

Issues

- Autonomic operation (fault tolerance)
- Minimize interference to applications
- Hardware support for new operating systems
- Resource management (global & local)
- Autonomic self-optimization (e.g., load balancing)
- Maximum utilization of the machine
- Collective run-time (collective system calls)
- Interactive parallel jobs
- Development strategies for using testbeds
- Large-scale development systems widely available
- Programming model interaction with OS on reliability
- Single-system image
- Graceful degradation of system
- Hardware support for off-loading operating systems services

Infrastructure (1)

- We want to provide broad access to large scale platforms for research
 - Parallel machines
 - Parallel simulation environments
- Useful testbed machines need configuration managers
 - Users can specify custom configurations
 - Nodes come and go much more frequently
 - Don't know what nodes are doing (e.g., Ron may be trying a Plan9 installation) – system is opaque
 - Security issues
- Useful testbed machines need more monitoring
- Research on system software for testbeds

Infrastructure (2)

- How to put high-performance testbeds in people's home
 - Build or show how to build very cheap parallel machines (32-128 processors)
 - Software in open source to build community
 - Reference platform for hardware/software
 - Few thousand's of dollars

Infrastructure (3)

- Need to establish usage strategy for testbeds
 - People that are testing user-level code: operate just like applications
 - People that are testing operating systems: access to low-level hardware features (power supply, network configuration, etc): requires better security, better configuration, better management
 - Ability to run large but short jobs to test scalability

Self-optimization

- Run-time system has to support tuning: instrumentation, measurement, performance debugging
- Run-time system for automatic mapping and load balancing
- Isolating programming concepts (data units and work units) from execution units (processors, memory, switch)
- Mapping virtual resource to physical resources
- Run-time system support for continuous optimization of an application:
 - Changing a library dynamically
 - Recompiling modules
 - Redistributing data and computation
 - Dynamic refining performance analysis

Interactive jobs

- Support for interactive jobs in large parallel systems
- Easier to use a machine in interactive mode
 - Computational steering – user driven optimization
 - Virtual reality
 - Human-machine interaction in simulations
- Enabling technologies
 - Run jobs on demand
 - Time multiplex jobs to reduce response time
 - More dynamic handling of jobs
- Better human-machine interfaces to make it easier to interact with parallel applications
- Mixing batch and interactive jobs

Fault tolerance

- Fault-management: fault anticipation, recovery, degraded performance
- Making system state information available to the application – how to handle in the programming model (visible or invisible)
- System initiated checkpoint/restart – also useful to improve system utilization
- Application-specific checkpoint – user programmed or compiler generated

Scalable single-system image

- What does it mean (to be defined):
 - Single process space
 - Single root file system
 - Single socket space
 - Single /proc
 - Shared memory segments (for functionality if not performance)
 - Definition depends on the usage (site and applications)
- Why do we want it:
 - Legacy applications (e.g., debuggers)
 - Ease of operation and administration
- What are the research areas:
 - Single-system image for clusters
 - Bproc and MOSIX have limitations
 - Performance issues are significant even for single-address space machines
 - Collective operations on a new abstraction (e.g., job)
- Reliability of system under single-system image
- Scalability issues of single-system image
- New abstractions are necessary for larger systems

Interference to applications

- Need single-node operating system with more deterministic behavior
 - Bounded operations by operating system
 - Real-time OS architectures
- Asynchronous applications are more tolerant to interference
 - OS support for asynchronous operations?
- Synchronized operations across machine – multiple synchronized OS
- Dedicated hardware for the operating system (ASCI Red, BG/L)
- Single user thread/hardware context eliminates context switches
- Dynamic, fast scheduling of threads based on communication operations can help (message-driven scheduling)

Security management

- Security management: intrusion detection and compartmentalization

Community building

- Better interaction with Linux community working on large clusters:
 - Provide testbeds, benchmarks, tools
 - Cheap MPPs at home
 - Understand tools that they use
- Better interaction with real-time OS groups:
 - Common problems of reducing interference, predictability
- More interaction with domain-specific application teams