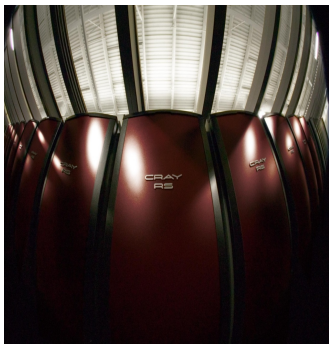
Exceptional service in the national interest

Sandia National Laboratories





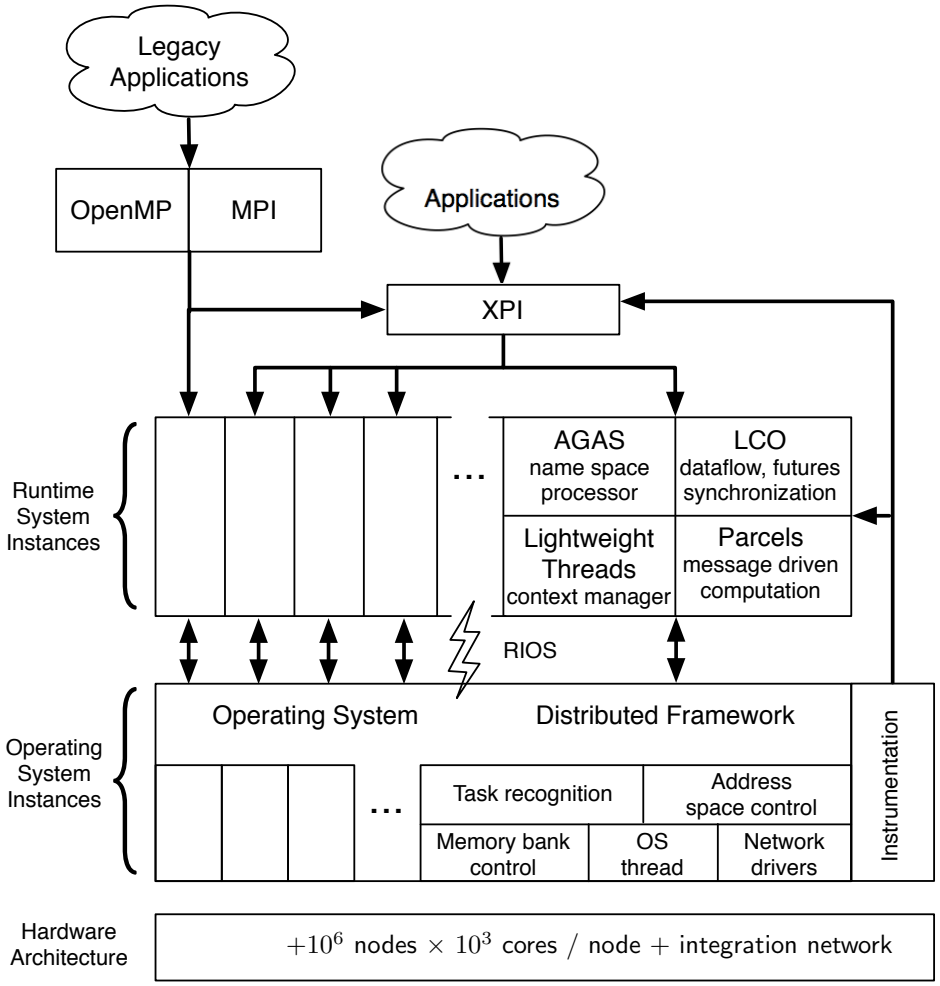

XPRESS Project Update

Ron Brightwell, Technical Manager
Scalable System Software Department

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.

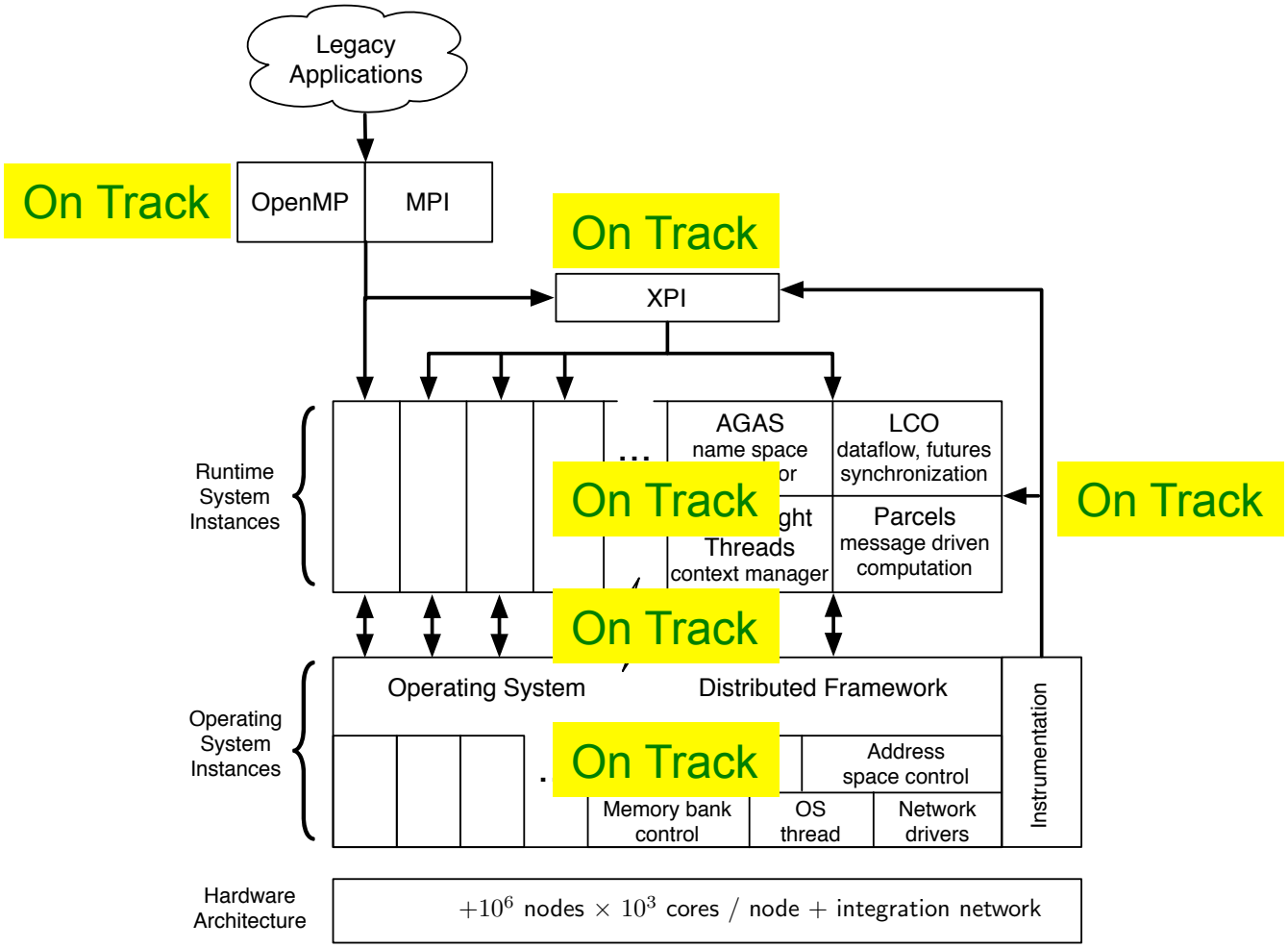
Vision - OpenX Software Architecture



Use Case Slide

- See Alice Koniges' talk at 3:15p

Status of Project – Overlaid With Vision



Compare Your New Technologies With Current State-of-the-Art – Particularly MPI+OpenMP

MPI+OpenMP

- Communicating sequential processes execution model across nodes
- Shared memory threading within nodes
- Hybrid (distributed + shared memory) programming model
- Largely static allocation and management of resources

HPX

- ParalleX execution model
- Adaptive Global Address Space programming model
- Dynamic allocation and management of resources

Questions?

Backup

Project Goal

- R&D OpenX software architecture for exascale computing
- Four thrust areas
  - HPX runtime system based on the ParalleX execution model that supports dynamic resource management and task scheduling
  - LXK lightweight operating system based on the Kitten OS that exposes critical resources to HPX runtime system
  - Runtime Interface to OS (RIOS) definition and description of the interaction between HPX and LXK
  - Support for legacy MPI and OpenMP codes within OpenX

Uniqueness of XPRESS

- Guided by ParalleX holistic parallel execution model
- Open-X software spans entire stack
  - Lightweight OS designed for dynamic adaptive runtime systems (LXK)
  - Runtime Interface to the OS (RIOS)
    - Enables efficient interaction between runtime and OS for event-driven adaptivity
  - Dynamic, adaptive runtime system (HPX)
  - Integration of performance instrumentation and control throughout software stack (APEX, RCR)
  - Low-level application programming interface (XPI)
  - Support for native applications and legacy applications
- Targets full system capability

Year 2 Activities

Major Task	Details
ParalleX Execution Model	• Finish description of ParalleX document • Incorporate locality management and introspection
HPX-3	• Continue to update, develop, and improve HPX-3 • Implement HPX-3 with XPI • Develop the HPX-4 threading system • Continue developing HPX process • Adopt IU Parcel port • Work with RENCI to integrate power management into HPX • Work with OU to make HPX APEX aware • Work with SNL to implement HPX on LXK • Work with UH to provide legacy support
HPX-4	• Implement Parcels • Implement Local Control Objects • Implement GAS • Implement interface to LSU's thread package • Implement interface via RIOS to LXK
LXK	• Deploy LXK virtual cluster environment • Demonstrate HPX-4 on LXK • Integrate instrumentation capability into LXK
RIOS	• Refine RIOS specification • Protocol specification • Use specification to guide design of interface LXK and HPX-4

Year 2 Activities (con'td)

Major Task	Details
APEX	• Implement more robust version of APEX with HPX-4 • Create a performance data access API for evaluating performance metrics mapped for lower-level measurements during execution to allow for performance data introspection • Develop interfaces for XPI to specify performance requirements and create performance data views • Define performance introspection requirements and architecture
Introspection	• Develop and integrate contention/energy models into HPX and APEX • Improve and increase data source for models by integrating into LXK • Finish design and start implementing multi-node data collection and contention/energy models
Legacy Migration	• Implement in OpenUH for supporting OpenMP 3.1 using HPX runtime • Evaluate performance of HPX-OpenMP and OpenUH-OpenMP using benchmarks and applications • Enhance performance of OpenACC complier and OpenMP 4.0 accelerator support in OpenUH • Data-driven computation model across cluster nodes • Develop MPI collective communication operations based on HPX/XPI operations • Finalize runtime support for MPI libraries in HPX
Applications (Year 2 start)	• XPI mini-apps • Fusion energy apps • Climate science and nuclear energy apps • Collaboration with Co-Design Centers • Collaboration with other X-Stack projects
Software Integration	• Develop plan for integrating software components • Deploy and support application development and evaluation testbed

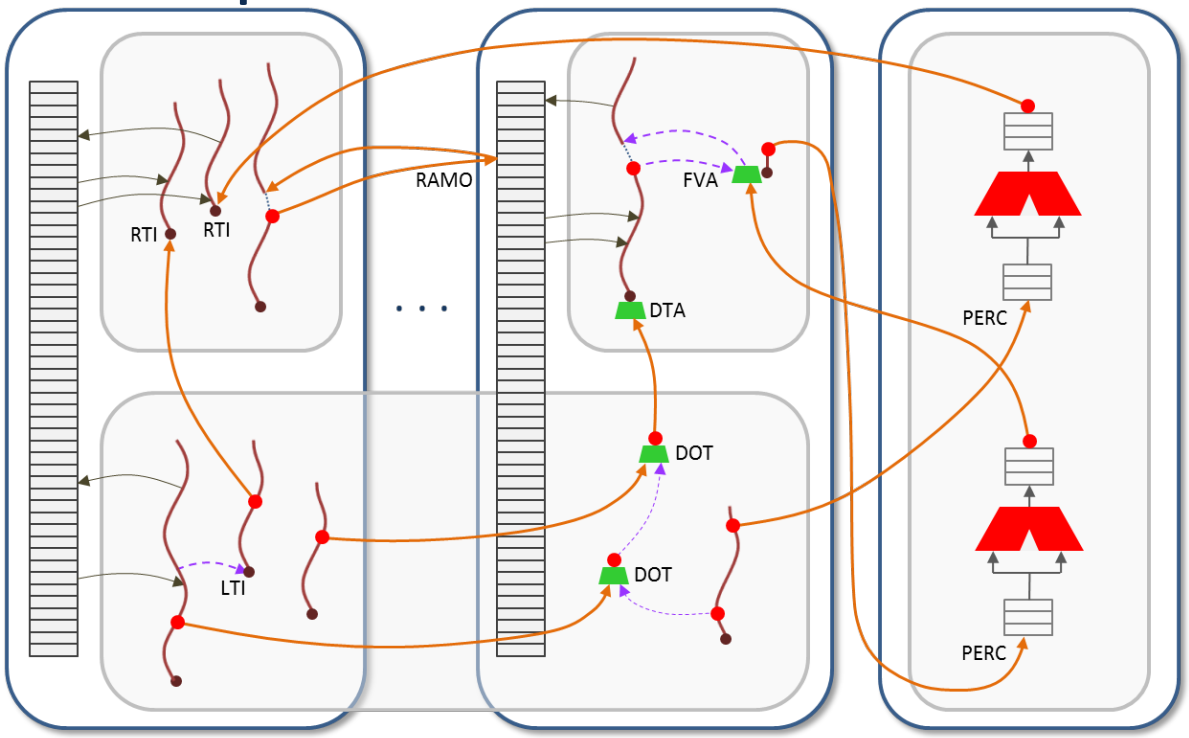
Status

- Met all Year-1 milestones and deliverables
- On track to meet all Year-2 milestones and deliverables
- Several key documents created and/or updated
  - Significant progress on RIOS specification
  - XPI Draft Specification version 1.0
  - Updated draft of ParalleX Execution Model specification
- Significant progress on several software components
- Several cross-institutional meetings and interactions
- Application work underway

Key Strategic Points/Accomplishments

- On schedule (SOW) with research, development milestones and deliverables
- Runtime System Software Prototypes
  - Working cross-nodes parcels (message-driven) transport layer
  - GAS cross-node framework
  - Multi-threads scheduling package
- XPI specification
- XPI first function implementation with micro-apps
- Software architecture: module functionality and interfaces
- Enhanced execution model description
- Interrelationship with lightweight kernel OS
- Experimental evaluation for validation and performance
- Relevancy/engagement with other DOE Programs
  - Co-design, Execution models, OS/R, PSAAP2

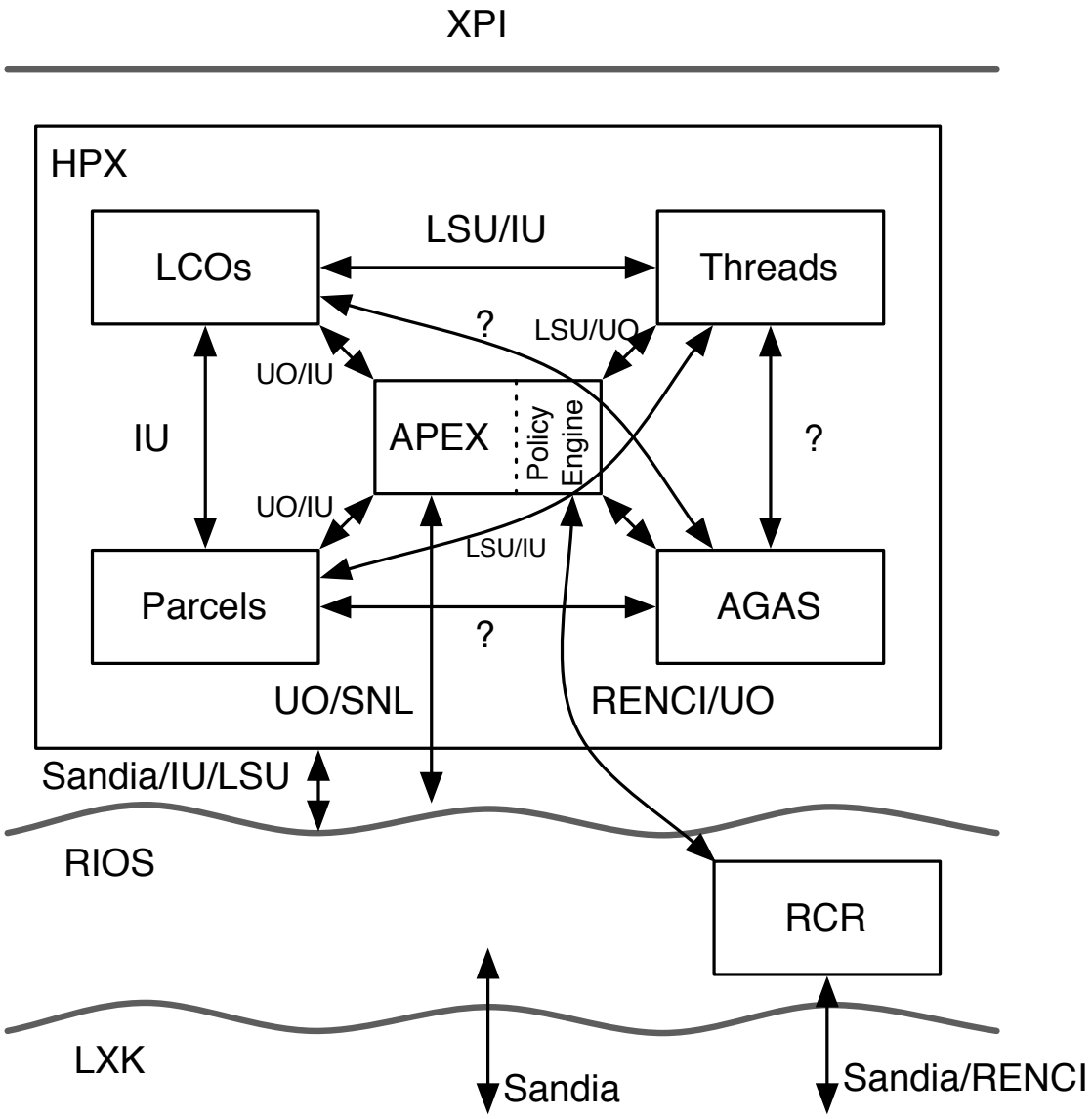
Semantic Components of ParalleX



Locality
Process
Local Memory
LCO
Accelerator

Thread
Suspended Thread
Local Memory Access
AGAS Address Lookup
Local Action
Parcel

LTI: local thread instantiation
RTI: remote thread instantiation
RAMO: remote atomic memory operation
DTA: depleted thread activation
DOT: dataflow object trigger
FVA: future value access
PERC: percolation


XPI
HPX
LCOs
LSU/IU
Threads
?
LSU/UO
UO/IU
IU
APEX
Policy Engine
?
UO/IU
LSU/IU
Parcels
AGAS
?
UO/SNL
RENCI/UO
Sandia/IU/LSU
RIOS
RCR
LXK
Sandia
Sandia/RENCI

HPX Progress

- Overall architecture that integrates:
  - Threads
  - Local Control Objects (LCOs)
  - Performance Measurement
  - Parcels
  - Global Address Space (GAS)
- The network layer has been updated
  - Made it reentrant
  - Implemented a transport channel for large messages
- We implemented remote thread spawns & future setting over the network
- We created a new type of LCO called *versioned gates* that control groups of futures

HPX - Threading Subsystem

- Fully refactored code base to improve performance, modularity, and maintainability
  - Performance improved by 20-30%: overall thread (scheduling) overheads down to 900 ns per thread (create, schedule, execute, and delete thread)
  - Different schedulers are available (LIFO, FIFO, with or without thread priority, etc.)
  - Can be switched at runtime, easily extensible by user policies
- Worked with RENCI to integrate power and contention management into schedulers
- Worked with UO to instrument threading subsystem and integrate wit APEX

HPX - Instrumentation

- HPX exposes a large set of Performance Counters
  - Uniform framework for exposing arbitrary system information
  - Examples: AGAS, threading/scheduling, parcel transport
  - Allows for runtime introspection as a precondition for runtime dynamic resource management
- APEX uses those to expose diverse set of information to tools like TAU
- HPX hooks into commercial tools like Intel® Amplifier and Intel® Inspector
  - Debugging support: memory checking, threading (deadlock) analysis, race condition checks, etc.

Impact

- There is a path forward for loosely coupled integration of software components across the entire XPRESS project team
- HPX-4 runs in distributed environments now
  - Up to 512 cores (16x32) on InfiniBand
  - Up to 64 cores (4x16) on Cray Gemini
- Network performance has increased by an order of magnitude
- Applications have more data structures available to them using futures
  - That suit the needs of real-world scientific applications

Parcels

- Software component for cross-nodes transport layer
  - Transport layers:
    - Photon (infiniband and Gemini)
      - Experiments with unified runtime/network software architecture
    - Portals-4
      - Final transport for commonality, portability, stability
    - TCP/IP
  - Platforms
    - X86 cluster (Cutter)
    - Cray XE6/XK7 (Big Red II)
- Message-driven computation
  - Instantiates threads on remote localities (nodes)
- Global data access
  - Able to directly read and write data from all localities
- Current support for PGAS
  - Temporary static implementation
  - Will be replaced with AGAS for dynamic migration of virtual data
  - Distinction is transparent to user
- Performance studies, measurements, tradeoffs underway

XPI Specification (r313)

- Currently supported capabilities
  - Processes
  - Parcels
  - Threads
  - AGAS
  - LCOs
- Current and Future development
  - Locality management
  - Integrated error handling
  - Customization (thread priorities, distribution, etc)
  - Introspection (load balancing, locality management, etc)
  - IO
- Public Specification is in Beta
- Implement XPI - Released mid-March: HPXPI 0.1
  - Almost complete, fully open source implementation of XPI specification on top of HPX
  - Benefits from high performance characteristics of HPX

libXPI Native Implementation

- Implements (targets HPX)
  - Parcels
  - Lightweight Threading (with local task sharing)
  - PGAS (with block cyclic distribution)
  - LCOs (futures, semaphores, and gates)
  - Networking (SMP, MPI, Portals, Photon)
- In development
  - Process termination detection
  - Continuation stacks
  - Lazy thread stack binding
  - AGAS
  - Test suite
- Initial release this spring
  - Concurrent with XPI Specification 1.0 gold release

LXK/RIOS Research Goals

- XPRESS aims to increase synergy of compute node OS kernel and user-level runtime systems
  - Today: Runtime must work around host OS, assume worst case
  - Vision: Runtime cooperates with host OS, delegated more control
- Key RIOS drivers
  - Runtime needs guarantees about resource ownership and behavior
  - OS needs way to shift resources between multiple runtimes
  - Two-way interfaces needed for key resources
    - Runtime tells OS what it needs, OS tells runtime what it gets
    - OS remembers original request, notifies runtime if more resources become available. Notifies runtime of resources need to be reclaimed.
  - Event-based protocol to notify of dynamic events (e.g., power state change, transient error)
- LXK = Kitten + RIOS

Areas Covered by RIOS

- Legacy support services
- Job management
- Thread management
- System topology and locality
- Introspection
- Memory management
- Network interface
- File I/O
- Energy management

RIOS Current Status

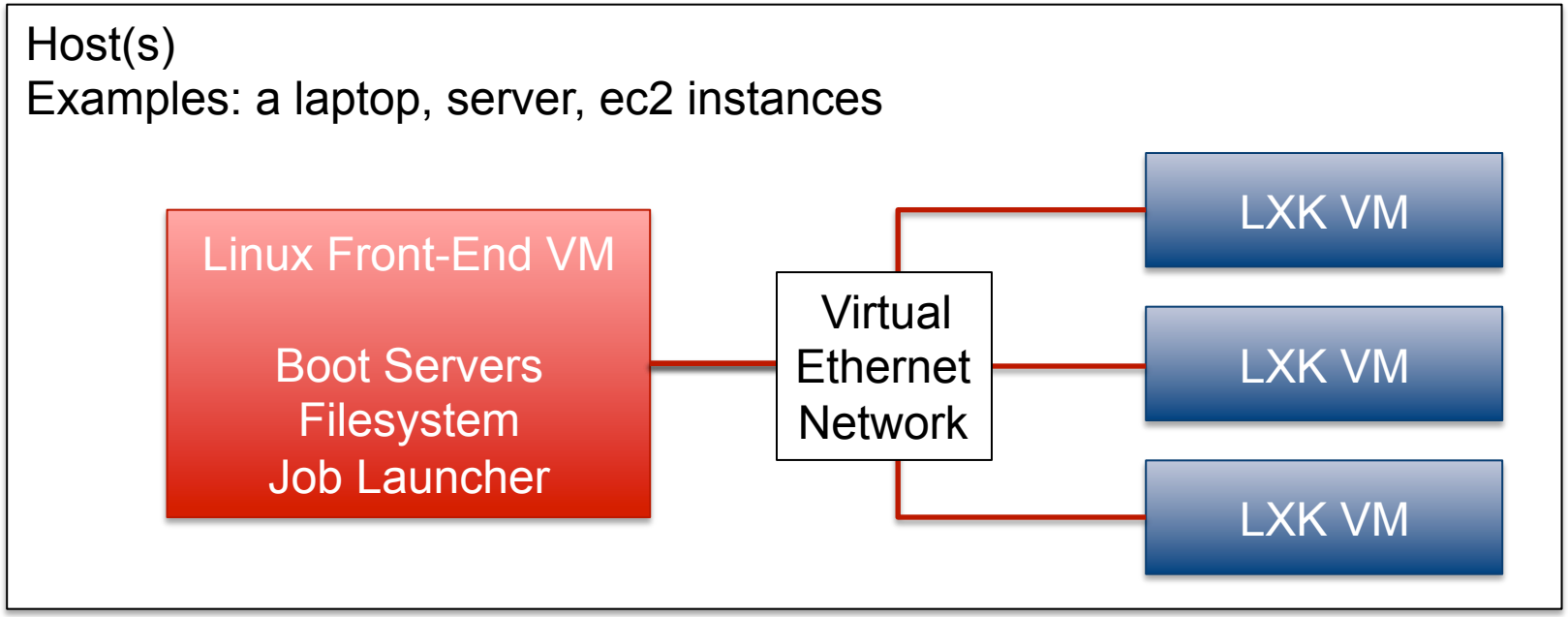
- Current draft outlines requirements and gives prose description of interfaces and protocols
- FY14 goal to formalize into a real specification that somebody could implement
- Recent focus on thread management and memory mgmt.
  - Created several working documents to drive discussions
    - “Thread Management in Kitten”, “Memory Management in Kitten”
    - IU write ups describing ParalleX process model and AGAS
  - LSU providing input on HPX-3 thread management requirements

Porting / Re-Architecting RCR for LXK

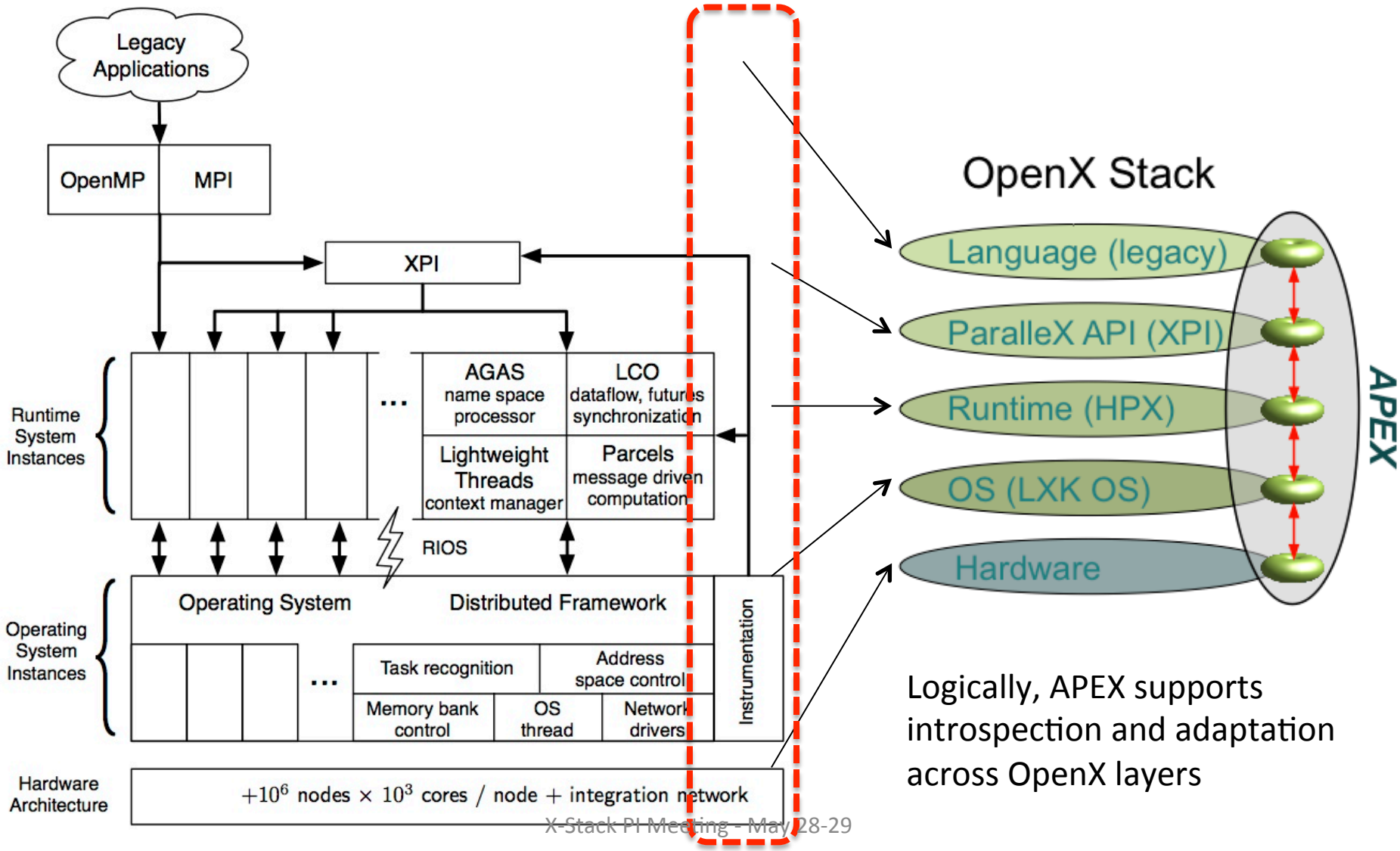
- Current RCR architecture (pre-XPRESS)
  - User-level daemon with root privilege
  - Makes high-frequency system calls to Linux (polling)
  - Relatively high overhead (~20% of a Xeon core)
- Goal: Dramatically reduce overhead, eliminated daemon
- Plan: Split RCR into two pieces
  - 1) LXK internal module, timer interrupt used to poll counters, include simple logic to calculate derived metrics / statistics
  - 2) User-level library, implements interfaces for configuring LXK module, reading collected data via shared memory mapping with kernel, interface with RIOS/APEX
- Targeting end of April for initial implementation

Virtual LXK Development Environment

- Needed easier way to deploy LXK to collaborators
  - Use bundled set of virtual machines: 1 Linux, N LXK
  - Goal to provide functional development environment, reasonable performance
  - Initial development in FY13, continuing in FY14
    - Brian Kocoloski (U. Pittsburgh) – Portals4 networking and job launch
    - Jorge Cabrera (Florida Intl. U.) – I/O forwarding layer from Kitten to Linux



OpenX Architecture and APEX/RCR's Role

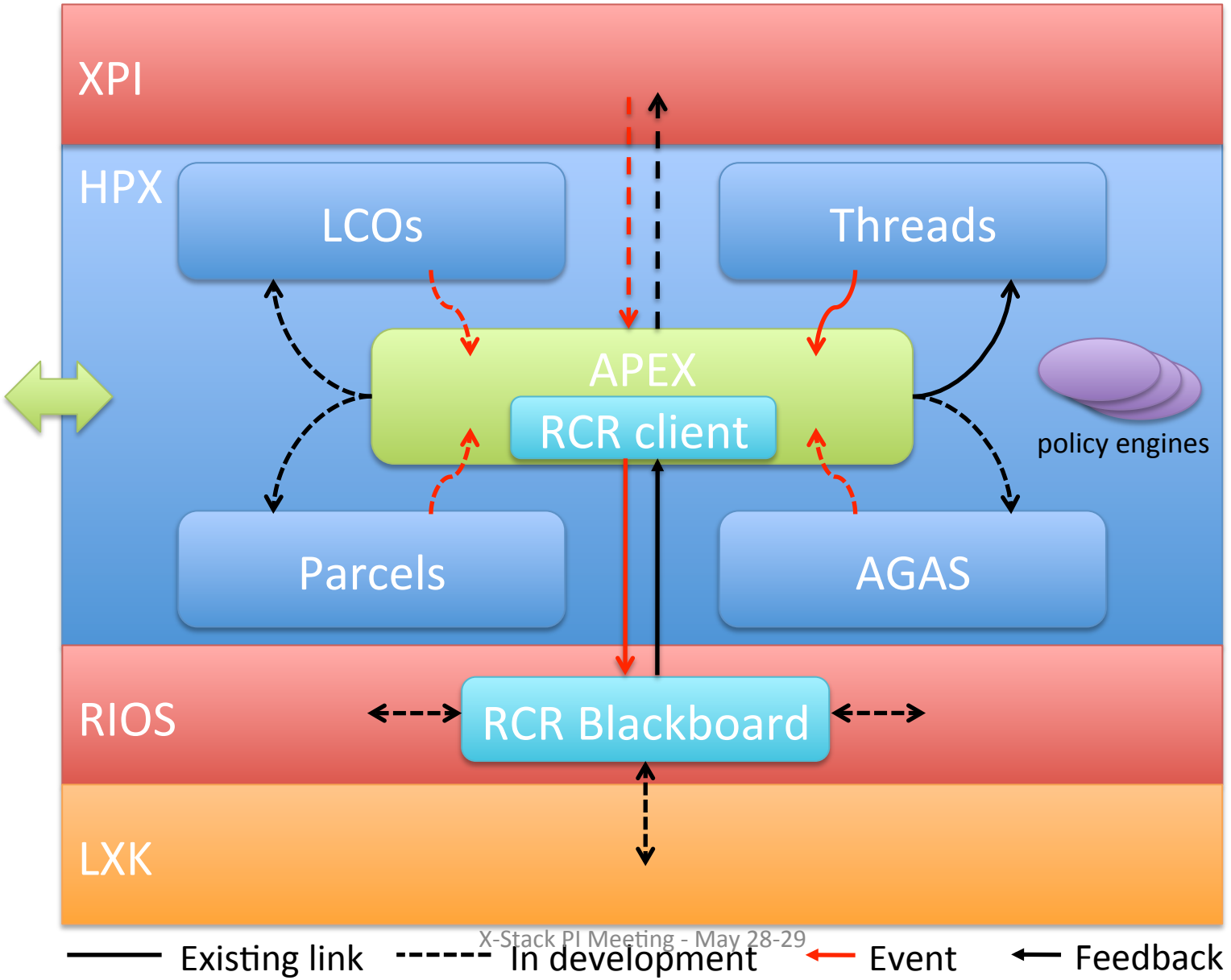


Logically, APEX supports introspection and adaptation across OpenX layers

Introspection Overview

- The XPRESS project provides the opportunity for all layers of the software stack to not just inter-operate, but also enable introspection between layers
- The APEX component encapsulates the mechanisms for introspection and for dynamic adaptation at runtime
- APEX is designed to maintain and/or observe the running performance state of all XSTACK layers:
  - Application level (e.g., XPI interface, events in HPX)
  - System level (e.g., RIOS interface, LXK, hardware)
- APEX provides an interface for:
  - Submitting state changes (observations to analyze)
    - Observable HPX state changes
    - RCRblackboard to observe OS/system changes
  - Registering adaptive policy criteria with a Policy Engine
  - Produce events to change performance
    - Events to HPX
    - Use RIOS to inform OS

XPRESS Interfaces : APEX interactions



APEX architecture

- Two key components provide APEX infrastructure
- Event-driven execution model
  - Thread Manager currently supported
    - thread registration, counter sampling, activity start/stop events
  - Plan to add LCOs, Parcels, AGAS, XPI events
- Periodic interruption / interrogation of the overall system state by an out-of-band thread
  - Process-wide view of thread activity, system health
- Global APEX operation provided HPX
  - Enables APEX to develop monitoring, in situ analytics, and policy responses that will run across system

RCR Integration

- Provides “whole node” view of performance state
- RCR has the same lifetime as the Operating System, most of APEX is associated with an application’s lifetime
- RCRblackboard provides shared memory region for communicating across process boundaries
- LXK monitors OS resources, hardware resources, power consumption, and so on
  - Reads, writes to/from RCRblackboard
- The APEX RCR Client runs in user space, monitors OS & HW state, reports HPX, XPI and Application state (through API)
- Policy Engine determines when to respond to state changes
  - As specified in policy definitions

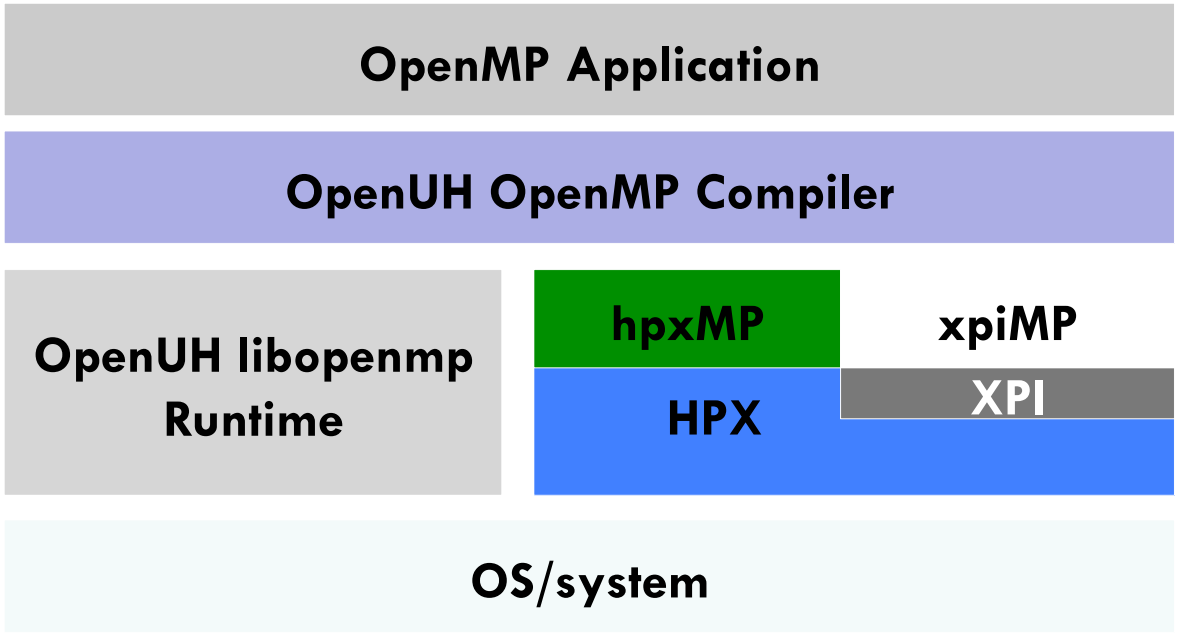
Policy Engine

- The Policy Engine subcomponent will also be one of the internal event listeners registered in APEX
- Policy is set in APEX by registering test and action functions with the engine
- Either periodically or on an event, the Policy Engine executes the test functions, and if they return true the action function is executed
- Any XSTACK layer component will have the capacity to respond to the performance status of any other layer as necessary

XPI, Application Interface Support

- XPI will also report state changes to APEX, respond to policy actions when appropriate
- Application developers may opt to define policy
- Need to specify user-level API for defining, registering application policy

Legacy Migration:
Replacing OpenMP runtime with HPX and XPI



- Next steps
  - OpenMP runtime on top of XPI
  - OpenMP interoperability with HPX and other runtime

Support for Open MPI on HPX/XPI

- Work focuses on taking advantage of HPX features in Open MPI Runtime
  - Application would be able to utilize features of MPI and HPX simultaneously
- Two options being evaluated:
  - Replace entire RTE of Open MPI with a slim layer based on HPX -> daemonless approach
  - Add new components in ORTE to support HPX
- Operational prototype expected by the end of the summer

Summary

- On schedule to meet research goals
- Working prototypes of all major components
- Dramatically improved efficiency and performance for several software components
- Demonstrated distributed message-driven computation
- Kitten/LXK on track to support HPX-4 by end of Year-2
- Preparing to do Sandia-led software integration
- Applications work underway
- Important pathfinding discoveries along the way
  - Performance limitations of model, runtime system, hardware
  - Learning how OS and runtime serve each other

http://xstack.sandia.gov/xpress