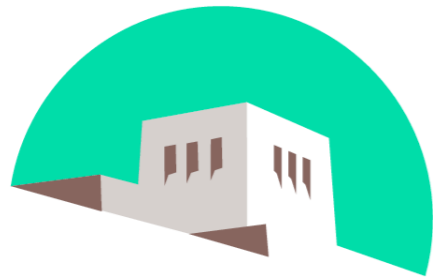


Review of Previous FAST-OS Activities

Arthur B. Maccabe

Computer Science Department & HPC@UNM
The University of New Mexico



What is FAST-OS?



Forum to Address Scalable Technology for runtime and Operating Systems

(Blame Ron Brightwell, I do)

- Observation: applications on ASCI/Red have 60% scaling efficiencies of on 9000+ processors; other systems show scaling efficiencies near 10% -- why?
- Scaling must be addressed at all levels: architecture, operating system, runtime system, and application
- Distinction between operating and runtime systems
 - ◆ OS is about protection (competition for shared resources)
 - ◆ RT is about environment for applications (abstract resources)
 - ◆ We should not make too much of the distinction



Activities



February 2002	Bodega Bay	WIMPS
July 2002	Chicago	FAST-OS
March 2003	Durango	SOS
June 2003	Washington	HECRTF
June 2003	Washington	SCaLeS
July 2003	Washington	FAST-OS



Goals



- Runtime and operating systems represent a potential obstacle to the development of next generation (and beyond) systems
- (Re)Build OS research community focused on high end computing systems
 - ◆ Include industry, labs, and academics
- Should identify and address limitations to scalability



Foundations



- Focus is on capability
 - ◆ Capacity is important, but secondary

- Focus is on next generation systems
 - ◆ Further out is important, but secondary

- FAST-OS is **not** lightweight kernels
 - ◆ Lightweight kernels may play a role, but other approaches should be considered

- Open source



Issues



■ Fault tolerance

- ◆ checkpoint - restart (system, application, compiler supported)
- ◆ run through failure, other forms of application fault tolerance
- ◆ migration (run away)

■ OS Structure

- ◆ global/local OS split, OS/runtime split
- ◆ adaptability, composibility
- ◆ extending lightweight approaches
- ◆ protection boundaries and virtualization

■ APIs

- ◆ Application/runtime, Runtime/OS, OS/compiler, architecture
- ◆ Tool interfaces (e.g., debuggers)
- ◆ Environment information



Issues (continued)



■ Specific functions

- ◆ process management, file systems, scheduling, security
- ◆ QoS
- ◆ support for invariants (debugging)

■ Scalability

- ◆ what is it? do we even know it when we see it?
- ◆ which services need to scale?

■ Interactive systems

■ Hardware innovation

- ◆ do we need multiple solutions?

■ Hardware support for OS/runtime

- ◆ protection, reliable networks, collective operations, atomic memory operations, transactional memory...



Issues (continued)



- Application requirements
- Metrics
- Programming models (FAST-OS?)
 - ◆ models for 100,000 processors
 - ◆ breaking the legacy straightjacket
 - ◆ what comes after MPI?
 - ◆ path for existing, scalable applications to petaflop systems
- Testbeds & Simulation



The Death of OS Research



■ The curse of Mach

- ◆ You can do anything you like, as long as what you like is Mach

■ Investment

- ◆ 100's of man years

■ Services & Standards

- ◆ Users demand a rich set of service--even when developing for embedded and HEC environments

“To be a viable computer system, one must honor a huge list of large, and often changing, standards: TCP/IP, HTTP, HTML, XML, CORBA, Unicode, POSIX, NFS, SMB, MIME, POP, IMAP, X, A huge amount of work, but if you don't honor the standards, you're marginalized” Rob Pike



Death (continued)



■ Hardware access

- ◆ OS developers rarely get access to larger systems
- ◆ OSF only had access to 32-node systems

■ Moore's Law

- ◆ OS development focuses on features, not implementations
- ◆ OS becomes more complex due to poor implementations

■ Linux

- ◆ Structure: 1,000's of lines of code know the socket structure
- ◆ Acceptance: metric is performance on servers and desktops
- ◆ Culture: Linux hackers don't acknowledge OS research



OS Challenges



■ Architecture

- ◆ Architectural innovation is critical for HEC revitalization
 - multiprocessor cores, PIM systems, power-aware systems
- ◆ Current OSes stifle architectural research
 - Linux page table abstraction *is* x86

■ Multiple management strategies

- ◆ Resource constrained applications
- ◆ OS/Application management mismatch
 - Applications re-invent resource management
- ◆ OS adaptability

■ Specialized HEC needs

- ◆ e.g., new programming models
- ◆ I/O services



Challenges (continued)



■ Sophisticated services

◆ System complexity

- vector to distributed memory to SMP + message passing to ...

◆ Visualization

■ Security

◆ Connecting HEC systems to the Internet (DTF)

◆ Multilevel secure

■ Fault tolerance

■ Consolidation

◆ Tap into broader community

◆ Avoid inventing new wheels



Suggested Research Agenda



Goal: (re)build research community

- What constitutes OS research?
 - ◆ Need to extract investment from the code
- HEC must define interesting problems
 - ◆ Meaningful application benchmarks
 - ◆ Publish challenges
- Provide testbeds
 - ◆ Small, medium, and large
 - ◆ Give us the new stuff: SMP, distributed memory, PIM systems
- Fundamental innovations
 - ◆ Small, start from scratch projects are critical
 - ◆ Research in component strategies for OS structure



Agenda (continued)



■ Low-level mechanisms

- ◆ Resource protection and mediation are fundamental OS issues
- ◆ OS-bypass is evil
- ◆ QK & Hypervisors

■ Bridge research and development

- ◆ Need a seamless path from research to development to deployment
- ◆ Build mutual trust/respect between communities

■ Reduce legal barriers

- ◆ Open source, open source, open source
- ◆ Need to deal with intellectual property in a meaningful way
- ◆ SCO lawsuit



■ APIs

- ◆ POSIX APIs not adequate for future systems
 - lack of performance transparency
 - global state assumed by POSIX semantics
- ◆ Fund research in non-POSIX compliant APIs

■ Hardware abstractions

- ◆ needed for portability and improved resource management
- ◆ remove dependence on physical configuration
- ◆ determine best candidates for virtualization
 - virtual processors, virtual PIMs, etc
- ◆ make abstraction layers replacable when needed



HECRTF (continued)



■ Scalable resource management

- ◆ system and node scheduling are critical
- ◆ memory hierarchy is becoming more important
- ◆ OS support required for shared resources
- ◆ space-sharing is assumed
- ◆ give users control whenever possible
- ◆ dynamic creation/management is increasingly important

■ File systems

- ◆ develop non POSIX API
- ◆ may need active file system
- ◆ consider approaches that move data processing into the I/O path



HECRTF (continued)



■ Parallel and network I/O

- ◆ lots of parallelism
- ◆ Grid style interconnects

■ Fault management

- ◆ prediction
- ◆ autonomic approaches

■ Configuration management

- ◆ automatic local consistency
- ◆ interruptable update schemes (steal from database community)

■ OS portability

- ◆ reuse code to improve productivity
- ◆ improve components and modularization in OS research
- ◆ support unification where possible and performance permits



HECRTF (continued)



■ Security

- ◆ 30 year old Unix model has significant problems
- ◆ do we need “orange book” (MLS) security?
- ◆ rootless, uidless, Eros, Plan 9
- ◆ fund specifically non-Unix security research

■ Programming model support

- ◆ need to push beyond MPI
- ◆ UPC, CAF
- ◆ tools (debugging and performance)

■ Testbeds

- ◆ lack of testbeds has hindered research

■ Open source

- ◆ open community?



Why we're here



- We have already made the case for OS research; now what do we do?
- Think about relations among issues
- Which are best addressed in basic research? development? deployment?
- Which are best addressed by academics? labs? industry?
- Dependencies among issues
- Don't need (or want) full prioritization

