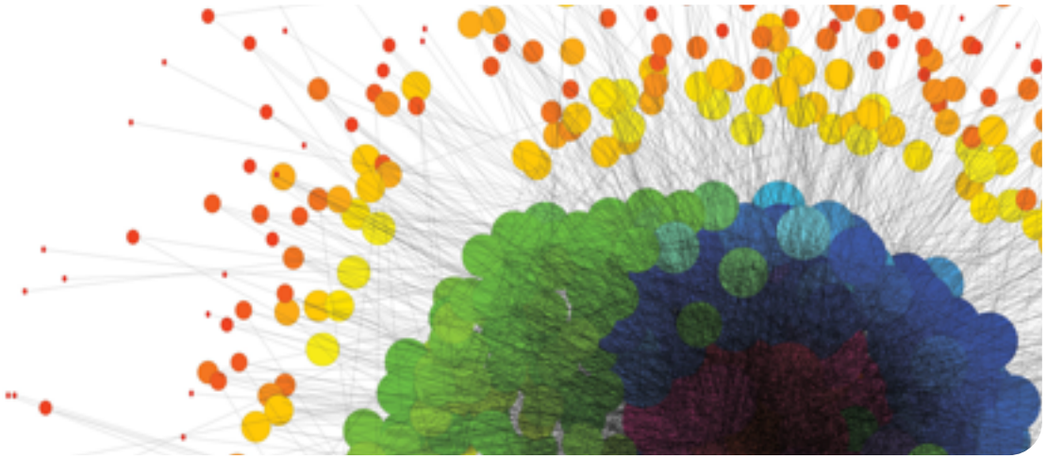
Exceptional service in the national interest

Sandia National Laboratories



A Strategy For Exascale Computing

Dylan Stark & Kyle Wheeler

HPC User Forum 2012

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.

SAND: 2012-7850P

The Exascale Challenge

- **What does an exascale machine look like?**

- What does system software for an excascale machine look like?
  - Collaborate with the level above and the level below
  - Leverage technology trends
  - Rethink application space (what will be important in a decade?)
  - *Be metric-focused!*
    - Picojoules, picojoules, picojoules ... and time too!

There's no THERE there!

- **What does the future look like?**
  - 125 MW+ exascale platform
  - Cannot cool aggressive clockrates as feature size scales
  - All future performance gains through parallelism (3–5 orders of magnitude!)
  - DATA MOVEMENT and not FLOPS dominates energy performance
- **Apps increasingly unstructured**
  - DS&A
  - Physics
- **Given a limited investment portfolio, where do we invest?**
  - Data movement!
  - Parallelism!
    - Gains needed to match today's performance power efficiently
    - Plus 3 orders of magnitude to achieve exascale
- **The current commodity trajectory is insufficient to hurdle the energy wall**
  - Must influence more than the interconnect
  - The programming model is insufficient to meet the challenges

From where we sit...

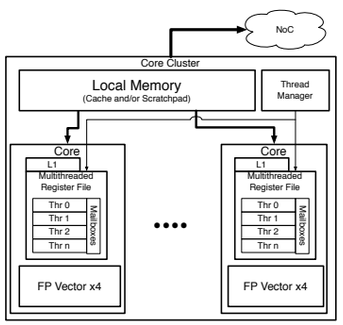
Sandia National Laboratories

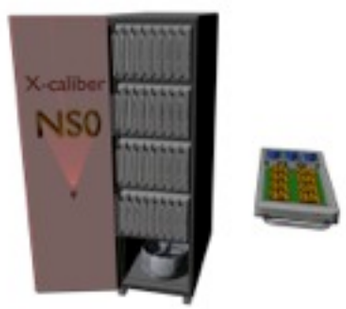
Safety and Security | **Reentry** | **Circuitry** | **Graph** | **Stream**

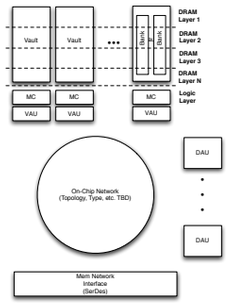
Application Drivers

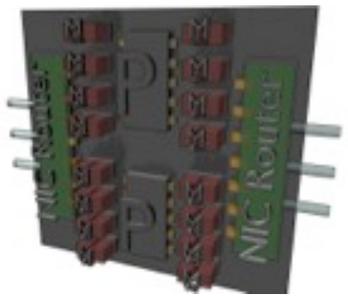
System Software - enabling a new model of computation

Architecture - Coping with Concurrency and Data Movement

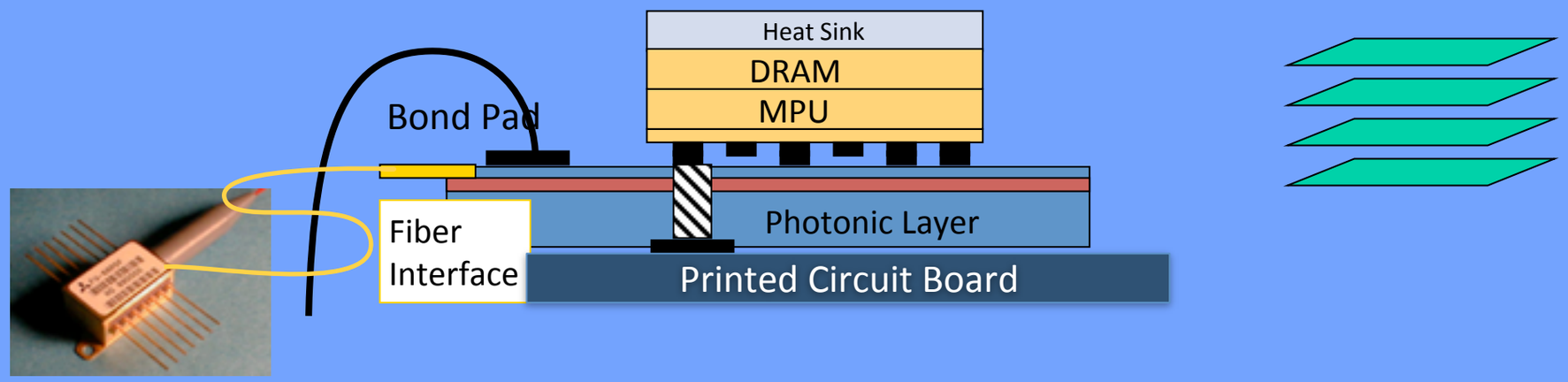






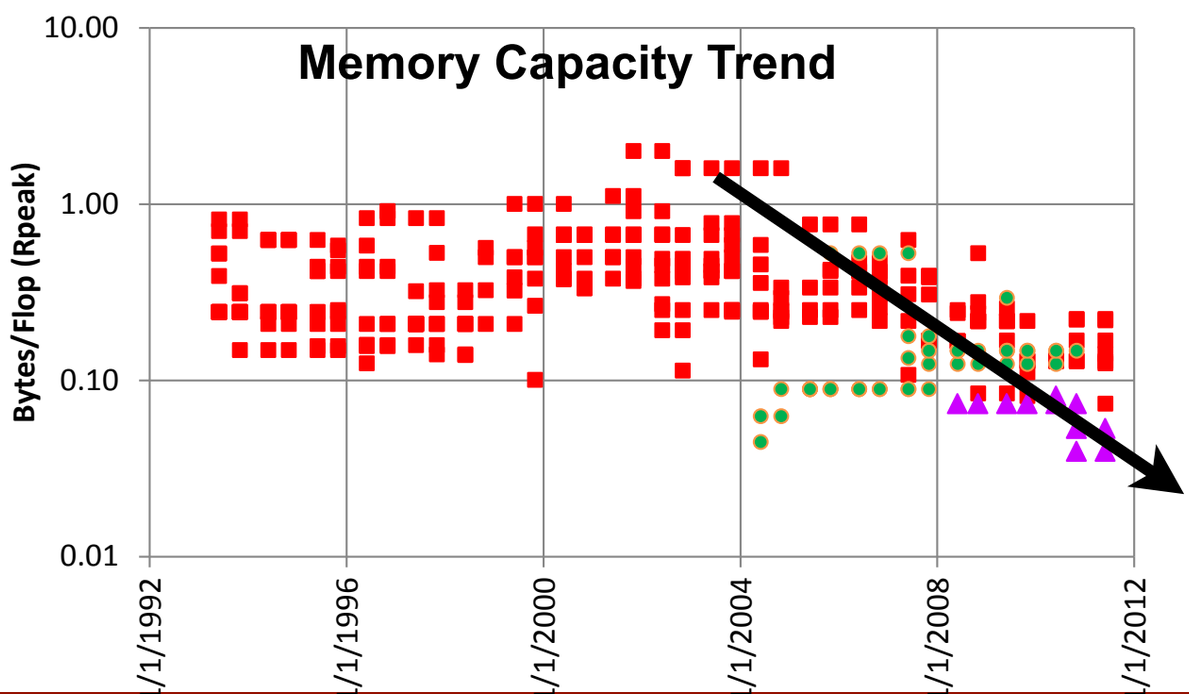


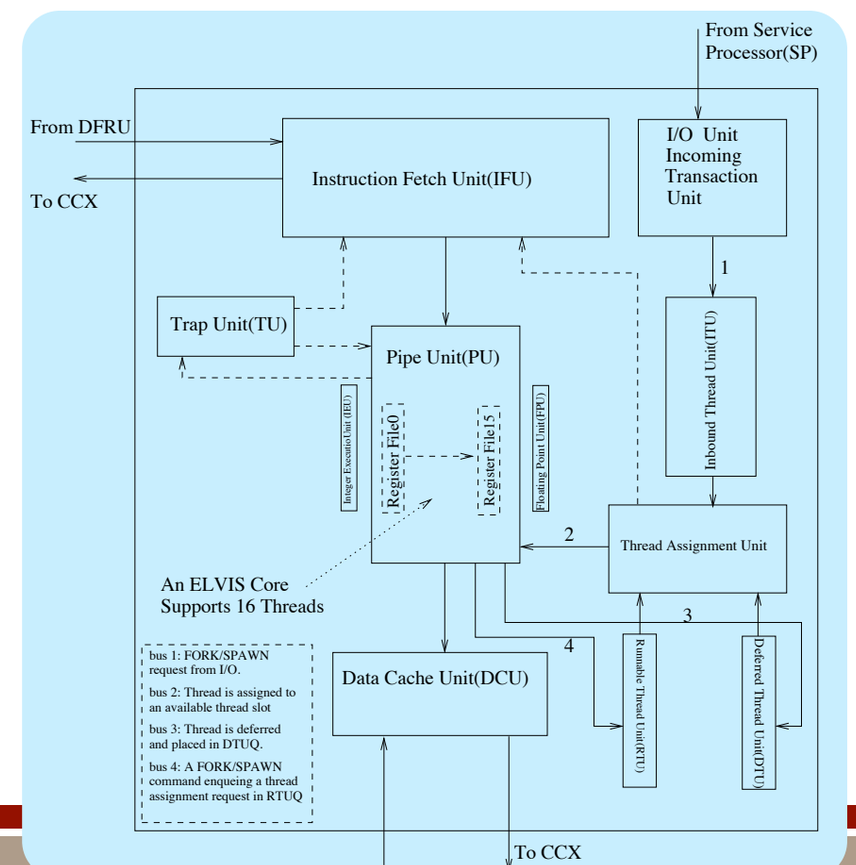
Microsystems - Key Data Movement Enabling Technologies



Trouble Down Below: Hardware

- Exponential increase in node-level parallelism
- Lower memory capacity per core
  - Weak scaling will be insufficient!
- Significantly lower network to memory bandwidth ratios
  - HIGH latency difference
- Need for system software to have finer-grained control of hardware resources

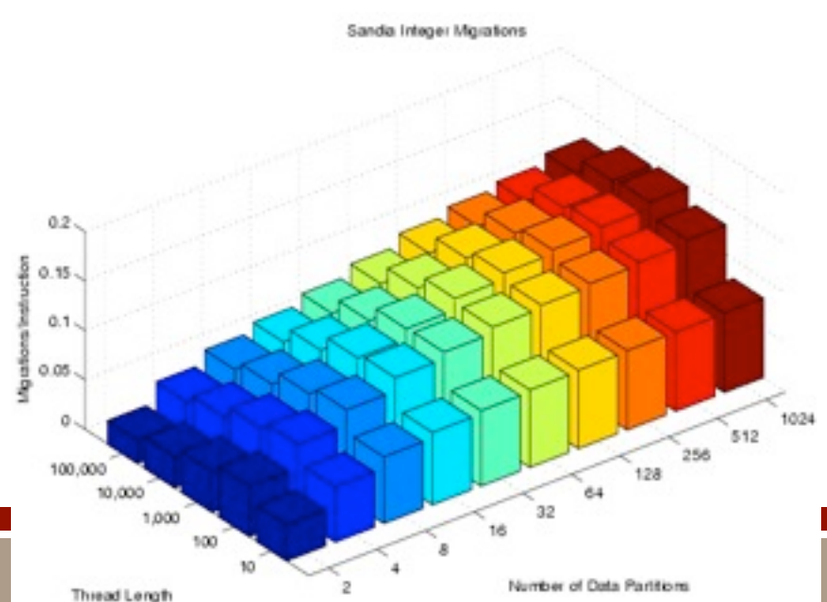


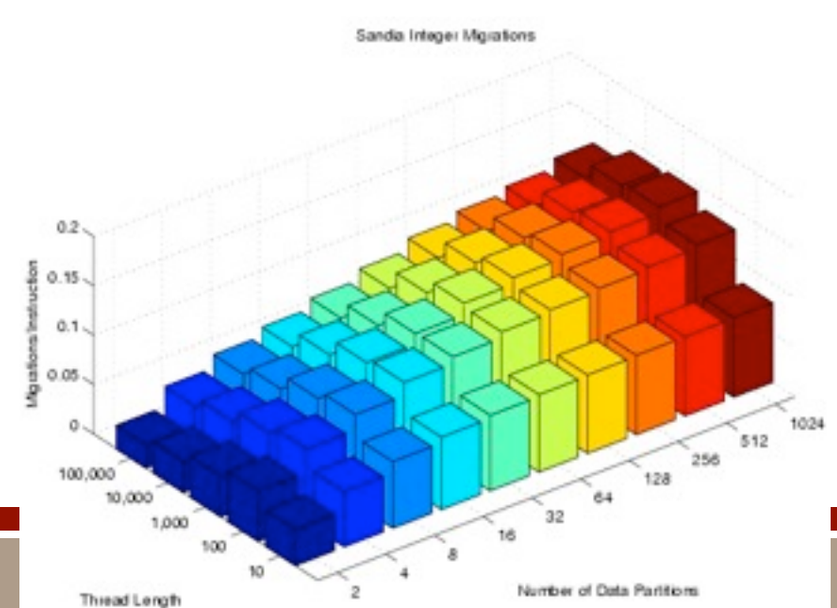


■ Heavyweight ● Lightweight ▲ Heterogeneous

Programming Model Falls Down

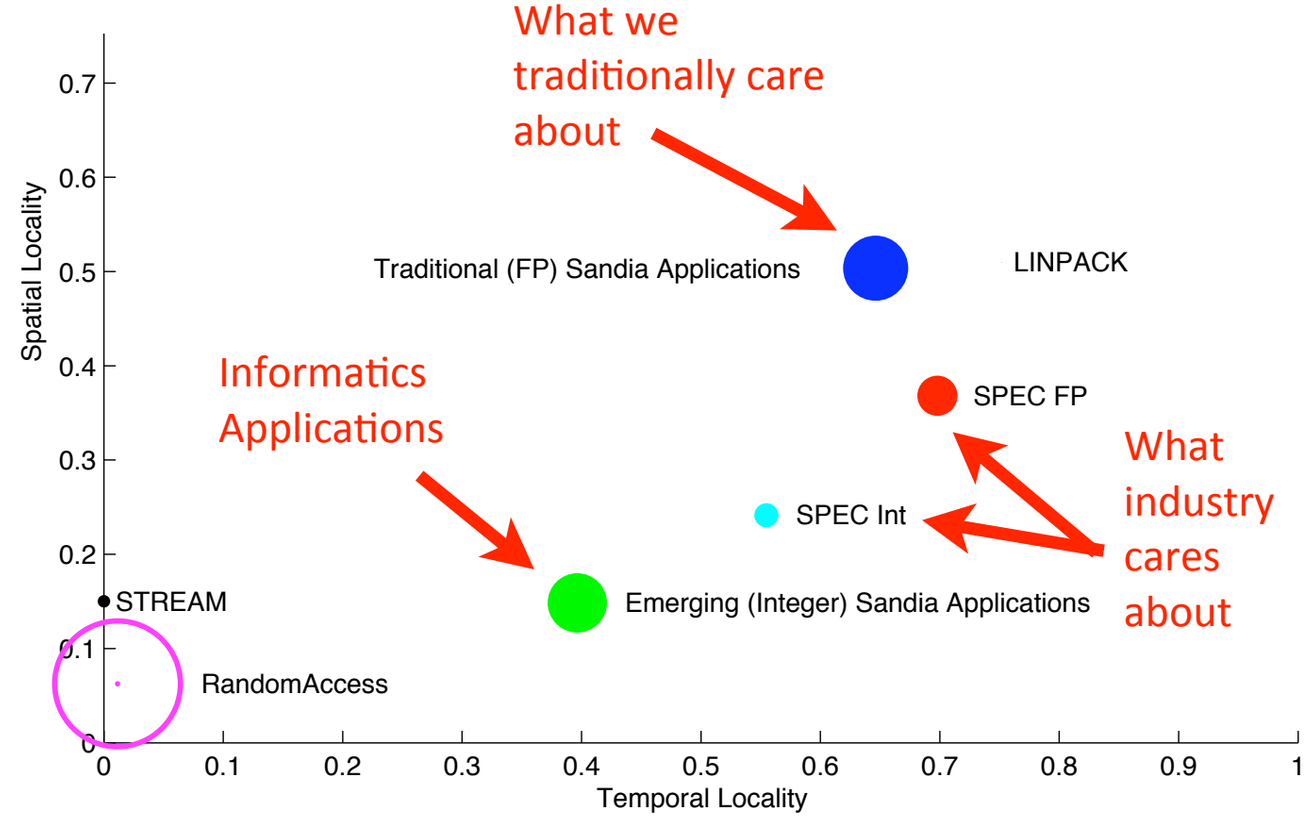
- **Every core in the system has to know about every other core in the system**
  - We can't afford the energy for today's loose coupling
  - We can't manage the concurrency (even locally) with today's model and today's coupling — the synchronization problem is too hard
- **New models are required**
  - 5–400x improvements in data movement over the application suite
    - **THIS IS WHERE YOUR ENERGY GOES!**
  - Reduced thread state size (15% of a modern register file)





Death from Above: Applications

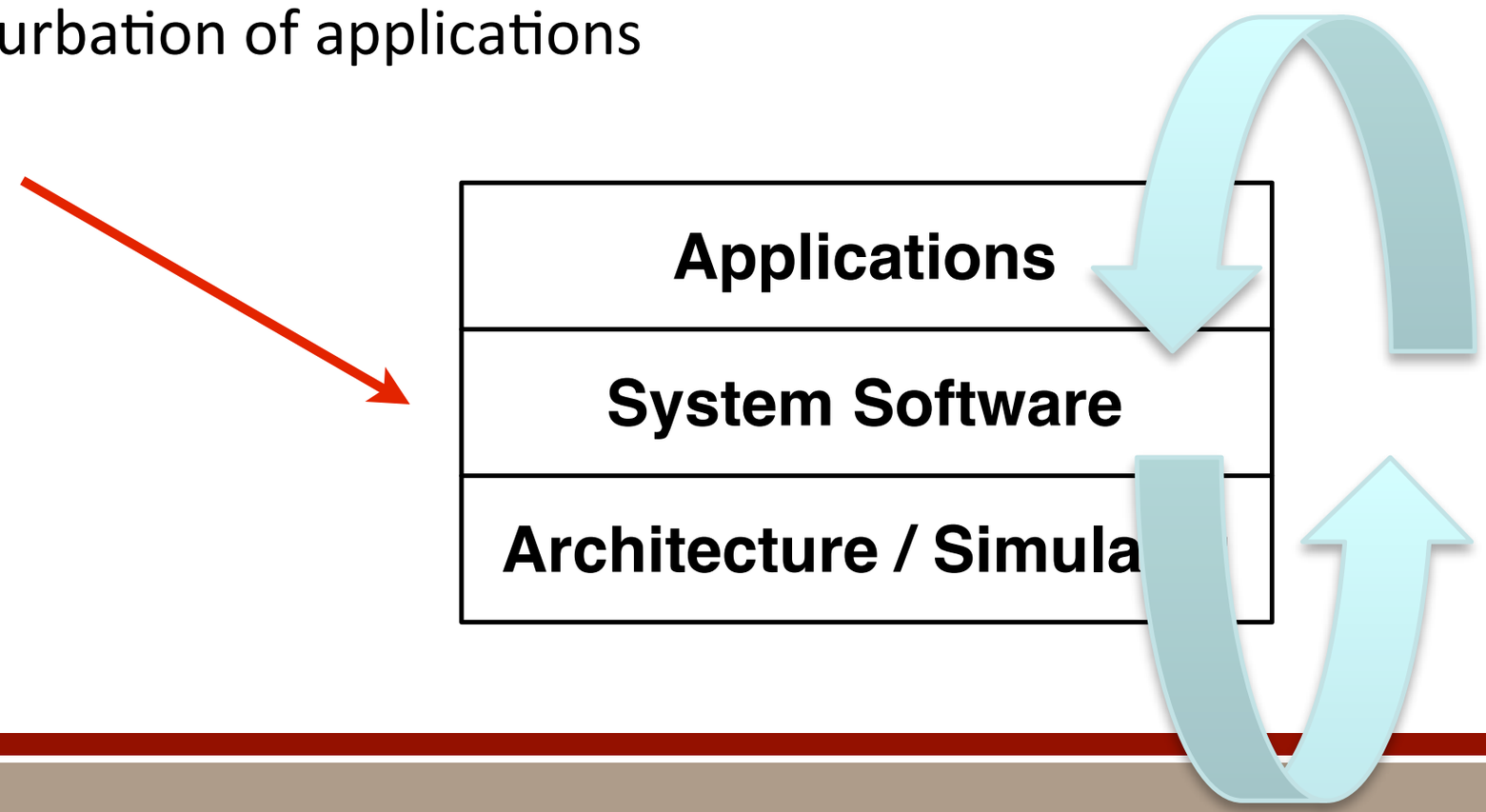
- **Huge variety in programming models and run-times emerging**
  - Evolutionary BSP-originated applications
  - Revolutionary programming models
    - Everyone's got one, and they're all the best
    - 101+ actively developed parallel programming languages
  - Lots of new application varieties
  - Flexibility is key
- **Multiple optimization points**
  - Time to solution
  - Energy to solution
  - Money to solution
  - Total system efficiency



From: Murphy and Kogge, *On The Memory Access Patterns of Supercomputer Applications: Benchmark Selection and Its Implications*, IEEE T. on Computers, July 2007

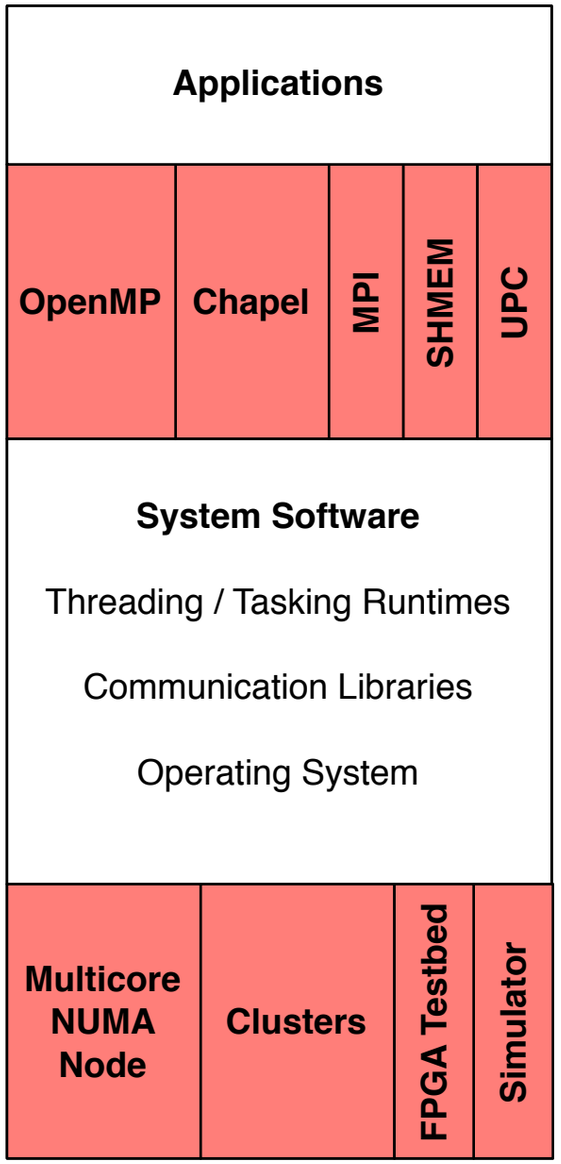
Foundational Knowledge

- Distributed systems scaling determined by:
  - Ability to move data
  - Synchronization
- Lightweight System Software WORKS
  - ASCI Red, ASC Red Storm, BG/{L,P,Q}
  - Manageable perturbation of applications & architectures



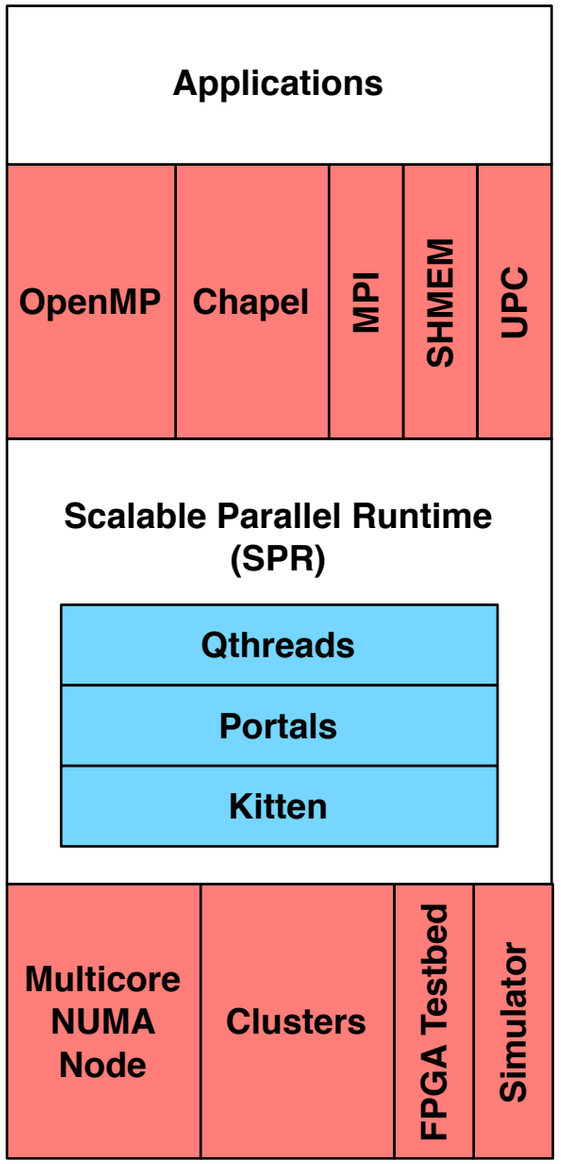
Necessity is the Mother of Invention

- Need insight into:
  - Trade-offs between different data/work movement strategies
  - Cost of synchronization/protection mechanisms with real applications
  - How much automation/adaptivity is necessary in large scale applications?
- Research is slowed by lack of experimental platform
- Use both clusters and simulation as foundational experimental platforms!
- Combine Kitten, Portals4, and Qthreads to build a **multi-node multi-threaded runtime** for experimentation (SPR)



Scalable Parallel Runtime (SPR)

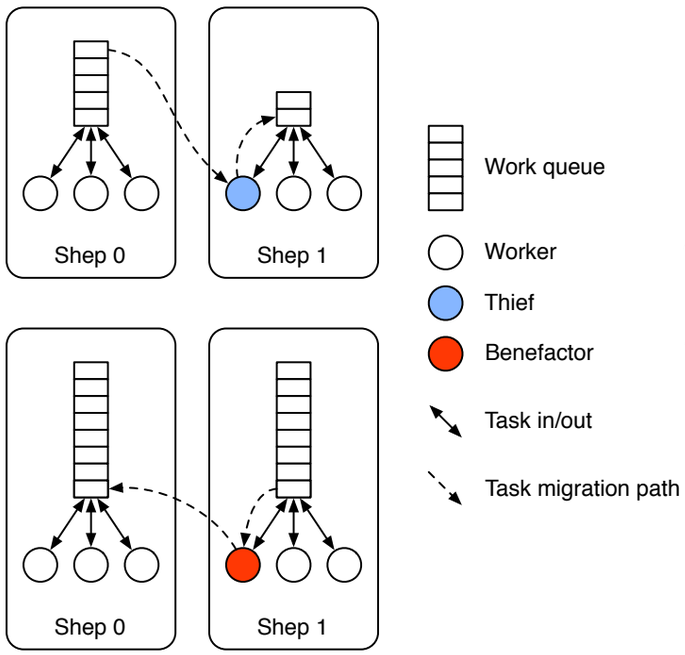
- **Kitten: Lightweight OS kernel**
  - Builds on lessons from ASCI Red, Cplant, Red Storm
  - Utilizes scalable parts of Linux environment
  - Primarily supports direct hardware mapping
- **Portals 4: Lightweight communication interface**
  - Semantics for supporting both one-sided and tagged message passing
  - Small set of primitives, allows offload from main CPU
  - Supports direct hardware mapping
- **Qthreads: Lightweight threading interface**
  - Scalable, lightweight scheduling on NUMA platforms
  - Supports a variety of synchronization mechanisms, including full/empty bits and atomic operations
  - Potential for direct hardware mapping



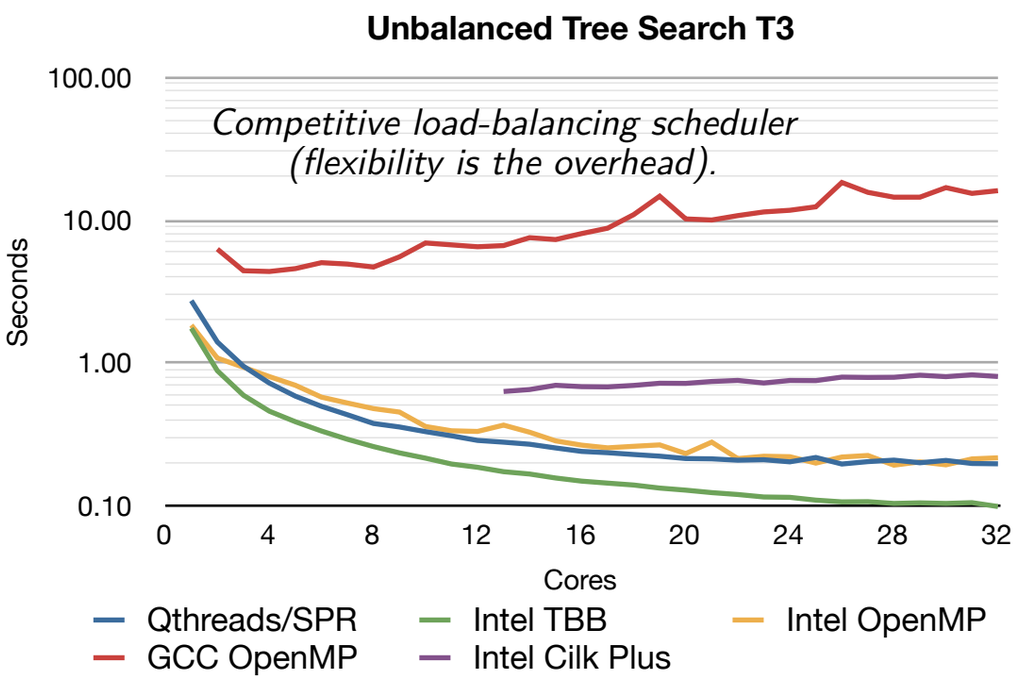
Open, Optimized Component Technologies

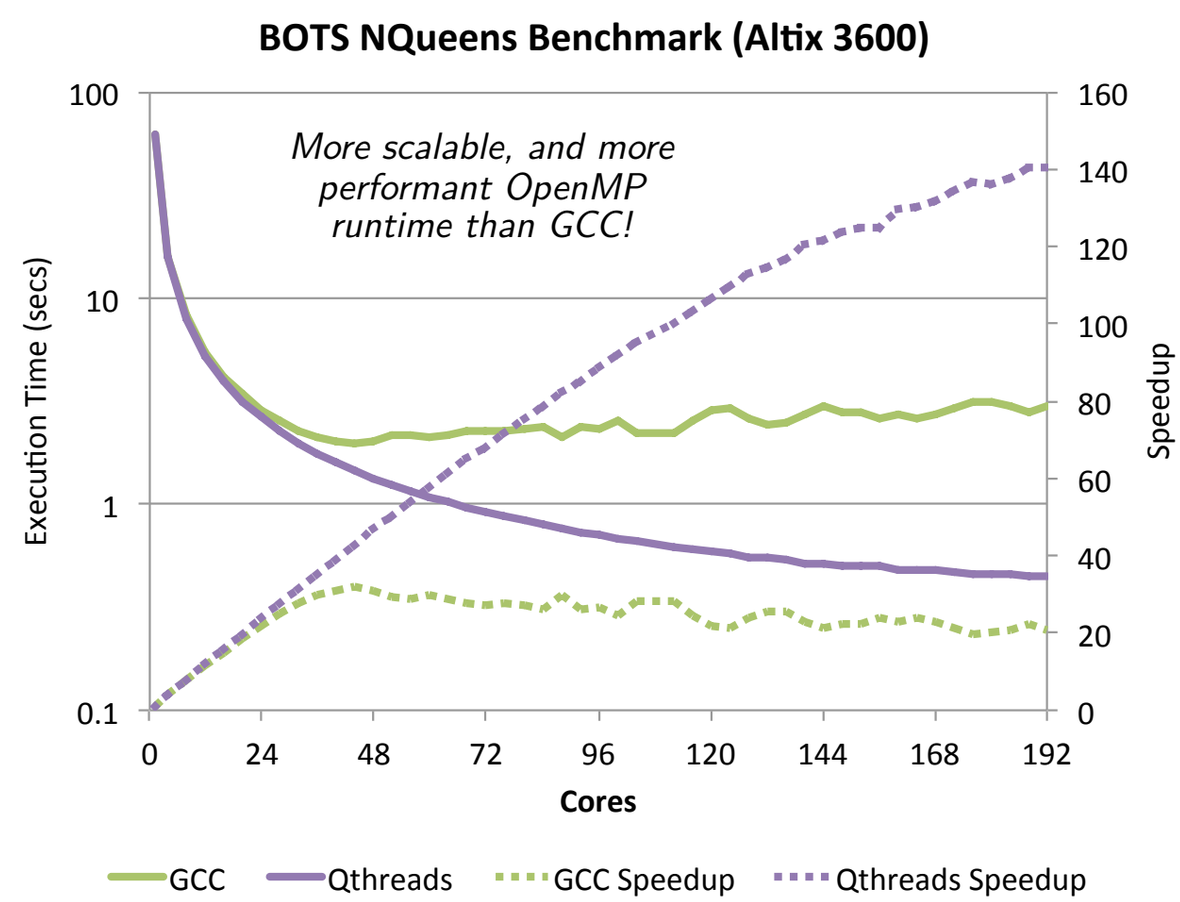
Qthreads Lightweight Threading

- **Simple task-based runtime**
  - Tool for programming model research
  - Supports both **OpenMP**-like models and more complex **Chapel**-like models
  - Presents simplified model of system to the application
  - High-performance scheduler
- **Current R&D**
  - Priority scheduling
  - Remote task launch efficiency
  - Efficient, flexible collective operations



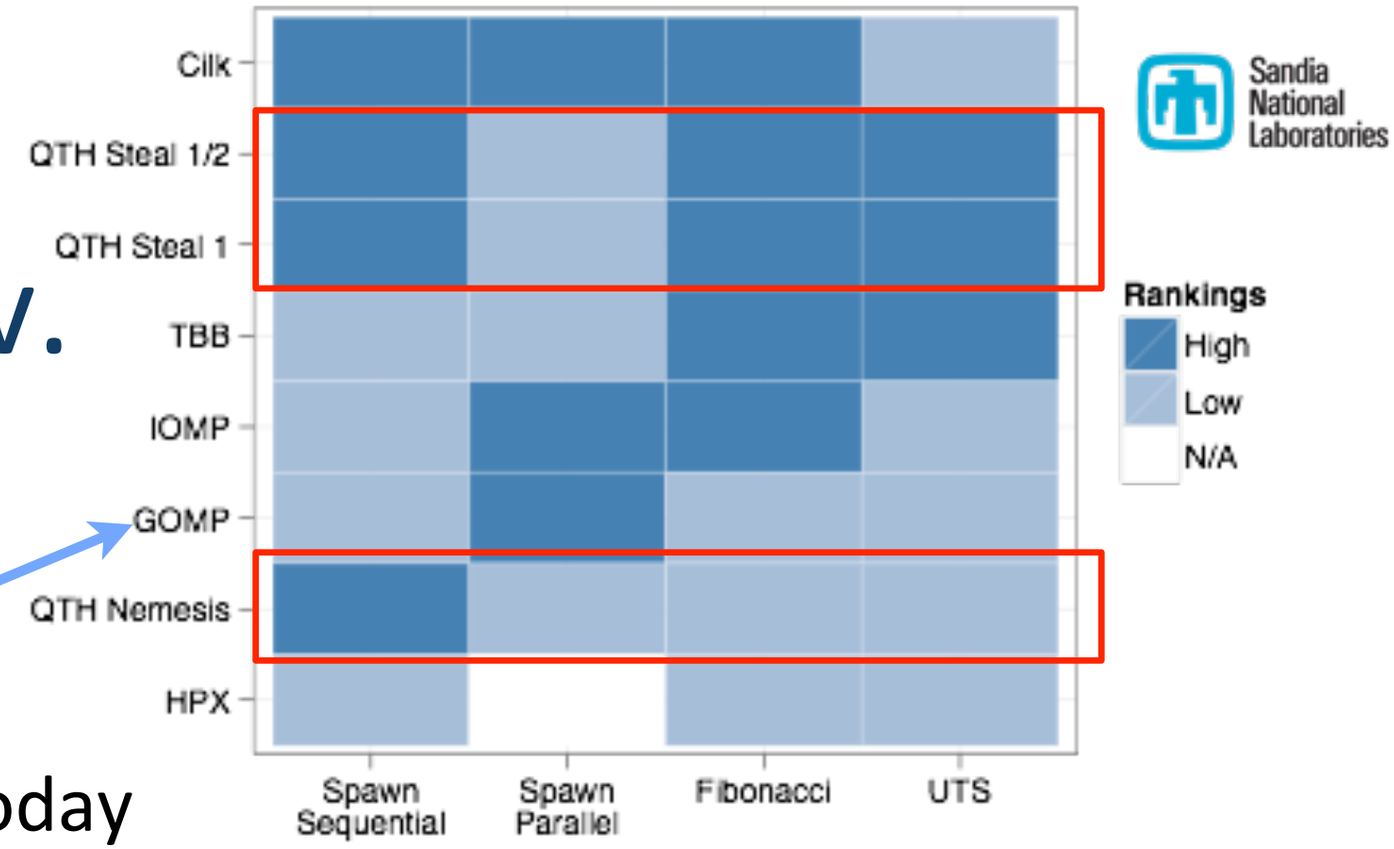
High performance "sherwood" work-stealing scheduler effectively balances cache efficiency with load balancing.

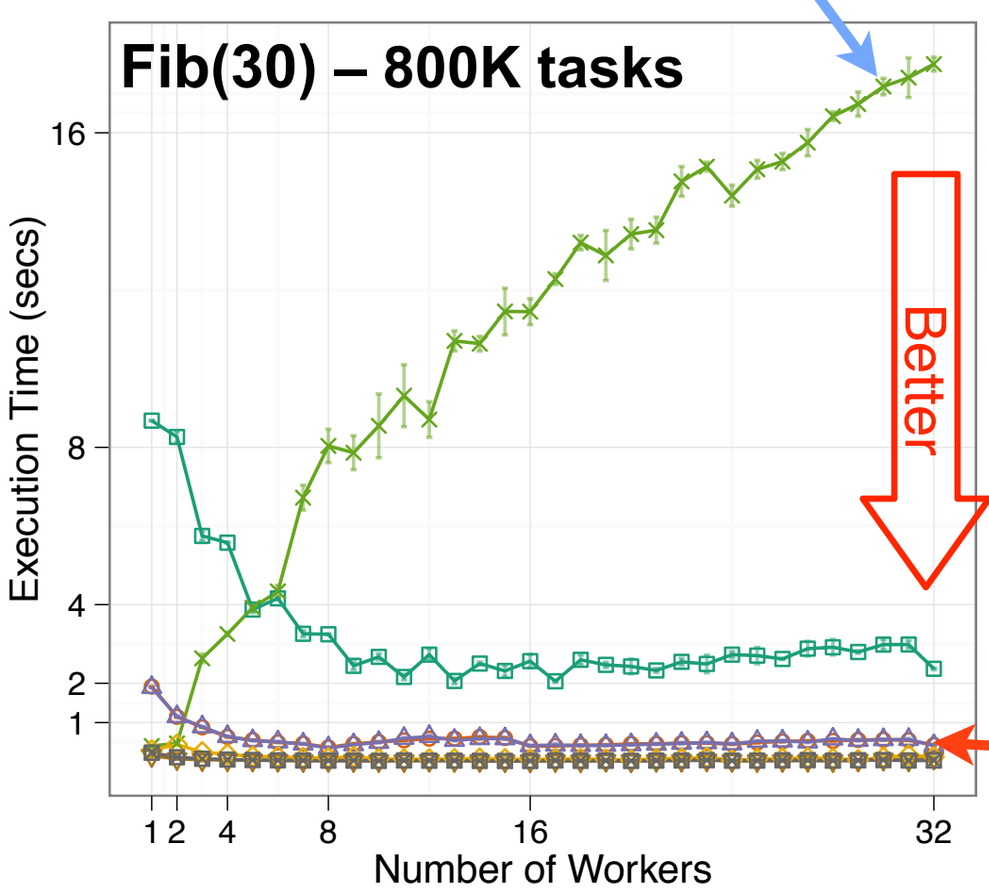


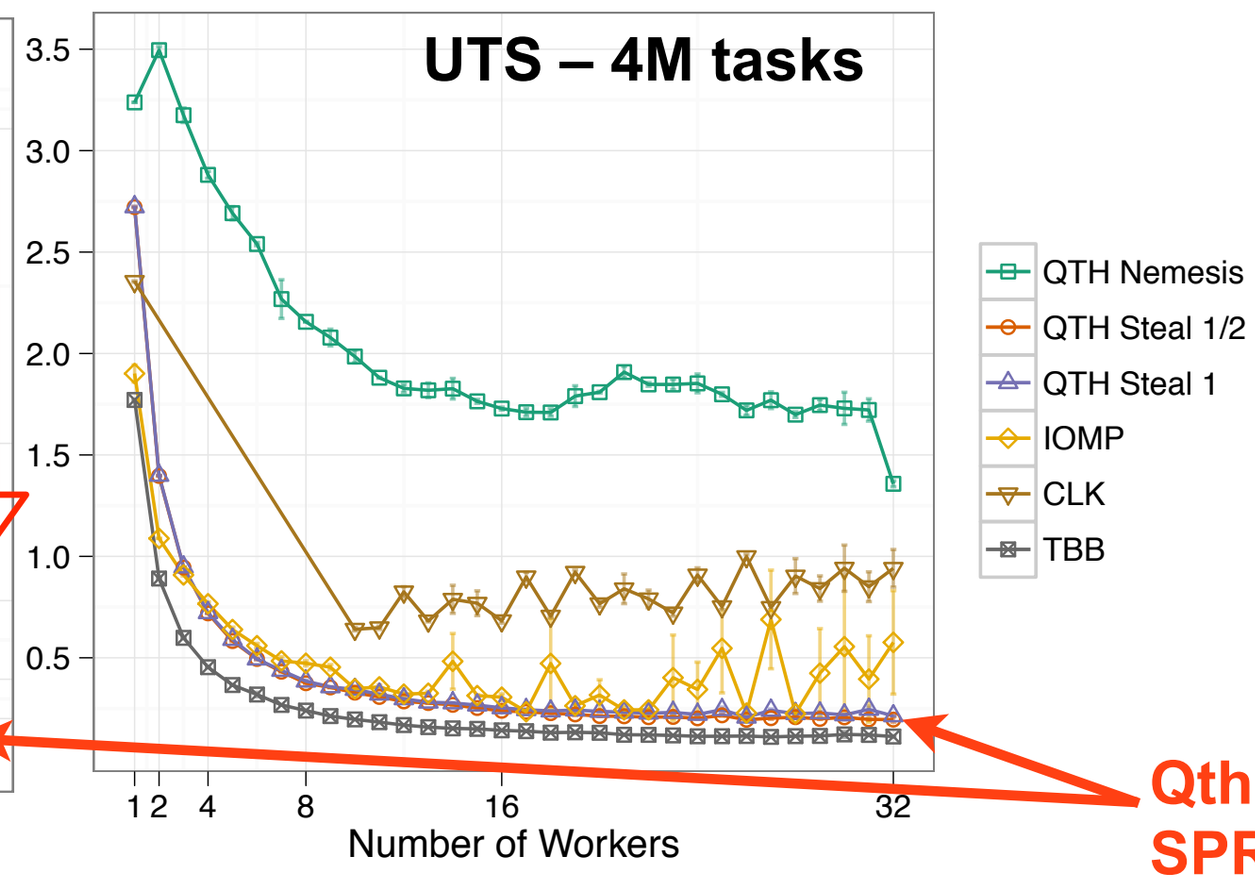


A *Vehicle* for Research *and* Dev.

- Ongoing runtime research
  - Highly configurable
  - Not one-size-fits-all
- Performance-competitive today



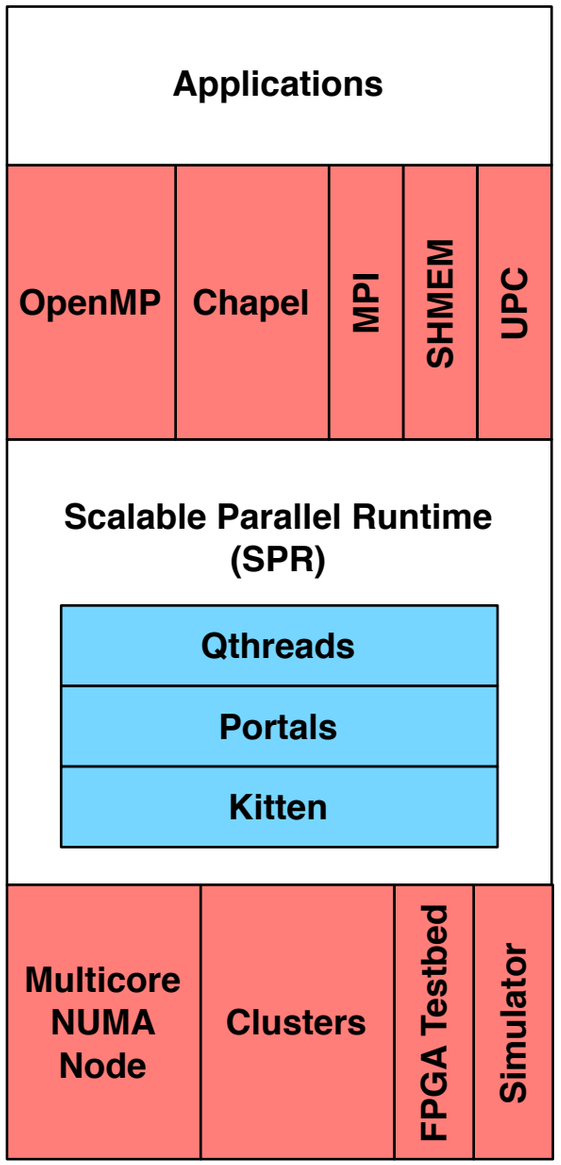




Qthreads/ SPR

Open Research Questions!

- **How will threads evolve to be more lightweight and match hardware semantics?**
  - What will hardware threading semantics be?
- **What synchronization primitives are necessary for highly asynchronous applications?**
  - Fast, Free, Infinite
- **What memory consistency models are necessary?**
  - … or even useful?
- **What communication primitives are necessary for evolving applications?**
  - Probably not six-function MPI



SPR is an Experimental Platform for Exascale R&D

http://code.google.com/p/qthreads

http://code.google.com/p/portals4

http://code.google.com/p/kitten

THANK YOU!

Runtime Architecture / Experimental Platform