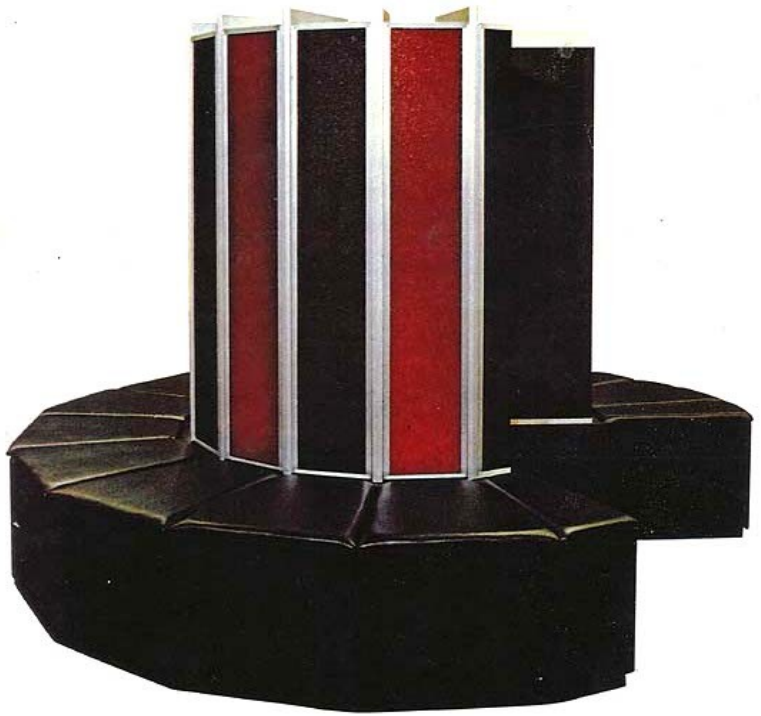
Linux and High-Performance Computing



The Cray 1, early vector processor and universal icon of supercomputing. (Thanks, Cray 1 Hardware Manual Cover)

Outline

- Architectures & Performance Measurement
- Linux on High-Performance Computers
  - Beowulf Clusters, ROCKS
  - Kitten: A Linux-derived LWK
  - Linux on I/O and Service Nodes
- Free Software for Parallel Computing
  - Resource Management
  - MPI (Message Passing Interface)
  - OpenMP

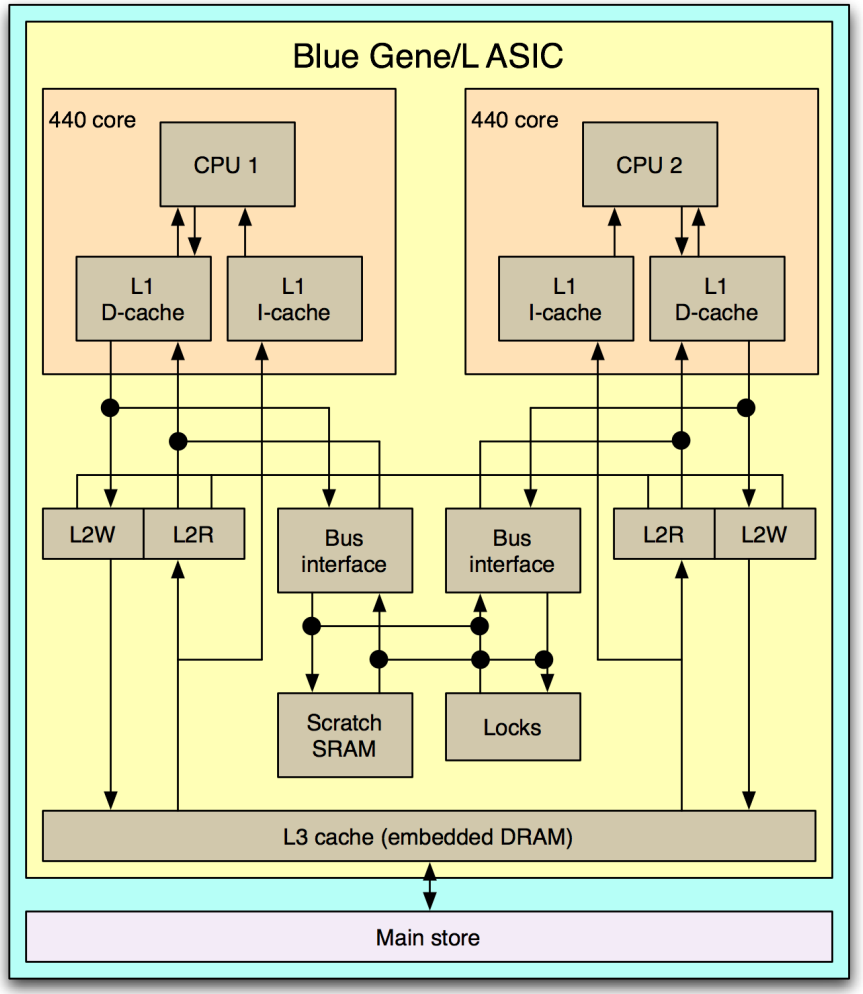
Performance Measurement

- Linpack FLOPs are the performance standard used for consideration in the TOP500
  - The specific benchmark is HPL
  - Asks “how many FLOPs do you get while solving an NxN series of linear equations, starting at N=1000?”
  - Linpack is a commonly targeted application for two reasons:
    - It is useful in engineering.
    - Being on the TOP500 is even more useful in marketing.

Architecture: Where does the performance come from?

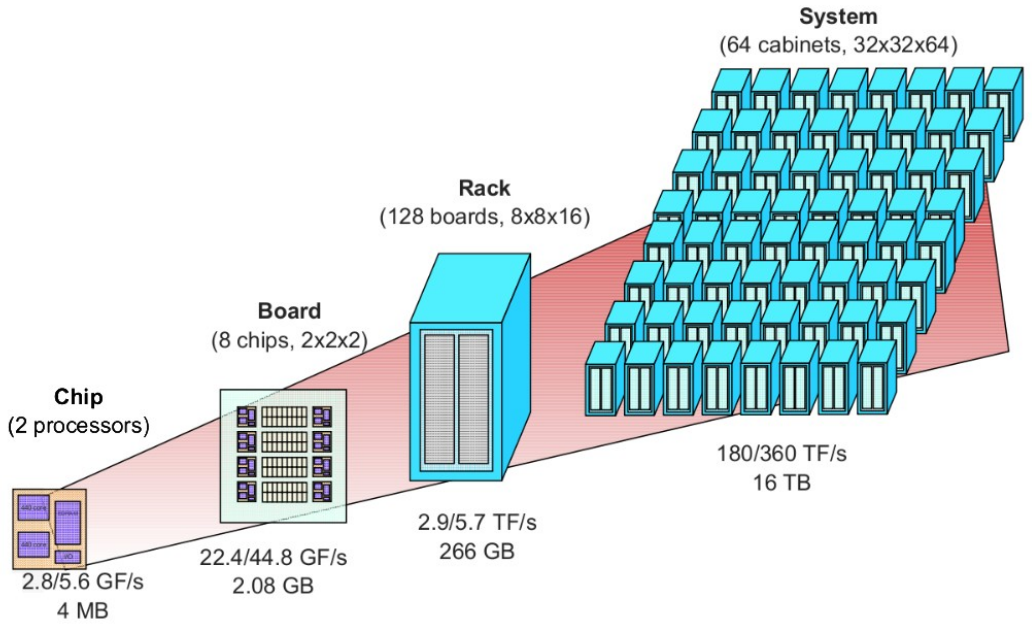
Node-level

- Clock rate (arguably the least important.)
- CPU Microarchitecture (OoO? SMT? How many pipeline stages?)
- Memory architecture (Cache? How many CPUs share memory? NUMA/NUCA/ COMA?)
- ALU Features (SIMD, vector units)
- Software Features (How good is the compiler? Random timing fluctuations in OS?)



Single processor chip from a Blue Gene/L. (Thanks, Wikipedia)

Architecture: Where does the performance come from?



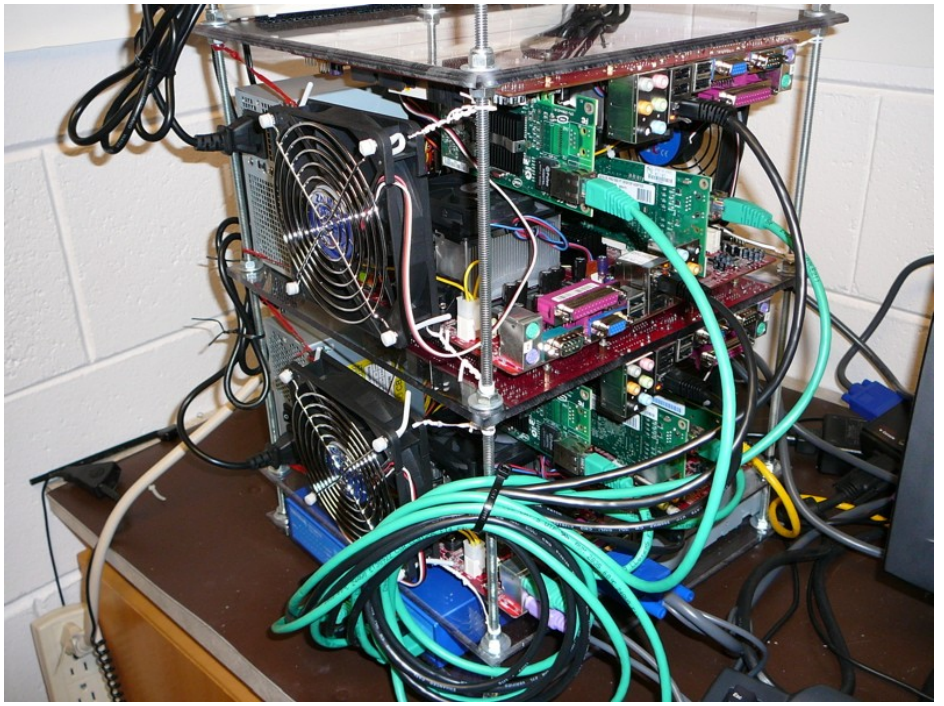
Blue Gene/L System (Thanks, IBM)

System-Level

- Network
  - Latency
  - Bandwidth
  - Topology
- Power
  - What fraction of cores can be powered at a given time?
- Memory system (again)
  - Is memory distributed? If so, what are the latencies?

The Low End: Clusters

- Normal PC hardware on low-latency (or even Ethernet) backplane.
- Often running straight Linux distribution.
- Typically run MPI-based parallel applications (most supercomputers do, too).
- Least FLOPs/watt (doesn't scale well)
- Greatest FLOPs/dollar (inexpensive entry-level HPC)



Microwulf: a “personal, portable Beowulf cluster” (thanks Calvin College)

The High-End: Supercomputers

Cray XMT, a modern supercomputer architecture.(Thanks, Cray)

- Can have custom processor cores (like Cray XMT) or commodity parts (like Red Storm)
- At the very least, have custom board-level design with scalability in mind.
- OS development/porting requires major effort.
  - But less effort that porting Linux to a new workstation.
- More FLOPs/watt than clusters.
- Application-matched network/memory systems.
- Greater initial purchase price.

Linux for Clusters: The Beowulf

- ROCKS: Linux distro with deployment on clusters in mind.
- Based on CentOS (formerly based on Red Hat)
- Comes with:
  - MPI distributions.
  - Resource management utilities (like Torque)
  - Installer built for cluster-wide deployment.

Lightweight Kernels

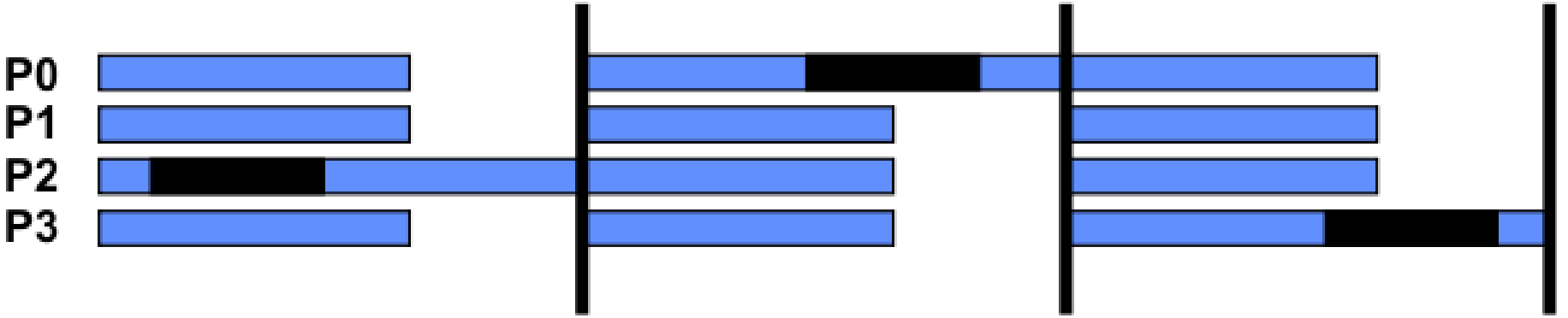
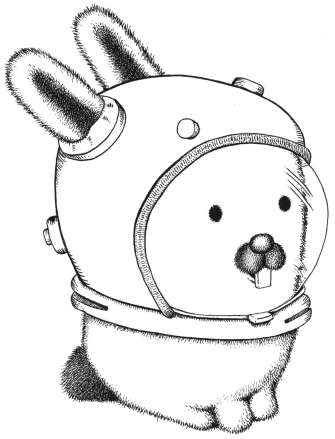


Illustration of effect of OS scheduler noise over a series of barriers.
(Thanks, Kevin Pedretti, Sandia National Labs)

- Compute nodes in supercomputers don't typically run desktop Oses.
  - Too much overhead.
  - Unpredictable behavior degrades performance (see Petrini, et al., “The Case of the Missing Supercomputer Performance”, accompanying chart)
  - Features like virtual memory aren't necessarily useful on large-scale batch systems.

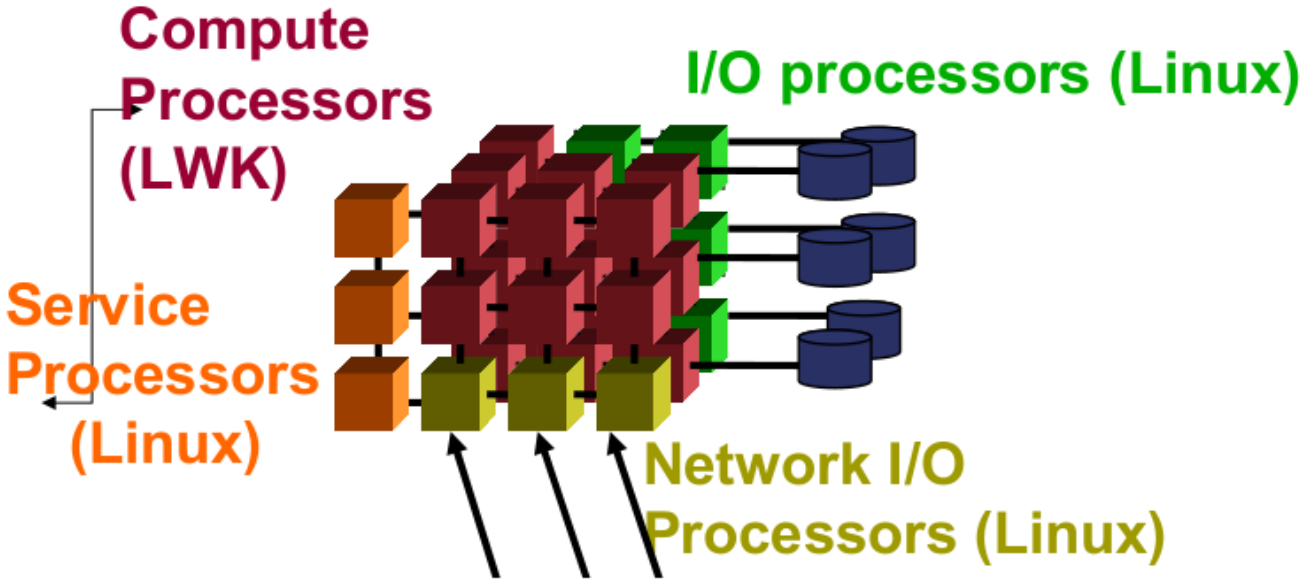
Lightweight Kernels

- Kitten: A Linux-based LWK
  - Uses Linux Kernel code “where it makes sense”; no need to rewrite code like drivers, and no need to maintain compatibility with the mainstream Linux kernel.
  - Under active development at Sandia.
  - Replacement for Catamount, the LWK used on Red Storm.



- Plan 9 from Bell Labs
  - Not Linux-based (here for contrast)
  - Microkernel architecture allows it to be run on service and compute nodes.
  - Has been ported to Blue Gene/L, but since it's Plan 9, no one cared.

Linux on I/O and Service Nodes



Example of a supercomputer with four classes of node. (Thanks Kevin Pedretti, Sandia National Labs)

- Red Storm: Runs LWK Catamount on compute nodes, uses Linux for network-facing nodes.
- Linux nodes handle external networking, job scheduling, software development environment.
  - Are also sometimes used for disk I/O
- Standard server facilities (OpenSSH, FTP) provided by Linux I/O nodes.
- LWKs don't have to provide OS features like sockets that these apps require.

Free Software Used in HPC

Resource Management

- Not much to say here. When you get access to, or have to set up, a high-performance machine, the manual for one of these will likely be at the top of the pile on your desk.
- OpenPBS (Portable Batch System)
  - Batch scheduling system first released by NASA in 1990.
- Torque
  - Considered a replacement for OpenPBS
  - Similar interface to user:
    - Jobs submitted with **qsub**, checked with **qstat**.

Free Software Used in HPC

MPI

- Framework for distributed applications.
  - Writing an MPI application is like writing a networked application, except MPI also handles the launching of multiple processes across nodes.
- Implemented entirely as a library (no compiler support needed.)
- Deals (minimally) with the fact that programming a modern supercomputer is like programming thousands of workstations.
  - But it still is like a cluster of workstations. In fact, so much like a cluster of workstations that you can actually use a cluster of workstations as a high performance machine. Thus the Beowulf.

Free Software Used in HPC

MPI: Semantics

- Each instance of a process in an MPI system is a “rank”, typically distributed one per node.
- Grunt work handled by **send** and **receive** functions:

  MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm**)**;

  MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status **)**;
- **MPI_Comm** is usually **MPI_COMM_WORLD**; **MPI_Datatype** can be one of **MPI_CHAR**, **MPI_DOUBLE**, **MPI_COMPLEX**, etc.
- Multiple sends and receives can be combined (scatter/gather I/O) into a single operation. **MPI_Send_init**, **MPI_Recv_init**, will prepare for a send or receive, and then **MPI_Startall** or similar can be used to complete the transaction.

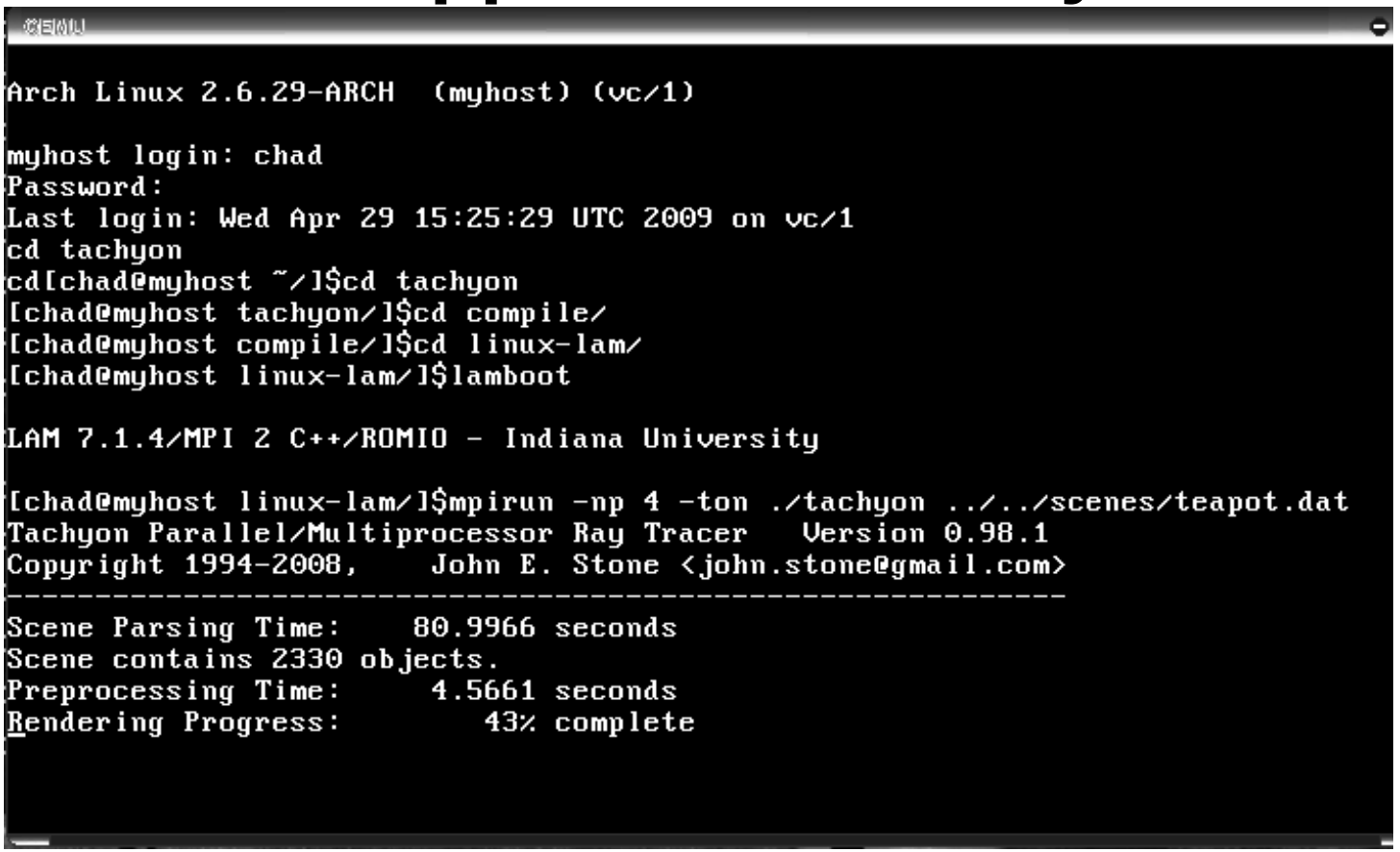
Free Software Used in HPC

MPI: Implementations

- MPICH
  - First MPI Implementation
  - Currently maintained by Argonne National Lab
  - Still commonly available on Linux distros.
- LAM/MPI
  - Originally conceived as a parallel runtime for the Transputer.
  - Another early MPI (although MPI support was ironically added later in its development)
  - Also still commonly available.
- OpenMPI
  - LAM/MPI and several other development teams combined their efforts to create OpenMPI.
  - Should be target for all new MPI development.

Free Software Used in HPC

MPI Application: Tachyon



Launching Tachyon in a VM, shows use of **mpiboot** and **mpirun**.

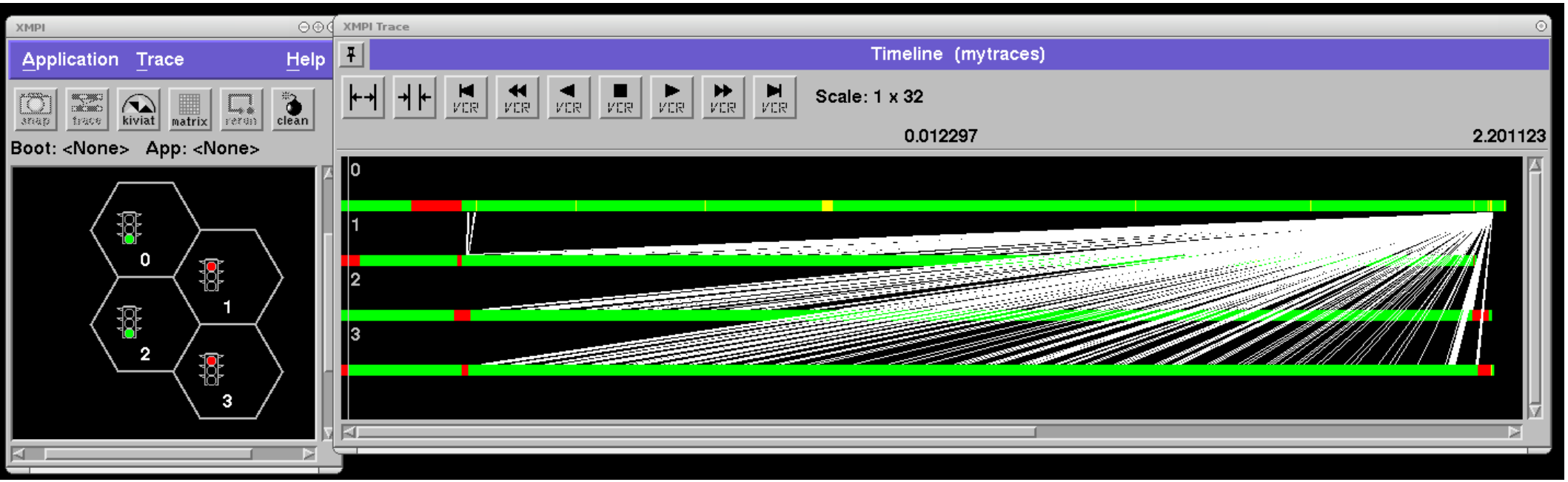
Free Software Used in HPC

MPI Application: Tachyon

The Utah teapot, rendered by Tachyon
across 4 Qemu VMs.

Free Software Used in HPC

MPI Application: Tachyon



View of Tachyon trace in XMPI. Chunks are rendered by each rank and all sent back to Rank 0, which then emits the image file. The trace can be played back with the “VCR” feature.

Free Software Used in HPC

OpenMP

- Used to parallelize applications at the node level.
- Requires compiler support.
- Includes library for high-level thread management.

Free Software Used in HPC

OpenMP: Semantics

- Annotate serial code with hints for the compiler.
  - Tell the compiler which parts are parallel and how.
- Example (with no dependencies):

```
#pragma omp parallel for schedule(dynamic, 10)
for ( int i = 0; i < 1000; i++ )
  a[i] = (b[i] + c[i])/2;
```

- Optional schedule directive gives scheduling type and chunk size.
- Also supported are thread-local and shared variables, atomic operations, and barriers.
- Much easier than pthreads for fine-grained loop parallelizations.

Free Software Used in HPC

OpenMP: GOMP

- The implementation of OpenMP in GCC.
- Includes a complete version of the runtime.