Linux never has been and never will be “Extreme”

Arthur B. (Barney) Maccabe
Computer Science Department
Center for High Performance Computing
The University of New Mexico

Salishan April 23, 2003
This talk was prepared on a Debian Linux box

http://wwwdebian.org

using OpenOffice

OpenOffice.org 1.0
The Open Source Office Suite

http://www.openoffice.org
Outline

- My background: lightweight operating systems
- Linux and world domination
- Adapting to innovative technologies
- What is Linux?
- OS Research
- Summary
SUNMOS: Sandia/UNM OS

- Lightweight, Compute Node OS
  - Developed for 1024 node nCUBE 2
  - Ran on Intel Paragon (1800+ nodes)
    OSF-1/AD didn't scale until a few years later

- Intel Paragon
  - SUNMOS 256KB
    OSF-1/AD 10-12MB
    16 MB memory / node
  - 4KB to 4MB page:
    25% application improvement
    4 TLB entries
Puma/Cougar

- Follow-on to SUNMOS

- Compute node OS for Intel Tflops, ASCI/Red
  - 4500+ compute nodes
    - 2 333MHz Pentium II/node
    - 256MB/node
  - Applications show 60-70% scaling efficiency
    - Is it the OS or the machine?
    - Rogue OS effects (daemons, etc)
CPlant (tm)

- 1500+ 466MHz Alpha EV6
- Myrinet LANai-7 and LANai-9
- Red/Black switching
- Re-create systems software from ASCI/Red
  - High-performance message passing (Portals)
  - Application launch
  - System management tools
- Linux (tm) on service and compute nodes
  “World's largest Linux cluster”
A Linux Mismatch

- **Partition model**
  - Specialization in hardware and software

- **Linux responds to application requests**
  - Resources do not initiate requests (inetd is a bit of a kludge)

- **Compute node OS is a slave to service nodes**
  - Cplant copies image to RAM disk and exec
  - Bproc uses process migration
**Linux on Cplant Compute Nodes**

- **Original plan:**
  - use Linux to start, build communication layers
  - port Cougar later

- **Linux turned out to be OK**
  - Compute to communication imbalance
  - Linux isn't horribly broken
  - Open source is a good thing
  - People want to talk about and work on Linux

- **It's not all roses**
  - Lots of distractions (see above)
Numbers
Scaling & Related Issues

- **ASCI/Red 60-70% scaling efficiency for applications**
  - Machine or OS?
  - How much do the apps contribute?

- **Horror stories:**
  - Typical scaling efficiency is closer to 10%
  - Barriers that take up to an hour!
  - “Rogue OS effects”
Comparing Linux and Cougar

- Port Linux to compute nodes of ASCI/Red
  - started with 2.4.18, now using 2.4.20
  - original version was to port Cougar to Cplant
- Direct comparison of Linux and Cougar
- Nighten
  - ASCI/Red development system
  - 144 nodes
- Nodes
  - Dual 333 MHz Pentium II's
  - 256 MB
CG

Mops/sec

Nodes

Linux
Cougar

2
4

0
5
10
15
20
25
30
35
40
45
50
55
60
65
70

Salishan April 23, 2003
More (good) Data

**MG**

![MG Graph](image)

**IS**

![IS Graph](image)
Lies, lies, lies

- **Bandwidth**
  - Cougar: >300 MB/s
  - Linux: <35 MB/s

- **Latency**
  - Cougar: 20 usec
  - Linux: 90 usec

- **MPI**
  - Cougar: MPI / Portals 2.0
  - Linux: MPICH 1.2.5 / P4 / TCP / IP / skbufs

Integrate Portals into Linux on ASCI/Red
World Domination
Linux and World Domination

- Largest system
- Largest reasonable Linux system
- Beowulf
- Cplant
- Pink/MCR
Hardware Trends Help Linux

- **Memory**
  - Paragon: 16 MB
  - ASCI/Red: 256 MB
  - Cplant: 1 GB

- **TLB entries**
  - Paragon: 4
  - ASCI/Red: 64
  - Cplant: 128(?)

- **Processor speeds**
  - Paragon: 50 MHz
  - ASCI/Red: 333 MHz
  - Cplant: 466 MHz

- **Relative networking**
  - Paragon: 200 MB/s
  - ASCI/Red: 400 MB/s
  - Cplant: 100 MB/s

Management of node resources is not as critical
Linux Helps Itself

- Easy to disable most daemons
  - Eliminate “Rogue OS” effects
- Really bad things can be turned off
  - malloc() uses mmap
  - out of memory killer
  - 1000 Interrupts/second on Alpha
- Good things being added
  - hugetlb pages
- Horrible things get fixed
  - Time goes backwards in 2.4.18 SMP mode
HPC Community Helps Linux

- System environments
  - Cplant(tm)
  - Scyld(tm)
  - Clustermatic(tm)
  - OSCAR(tm)
- Hardware support
  - Linux BIOS
  - Supermon
- Vendors
  - Drivers available
    - Myricom, Quadrics, SCI, etc.
  - Major vendors support Linux
    - IBM, HP, Dell
  - Specialized vendors
    - Linux Networx, Pro Micro, Atipa, Racksaver, ....
The Obvious Conclusion

World Domination

If you wait long enough, Linux will run well on your system
- Hardware improves
- Linux improves
- The community works

If you wait long enough, your application will run just fine on a sequential system
Cool Things with Linux

- Vertigo: Automatic Performance-Setting for Linux
  - Flauter (ARM) & Mudge (Michigan)
  - OSDI, December 2002

- Transparent superpages for FreeBSD
  - Navarro, Iyer, Druschel & Cox (Rice)
  - OSDI, December 2002
  - FreeBSD

Is the goal to show that Linux can work, or to build a working system?
Some Other Observations

- **Barney's favorite wine:**
  
  “The Linux community doesn't care about HPC”

  - We haven't made a the case for any single feature
  - The HPC community is hard to define:
    Extreme Linux forum was not so extreme

- **Linux direction is not focused on HPC**
  - Servers and desktops

- **Linux on Red Storm?**
  - How much risk? How soon?
Fundamental Conflict

- Working on Linux benefits more people
  - Broader code base
  - Well understood environment

- Specialized solutions work sooner
  - More readily adaptable
  - Designed specifically for the system
Obvious Response

- Work on Linux
- Work on specialized solutions
- largest system
- largest reasonable Linux system
Dealing with Innovation
Innovative Technologies

Time

System Scale

largest system

largest reasonable Linux system

DARPA HPCS
When will it run Linux?
What is Linux?
Linux is an API

The Linux API - Resource abstraction and management
Linux is an Environment

- MPICH
- sockets library
- stdio
- BLAS
- glibc
- The Linux API
- Linux Device Drivers
- GM library and drivers
- Application
- MPICH-GM
API: Syntax and Semantics

- Syntax: Operation signature
  - name (index) of system call
  - number and types of parameters
  - Linux has ~250 system calls

- Semantics: Relative costs
  - how much does fork cost?
  - how long does read take?
  - what does malloc really do?

Syntax is fairly easy, Semantics is hard
What else is there?
OS Research: History

- Synchronization is fundamentally hard
- File systems are neat
- Structure is the way to manage complexity
- You can do anything as long as it is Mach
  - structure is important
- 100's of man-years of investment
  - Middleware
  - Extensible OSes
  - OS Bypass
OS Design Approaches

- **Monolithic**
  - Modular
  - Lightweight
- **Micro-kernels**
- **Extensible**
- **Exo-kernels**
- **OS Bypass**
Monolithic Approaches

- OS controls access to all resources
  - Modular: for variety of resources
  - Lightweight: limit resources and features
Micro-kernels

- **OS routes messages**
- **Servers control resources**

Exo-kernels

- User level OS
- Resources manage themselves
- Applications run independent OSes
Extensible Kernels

- Run application code in the kernel
  - Direct access to resources
  - Avoid interrupt costs
  - Avoid syscall overheads
**OS Bypass & Splintering**

**OS Bypass**
- Bypass the OS for resources that are used intensively

**Splintering**
- OS remains in charge
- Control goes through OS
- Data transfer is direct
Close to the end
Why is OS work hard?

- **Design?**
  
  *That's the fun part*

- **Variety of applications?**
  
  *We don't care about all that many applications*

- **Variety of hardware?**
  
  *We don't really care about that much hardware: processors, memory, timer/clock, network cards, serial interfaces*

- **Buggy hardware?**
  
  *This is a big problem*
Dealing with Buggy Hardware

- **Start with Linux BIOS**  
  *(Thanks Ron Minnich)*

- **Steal Linux drivers, without modification, wherever possible**

- **Write specialized drivers where needed**
  - Communication
  - Memory
Summary

- Observation: Linux will always catch up (is Windows far behind?)
  - If you can afford to wait, you should
  - If you're waiting, work on improving Linux
- My goal is to build systems that work now
  - Strategy: use Linux and feedback into Linux
- OS structure research is important
  - It's not that expensive
  - Selecting a winner too early is destructive
  - Don't over value what you have
6/7/8/9
Operating Systems

- **Multics:** Imagine what we *could* do
- **Unix:** *This is what we can* do
- **BSD:** Wizards may play with the code
- **Mach:** We can do anything, with nothing
- **Windows:** We can make lots of *money*
- **MacOS:** Isn't this pretty?
- **Plan 9:** We can do *less* now
- **Linux:** We don't need no money. *Here's the code, have fun!*
Strategy

● Reductionism (in theory)
  - break a system into its parts and study the parts in isolation
  - the fun comes when you try to re-integrate all the parts

● Reductionism (in systems)
  - identify crucial features, build a simplified version of the full system
  - the fun comes when you try to add features
OS Concepts

- BIOS & High Level languages
  - stand alone machines
  - scheduling through reservations
- Multiprogramming
  - hide latency for long I/O operations
    - users are too stupid, lazy or unmotivated to figure out nonblocking operations
  - optimize processor utilization
- Timesharing
  - humans are really slow
  - optimize response time
What is “Extreme”? 

- Resource constrained computing
- For my desktop, the resources are applications and familiarity
- For my laptop it's battery life, screen size, applications and familiarity
- We probably want to talk about physical resources:
  - processors
  - memory
  - communication
Extreme Systems

- OS defines resource access mechanisms
  - required of all processes
- Frequently, mechanisms include policies
  - consider malloc
- Cannot tolerate abuse of critical resource
- Bypass, if possible
- Hack if possible and necessary
- Design and implement mechanisms that work