

2015 ECI Runtime Systems Workshop
*Parallel Session 1: Runtime Systems
Architecture (set 1)*

Execution Model

- Execution model: Discuss the governing principles of the strategy of computation that guides the runtime system (together with the other system component layers) in addressing the challenges and issues discussed here. Discuss the necessary features of a cogent, comprehensive, and self-consistent execution model upon which to base a runtime system design.
- Research topic: Define the responsibilities of different layers.
 - What can one layer expect from other layers?
- Bypass methods through a layer where applicable
- Concept of services, you can choose to use them (or not)
 - Some services cannot be “bypassed” without consequences
 - Services that are responsible for a global view need to know what is happening globally – hard problem

~~Asynchrony~~ (Performance Variability)

- **Asynchrony:** Discuss the dramatic increase on variability of latency of communication. Considering shared resources such as networking, memory banks, discuss the increased uncertainty of action/task response time requiring adaptive methods managed by the runtime. Discuss the semantics and control strategy in the presence of the uncertainty imposed by asynchrony. Identify the hardware support that facilitates task management. Identify the differences between supporting synchronous and asynchronous programming models.
- Today we view our world as regular, increasingly not true due to hardware variability, contention, algorithm irregularity, etc
- Asynchronous runtime systems can be used to address variability of communication (network, memory accesses), application irregularity
- Asynchrony may in turn increase performance variability
- Research issue: balance of coordination (implies communication) and asynchrony
 - May increase communication pressure and complicate resource management
 - Performance data collected for a representative set of tasks/nodes – not just one
 - Becomes a sampling problem
- Hardware to facilitate (Research question)
 - We need to know more before we can selectively optimize via hardware
 - Things like fast atomics, light weight task scheduling may be candidates, behavior of software on the hardware equivalent of global performance counters

System Fragmentation

- System fragmentation: Discuss whether a runtime system should operate across the total system span (or sub-span of the app) or if it should operate on separate local domains (nodes).
- Yes
- Runtime system has to operate across all nodes dependent on what the service provides
 - At least some of the services
 - Some services are between applications on the machine
 - Some are associated with the application
 - Some services may only be concerned with what is happening on the node

Relationship between OS and Runtime system

- Relationship between OS and runtime system: Given that the OS is persistent and is the surrogate for the hardware resources and that the runtime reflects the workload demands and properties of the application program, discuss the relative roles and responsibilities of these two in cooperation and what are the logical interfaces and protocols by which current information is exchanged.
- Distinction between services “used” by an application and services “used between” applications
- What is the visibility of the service?
- OS acting as hardware abstraction should be out of the critical path to accessing hardware resources

Relationship between Programming models and runtime systems

- Relationship between Programming models and runtime systems: Establish the basic requirements of future runtime systems in terms of their relationships with the programming interfaces and compilation strategies. Explore the nature of programming models that will employ runtime systems for management of dynamic adaptive execution to guide co-design of programming and runtime methods.
- Research topic: what is presented or exposed regarding the runtime to the programming model and or through to the compiler and application.
 - Balance of performance, portability, productivity and understandability
- The interaction between this wide range of services in addition to things like performance monitoring and data services

Compile ~~time~~ derived information, guidance and constraints, e.g. JIT compiler

- Compile time information, guidance, and constraints: Discuss new forms of compile time information that is fed to the runtime to help it better perform its responsibilities adaptively? This is a major new area of consideration, perhaps among the biggest innovations of future system design.
- Clearly a lot of potential research in this area, we need to have a mechanism to have more influence in this area – than we currently have
- How to have a two way street through service layers to influence workflow
 - How to transfer semantic information up and down the services stack

Evaluation

- Evaluation: Discuss the metrics, measures, benchmarks, and means for testing and evaluation of the runtime system architecture.
- Systems need to present the information we need
- How do we collect the information even if provided
- We don't want performance metrics alone
 - Want relative metrics to evaluate progress on research effort
 - Time to solution
 - Time to solution with failures
 - Time to solution with system variability
 - Time to solution under power/energy constraints
- Runtime overhead
 - Time (CPU overhead)
 - Space (memory)
- Benefit provided vs. overhead
- Portability of runtime
 - Runtime implementation, algorithms/data structures portable across multiple architectures
 - Runtime provides program portability