to describe:
  - **Properties of the Allocation** – persistence, reliability level required
  - **Properties of the Accesses** (perhaps for sections of code) – atomic, streaming, random, mixed, don't-cache (streaming loads/stores)
- Low-level functions and API are important but these are for system software/runtime developers and the application ninjas
- Like that some *properties* can be implied by a compiler
- **Simply not scalable to expect us to modify all our allocations and know what to do every time on every different machine**


Sandia
National
Laboratories
THE LONG ROAD
AHEAD...

How is this going to work out?



- **Simply just too much to work on right now (applications teams are overloaded)**
  - Turned out threading was much easier than memory systems, but even this is taking a *long* time
- My feeling is that for Trinity (and probably CORAL) we will see applications use caching (H/W or S/W (UVM)) while we work all this out (performance cost)

Experiences on GPU Systems

Sandia National Laboratories

- Mixture of approaches seen in our codes
  - Specific allocation of data structures on GPU
  - Use of S/W management with UVM

- General path – UVM which effectively limits our maximum problem sizes

- Performance can vary quite a bit but when it works, it has worked very well compared to contemporary dual-socket systems

- Clear that future H/W directions (CORAL) address many of the more pressing concerns (and make software *much* easier to write)

- Looking forward to NVLINK prototype systems very soon

Experiences on KNL

Sandia National Laboratories

- Initial work on KNL with mini-applications and some performance kernels (from Trilinos) going very well

- For some applications, greater improvement than the hardware specifications moving between memory

- Strongest application performance for some kernels on any GA-hardware we have ever seen

- API (memkind) bring up going well but we expect this to be low-level (users do not like this and want it hidden away)

- Lots under NDA but results will most likely be shown at ISC'16

Mem Kind: http://memkind.github.io/memkind/memkind_arch_20150318.pdf

Exceptional service in the national interest

http://www.github.com/sstsimulator