Working Group 4: Runtime and Operating Systems

Chair: Rick Stevens

Vice Chair: Ron Brightwell

WG 4– Runtime and OS Charter

- Charter
  - Establish baseline capabilities required in the operating systems for projected HEC systems scaled to the end of this decade and determine the critical advances that must be undertaken to meet these goals. Examine the potential, expanded role of low-level runtime system components in support of alternative system architectures.
- Chair
  - Rick Stevens, Argonne National Laboratory
- Vice-Chair
  - Ron Brightwell, Sandia National Laboratory

WG 4– Runtime and OS Guidelines and Questions

- Establish principal functional requirements of operating systems for HEC systems of the end of the decade
- Identify current limitations of OS software and determine initiatives required to address them
- Discuss role of open source software for HEC community needs and issues associated with development/maintenance/use of open source
- Examine future role of runtime system software in the management/use of HEC systems containing from thousands to millions of nodes.
- Example topics:
  - file systems, open source software, Linux, job and task scheduling, security, grid-interoperable, memory management, fault tolerance, checkpoint/restart, synchronization, runtime, I/O systems

Working Group Participants

- Ron Brightwell
- Neil Pundit
- Jeff Brown
- Lee Wand
- Gary Girder
- Ron Minnich
- Leslie Hart
- DK Panda
- Thuc Hoang
- Bob Balance
- Barney McCabe
- Wes Felter
- Keshav Pingali
- Deborah Crawford
- Asaph Zemach
- Dan Reed
- Rick Stevens

Our Charge

- **Establish Principal Functional Requirements** of OS/runtime for systems for the end of the decade systems
- **Assumptions:**
  - Systems with 100K-1M nodes (fuzzy notion of node) +-order of magnitude (SMPs, etc.)
  - COTS and custom targets included
- Role of Open Source in enabling progress
- Formulate critical recommendations on research objectives to address the requirements

Critical Topics

- Operating System and Runtime APIs
- High-Performance Hardware Abstraction
- Scalable Resource Management
- File Systems and Data Management
- Parallel I/O and External Networks
- Fault Management
- Configuration Management
- OS Portability and Development Productivity
- Programming Model Support
- Security
- OS and Systems Software Development Test beds
- Role of Open Source

Recurring Themes

- Limitations of UNIX
- Blending of OS and runtime models
- Coupling apps and OS via feedback mechanisms
- Performance transparency (visibility)
- Minimalism and enabling applications access to HW
- Desire for more hardware support for OS functions
- “Clusters” are the current OS/runtime targets
- Lack of full-scale test beds limiting progress
- OS “people” need to be involved in design decisions

OS APIs (e.g. POSIX)

- **Findings**:
  - POSIX APIs not adequate for future systems
    - Lack of performance transparency
    - Global state assumed in POSIX semantics
- **Recommendations:**
  - Determine a subset of POSIX APIs suitable for High-performance Computing at scale
  - New API development addressing scalability and performance transparency
    - Explicitly support research in developing non-POSIX compatible

Hardware Abstractions

- **Findings:**
  - HAs needed for portability and improved resource management
    - Remove dependence on physical configurations
      - virtual processors abstractions (e.g. MPI processes)
    - Virtualization to improve resource management
      - virtual PIMs, etc. for improved programming model support
- **Recommendations**:
  - Research to determine what are the right candidates for virtualization
  - Develop low overhead mechanisms for enabling abstraction
  - Making abstraction layers visible and optional where needed

Scalable Resource Management

- **Findings:**
  - Resource allocation and scheduling at the system and node level are critical for large-scale HEC systems
  - Memory hierarchy management will become increasingly important
  - Dynamic process creation and dynamic resource management increasingly important
  - Systems most likely to be space-shared
  - OS support required for management of shared resources (network, I/O, etc.)

Scalable Resource Management

- **Recommendations:**
  - Investigate new models for resource management
    - Enabling user applications to have as much control of low-level resource management where needed
    - Compute-Node model
      - Minimal runtime and App can bring as much or as little OS with them
    - Systems/Services Nodes
      - Need more OS services to manage shared resources
    - I/O systems and fabric need to be managed
  - Explore cooperative services model
    - Some runtime and traditional OS combined into a cooperative scheme, offload services not considered critical for HEC
  - Increase the potential use of dynamic resource management at all levels

Data Management and File Systems

- **Findings:**
  - The POSIX model for I/O is incompatible with future systems
  - The passive file system model may also not be compatible with requirements for future systems
- **Recommendations:**
  - Develop an alternative (to POSIX) API for file system
  - Investigate scalable authentication and authorization schemes for data
  - Research scalable schemes for handling file systems (data management) metadata
  - Consider moving processing into the I/O paths (storage devices)

Parallel and Network I/O

- **Findings:**
  - I/O channels will be highly parallel and shared (multiple users/jobs)
  - External network and grid interconnects will be highly parallel
  - The OS will need to manage I/O and network connections as a shared resource (even in space shared systems)
- **Recommendations:**
  - Develop new scalable approaches to supporting I/O and network interfaces (near term)
  - Consider integrating I/O and network interface protocols (medium term)
  - Develop HEC appropriate system interfaces to grid services (long term)

Fault Management

- **Findings:**
  - Fault management is increasingly critical for HEC systems
  - The performance impacts of fault detection and management may be significant and unexpected
  - Automatic fault recovery may not be appropriate in some cases
  - Fault prediction will become increasingly critical
- **Recommendations:**
  - Efficient schemes for fault detection and prediction
    - What can be done in hardware?
  - Improved runtime handling (graceful degradation) of faults
  - Investigate integration of fault management, diagnostics with advanced configuration management
  - Autonomic computing ideas relevant here

Configuration Management

- **Findings:**
  - Scalability of management tools needs to be improved
    - Manage to a provable state (database driven management)
  - Support interrupted firmware/software update cycles (surviving partial updates)
  - New models of configuration (away from file based systems) may be important directions for the future
- **Recommendations:**
  - New models for systems configuration needed
  - Scalability research (scale invariance, abstractions)
  - Develop interruptible update schemes (steal from database technologies)
  - Fall back, fall forward
  - Automatic local consistency

OS Portability

- **Findings:**
  - Improving OS portability and OS/runtime code reuse will improve OS development productivity
    - Device drivers (abstractions)
    - Shared code base and modular software technology
- **Recommendations:**
  - Develop new requirements for device driver interfaces
    - Support unification where possible and where performance permits
  - Consider developing a common runtime execution software platform
  - Research toward improving use of modularization and components in OS/runtime development

OS Security for HEC Systems

- **Findings:**
  - Current (nearly 30 year old) Unix security model has significant limitations
  - Multi-level Security (orange book like) may be a requirement for some HEC systems
  - Current Unix security model is deeply coupled to current OS semantics and limits scalability in many cases
- **Recommendations:**
  - Active resource models
    - Rootless, UIDless, etc.
  - Eros, Plan 9 models possible starting point
  - Fund research explicitly different from UNIX

Programming Model Support in OS

- **Findings:**
  - MPI has productivity limitations, but is the current standard for portable programming, need to push beyond MPI
  - UPC and CAF considered good candidates for improving productivity and probably should be targets for improved OS support
- **Recommendations**:
  - Determine OS level support needed for UPC and CAF and accelerate support for these (near term)
  - Performance and productivity tool support (debuggers, performance tools, etc.)

Testbeds for Runtime and OS

- **Findings:**
  - Lack of full scale test beds have slowed research in scalable OS and systems software
  - Test beds need to be configured to support aggressive testing and development
- **Recommendations:**
  - Establish one or more full scale (1,000's nodes) test beds for runtime, OS and Systems software research communities
  - Make test beds available to University, Laboratory and Commercial developers

The Role of Open Source

- **Findings:**
  - Open source model for licensing and sharing of software valuable for HEC OS and runtime development
  - Open source (open community) development model may not be appropriate for HEC OS development
  - The Open Source contract model may prove useful (LUSTRE model)
- **Recommendations:**
  - Encourage use of open source to increase leverage in OS development
  - Consider creating and funding an Institute for HEC OS/rumtime Open Source development and maintenance (keeping the HEC community in control of key software systems)