SAND2012-2034P

X-Caliber & XGC System Software Research & Development

Kyle Wheeler

SOS16

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Context

- **X-Caliber: our UHPC effort**
- **XGC: Exascale Grand Challenge LDRD**
  - (Laboratory Directed Research and Development)

- **Consistent Challenge: figure out what exascale system software looks 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!

X-Caliber Software Team

- **Sandia**
  - Brian Barrett, Kyle Wheeler
- **LSU**
  - Thomas Sterling, Dylan Stark, Hartmut Kaiser, Chirag Dekate
- **University of Illinois**
  - William Gropp, Marc Snir
- **USC/ISI**
  - Pedro Diniz

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

XGC Software Team

- **Sandia**
  - Brian Barrett, Ron Brightwell, Kevin Pedretti, Dylan Stark, and Kyle Wheeler

- **University of Illinois**
  - Vikram Adve, Bill Gropp, Marc Snir

- **RENCI**
  - Allan Porterfield

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

renci
RESEARCH \ ENGAGEMENT \ INNOVATION

NNSA
National Nuclear Security Administration

Sandia National Laboratories

XGC Thrust Areas

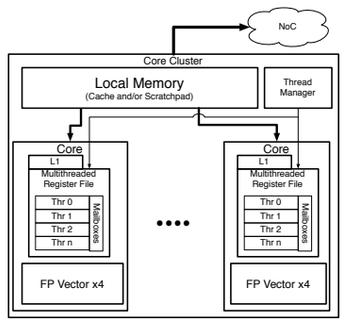
(1) Baseline - what happens if we do nothing?

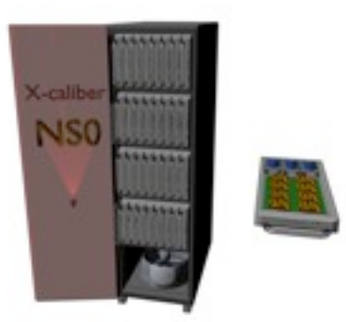
Safety and Security | **Reentry** | **Circuitry** | **Graph** | **Stream**

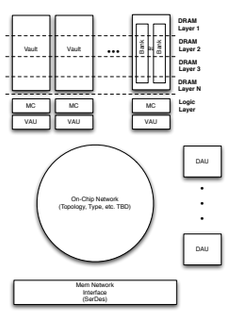
(5) Application Drivers

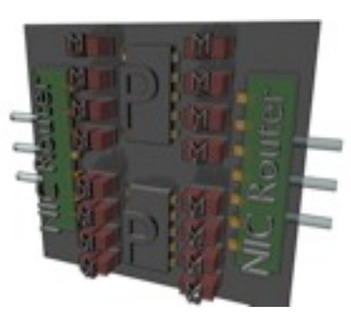
(4) System Software - enabling a new model of computation

(3) Architecture - Coping with Concurrency and Data Movement

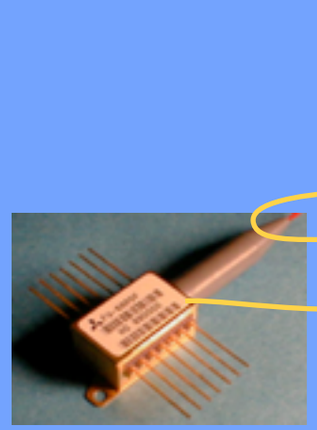


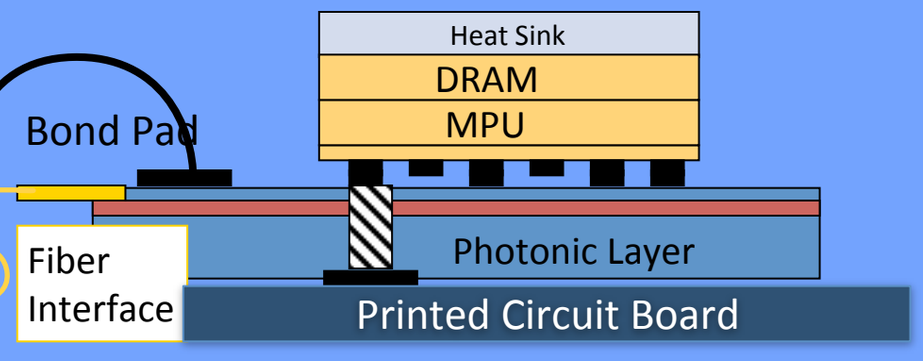


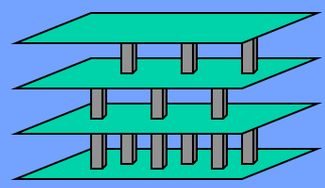




(2) Microsystems - Key Data Movement Enabling Technologies





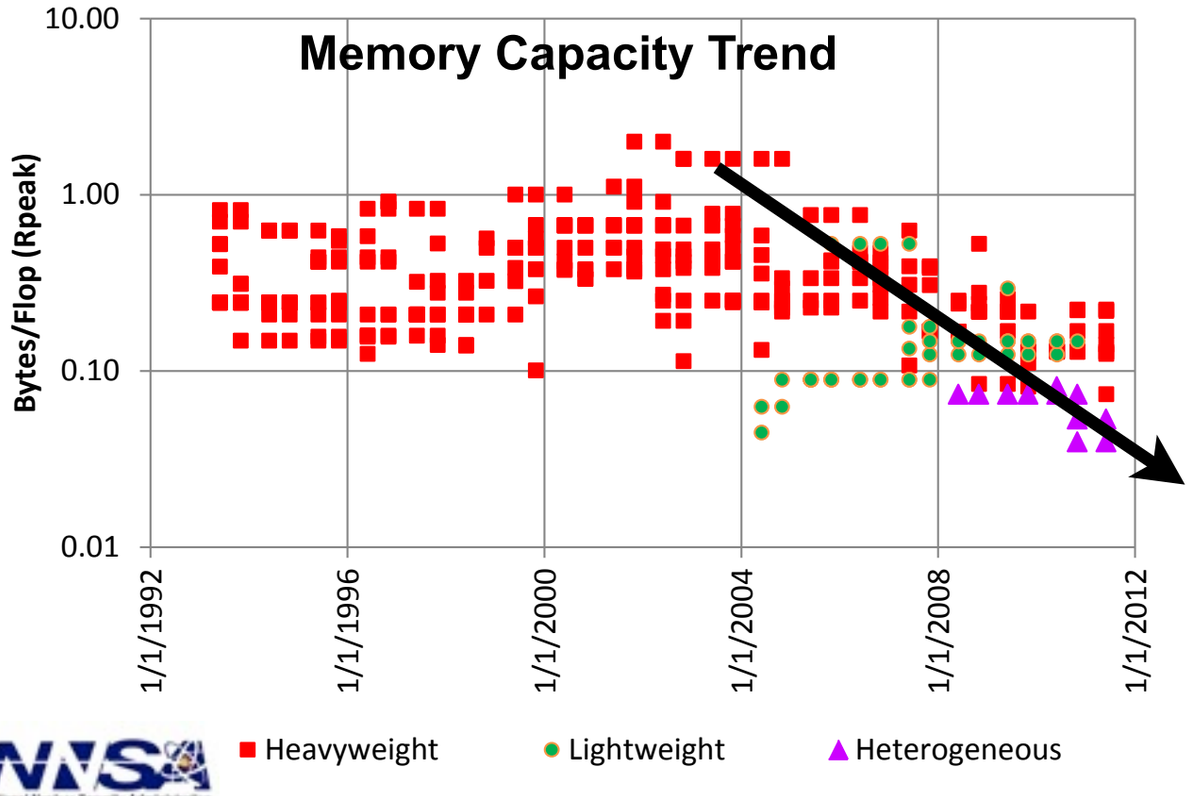


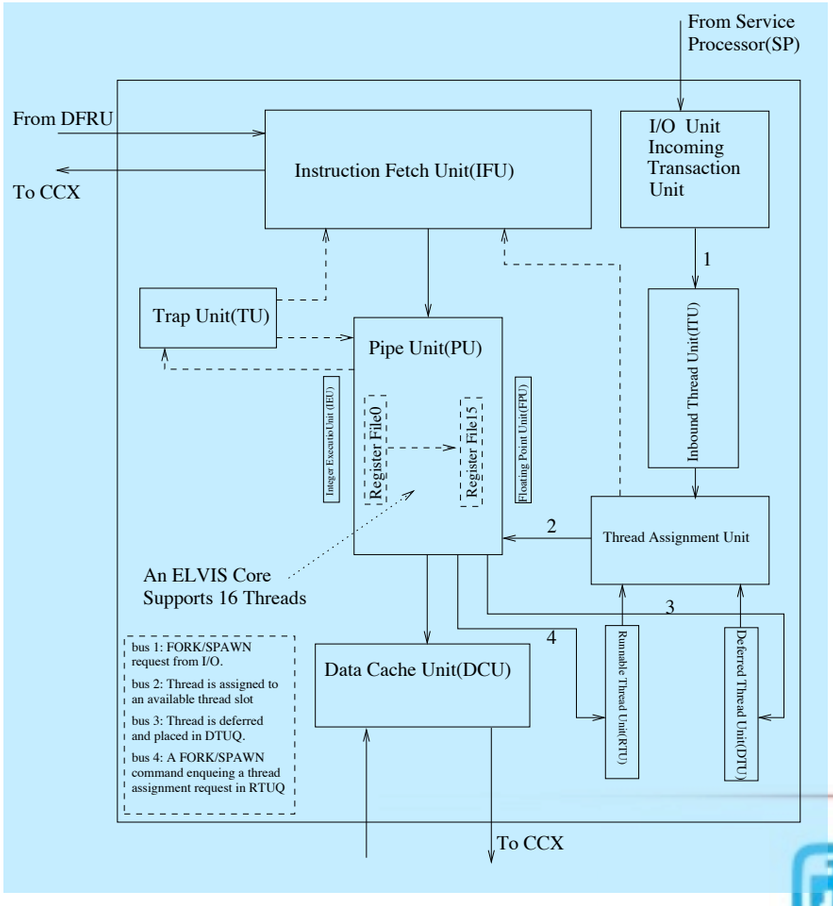
NNSA National Nuclear Security Administration

Sandia National Laboratories

Hardware Challenges

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



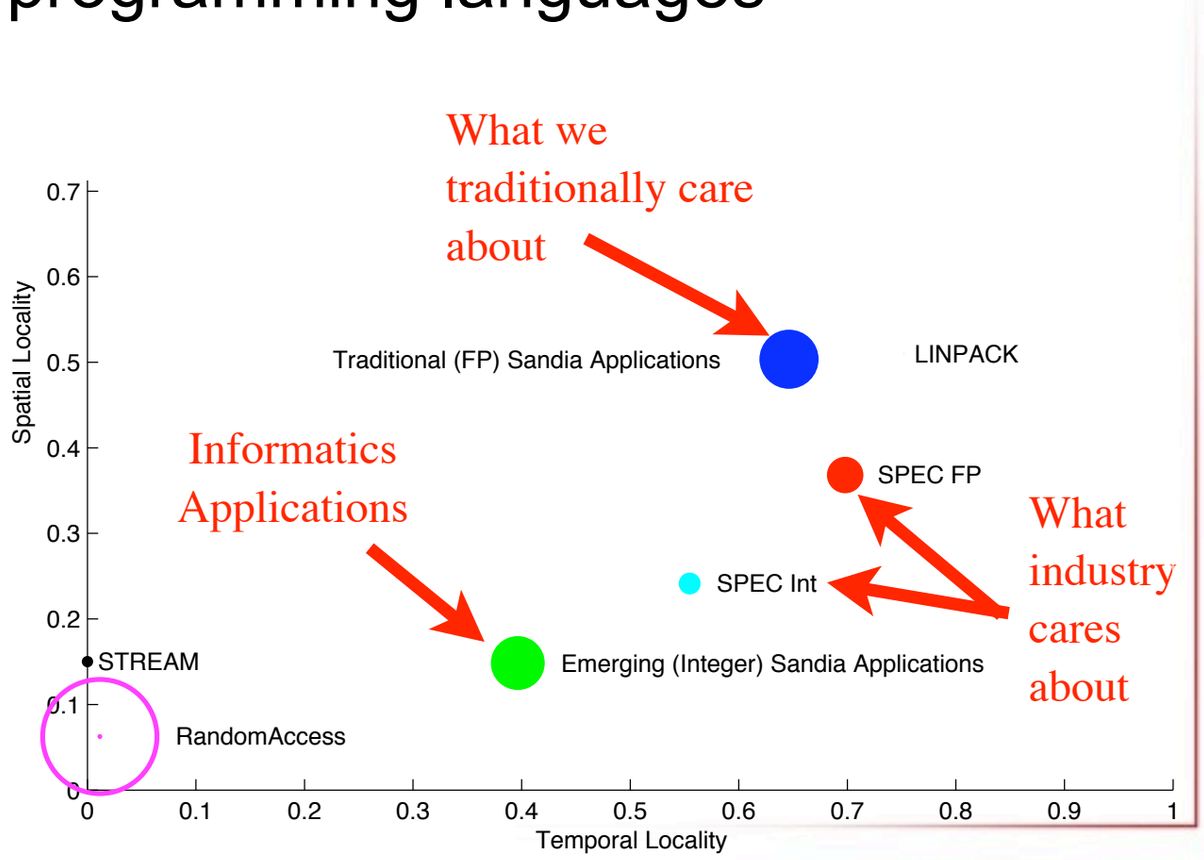


National Nuclear Security Administration

Sandia National Laboratories

Application Challenges

- **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
  - –Flexibility key
  - –Lots of new application varieties
- **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

NNSA National Nuclear Security Administration

Sandia National Laboratories

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}
  - Low perturbation of applications

- **Synchronization Costs**
  - Local and remote
  - Explicit and implicit

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?**
  - Free, Fast, Infinite

- **What memory consistency models are necessary?**
  - ... or even useful?

- **What communication primitives are necessary for evolving applications?**
  - Probably not six-function MPI

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 automaticity/adaptivity is necessary in large scale applications?

- **Research is slowed by lack of experimental platform**

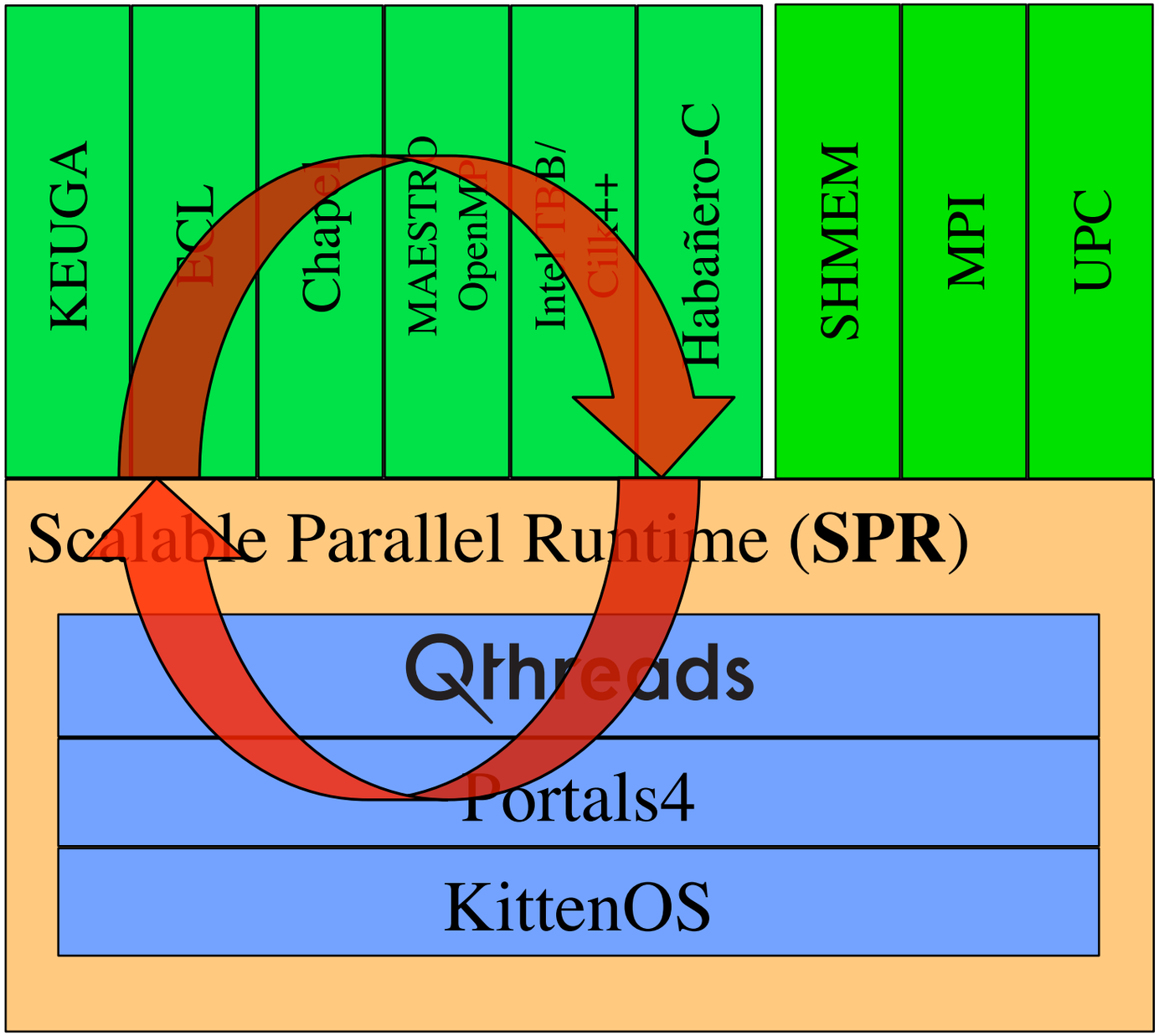
- **Use both clusters and simulation as foundational experimental platforms!**

- **Combine Kitten, Portals, and Qthreads to build a multi-node multi-threaded runtime for experimentation (SPR)**

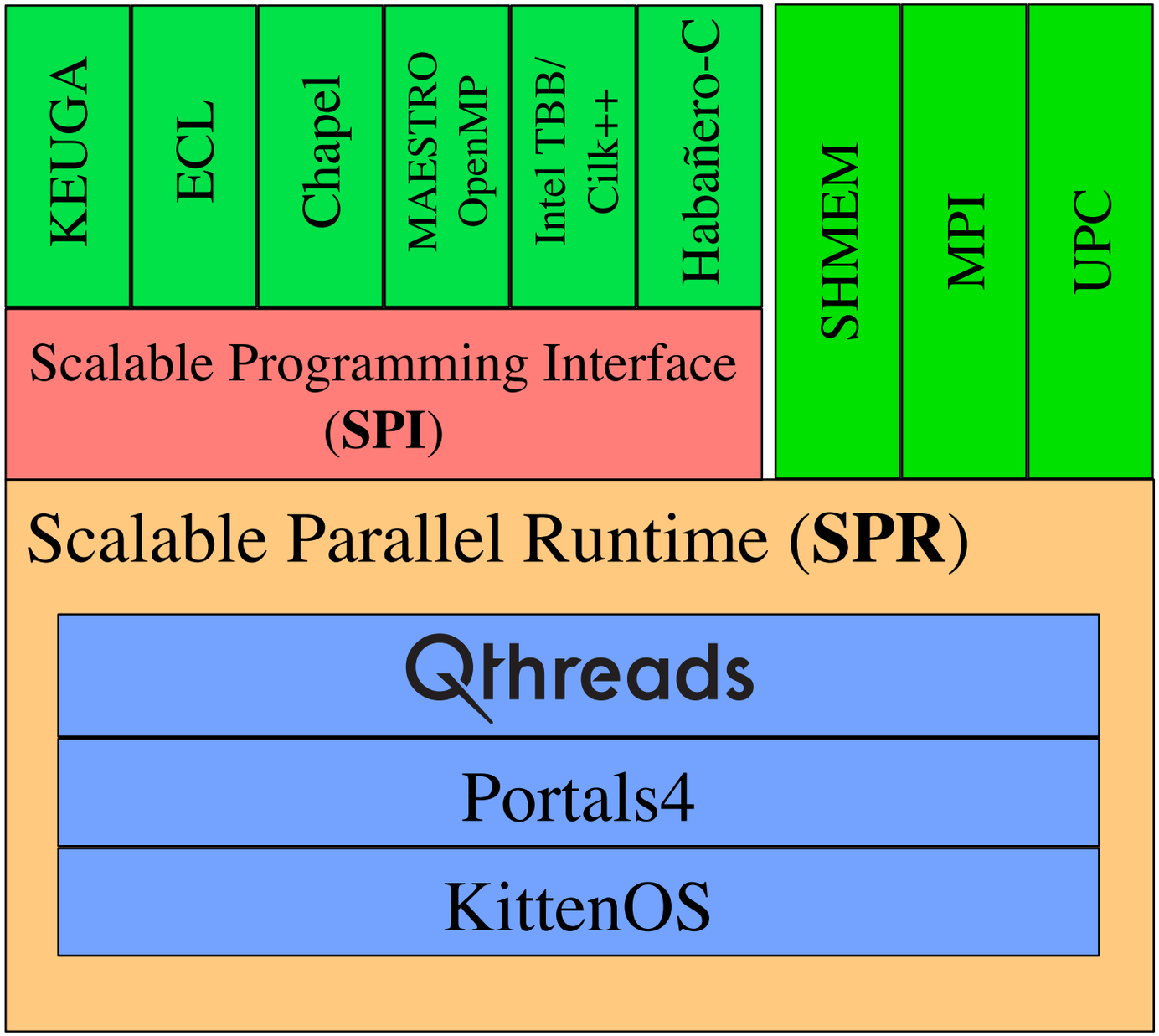
Scalable Parallel Runtime (SPR)

- **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
- **Portals 4: Lightweight networking API**
  - Semantics for supporting both one-sided and tagged message passing
  - Small set of primitives, allows offload from main CPU
  - Supports direct hardware mapping
- **Kitten: Lightweight OS kernel**
  - Builds on lessons from ASCI Red, Cplant, Red Storm
  - Utilizes scalable parts of Linux environment
  - Primarily supports direct hardware mapping

Runtime Architecture / Experimental Platform



Runtime Architecture / Experimental Platform

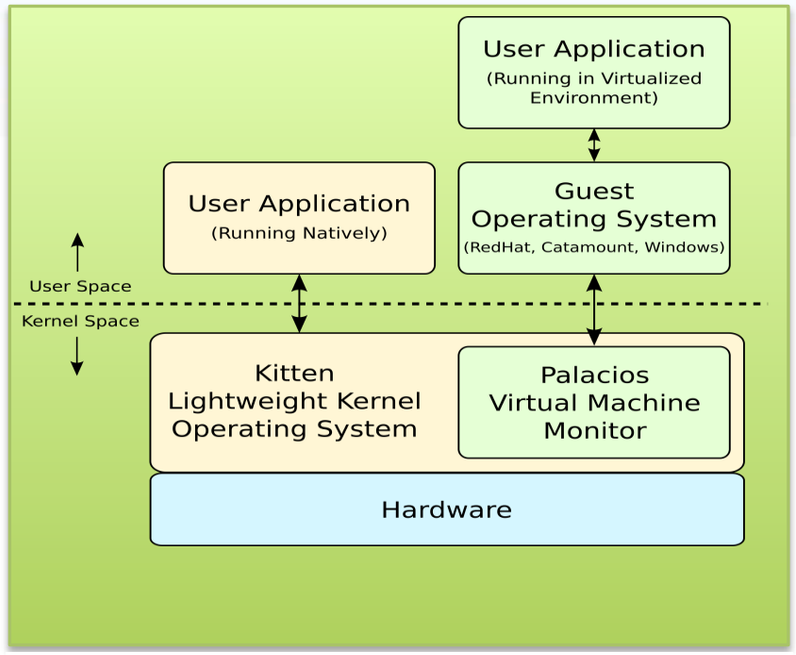


Kitten Lightweight Kernel

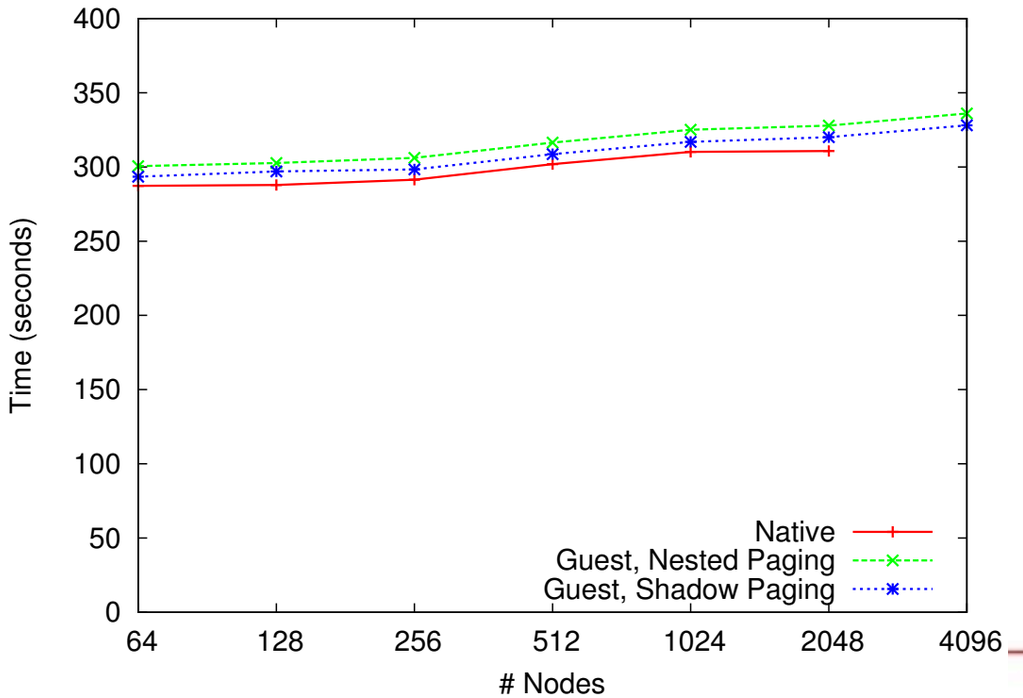
- **Simple compute node OS**
  - Tool for OS+runtime research
  - Looks like Linux to applications and tools
- **Current R&D**
  - Job launch via OpenMPI ORTE / mpirun
  - Support for Intel MIC, Arthur cluster at Sandia
  - System-call forwarding
  - Low-overhead task migration

Operating System	Round-Trip Task Migration Time (task on core A migrates to core B, then back to A
Linux 2.6.35.7	4435 ns
Kitten 1.3	2630 ns

Core-switching performance between two cores in the same Intel X5570 2.93 GHz processor. Kitten achieves a speedup of 1.7 compared to Linux, due to simpler implementation.



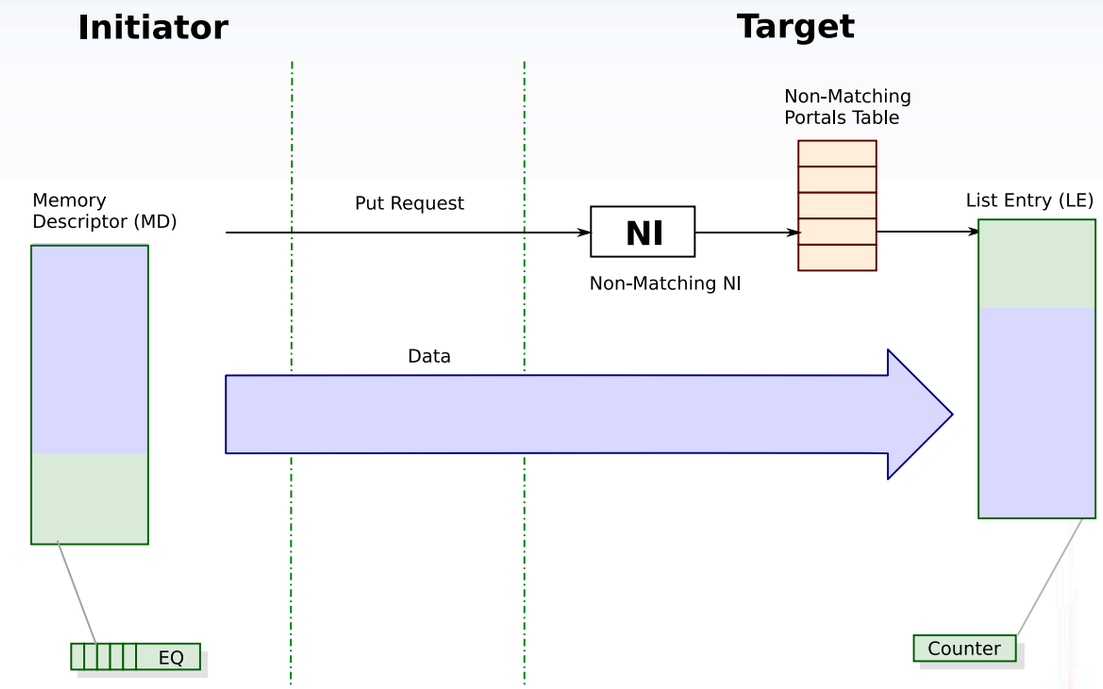
Kitten LWK supports running native applications alongside guest OSes.



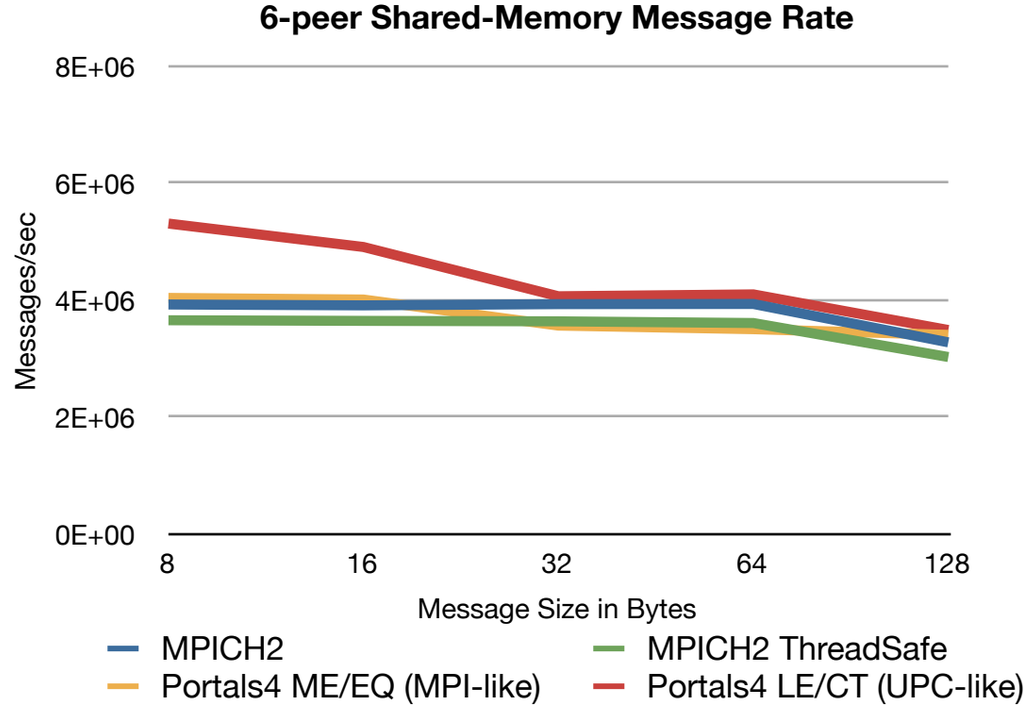
Weak scaling performance of Catamount guest OS is within 5% of Catamount native OS at 4096 nodes

Portals4 Lightweight Comm.

- **Simple low-level communication layer**
  - Tool for communication+runtime research
  - Thread-safe by design
  - Supports legacy and next-gen applications and tools
  - Common substrate to allow efficient use and sharing of resources among higher-level protocols
- **Current R&D**
  - Support for InfiniBand and efficient shared memory multi-core
  - Efficient blocking/waiting mechanisms



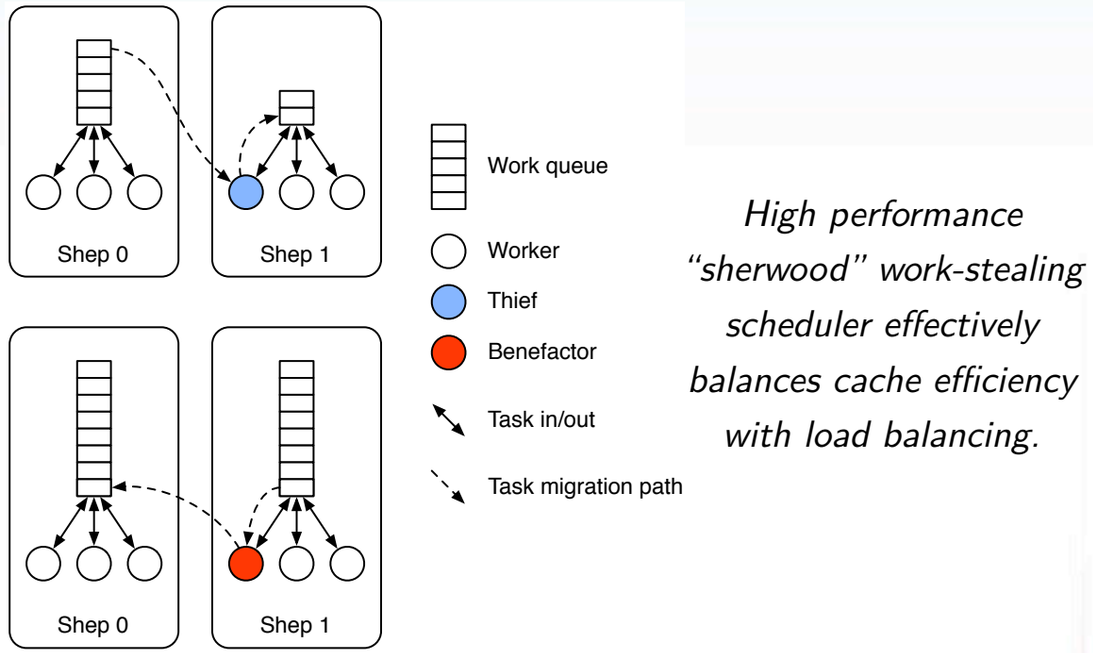
Much of the potential complexity gets out of the way for basic operations.



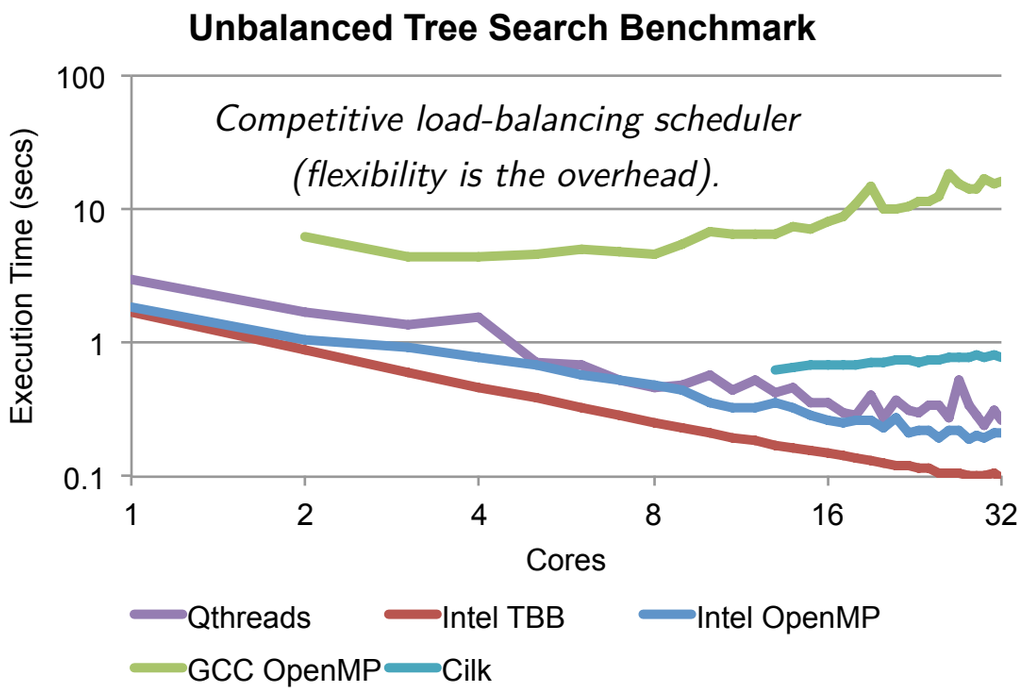
Message rates for small messages match MPICH2 performance under MPI-like conditions, and can even beat it for UPC-like conditions.

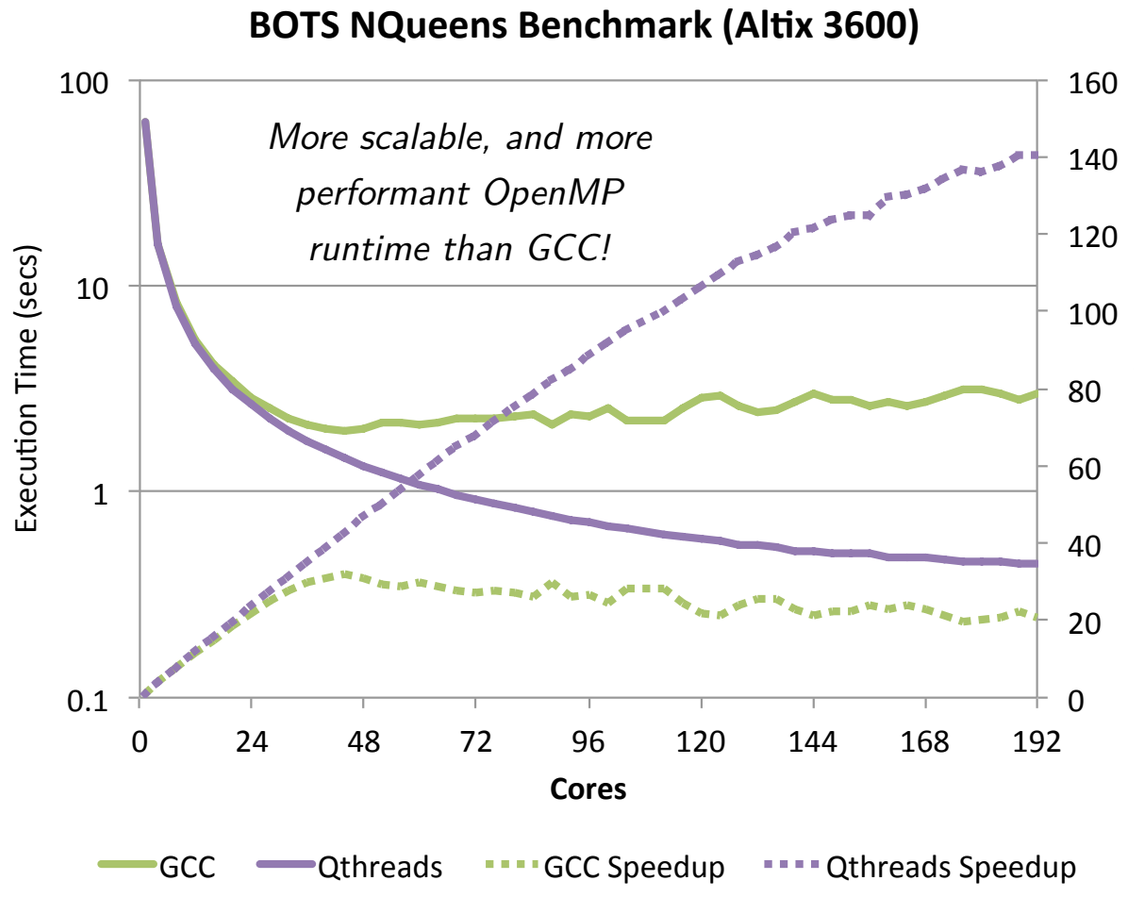
Qthreads Lightweight Threading

- **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 Qthreads R&D**
  - Task team and eureka support
  - Efficient, flexible collective operations
  - Remote task launch



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





The Lime in the Coconut

- **Research slowed by lack of applications**
  - Apps need programming environment vision
  - ...and an API, if possible
- **Experiment-driven SPI (Scalable Programming Interface) design-points:**
  - Environmental description (local vs global topology)
  - Naming needs (GIDs vs handles vs ?)
  - How much detail is necessary from the application to specify performant data/work movement?
  - How much detail from the runtime is necessary to enable specification of performant data/work movement?
  - What synchronization semantics are needed and/or useful? (Futures vs mutexes vs FEBs vs ?)
- **Use both experimental results and application programming effort to guide API development**

Current Status

- **Download Today!**
  - Kitten: http://code.google.com/p/kitten/
  - Portals4: http://code.google.com/p/portal4/
  - Qthreads: http://code.google.com/p/qthreads/

- **Stacked components work**
  - Portals4 on Kitten (with InfiniBand)
  - Qthreads on Kitten
  - Qthreads on Portals4

- **Multinode Threading Environment**
  - Remote spawn/sync
  - Multinode UTS, without work-stealing

Thank You!