Linux for Scientific Computing

Bill Saphir
Berkeley Lab
wcs@nersc.gov

Things you should know if you’re thinking about using Linux for Scientific Computing

Bill Saphir
Berkeley Lab
wcs@nersc.gov
Random thoughts on things you should know if you’re thinking about using Linux for Scientific Computing

Bill Saphir
Berkeley Lab
wcs@nersc.gov

Why?

Scientific research is one of the first areas where Linux has had a major impact on production, mission-critical computing.
Features of scientific computing

- Floating point performance is everything (well, almost everything)
- Users write their own codes
- Legacy Fortran is common
- Full-featured user-friendly GUI interface not required
- Goal is science, not computer science.

Who are the foolish zealots?

- Data analysis in experimental physics
  - Cern, Fermilab, SLAC, Brookhaven, DESY; Astrophysics
- Parallel computing on clusters
  - Sometimes called “Beowulf” clusters
  - Mini-supercomputers
- Thousands of random apps powered by graduate students
Outline

- Why Linux?
- Hardware
  - Computer architecture
  - Processors
  - Benchmarks
- Serial computation
  - Compilers
  - Libraries
- Parallel computation
  - SMP
  - Clusters

Why Linux?

- Access to cheap hardware
- Control
- Availability of software
- Convergence
- Access to cheap graduate students
- Alternative to NT
Computer architecture for HPSC

Von Neumann

Faster Processors

Memory bandwidth important. Cache not so important

Vector Processors no longer with us (in US)

SMP

DMP

Processor support in Linux

- These supported processors are useful for scientific computing:
  - x86
  - Alpha
  - Sparc/Sparc64
  - PowerPC
  - MIPS

- Coming up:
  - Power 3
  - Merced
Which processor?

- Three important criteria
  - Cost
  - Performance
  - Availability of software

Measuring Performance

- Peak
- Linpack
- STREAM (memory bandwidth)
- SPEC
- NPB and NSB
Current Peak Rates

<table>
<thead>
<tr>
<th>Name</th>
<th>MHz</th>
<th>Flop/cycle</th>
<th>Peak Mflop/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha 21264</td>
<td>677</td>
<td>2</td>
<td>1354</td>
</tr>
<tr>
<td>Alpha 21164</td>
<td>600</td>
<td>2</td>
<td>1200</td>
</tr>
<tr>
<td>Power 3</td>
<td>233</td>
<td>4</td>
<td>932</td>
</tr>
<tr>
<td>Sparc</td>
<td>450</td>
<td>2</td>
<td>900</td>
</tr>
<tr>
<td>PIII</td>
<td>550</td>
<td>1</td>
<td>550</td>
</tr>
<tr>
<td>R10K</td>
<td>250</td>
<td>2</td>
<td>500</td>
</tr>
</tbody>
</table>

Linpack

- The Linpack benchmark solves a dense linear algebra problem -- BLAS 3
- Can be run in serial or parallel
- Because BLAS 3 can be blocked, Linpack effectively runs in cache and gets a very high percentage of peak.
- Linpack is important for two reasons:
  - Good basic test of whether a machine (parallel) can run or not
  - Basis of Top 500 list (www.top500.org)
STREAM Benchmark

- Measures memory bandwidth
- Developed by John McCalpin
- 4 Tests
  - Copy (A = B)
  - Scale (A = s*B)
  - Add (A = B + C)
  - Triad (A = B + s*C)

Stream Results

<table>
<thead>
<tr>
<th>Processor</th>
<th>MHz</th>
<th>Peak (Mflop/s)</th>
<th>Triad (2*MW/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha 21264</td>
<td>500</td>
<td>1000</td>
<td>331</td>
</tr>
<tr>
<td>Alpha 21164</td>
<td>533</td>
<td>1066</td>
<td>73</td>
</tr>
<tr>
<td>Pentium II</td>
<td>400</td>
<td>400</td>
<td>79</td>
</tr>
<tr>
<td>Ultrasparc (UE10K)</td>
<td>400</td>
<td>400</td>
<td>74</td>
</tr>
<tr>
<td>MIPS (O2K)</td>
<td>300</td>
<td>600</td>
<td>48</td>
</tr>
<tr>
<td>Power-3</td>
<td>200</td>
<td>800</td>
<td>~250</td>
</tr>
<tr>
<td>Cray C90</td>
<td>1000</td>
<td>2375</td>
<td></td>
</tr>
</tbody>
</table>
SPEC95

- SPEC = Standard Performance Evaluation Corporation
- http://www.spec.org

- SPECint95
  - 8 integer-intensive codes written in C
- SPECfp95
  - 10 floating point-intensive codes written in Fortran
  - All are scientific computations.

SpecFP 95

<table>
<thead>
<tr>
<th>Processor</th>
<th>MHz</th>
<th>SPECfp95</th>
<th>SPECint95</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha 21264</td>
<td>500</td>
<td>48.4</td>
<td>23.6</td>
</tr>
<tr>
<td>Power 3</td>
<td>200</td>
<td>27.6</td>
<td>12.5</td>
</tr>
<tr>
<td>Ultrasparc</td>
<td>450</td>
<td>27.0</td>
<td>19.7</td>
</tr>
<tr>
<td>MIPS</td>
<td>250</td>
<td>23.2</td>
<td>15.1</td>
</tr>
<tr>
<td>Athlon</td>
<td>650</td>
<td>22.4</td>
<td>29.4</td>
</tr>
<tr>
<td>PIII/500</td>
<td>500</td>
<td>15.1</td>
<td>21.6</td>
</tr>
<tr>
<td>Alpha 21164</td>
<td>533</td>
<td>14.1</td>
<td>16.8</td>
</tr>
</tbody>
</table>
Multiprocessor machines

- x86/Alpha/Sparc/MIPS all available in SMPs
  - Cache coherent shared memory
  - Single copy of operating system
  - Well-supported by Linux up to about 4 processors

- OS support is not the limiting factor. Memory bandwidth is.
  - Low-end SMPs share memory through a bus
  - Nearly saturated by one processor. Two or more processes compete for memory bandwidth.
  - Expect 1.5x speedup max on Intel or current Alpha.

Software

- Compilers
- Libraries
- 3rd party software
Compilers

- Old standbys, available on all platforms
  - C: gcc
  - C++: g++
  - Fortran 77: g77

- Open source but:
  - g++ doesn’t handle complex C++ (e.g. heavy use of expression templates)
  - g77 is Fortran 77 only
  - no parallelization for SMPs
  - generated code is not very fast

x86 Compilers

- **Portland Group** (www.pgroup.com)
  - Fortran 90/95/OpenMP parallelization/Better performance (~10%)/HPF

- **Kuck and Associates** (www.kai.com)
  - Better C++/OpenMP parallelization

- **NAG** (www.nag.com)
  - Fortran 90/95/Tends to be picky

- **Absoft** (www.absoft.com)
  - F90/95/Includes IMSL (RH 5.2?)

- **Fujitsu** (www.tools.fujitsu.com)
  - C/C++/F90/F95
Alpha compilers

- Compaq/DEC compilers are available
  - Better performance (optimized for Alpha)
  - Full Fortran 90 (available now)
    - C/C++ later
- NAG
  - Fortran 95

Other compilers

- Ultrasparc
  - Nothing more available
- MIPS
  - Nothing more available
- Power-3
  - IBM is looking into putting AIX compilers under Linux
NAS Parallel Benchmarks

- Developed at NASA Ames Numerical Aerodynamic Simulation facility.
- Designed to measure performance of parallel computers
- 8 codes: 5 kernels and 3 pseudo-applications represent a CFD workload.
- 5 sizes: S, W, A, B, C.
- Two versions
  - NPB 1: pencil and paper (algorithm specified)
  - NPB 2: specified by source code
- NAS Serial Benchmarks (NPB 2-serial) are single processor versions of NPB 2.

A Few NSB results

<table>
<thead>
<tr>
<th>Proc</th>
<th>MHz</th>
<th>Cmplr</th>
<th>OS</th>
<th>FP Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>21264 ds20</td>
<td>500</td>
<td>DEC</td>
<td>Tru64</td>
<td>182.1</td>
</tr>
<tr>
<td>21264 ds10</td>
<td>466</td>
<td>DEC</td>
<td>Tru64</td>
<td>141</td>
</tr>
<tr>
<td>21264 xp</td>
<td>500</td>
<td>DEC</td>
<td>Tru64</td>
<td>154.1</td>
</tr>
<tr>
<td>21264 xp</td>
<td>500</td>
<td>DEC</td>
<td>Linux</td>
<td>132.1</td>
</tr>
<tr>
<td>21264 xp</td>
<td>500</td>
<td>gcc</td>
<td>Linux</td>
<td>100.0</td>
</tr>
<tr>
<td>21164</td>
<td>600</td>
<td>DEC</td>
<td>Tru64</td>
<td>65.9</td>
</tr>
<tr>
<td>PII</td>
<td>400</td>
<td>PGI</td>
<td>Linux</td>
<td>53.4</td>
</tr>
<tr>
<td>Celeron</td>
<td>400</td>
<td>PGI</td>
<td>Linux</td>
<td>45.1</td>
</tr>
</tbody>
</table>

- see http://www.nersc.gov/research/ftg/pcp/performance.html
Basic Free Numerical Libraries

There are many free libraries. Some of the more important (and industrial strength) ones are:

- Optimized BLAS for x86
  - http://www.cs.utk.edu/~ghenry/distrib

- FFTW: Fastest Fourier Transform in the West
  - http://www.fftw.org

Don’t use numerical recipes!

Basic libraries - Commercial

- X86
  - NAG (www.nag.com).
  - IMSL (www.vni.com/products/imsl)

- Alpha
  - Compaq Portable Math Library (CPML) -- libm replacement
  - Compaq Extended Math Library (CXML)
More software

Two excellent sources of information.

- Scientific Applications on Linux at Kachinatech: http://sal.kachinatech.com
- Steven Baum’s Linux List http://stommel.tamu.edu/~baum/linuxlist/linuxlist/node6.html

Parallelism

- 2 types of concurrency in parallel applications
  - Embarassing parallelism
    - Little/no coupling between tasks
    - Independent processes can be executed in parallel
    - seti@home; analysis of event data from colliders; monte carlo simulations.
  - Everything else
    - parallelism is fine-grained
    - data distribution is fine-grained
    - frequent communication
    - main application focus of the rest of this talk
Parallelism

Three viable programming models
- Compiler-generated parallel code
  - SMP only
  - Not (yet?) widely used with Linux
- Threads
  - SMP only
  - Not widely used for scientific computing
- Message passing
  - Distributed memory or SMP
  - Widely used on clusters
- Non-viable alternatives: HPF, distributed shared memory

Compiler parallelization

- Compiler detects concurrency in loops and distributes work in a loop to different threads.

```c
for (i = 0; i < 1000000; i++)
    a[i] = c[i] * (b[i+1] - 2b[i] + b[i-1]);
```
- Requires cache-coherent shared memory in general
- Compiler is usually assisted by compiler directives.
- OpenMP is the standard for Fortran and C
  - KAI
  - Portland group
Message Passing

- Programming model:
  - Separate processes with separate address spaces
  - Communication by cooperative send/receive
  - Mixed MPI/threads possible in theory, but not supported in Linux implementations.
- MPI (Message Passing Interface) is the industry standard.
- PVM should be used only when MPI can’t do the job.
- Hardware
  - Distributed memory (cluster)
  - Shared memory
  - Mix of shared/distributed

Clusters

- A *cluster* is a collection of interconnected computers used as a unified computing resource. (Pfister)
- Clusters can offer
  - High performance
  - Large capacity
  - High availability
  - Incremental growth
- Clusters used for
  - Scientific computing
  - Making movies
  - Commercial servers (web/database/etc)
“Beowulf” clustering

- Clustering of x86-based Linux machines for scientific computing was popularized by the Beowulf project at Caltech/JPL.

- “Beowulf-class” is a slippery term, but usually implies:
  - Off-the-shelf parts
  - Low cost LAN
  - Open source OS

- National labs are looking at highly-scalable non-beowulf clusters for next generation of supercomputing.

How to build a cluster

- Building/maintaining a cluster is a lot of work

- Type of cluster depends on the type of job.
- Tightly coupled applications have more stringent requirements.

- Expect a flood of software and documentation to appear over the next year that makes it much easier to put together clusters.
Architecture

Network setup

- Private network
  - Cluster security/setup/administration much easier
  - Application cannot interact with outside world

- Public network
  - Security/setup/administration difficult. IP addresses needed.
  - Interaction possible

- Firewall
  - Most flexible
  - Experts only
Local install or diskless?

- **Local install**
  - Most natural if you’re used to installing desktops
  - N separate copies of Linux to maintain
  - Works best in completely homogeneous system

- **Diskless install**
  - 1 copy of Linux to maintain
  - Requires special tools to manage
  - For many applications, scales up to 32 or 64 nodes

Node classification

- **Interactive nodes**
  - Attached to external network
  - Compile/edit/debug

- **Fileserver nodes**
  - Global file systems (e.g. home directories)
  - Remote filesystems for diskless clients

- **Other service nodes**
  - Batch server/YP server/Security server

- **Compute nodes**
  - Space-shared by parallel applications
Other cluster infrastructure

- YP (NIS) for user management
- BOOTP for IP address management
- Global filesystem.
  - Necessary and expected
  - Most important unsolved problem of clusters.
  - No viable solution except NFS
  - See http://pdsf.nersc.gov/talks/nfs/index.html

Other Hardware

- Network
  - Fast ethernet. By far the most common.
  - Gigabit ethernet. Expensive, not much faster
  - New networks on horizon: Giganet, Servernet II
  - Virtual Interface Architecture may make high performance networks more accessible and available.

- Serial console management
  - Cyclades, Rocketport (comtrol.com) multiport serial cards
Other software

- MPI
  - Get MPICH from http://www.mcs.anl.gov
  - LAM is another free implementation, but no compelling reason to use it.

- PBS
  - Batch management system developed at NASA Ames
  - Space shares the cluster; manages multi-user system
  - Easily integrated with MPICH
  - http://pbs.mrj.com

Task Farms

How would you do things differently for a task farm (embarrassingly parallel application)?

- Consider MOSIX to transparently load balance processes
- Switched network not necessary
Good news

More good news
The Network Matters

More info on clusters

• **How to Build a Beowulf.** Sterling Becker, et. al.
  MIT Press, 1999

• **In Search of Clusters.** Gregory Pfister.
  Prentice Hall, 1998 (2nd edition)

• The Beowulf mailing list: “subscribe” to
  beowulf-request@beowulf.gsfc.nasa.gov

• HOWTO:
  http://www.beowulf-underground.org/doc_project/index.html
Open source presentation

http://www.nersc.gov/~wcs