Parallel Session VI: Runtime Systems Research Questions

Andrew A. Chien
Zoran Budimlić
Josh Fryman
Charter

• “Identify research questions that need to be resolved within the context of current experience and knowledge.”

• Prompt: Sections of the Runtime Systems Report
  – Sorted: short ones first

• Guidance: Tersely, in the form of a question.
  – How to balance... What is the right... What features to support...

• Every section that follows is implicitly “Services”
General (cross-cutting) questions

• How should the runtime services exploit underlying OS services to ensure security?
• How do you control and optimize a non-deterministic system?
• How do you propagate information up or down the runtime layers?
• How do runtime services interact across applications?
• How does the need for resilience affect the implementation of all runtime services?
Resilience

- **What types of resilience** interfaces are appropriate for Exascale runtime systems?
  - Application, Task-based PM/Es, High level PM/Es?
  - Multiple runtimes or stacks of runtimes working together?
- **What types of Reliable Stores** (local, regional, global, ..) are appropriate?
  - Efficient implementations – flexible redundancy structure, size and speed
  - How to exploit the **wide variety of memory types** (near/far dram, nvram, burst buffers, ssd, filesystems, etc.)
- **What types of scheduling** (static, dynamic, intra-node/inter-node, ...) are appropriate?
  - Work-stealing task schedulers can support task migration, and can adapt to different numbers of workers
  - Data-driven tasks can be supported with highly asynchronous checkpointing algorithms
- **What types of Reliability Protocols** are appropriate? (**resilience composition – resilient resiliency**)
  - Efficient lightweight protocols suitable for HPC environments
  - Suitable for Runtime internals, applications, perhaps other parts of the system
- **What types of error detection and notification** services?
  - How to reduce/eliminate errors?
  - What types of errors and characterization of frequency, occurrence structure, etc.
  - What are the limits of prediction of errors?
- What are the right **divisions of labor between runtime/application/OS?**
  - Application/runtime collaboration
  - Mitigation strategies/approaches?
  - Should resilience mechanisms impose restrictions on programming models?
    - Cost/benefit analysis
    - How to minimize restrictions on programming models?
- **What is unique to HPC** vs. cloud computing and **what can we learn/leverage** from existing models
- Can we do **root cause analysis of problems, and how?**
Introspection (information)

• What is the right tradeoff between comm/power/info-quality/SLA/QoS
  – What are the right metric to measure effectiveness?
  – At which levels of the system should introspection happen (up/down)?
• What information should introspection service provide?
  – How is the information going to be used
  – What is the utility of the information (even beyond the life of the application)?
  – How to evaluate if prediction is correct?
  – How do we correlate information across levels?
  – Is there a standard schema for this kind of information?
• What is the scope of aggregation feasible?
  – How do we allow users to customize aggregation?
• How do we apply introspection to storage/interconnect too?
Naming

- **What virtualization of naming is needed** to support <feature>?
  - Features = resilience/elasticity/load balancing/polymorphism/…
  - Can we eliminate (virtualize) pointers?
  - Distinction between Global Address Space and Global Name Space

- **What’s in a name?**
  - Is there some structure to names?
  - What should be named (objects, tasks, data, locations)
  - Will persistent storage be part of naming?

- **What forms of local/global/regional naming are appropriate?**
  - Name scope as well (spatial, temporal = how long is the name visible, ..)?
  - How do you track names and changes of names and things that are not named?
  - What support should be there for discovery within applications and between them?

- What kinds of **names should be application/runtime/hardware visible**?
  - What’s the cost and what hardware support is appropriate?
Location

• **What’s in a location?**
  – How do you describe a location?
  – What is the information quality of location services?
  – Are **these 1-1 mappings or something more** flexible

• What are appropriate **interfaces to other runtime services**?
  – Notification, callback...
  – How do we express and discover affinity?

• **How much is the runtime’s responsibility?**
  – Who is allowed to change the location?

• **What is different for exascale?**
  – extreme scale, resilience
Communication

• **What sets of communication primitives** should be supported?
  – Point to point, one to many, many to one, third party
  – How can SW control/specify/reconfigure communication services?
  – Can communication primitives be broadened to encompass all data movements e.g., including those needed for heterogeneous devices?

• What **inter-application communication services** should be supported?
  – And in-situ workflows
  – How do we support communication across nodes?

• What **inter-runtime communication services** should be supported?
  – How is communication service within an application **virtualized**?

• How are **quality of service issues** handled?
  – What communication services are resilient?
  – Is it app->runtime, or runtime<->runtime, or something else?

• **What kinds of introspection information** should be provided by communication services?
Concurrency Control

• How to do global termination detection?
• What concurrency primitives should be supported on node and across nodes?
Scheduling and placement

• What does it mean to **prefetch on exascale machine (from where to where)?**
  – Can a user customize data movement?

• What do we expose as **controls to other layers**?
  – Hints vs directives?
  – What underlying hardware support do we expose?
  – How do we **schedule to optimize** power/performance/memory cost?
  – How do you specify/ **control layout to optimize** power/performance...?
  – How do you do critical path scheduling?
    • How do you detect critical path?
    • How to avoid getting the scheduler on critical path?
  – How do you generate and represent large-scale DAGS efficiently?
    • How to maximize useful parallelism?
  – Should the scheduler **exhibit back-pressure** to/from other levels of the system?

• What **changes in system-level scheduling and resource management** are required to support task and workflow execution models (elasticity...)
  – Can the system also support User-level threads?
  – What kinds of over-provisioning make sense?

• How do you **compose multiple schedulers and placement services**?
  – Can you build composable schedulers and placement services (as peers or hierarchical)?
  – How do you do scheduling/data movement for different and complex memory hierarchies?
Resource Management (Power)

• How do you best allocate power within <unit>?
  – jobs/nodes/hardware on a node/chassis/rack/net/storage/..
  – What granularity is useful to be able to manage power at <unit>?

• What control should the runtime system have over power
  – When should you load balance and when should you power balance?
  – How binding is a user request?
  – What controls to expose up/down the software stack for apps?

• What application characteristics to pass to the runtime so that it can reason about power/energy?
  – How do you model power implications of application choices/design?
  – Can software demonstrate that it can schedule for power?
  – How do you visualize power?
  – How do you transmit repetitive or long-term power need changes to the runtime system?
Resource Management (cont’d)

• Load balance
  – How should the resource-level interact with the application-level?
  – How do you quickly make room for adjacent jobs?
  – What granularity for runtime components and should it be tuneable?
    • Task models concurrently with big MPI ranks
  – Does load balancing impose requirements on location/name services?

• Locality discovery and management
  – What are the dimensionality factors of locality?
    • spatial, temporal, power, bandwidth, latency, resilience, QoS/SLA, . . .
  – How do you measure locality, or its “goodness”?
    • When is it useful to change locality (getting more or less)?
  – How do you quantify the “working set” requirements of exascale applications at different levels of the memory hierarchy?
  – What are the non-NP-Hard ways to evaluate all this?
  – How do you expose lower-level information?
  – How do you push the control into the lower levels?
  – Can SW reason about and exploit locality controls at exascale?
    • If using a dynamic task model?
Resource Management (Adaptive Control)

• **What do you want (and do not want) to adaptively control?**
  – Should the user be able to set binding limits to the adaptation?
  – How expensive are the controllers (power, performance, data), and what are the tradeoffs?
  – **How do you prevent feedback loops?**
• At what point **does your adaptive control logic need game theory**?
  – Hierarchy, government system, welfare system?
  – **Do you want to ensure fairness** and how?
    • How do you express priorities to the controller?
    • **Disambiguation of jobs**
    – Do you give priority to what you’re learning from the hardware or from the higher layers?
• How do you **handle user input**?
• How do you **integrate external adaptive control** (libraries / HW) with the runtime?