Runtime Systems Research Questions Working Group Out-brief

Session Chair: Thomas Sterling

March 13, 2015

Charter: Session V – Research Questions for Runtime Systems

- Core **product** of this workshop
- Derived from **context** of presentations, breakouts, and discussions
- Research **Meta-Questions**
  - What we don't understand
  - What requirements that we have to know how to address
  - What we don't know how to do
  - What we disagree about in terms of issues and concepts
- We don't have to **agree**; conflicts can be recast as questions yet to be resolved
- We don't need to **answer** the question here: that is for the research program

Focus Area 1: System Architecture

- Execution model
- Asynchrony
- System fragmentation
- Relationship between OS and runtime system
- Relationship between Programming models and runtime systems
- Compile time information, guidance, and constraints
- Evaluation

Focus Area 2: Runtime System Design

- Memory models, name space, and address space
- Introspection interfaces, policies, and control
- Contribution to tools
- Parallelism forms, granularity, and synchronization
- Contribution and responsibility to reliability
- Contribution and responsibility to energy/power management
- System hardware control
- Evaluation

Strategic Research Questions

- What issues/questions are actionable and what are those actions?
- What's *Next*? How do we make the next steps to planning?
- What is/is-not a runtime system in terms of roles and responsibilities with respect to OS and PM/E?
- What are suitable runtime system attributes that will address the challenges of **exascale** programming?
- Note: A runtime system is NOT a Lego-set: it comprises a complex set of interoperable mechanisms and services that manage resources and schedule tasks on behalf of the application that address performance, energy, reliability, portability, and user productivity; hopefully enables generality, performance portability, and interoperability for composability

Sterling's list:

- What is the **performance model** that will drive the development of the runtime systems?
- How does the runtime system achieve performance portability?
- What is the introspection strategy for dynamic control of resources and tasks
- In what way are policies for usage management and prioritization incorporated and provided?
- Name space support and employment?
- What hardware support would be critical to accelerating the operation of the runtime system?

Overriding Research Questions

- Runtime research issues
- Dynamicism
- Runtime systems
- Domain specificity
- Storage
- System interaction

Runtime Research Issues

- The runtime has to be well integrated, but some features may not exist:
  - On all hardware
  - 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

Dynamicism

- Is a dynamic runtime capable of delivering exascale performance?
  - How much parallelism should be exposed to the runtime
    - You may expose all parallelism available to the application, but have it throttled by the programming model implementation
    - Sometimes the amount of parallelism is data dependent
- Would dynamic control yield a significant improvement in operational concurrency?
- Would the cost overwhelm the benefits?

- Is a dynamic runtime capable of delivering exascale performance?
- Is it required for exascale performance?
- Does it make programming easier?

Runtime systems

- What do we mean by runtime?
  - User space vs. privileged
  - The interaction with the OS is clearly important
  - We will be focused on user space for this discussion
  - We need to think carefully about cloud computing and the use of virtualization (i.e., the runtime may have privileges)
- Later decide there are two levels:
  - Within the application
  - Management of “executables” across the system

Domain specificity

- Should the runtime be domain-specific
- How is domain knowledge passed to the runtime?
  - How to deal with load balancing for specific problems
- What mechanisms can be domain-independent?

Storage

- Relationship between runtime and data
- How does the runtime interact with the storage subsystem
- There is a connection between workflows and runtimes
- What runtime support is required for complex workflows

System Interaction

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

Detailed Runtime Characteristics

- Tasking Behavior
- DAGs
- Namespace and address space
- Messaging
- Resilience
- Energy

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. pre-emptible
  - Do they allow mutable external state ?
  - Is it legal to communicate between tasks?
  - Is it legal to synchronize?
- If tasks may be de-scheduled 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 computation be adaptive?*
- *Should messaging be active (active message)*
  - *Moving work to data?*
- *Should scheduling be message driven?*
- *Should computation be relocatable?*
  - *Does a task have a global id that works if the task moves?*
  - *Should tasks be first class (nameable)?*
- 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

Resilience and Energy

- What failures (if any) can the runtime system:
  - Hide from higher level software / programming?
  - Contain from other parts of the computation
- How should the runtime report failures?
- What are the most important types of failures and their likelihood? (input to runtimes)
- How does the runtime contribute to resilience in combination with other layers?

Programmatic Questions

- Program Questions
- What we need to do

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 we need to do?

- 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)