

Runtime Roadmap

Chair: Kathy Yelick

Roadmap focus points

- Bounding the scope of Runtime
- Convergence
- Research roadmap
- Industry Integration – buy-in and help

Bounding Scope: What makes sense organizationally?

- Programming environment runtime
 - Application Facing Runtime
 - Needs to include communication
 - May need to co-exist and interact (progress, etc.)
- System Facing Runtime
 - OS, Long-lived, Workflow tools, in-situ
 - Provides protection between application
 - Hypervisors, virtualization, dockers, etc. complicates this discussion
 - Node OS, System OS, Enclave OS
 - Needs to arbitrate and ultimately own resources

- There shouldn't be a hard boundary
- But we need to understand what we're talking about

What is the process for convergence?

- Can we identify the “motifs” of resource usage?
 - Static, semi-dynamic, fully-dynamic?
 - Progress strategies
 - Embeddable into complex distributed systems:
Relevant to HPC vs. commercial world?

OS/R Grand Challenges

- Can your runtime work with multiple PM/Es?
- Can different runtimes (or components) use
 - Shared hardware resources
 - Shared data from multiple applications
- Different usage models (application patterns)
- Can your runtime be library-ized?
 - Don't get to own main()
- How does data move between runtimes?

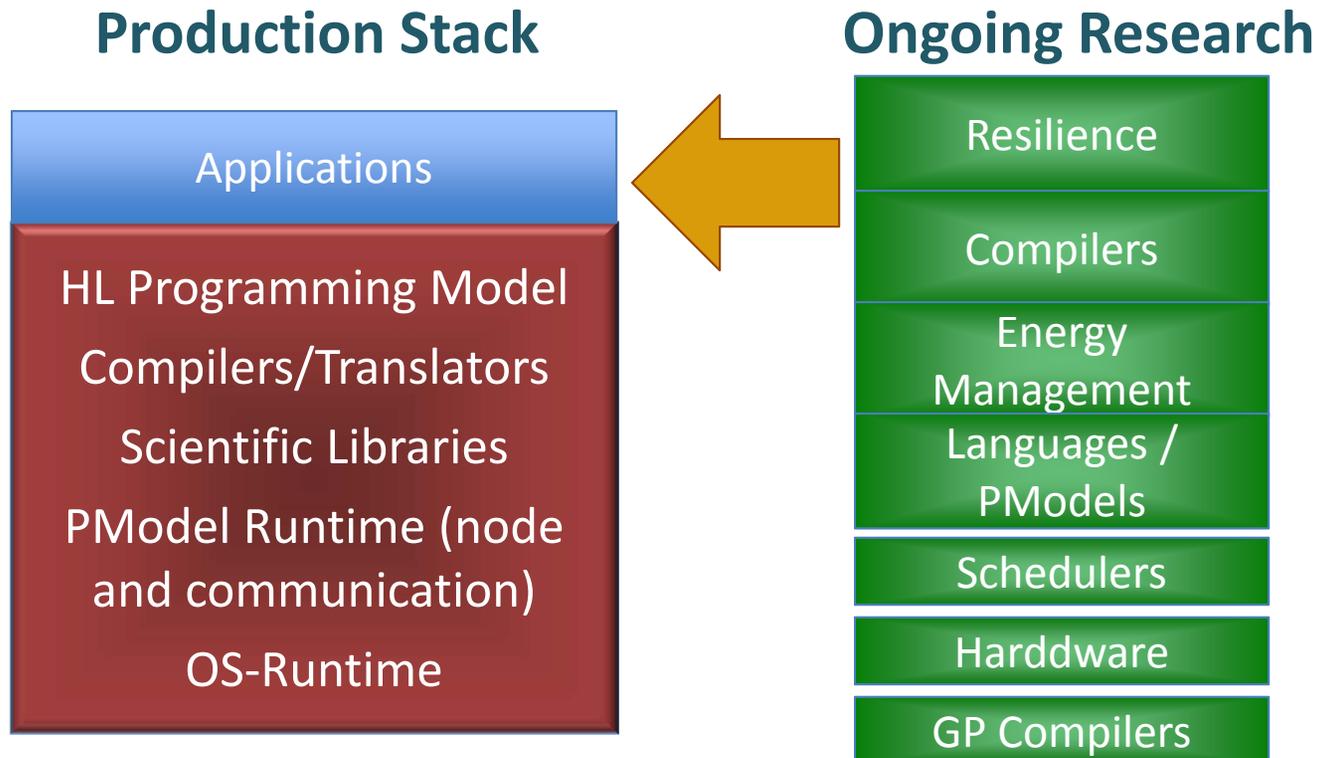
Convergence

- Need to map efforts to scoping – take inventory
 - What are the unique aspects of each
 - Is it application / hardware specific?
- How do we converge efforts and leverage
 - Need to start by agreeing on a hardware arbitration interface
 - Key to adoption (interoperation with existing runtimes)
 - Converge on a few runtimes that interoperate
 - Does this constrain solutions too much?
 - It would be nice if you could build a new runtime?
 - At what level do they need to interoperate (space / time sharing)?
- Does a composable service model address this?
 - Still need to define layers and interfaces

How to get interoperability?

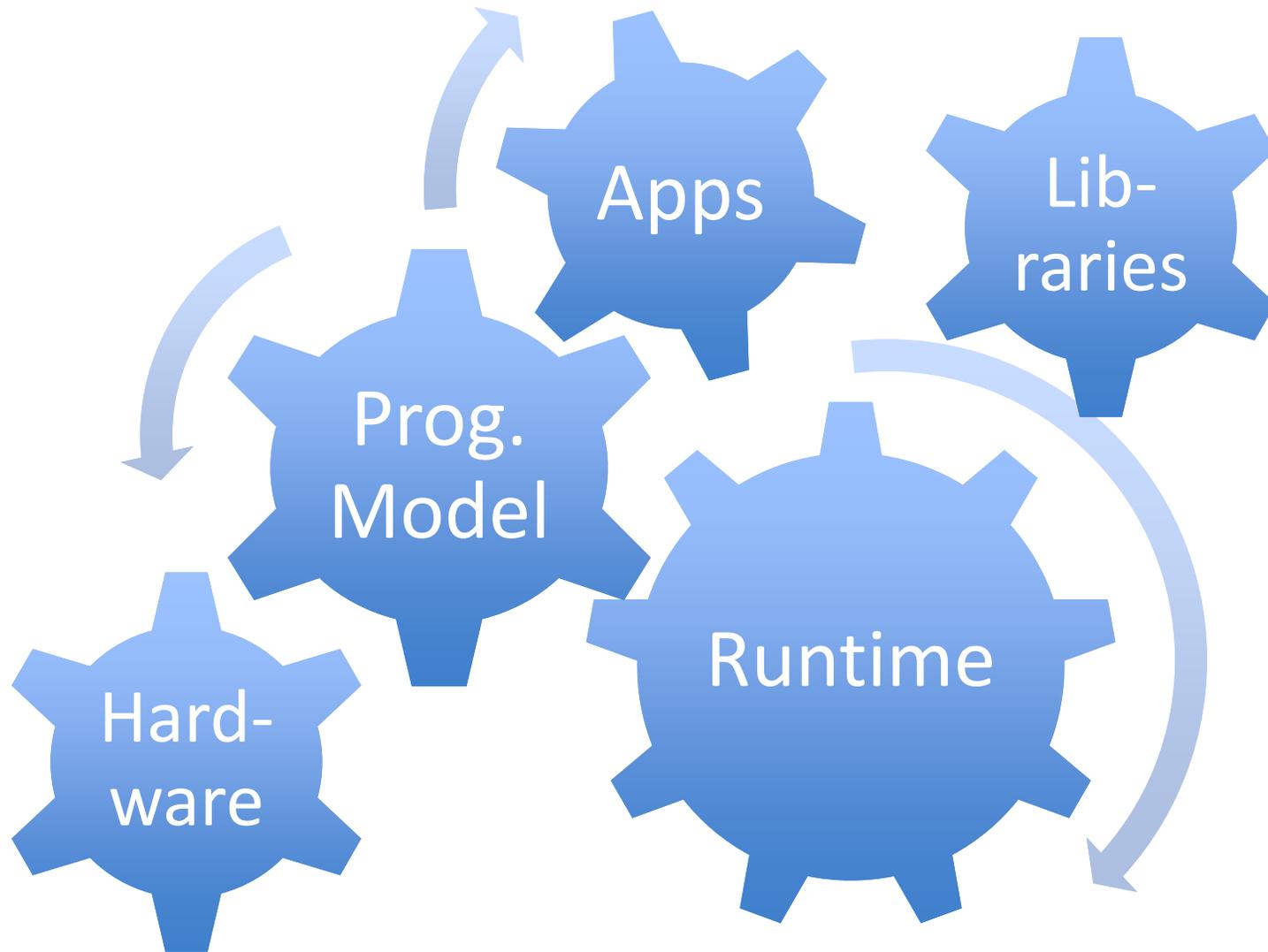
- Need enough experience to define APIs
 - Backplane in Argo / Hobbes is good example
 - Everyone wants to understand underlying hardware (locality / processing)
 - Interconnects are harder (how to virtualize endpoints)
 - New types of memory are not well understood yet
 - Types of user-level threads need to be understood
- Redundancy will become clear from process
- Get narrow communities together to understand their similarities and differences

How to Transition from Research to Development



- New programming constructs and implementation techniques transition from research to production based on:
 - Demonstrated application need and feasibility (performance,...)
- Research provides risk tolerance and upside potential

How to Stage Research and Development?



Industry Integration – buy-in and help

- How do we integrate with other ECI efforts?
- How do we integrate / leverage with industry?
- Would like to separate:
 - Are there open research questions?
 - Who will build these

Industry interactions

- Industry needs to be involved (as in co-design)
 - Companies vary on willingness to sharing ideas and collaborators
 - What role do we have in the OS?
 - Influence industry
 - What can we leverage
- Phases
 - Research phase: work with vendor to understand what both industry / us are researching
 - Demonstrate: Would like to boot our own research OS as testbeds (requirements in RFPs)
 - Deployment and support: We require functionality developed in research on vendor-provided systems
- Resilience affects everything
- Timing of influence is important: silicon longer lead time than operating systems, etc.

Roadmapping

- Start with process to understand differences and similarities for narrow problem
 - Terminology for communication
- Identify opportunities for
 - Common interfaces; may have different ones in context
 - Common implementations; may not want to rebuild something that others have
- Need to do this without constraining innovation
- People need to come back with innovations and convince the rest of the community of value
 - Decisions on common APIs, etc. may need to be revisited

Age-Old Debate



We need to be more precise in our terminology and discussions

Overarching Questions

What do we mean by Runtime?

- There are two levels:
 - Within the application
 - Management of “executables” across the system
- There was also a distinction between:
 - User space
 - Privileged
- Focused on user space / within an application, but noted:
 - The interaction with the OS is clearly important
 - We need to think carefully virtualization as in commercial clouds data center
 - Had some discussions on issues related to storage and dataflow, which are external

How Dynamic?

- How dynamic should an exascale runtime be?
 - When are decisions made?
 - What information is used to make load balancing and scheduling decisions?
 - Examples: Before runtime, at job launch time, at discrete points mid-execution (which can be globally), continuously on-the-fly (which probably means locally or at least hierarchically)
- Is a dynamic runtime capable of delivering exascale performance?
 - Or does the cost overwhelm the benefits?
- Is a dynamic runtime required for exascale performance?
 - Would dynamic control yield a significant improvement in operational concurrency?
- Does a dynamic runtime make programming easier?

How much parallelism?

- How much parallelism should be exposed to the runtime?
 - A programmer may expose all parallelism available in the application, but have it throttled (chunked) by the programming model implementation
 - Sometimes (usually?) the amount of parallelism is data dependent, so cannot be determined statically
- Both extremes are probably impractical:
 - Static: runtimes are data size dependent already
 - All concurrency: at the level of individual instructions, without an static information, e.g., partitioning, this would overwhelm a runtime

Domain specificity

- Should the runtime be domain-specific?
 - What mechanisms should be domain-independent?
 - What mechanisms are domain-specific?
- How is domain knowledge passed to the runtime?
 - How to deal with load balancing for specific problems? Is it a generic DAG scheduling problem, or is it specialized to a class of DAGs?

Runtime Decomposition

- The runtime has to be well integrated, but some features may not exist:
 - On all hardware
 - In all applications
 - Or in all parts of the applications
- How can you make it composable?
 - What is the smallest runtime you can give me and can it be used for one task?
 - How to make it decomposable? E.g., Burst buffers

Storage System Interaction

- What is the relationship between programming model runtime and external data services?
 - How does the runtime interact with the storage subsystem?
- What runtime support is required for complex workflows?
- Should complex workflows be supported
 - On Exascale technology systems?
 - At full scale?
 - Is this a productivity issue for scientists?

Requirements
gathering

System Interaction

- What does other systems software expect from the runtime (fault detection, etc.)
- What does the runtime expect from the other system software?

Resilience

- What failures (if any) can the runtime system:
 - Hide from higher level software / programming?
 - Contain from other parts of the computation
 - What are techniques for hiding
- What failures (if any) should the runtime expose?
 - How should the runtime report failures?
- What failures (if any) should the runtime ignore?
- What is acceptable behavior?
 - E.g., Self-driving cars; Mars mission
- What are the most important types of failures and their likelihood? (input to runtimes)

Requirements
gathering

Constraints
gathering

Energy

- Should the runtime (which one) manage energy?
Should energy management be:
 - Hidden from higher level software / programming?
 - Contained from other parts of the computation
- How should the runtime report energy issues (DVS, etc.) to the application?
- Do application programmers or P-model implementers want to control energy?
- What are the knobs and performance payoffs?

Requirements
gathering

Constraints
gathering

Technical Analysis of Existing Work and Remaining Questions

Program Questions

- Why aren't we answering the questions within the context of the Xstack prototypes and other runtime research that was done?
- What remains from what was done?
- What do we agree and disagree on?
- How do the application drivers benefit (or not) from the runtime system
- Old approaches are not working, but what do you do? Runtime systems are a candidate.

What Are Next Steps?

- Need for analysis of common and distinct concepts; similarities and differences
- Need for case studies of applications
- What requirements are we trying to satisfy?
 - Is there a particular class of applications being addressed?
 - What can the applications people tell us about their apps?
- Need for studies of “emulated” or “projected” exascale systems
- Need for studies of runtimes on existing systems and emerging systems (e.g., scratchpad and NVRAM)

Requirements gathering

Constraints gathering

Tasking Behavior

- What is appropriate task granularity
 - Is this a useful knob?
 - Given a software implementation, what is the finest granularity that can be effective?
 - What techniques can be used to lower overhead of tasking?
- Task characteristics
 - Run-to-completion vs. preemptible
 - Are they allow to mutable external state ?
 - Is it legal to communicate?
 - Is it legal to synchronize?
- If tasks may be descheduled is it:
 - Based on time slicing (non-cooperative)
 - Only voluntarily for communication
 - For synchronization

What are the implications for the programming model and application?

What subsets / combinations work together?

DAGs

- DAG characteristics
 - Should DAGs be static or dynamic?
 - Should they be explicit or implicit

Namespace and Address Space

- Should objects identifiers be global or local?
- Should that be a relocatable name or physical address?
 - What is advantage of relocatable names
 - What is the performance implication of data movement, memory footprint, energy, etc.
- How does the runtime system engage?

Messaging

- Should messaging be active (active message)
 - Are there limits in what can be “activated”?
 - How are computational / memory resources managed to execute the active computations?
- Should scheduling be message driven?

Layout and Load Balancing

- Should computation be relocatable?
 - *Is relocation global (across the system) or local (within a domain, e.g., node)?*
 - *Should tasks be first class (nameable)? Do we need a naming services for tasks as with objects?*
 - *Are tasks and objects the same entity?*
- Should the control layout be matched to or independent of the data layout?
 - Should computation follow data?
 - Should data follow computation?
 - Or something in between?