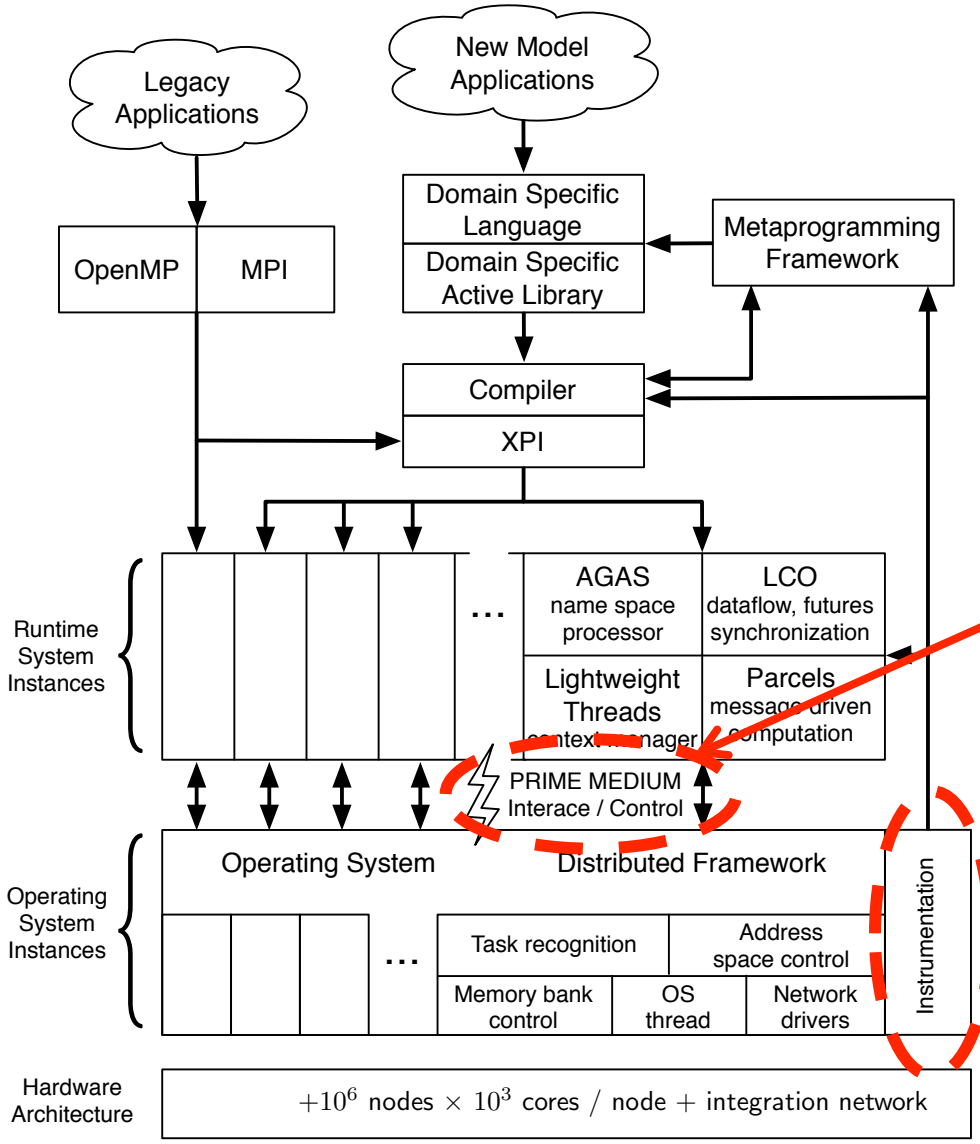
APEX – Design for XPRESS

Performance Research Lab
University of Oregon
Allen Malony, Sameer Shende, Kevin Huck
{malony, sameer, khuck} @cs.uoregon.edu

UNIVERSITY OF OREGON

Open-X Model



Aside: What's the plan for the Prime Medium?

Apex + RCRToolkit

APEX Overview

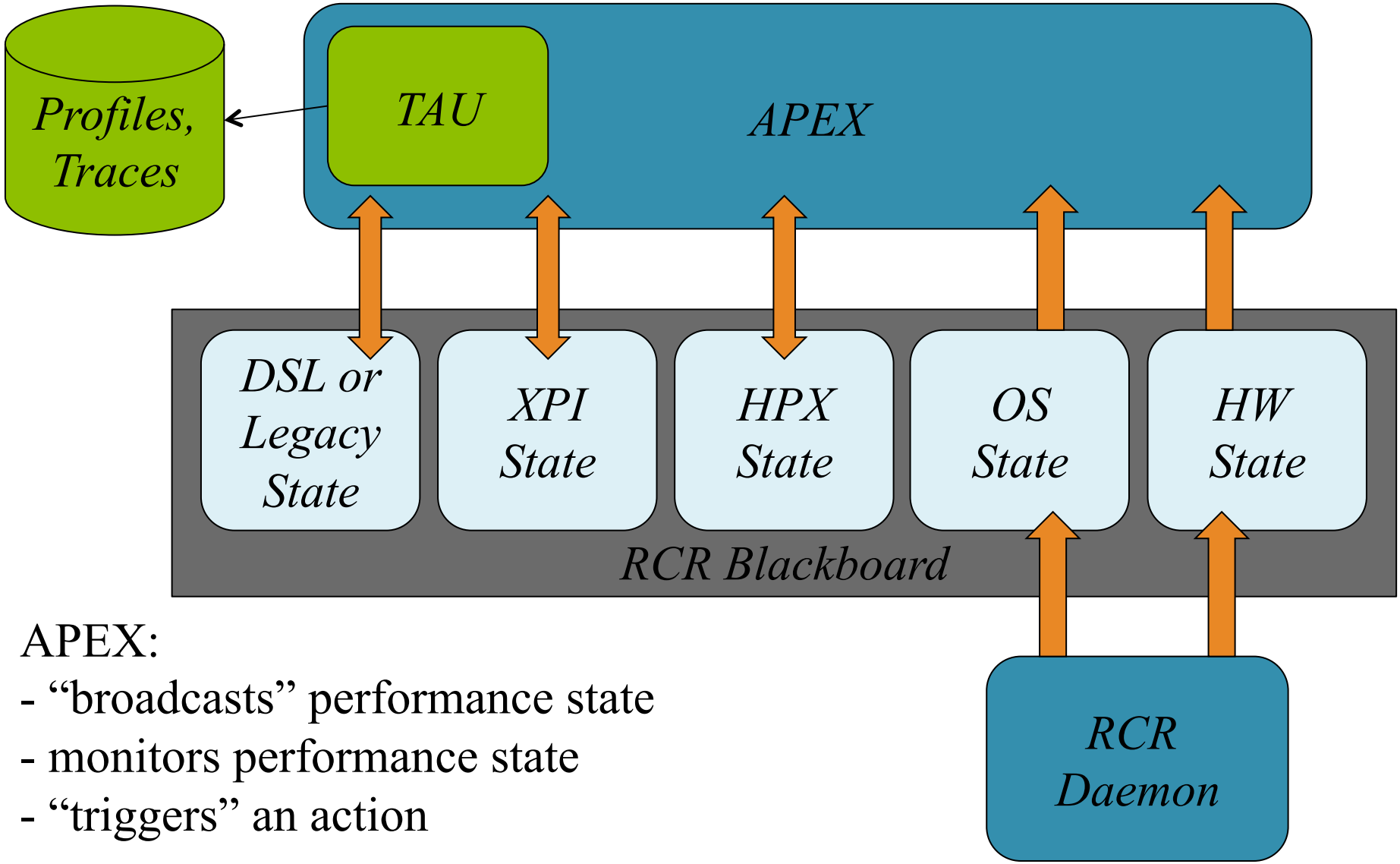
APEX

- Integrated first and third person model
- No pre-defined context (RCR "state space")
- Directly observed layers reported to RCRBlackboard
- Indirectly observed layers compared to thresholds
  - Violations "trigger" placement of action request in RCRBb
- Threshold processing dynamically controllable
- Distributed aggregation of observations
  - Adjustable resolution based on thresholds

APEX Performance Model

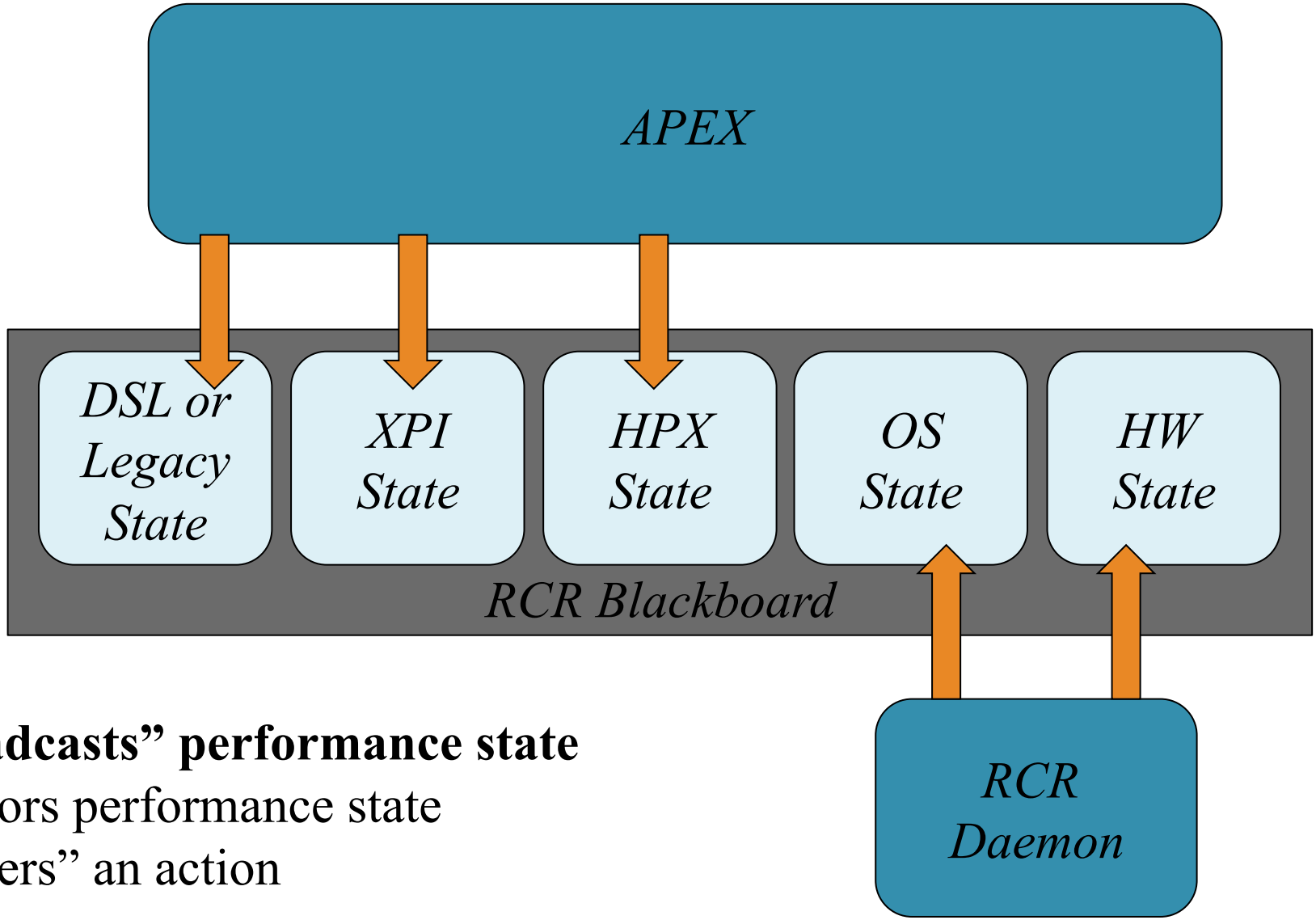
- Event driven model - "Observation with context"
- Event types
  - Instrumented measurement (explicit)
    - code insertion, interface wrappers, …
  - Observed value (counters)
    - Bytes read/written/transferred, thread state, …
  - Generic state transition
  - Program samples (sampling of PC) ?
- Threshold violations trigger an event in event queue

APEX: monitoring and writing to RCR



APEX:
- "broadcasts" performance state
- monitors performance state
- "triggers" an action

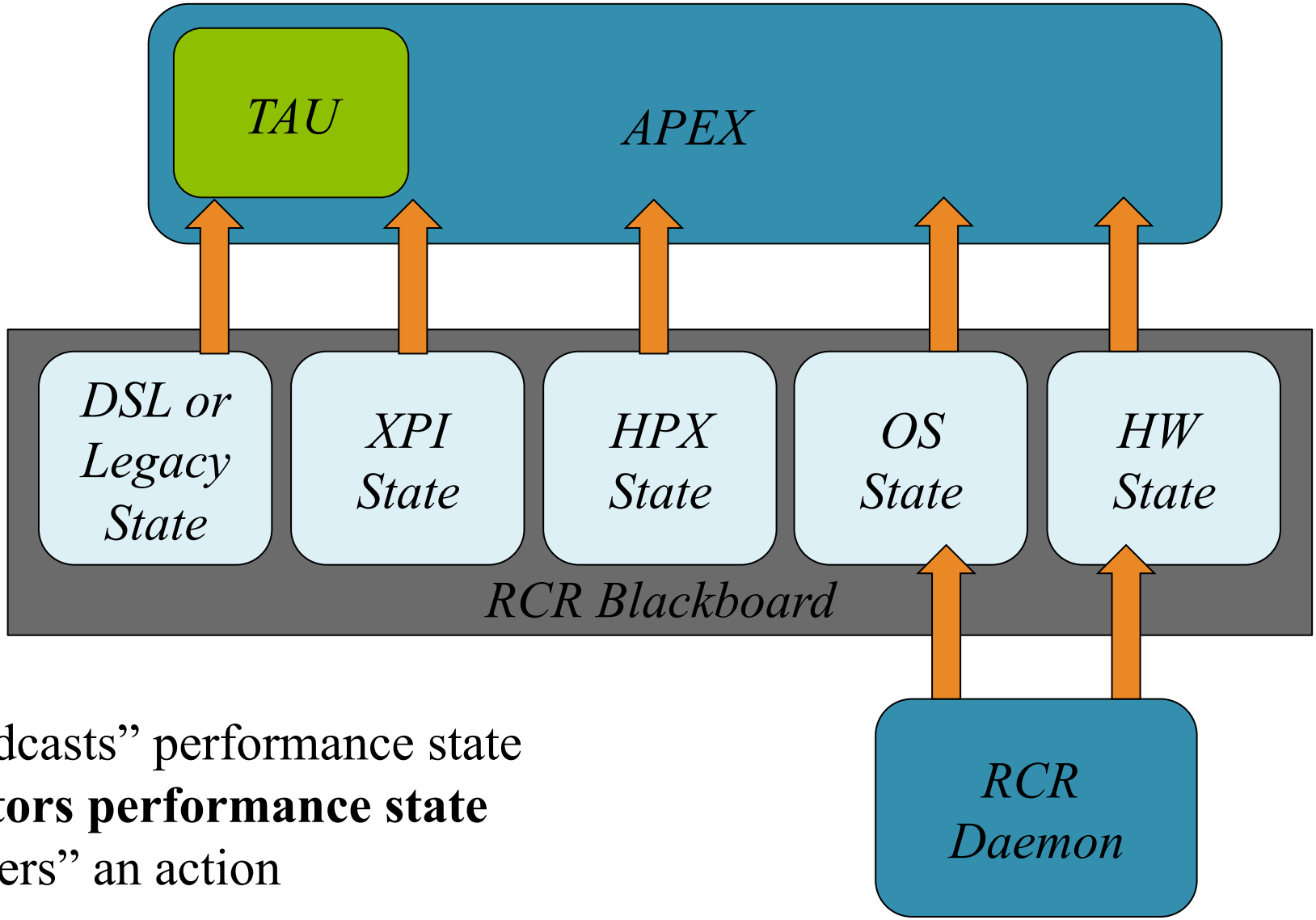
APEX Performance Events



APEX:

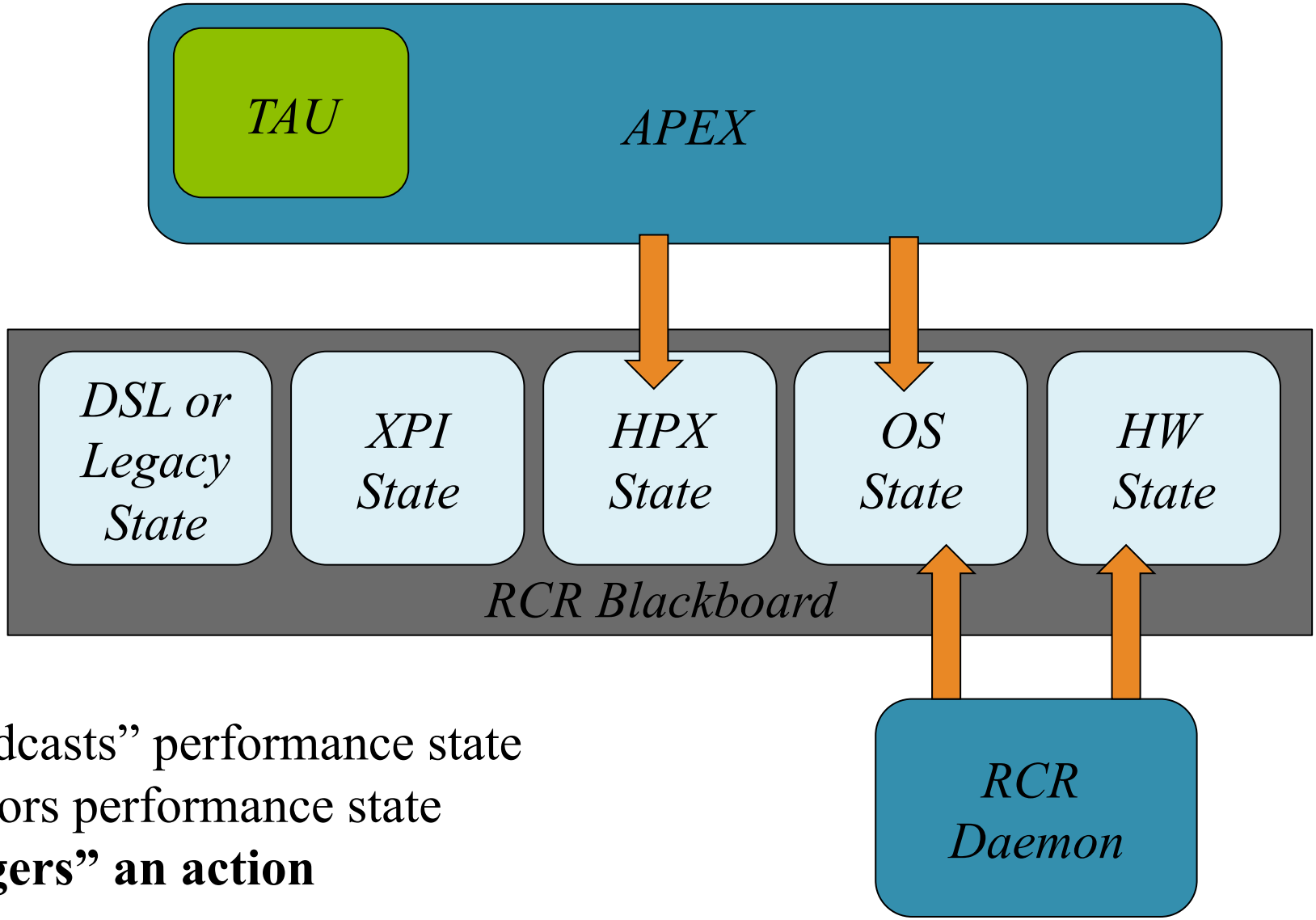
- **"broadcasts" performance state**
- monitors performance state
- "triggers" an action

APEX Performance Events



APEX:
- "broadcasts" performance state
- **monitors performance state**
- "triggers" an action

APEX Performance Events



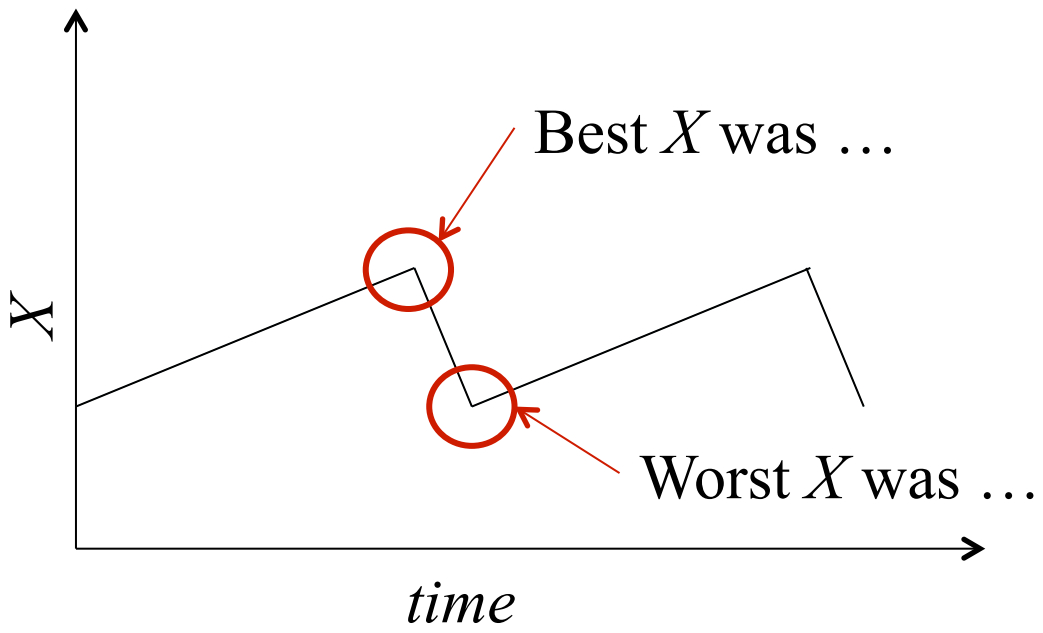
APEX:
- "broadcasts" performance state
- monitors performance state
- **"triggers" an action**

APEX and RCRToolkit

- APEX RCRToolkit client interface
- Any layer of the stack can examine data from other layers
- All layers know about anomalous states in other layers
  - OS layer knows about problems in the hardware
    - Contention, latency, failure, power drain…
  - Runtime layer knows about problems in the OS layer
    - Contention, latency, deadlock…
  - DSL layer knows about problems in the Runtime
    - Starvation, latency, overhead, contention…
- Apex provides interaction with RCRToolkit, decision logic for reporting anomalies
  - Report anomalies relevant to current layer

APEX profiler / logger / timeline generator

- Macroscopic view of performance
- Post-processed metrics (IPC, FLOPS, idle thread count, etc)
- Usual statistical suspects (mean, stddev)
- Mode(s), min(s), max(s) also include **full** context (where did it happen?)



APEX requirements

- Low overhead
- Reconfigurable at runtime
  - Change counter groups (HPX?)
  - Adjust resolution
  - Enable / disable observations (filtering, throttling)
- Application of performance thresholds to observation
- "Event driven" model with multiple broadcasters / listeners
  - Performance validator
  - RCRToolkit reporter / observer
  - APEX Profiler / logger / timeline generator

APEX concerns

- Event driven model implementation
- Broadcaster / listener complexity?
- Threading complexity (OS threads, HPX threads)
  - Suspended threads
- Lag between observation and RCR context capture (sliding window approach?)
- Processing overhead
- Sample targeting
  - HPX, XPI, DSL, MPI, etc. share a process space
- “Unknown unknowns”

APEX and HPX

- HPX "will track threads, queues, concurrency, remote operations, parcels, and memory management." HPX can be explicitly instrumented
- Key state changes
- Selected HPX API calls can be wrapped / instrumented
- AGAS performance properties
- Thread management states (idle/busy)
- Local Control Object (LCO) properties
- Parcel Transport properties
- Future synchronization
- "Event" triggers for runtime adjustment/optimization

APEX and Kitten/LXK

- LXK "OS will track system resource assignment, utilization, job contention, and overhead."
- Apply experience from KTAU research?
- OS Process performance observations
- OS Thread performance observations
- IO, Network, Memory performance observations
- Observe "Event" triggers for runtime adjustment/ optimization

APEX and Legacy Codes

- ParalleX, “DSLs and legacy codes will allow language-level performance semantics to be measured.”
- OpenMP, MPI interposition library support
  - PMPI interface
  - OpenMP Collector API, GOMP wrapper
- XPI interface can be wrapped
- Provides “top-down” performance context

APEX and DSL

- ParalleX, “DSLs and legacy codes will allow language-level performance semantics to be measured.”
- DSL compiler instrumentation support
- XPI interface can be wrapped
- Provides “top-down” performance context

Questions going forward

- Distributed APEX/RCRBlackboard
  - Inter-node introspection
  - Adjustable thresholds
  - Adjustable resolution
  - Client interface?
- Distinction between HPX user threads and OS threads
- Support for HPX suspensions, resumptions
- Others…

Challenges

- Design of APEX will enable closed-loop performance optimization for the ParalleX execution model
  - Performance portability through dynamic adaptivity
- Creating (application-level) performance abstractions
  - Defines/constrains what is (of importance) to be observed
  - Represent performance model target for analysis/control
- Building efficient mechanisms for on-the-fly analysis
  - Performance derived metrics (based on context)
  - Asynchronous operation
- Context creation and tracking
- Factors concerning effectiveness
  - Overhead of measurement
  - Cost of analysis
  - Latency of feedback (especially cross-machine)