Experiences in Developing **Lightweight System Software** for Massively Parallel Systems

Barney Maccabe
Professor, Computer Science
University of New Mexico

June 23, 2008 USENIX LASCO Workshop Boston, MA

Simplicity

Butler Lampson, "*Hints for Computer System Design*," **IEEE Software**, vol. 1, no. 1, January 1984.

- Make it fast, rather than general or powerful
- Don't hide power
- **Leave it to the client**

"Perfection is reached, not when there is no longer anything to add, but when there is no longer anything to take away."
A. Saint-Exupery

MPP Operating Systems

MPP OS Research

1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005

SUNMOS
message passing

Intel Paragon

Intel ASCI/Red

Cray Red Storm

Catamount
re-engineering of Puma

LWK
direct comparison

Puma/Cougar
levels of trust

Cplant (Portals)
commodity

Sandia National Laboratories

Unified
features

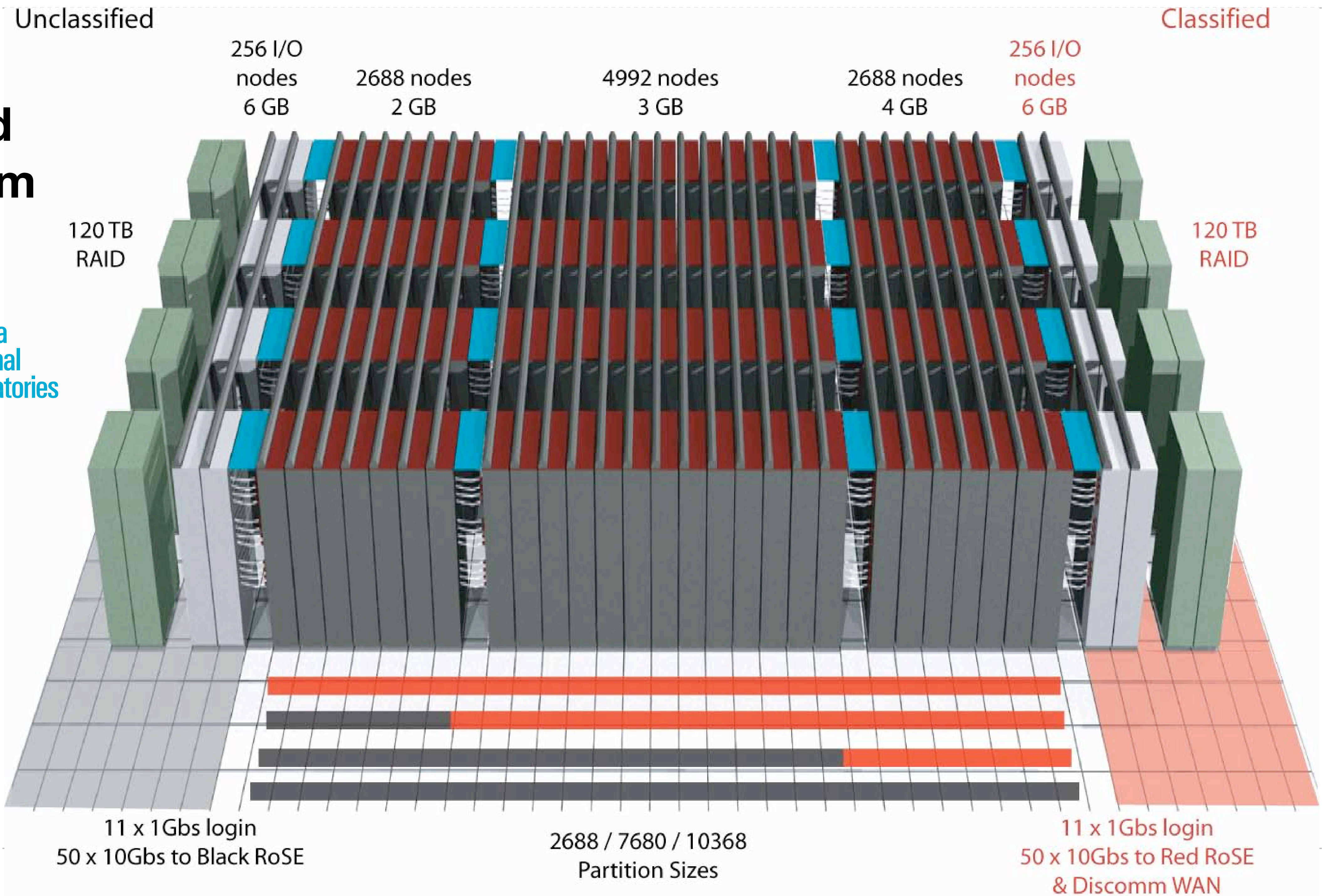
JRTOS
real-time

Cplant

Config
application driven

Partitioning for Specialization

Red Storm

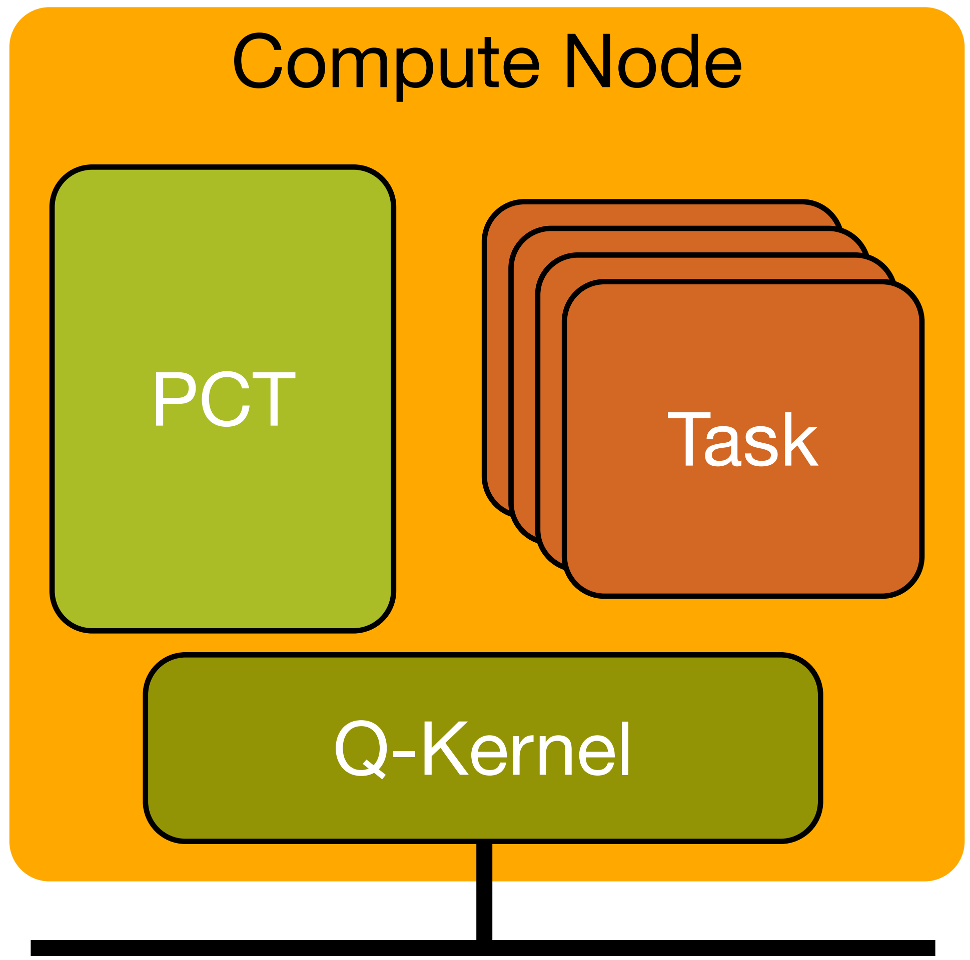


Functional Partitioning

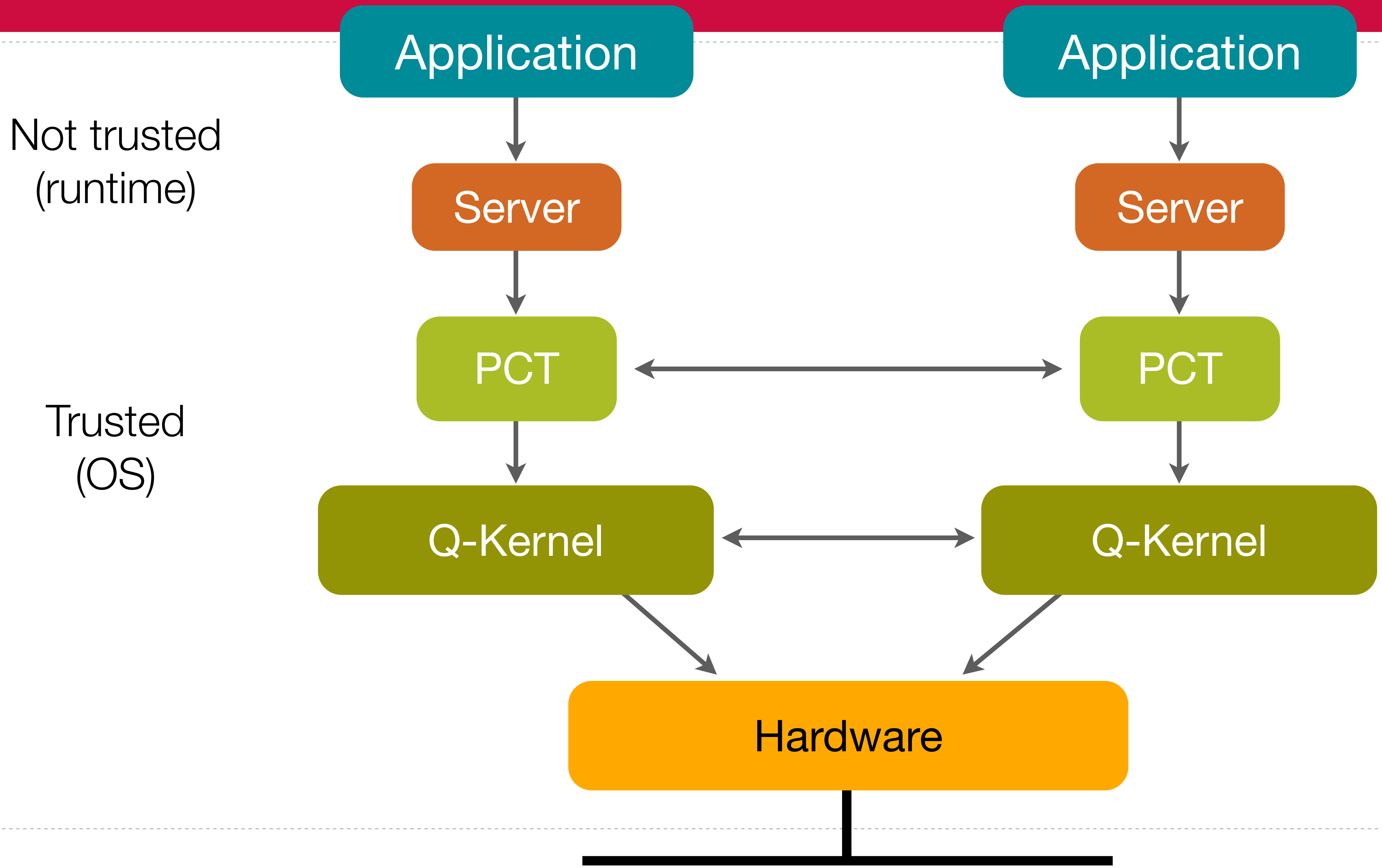
- Service nodes
  - authentication and authorization
  - job launch, job control, and accounting
- Compute nodes
  - memory, processor, communication
  - trusted compute kernel passes user id to file system
  - isolation through communication controls
- I/O nodes
  - storage and external communication

Compute Node Structure

- QK – mechanism
  - Quintessential Kernel
  - provides communication and address spaces
  - fixed size–rest to PCT
  - loads PCT
- PCT – policy
  - Process Control Thread
  - trusted agent on node
  - application load
  - task scheduling
- Applications – work



Trust Structure



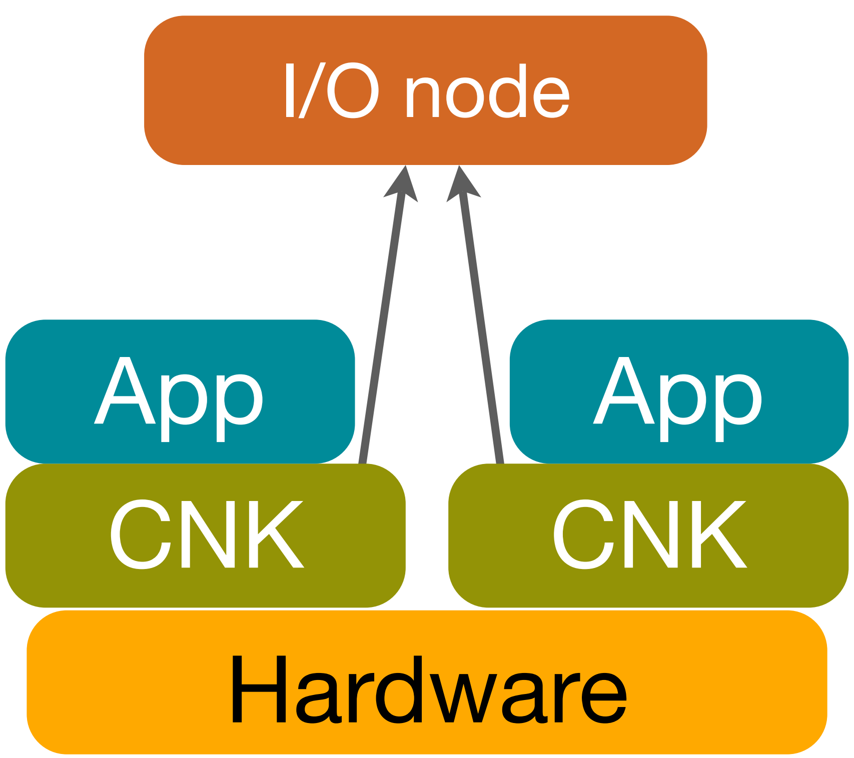
Is it good?

- It's not bad...
  - Intel Paragon 1993: 1,842 compute nodes
    - #1 6/1994–11/1994
  - Intel ASCI/Red 1997: 9,000 processors
    - First Teraflop system
    - #1 6/1997–11/2000
    - 40 hours MTBI
  - Red Storm 2005 (Cray XT3); 10,000 processors
- Other things are bad...
  - OSF-1/AD was a failure on the Paragon
  - OS noise when using full-featured kernels
    - Livermore and LANL experiences

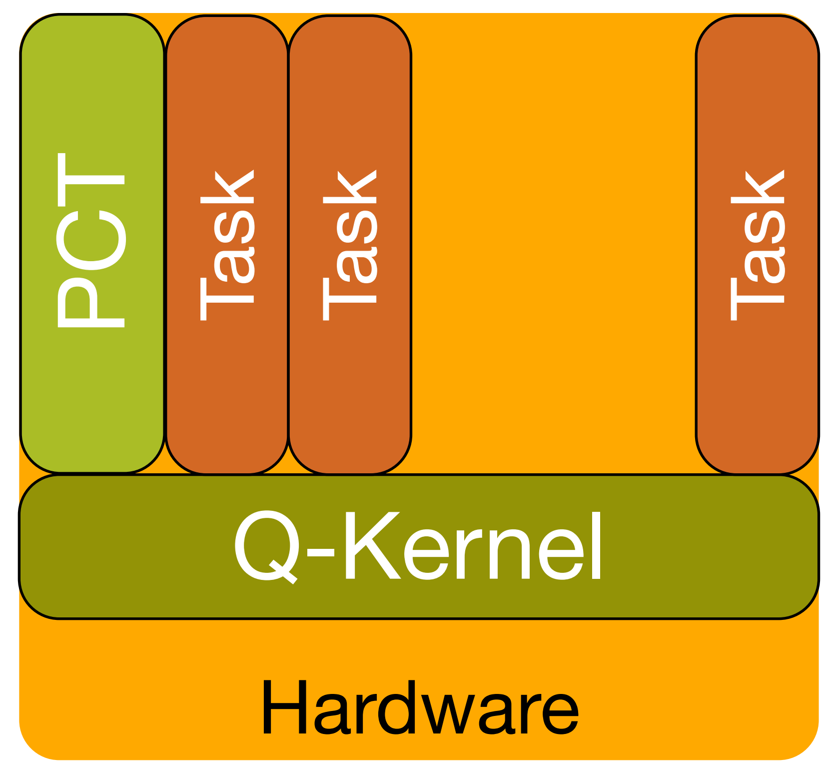
Historical problem: OS researchers only got to study broken systems at scale

Compare to Blue Gene/L

- BG/L
  - I/O nodes (servers)
  - CNK trampoline



- Catamount
  - QK = Hypervisor
  - PCT = Dom 0



OS Noise

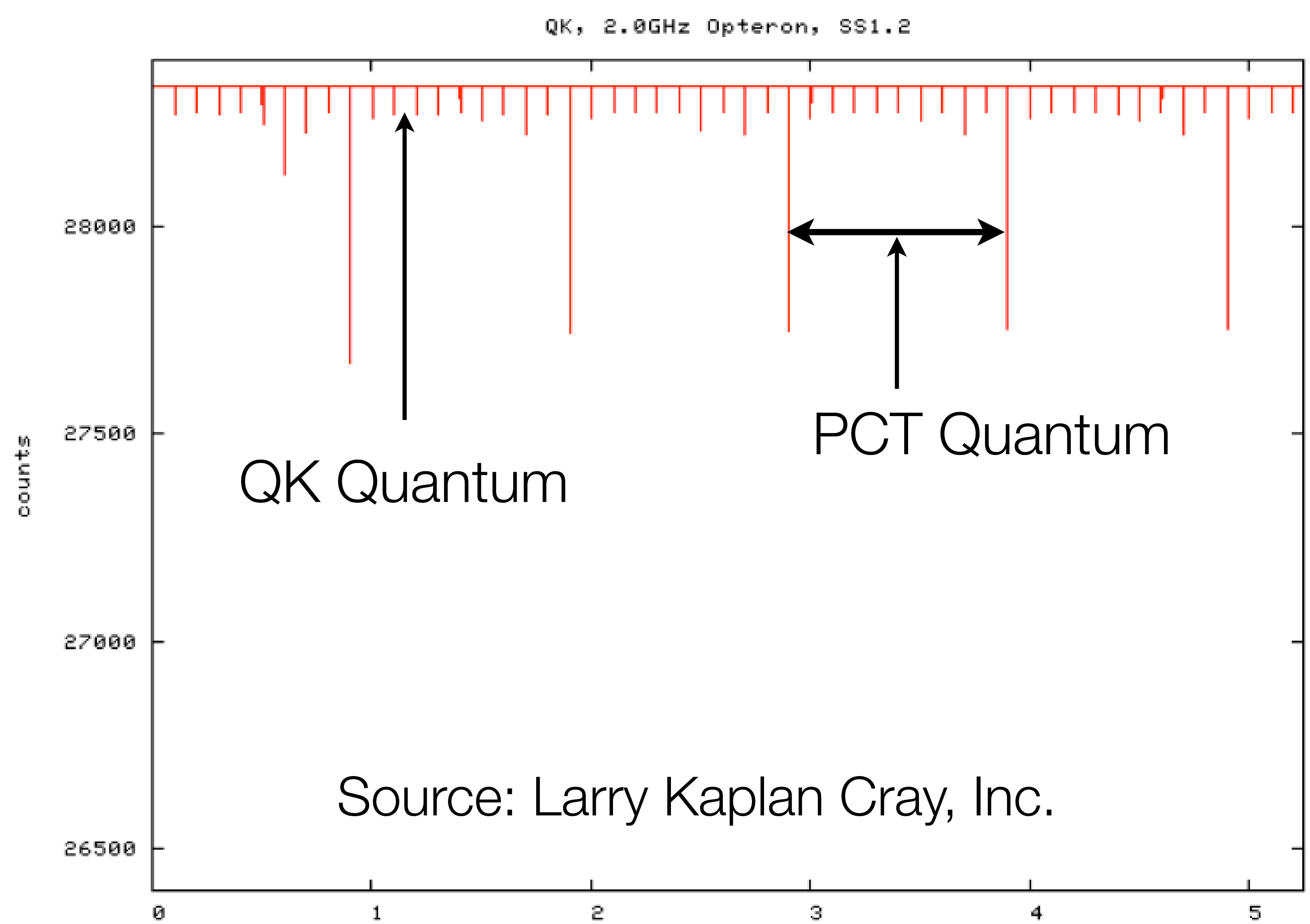
OS **Noise**

- **OS interference**: OS uses resources that the application could have used to do things not directly related to what the application is doing
    - Does not include things like handling TLB misses
    - May include message handling (if the application is not waiting)
- **OS Noise (Jitter)**: the variation in OS interference
    - Fixed work (selfish): measure variation in time to complete
    - Fixed time (FTQ): measure variation in amount of work completed
    - e.g., garbage collection–noise is usually there to do good things

FTQ (Fixed Time Quantum)

- FTQ
  - Fixed Time Quantum
  - Measure application work in fixed time quantum
  - Matt Sottile, LANL (now at Google)

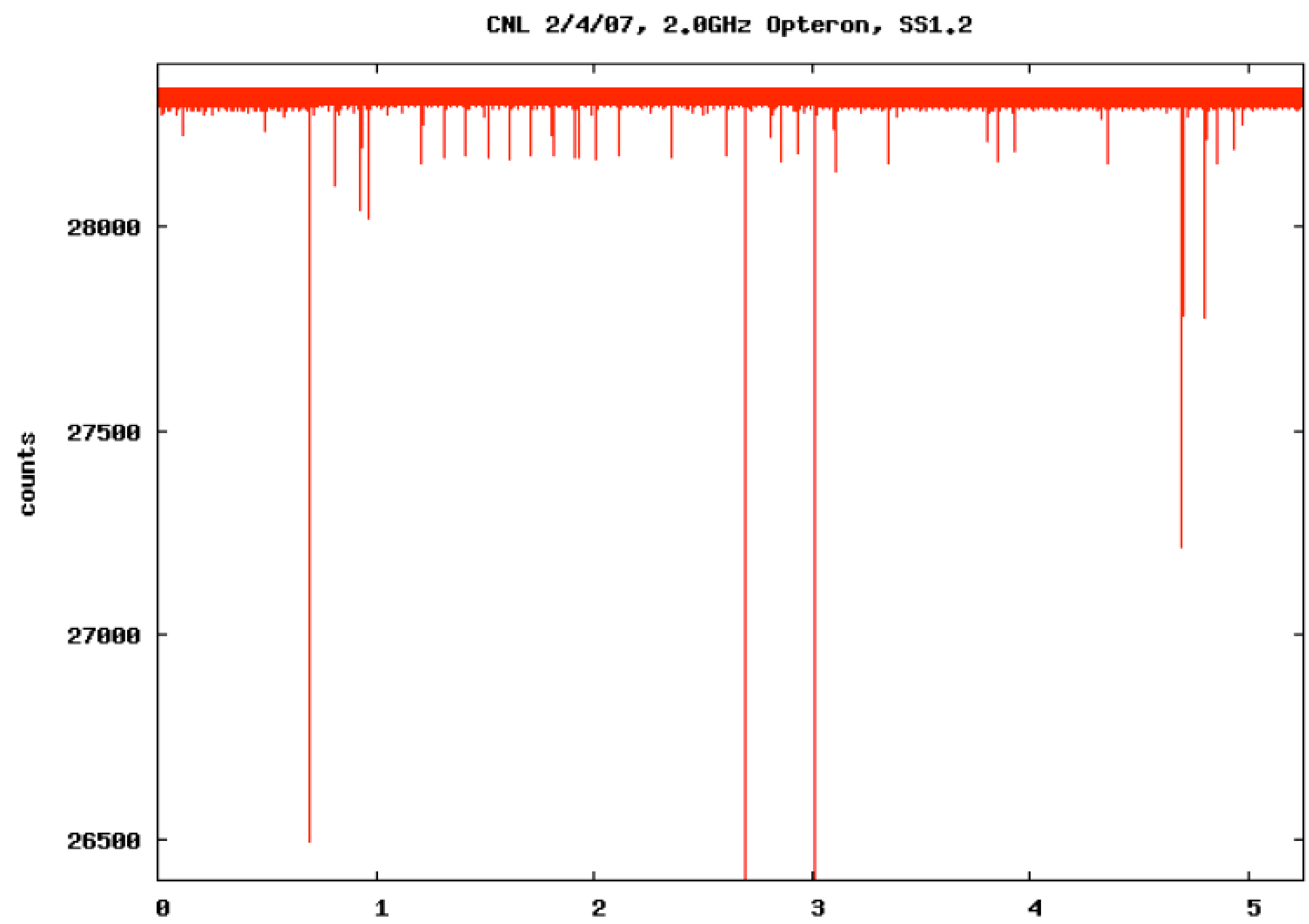
FTQ on Catamount



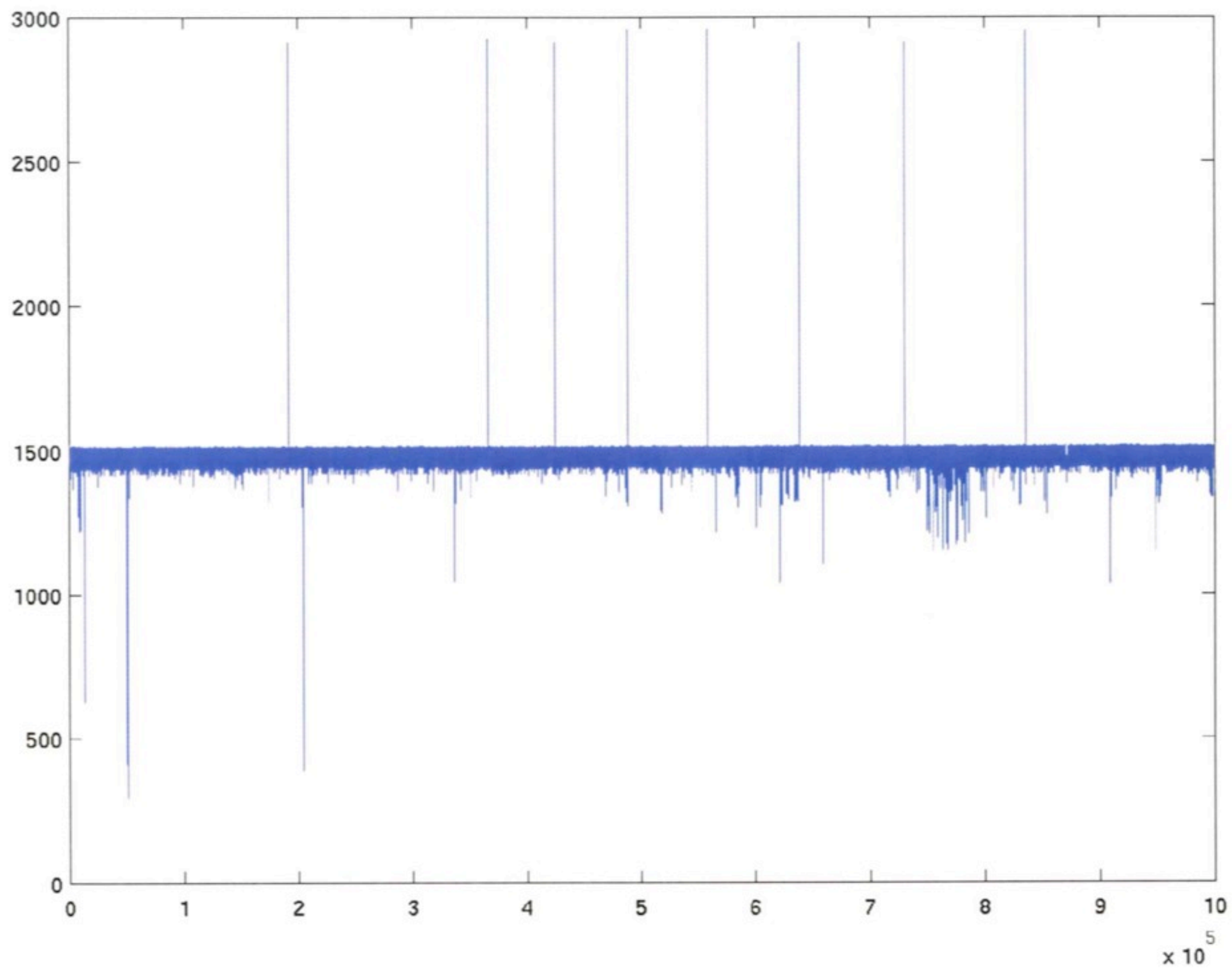
FTQ on **Linux**

Source: Larry Kaplan Cray, Inc.

FTQ on Linux

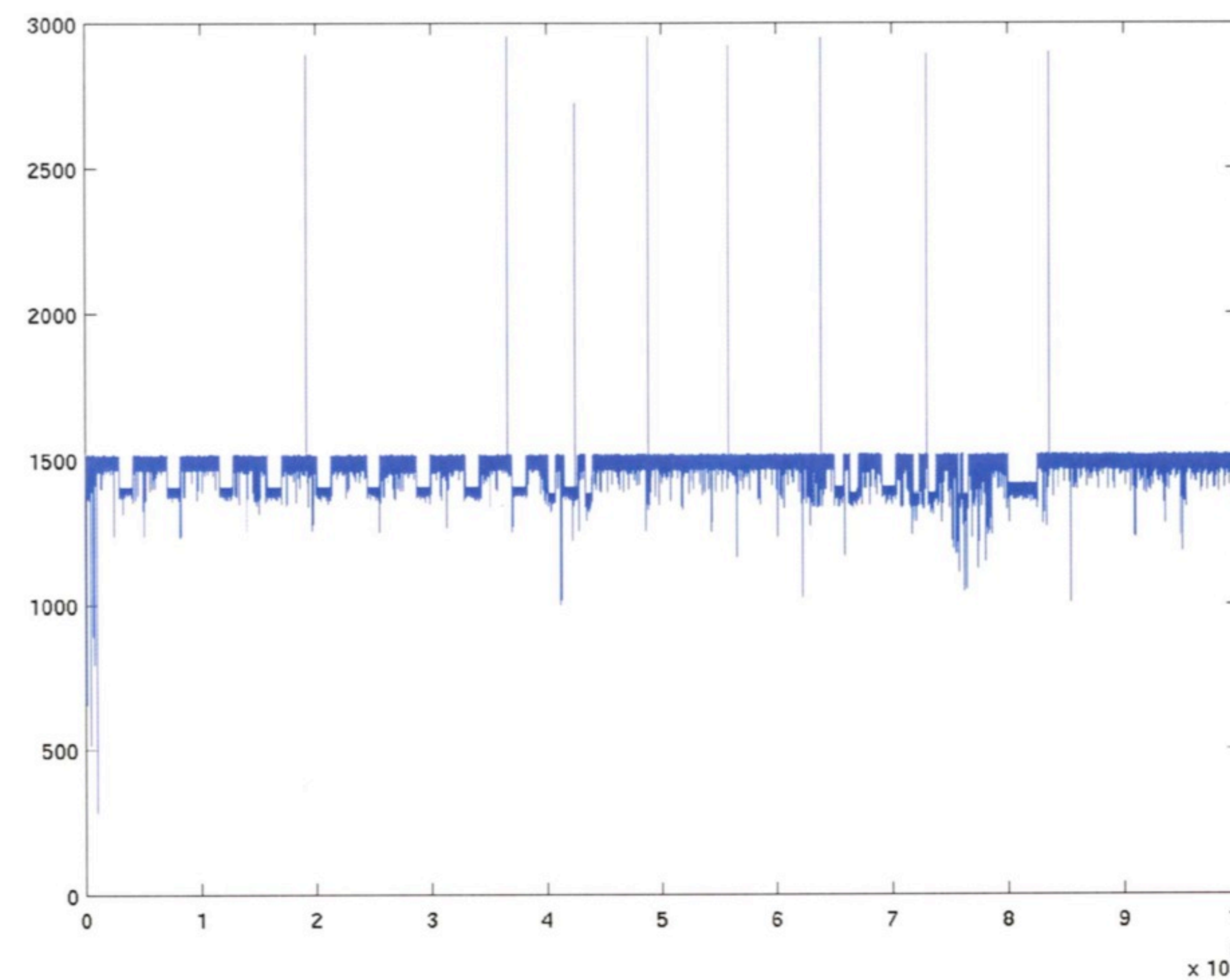


FTQ on **ASC Purple**



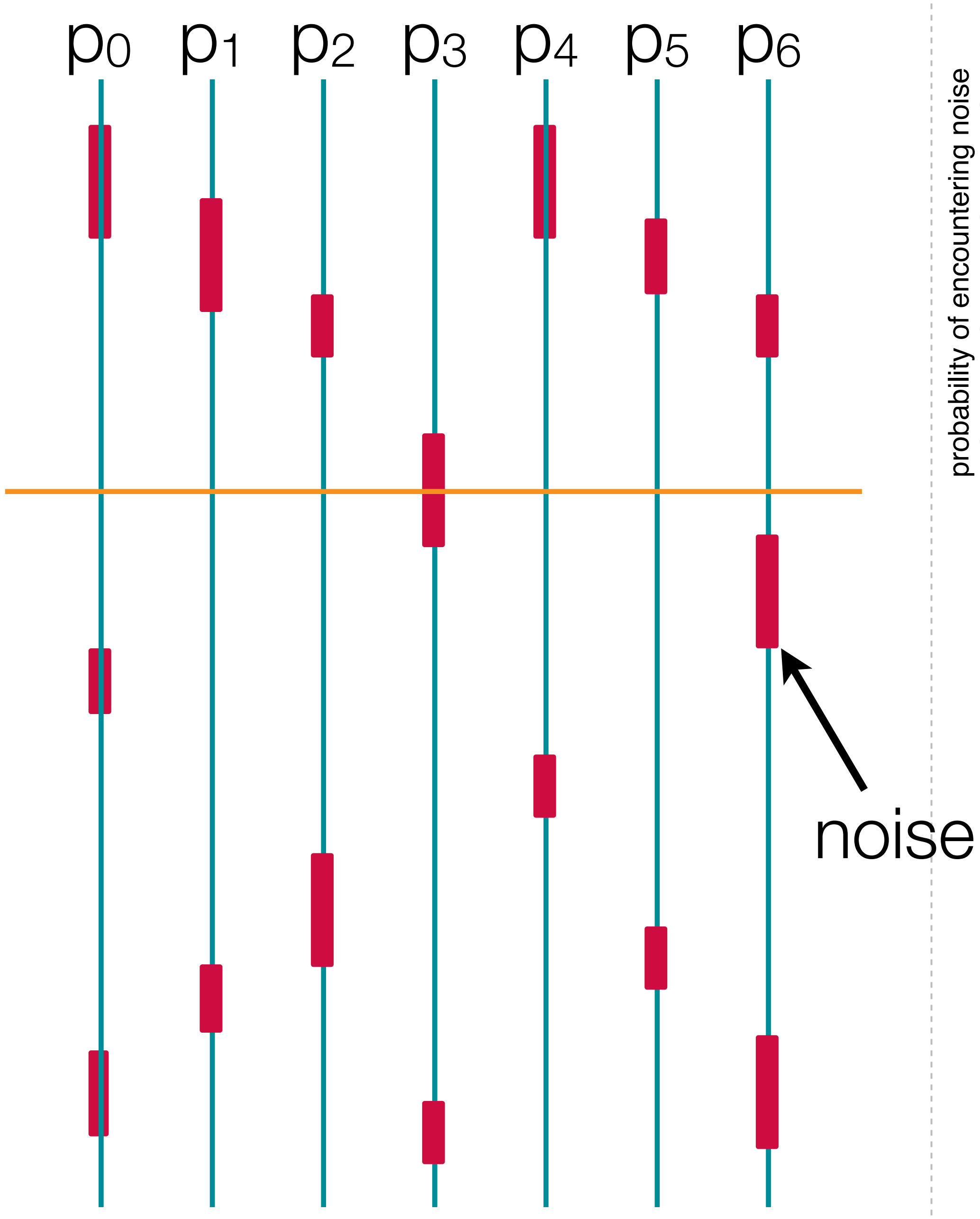
Typical FTQ

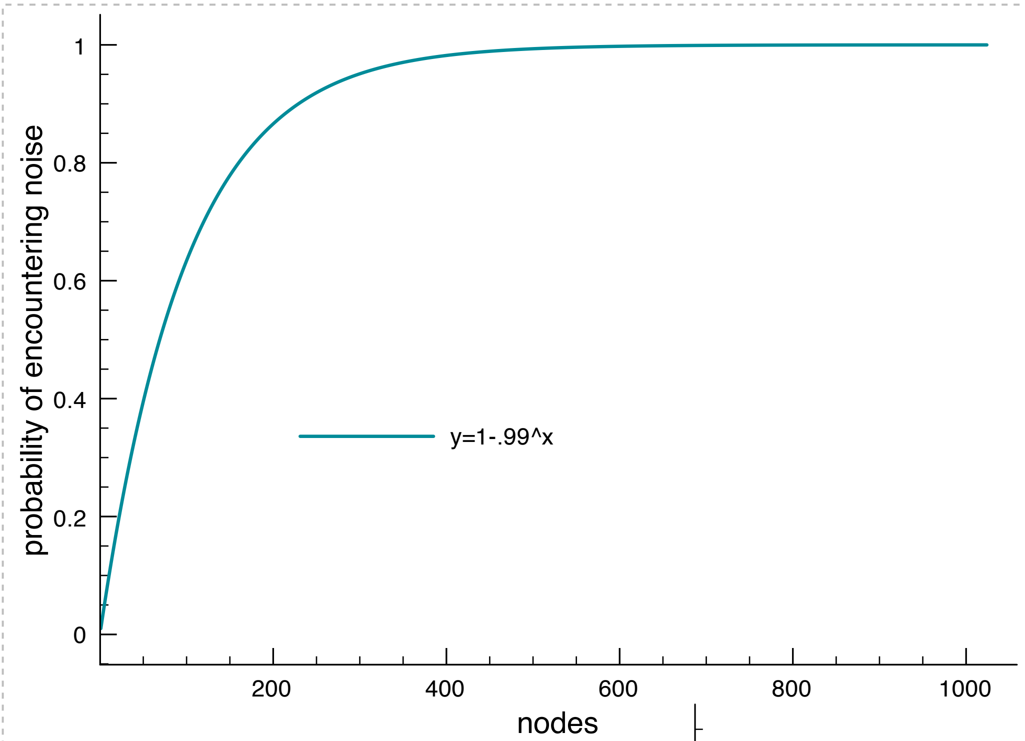
After Firmware upgrade
(VM is using cycles)

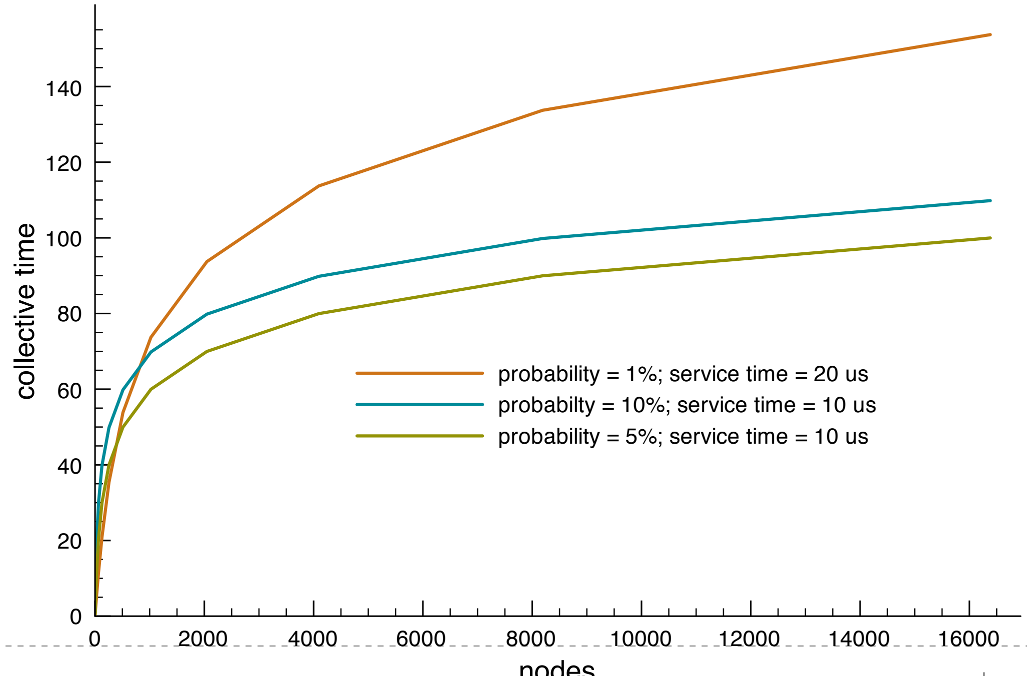


Source: Terry Jones, LLNL

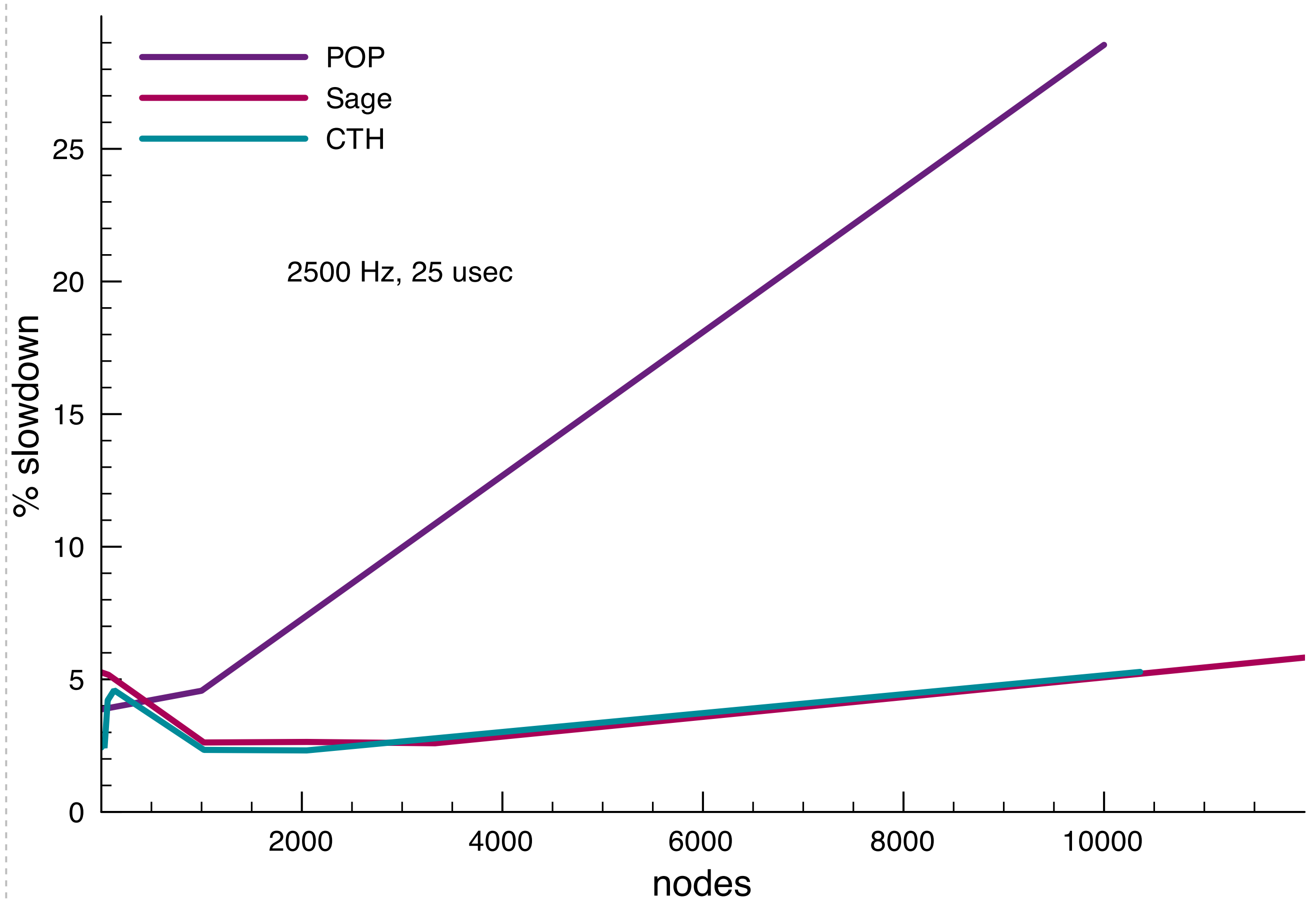
What's the big deal?



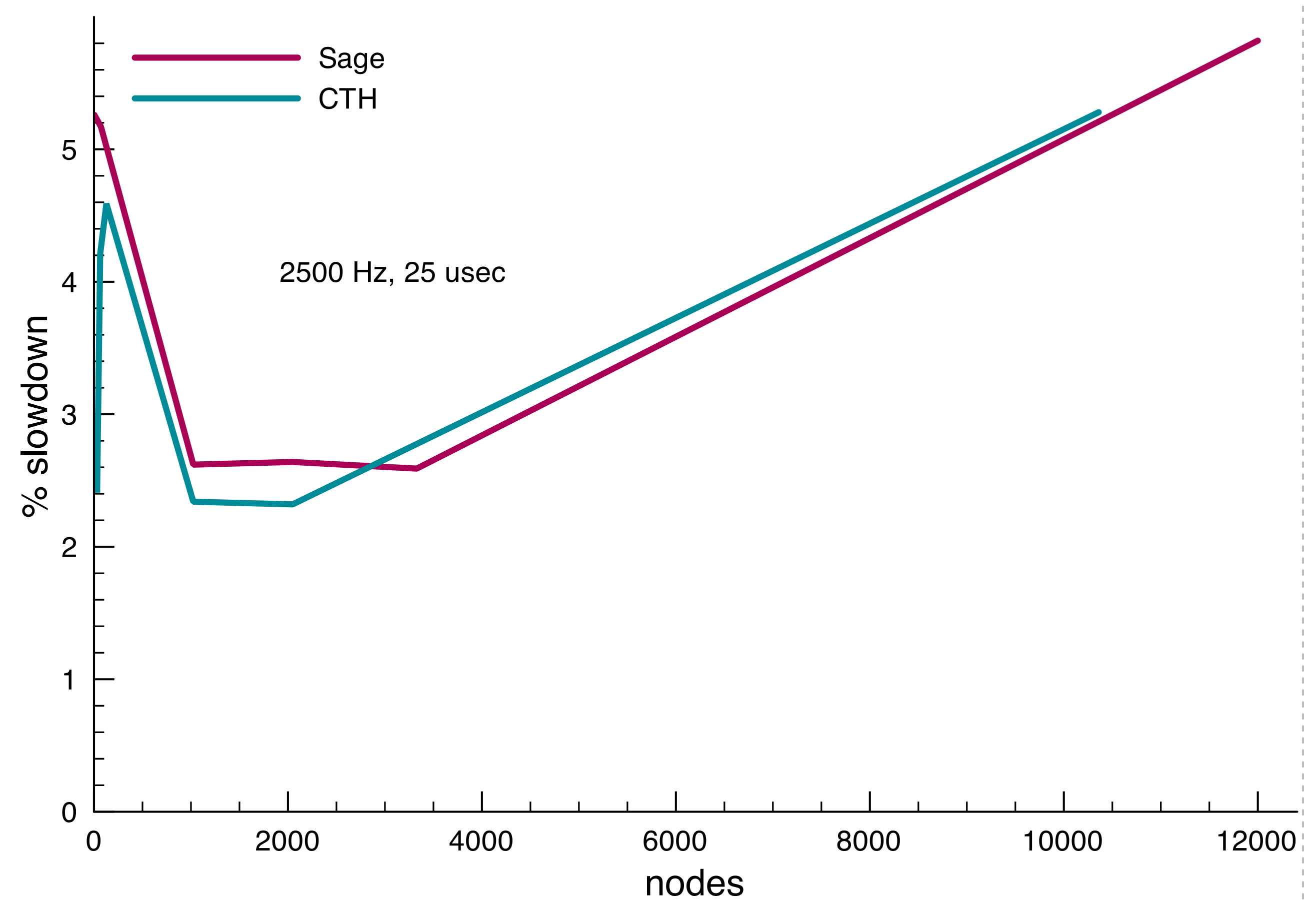




Noise does matter

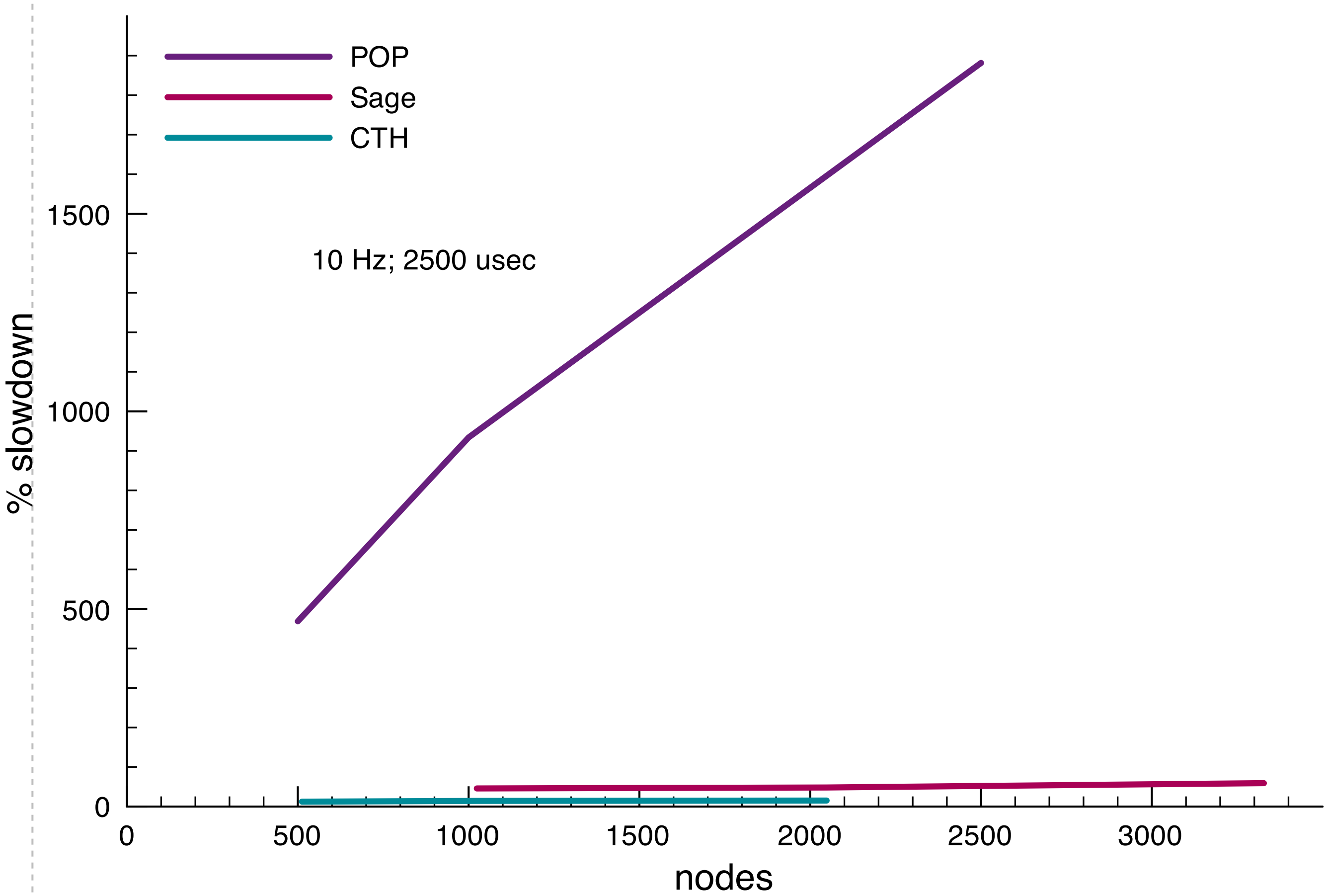


Source: Kurt Ferreira, UNM

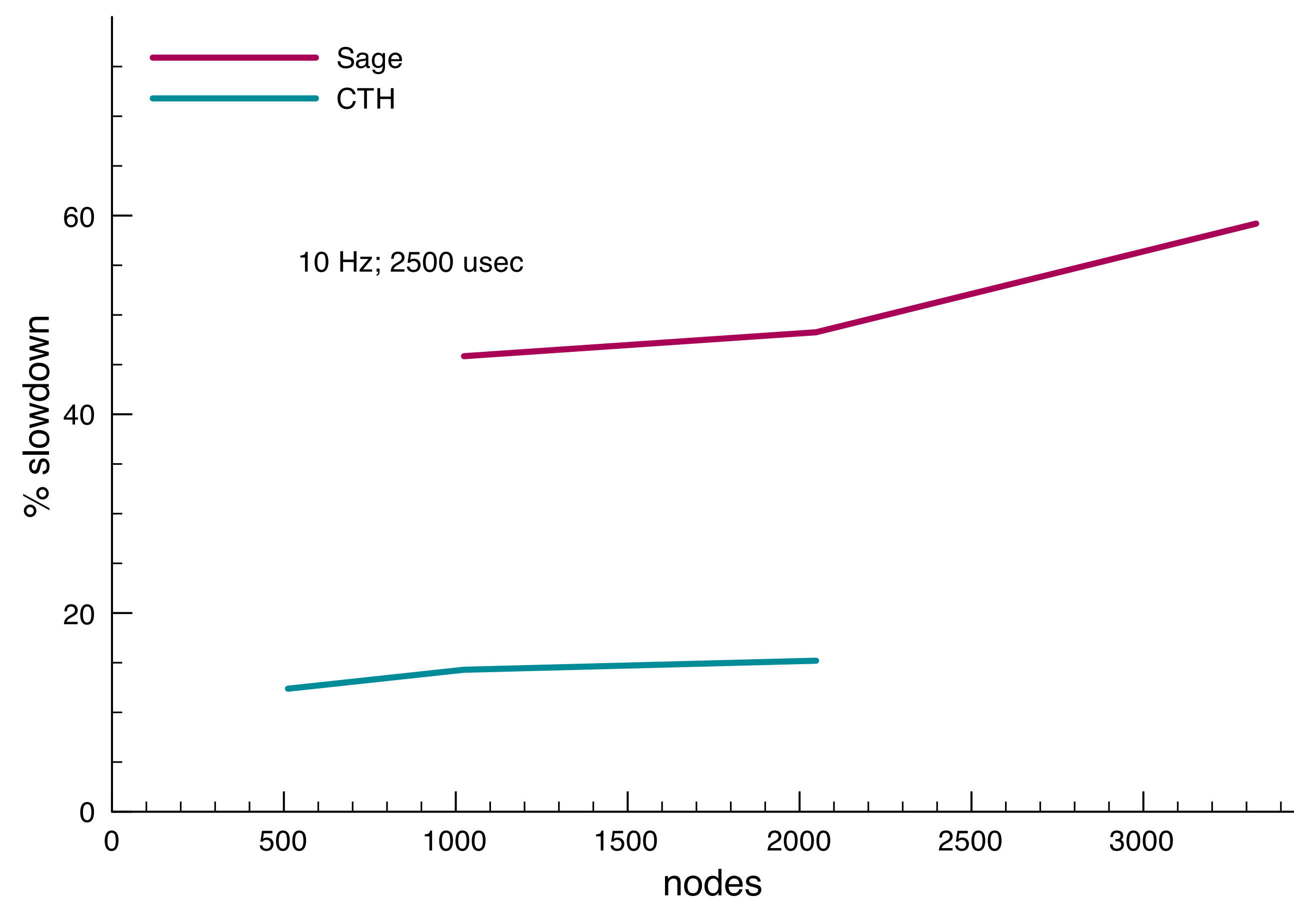


High Frequency, Low Duration
2.5% total noise injected

Noise does matter–really



Source: Kurt Ferreira, UNM



Low Frequency, High Duration
2.5% total noise injected

Dealing with noise

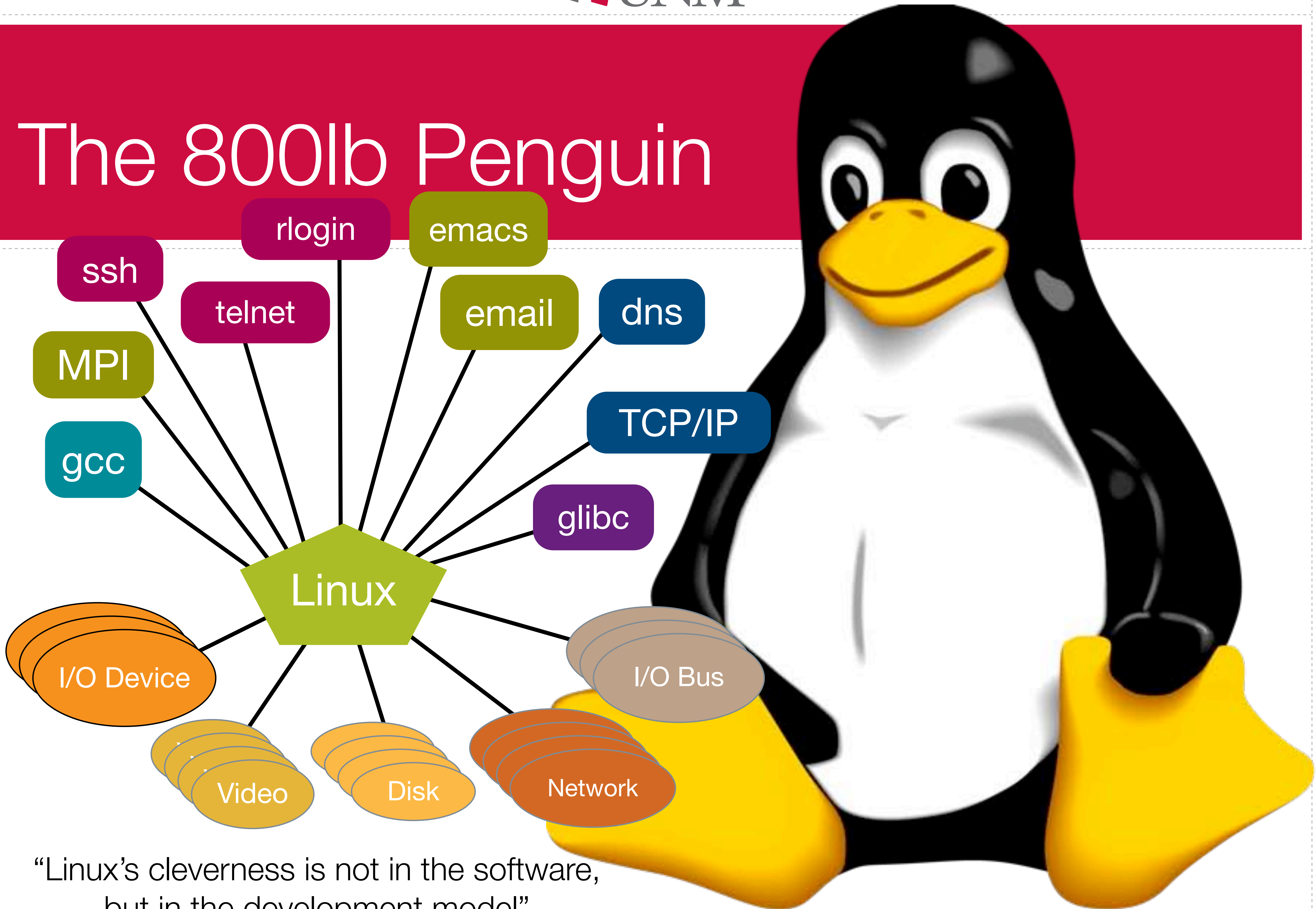
- Minimize noise
  - Lots of short noise is better than small amounts of long noise
  - Make “noisy” services optional
- Block synchronous systems services
  - synchronizing tens of thousands of nodes is hard
- Hardware support
  - for noisy operations (e.g., global clock)
  - for operations affected by noise (e.g., collective offload)
- Develop noise tolerant algorithmic approaches
  - equivalent to latency tolerant and fault oblivious approaches
    (i.e., accept that noise will eventually dominate all other things)
- Define how applications can be noise tolerant (e.g., avoid ALLREDUCE)

Linux

Linux

What was the question?

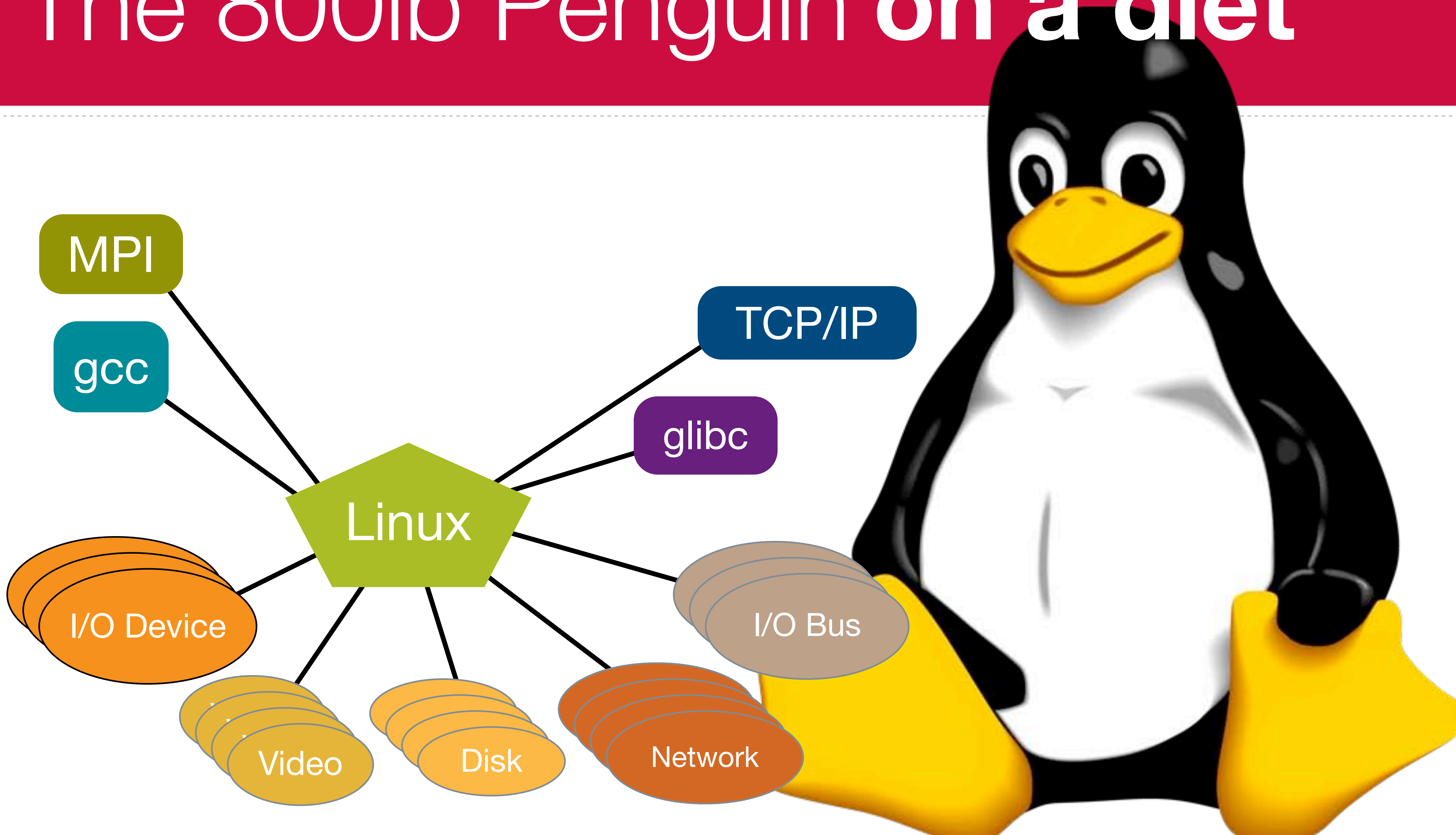
The 800lb Penguin



"Linux's cleverness is not in the software, but in the development model"

Rob Pike, "Systems Software Research is Irrelevant," 2/2000

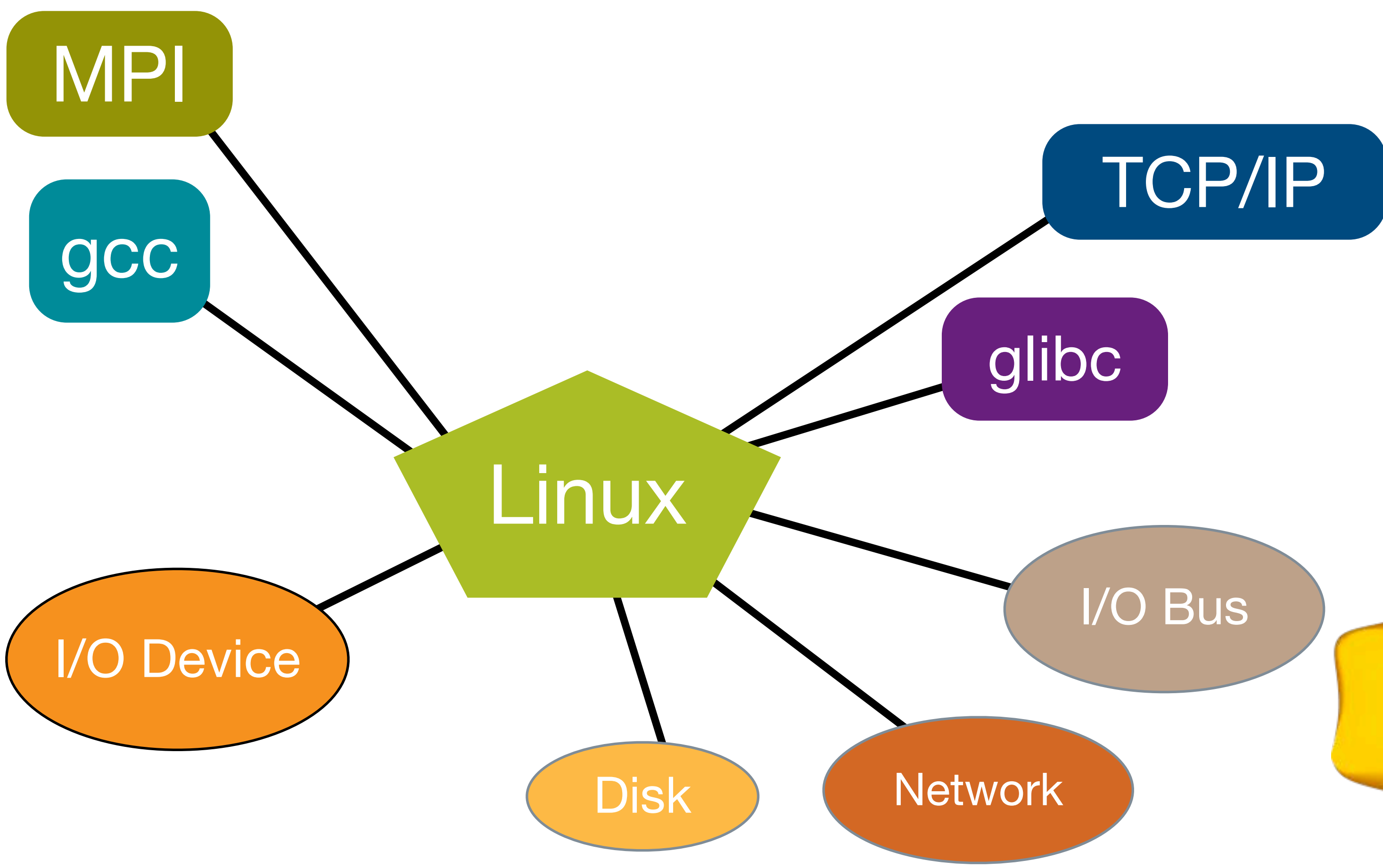
The 800lb Penguin **on a diet**



"Linux's cleverness is not in the software,
but in the development model"

Rob Pike, "Systems Software Research is Irrelevant," 2/2000

The 800lb Penguin **on a diet**

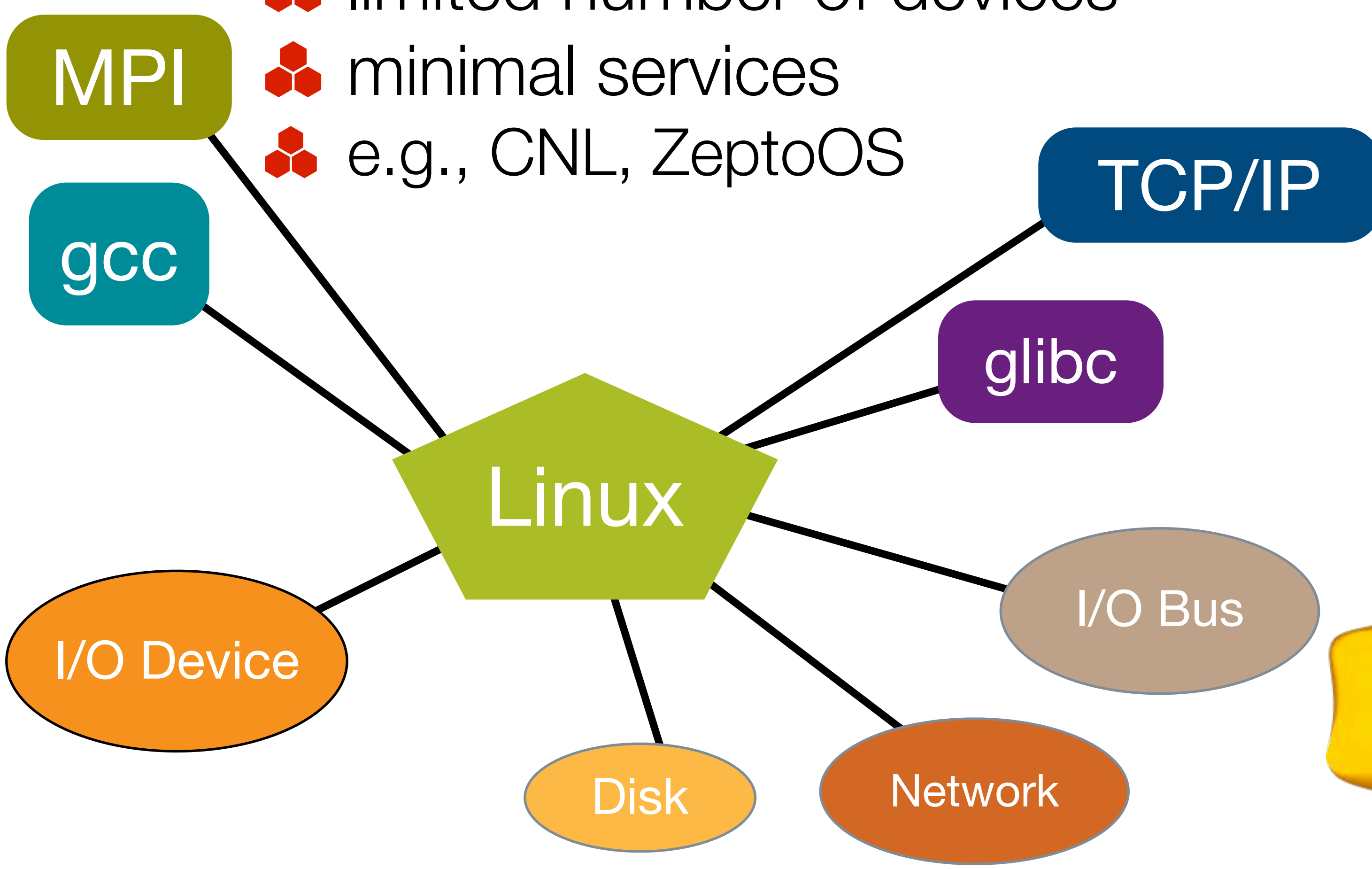


"Linux's cleverness is not in the software,
but in the development model"

Rob Pike, "Systems Software Research is Irrelevant," 2/2000

The 800lb Penguin **on a diet**

- no “broken” hardware
- limited number of devices
- minimal services
- e.g., CNL, ZeptoOS

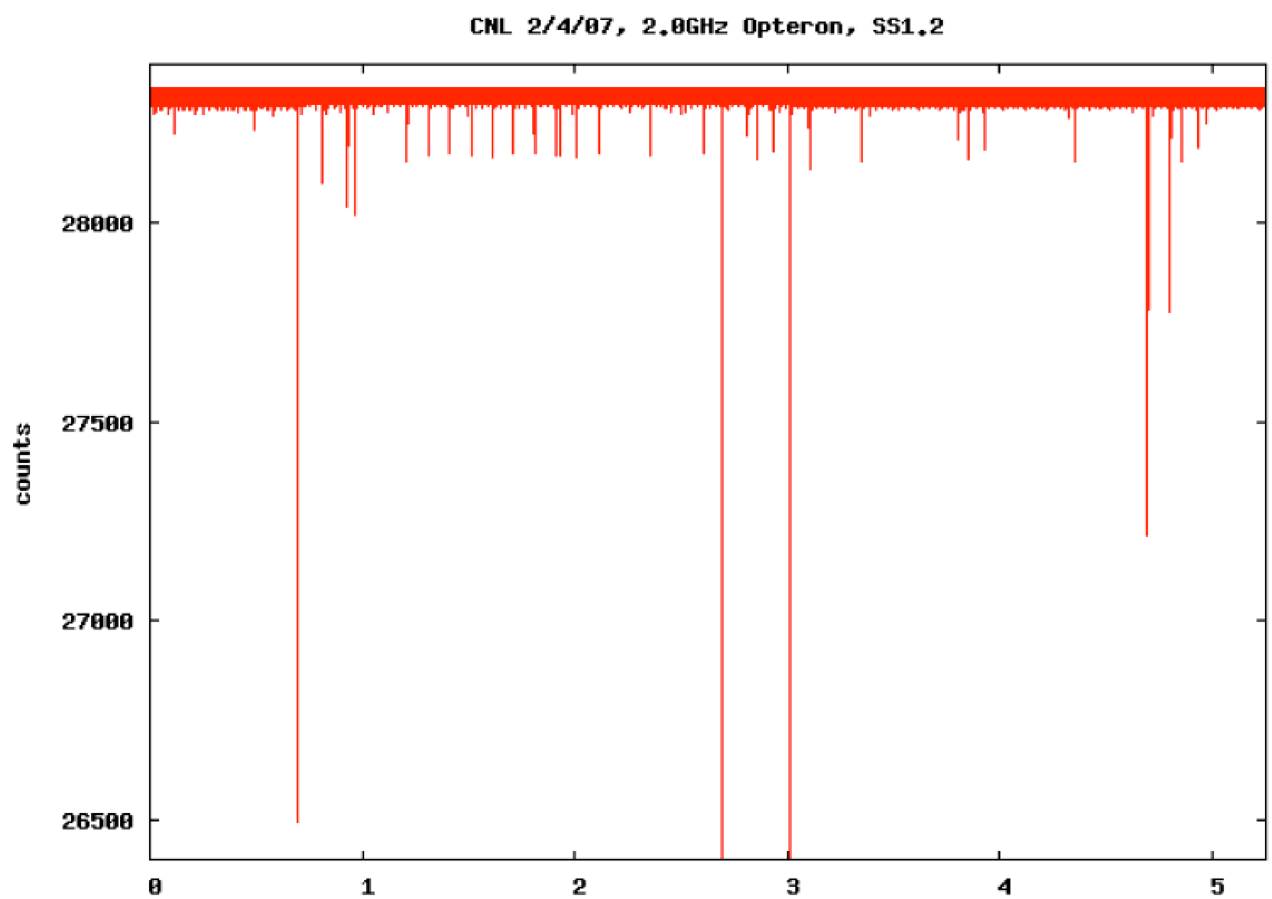


“Linux’s cleverness is not in the software,
but in the development model”

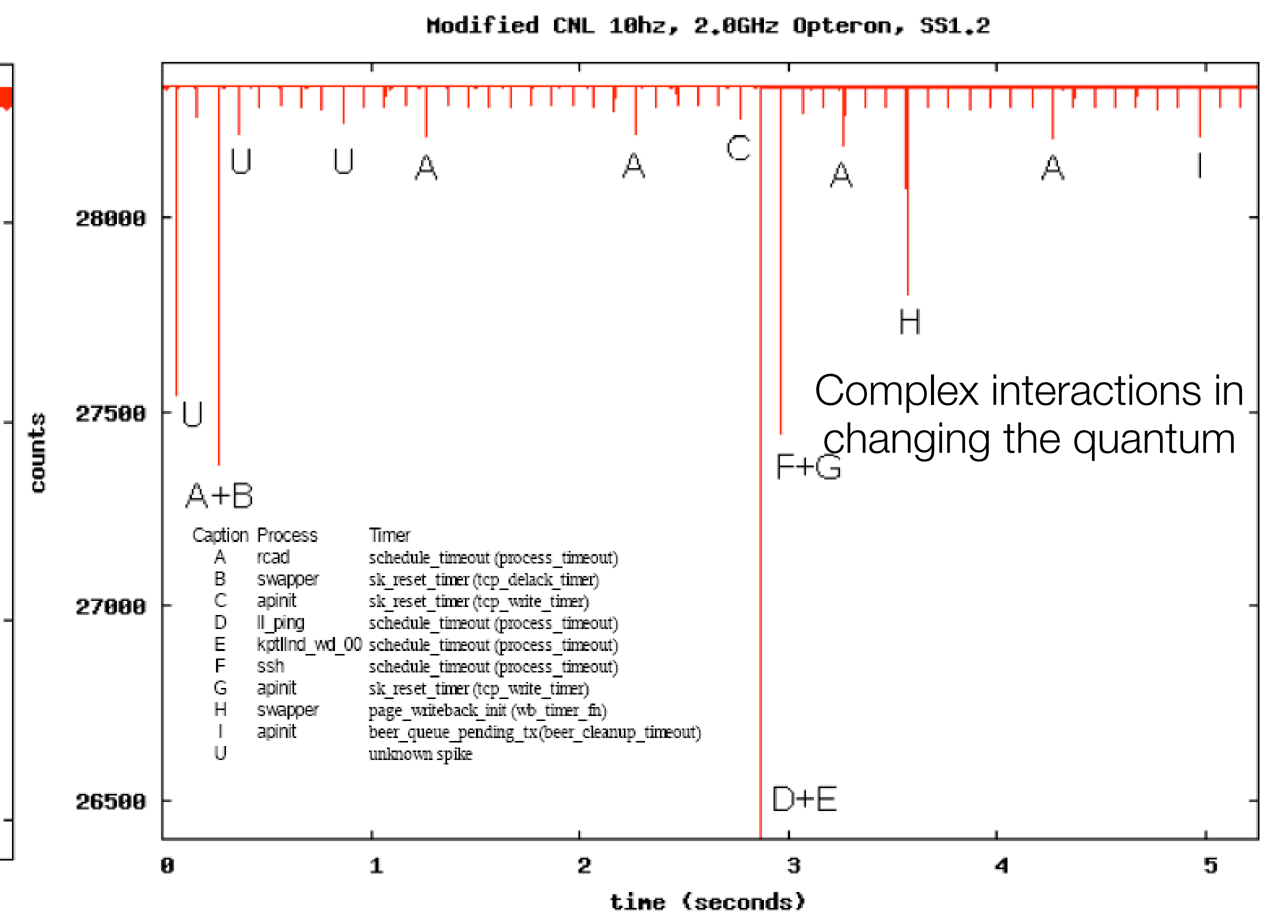
Rob Pike, “Systems Software Research is Irrelevant,” 2/2000

Building **Compute Node Linux**

FTQ on Linux



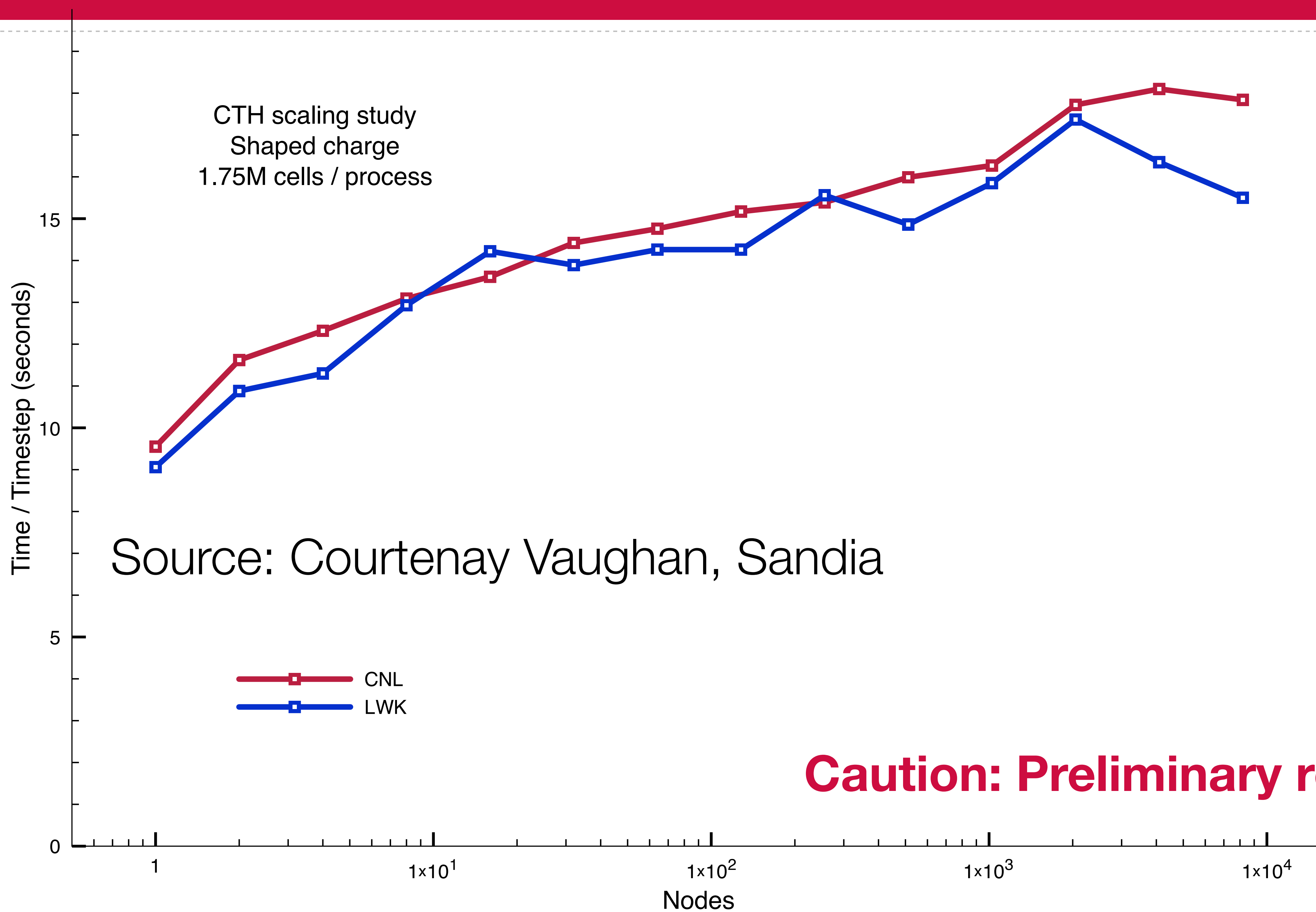
FTQ evolving on CNL



- BusyBox (embedded)
- no remote login
- add “capacity” features as needed by application
  - Libraries
  - NFS, LDAP

Source: Larry Kaplan Cray, Inc.

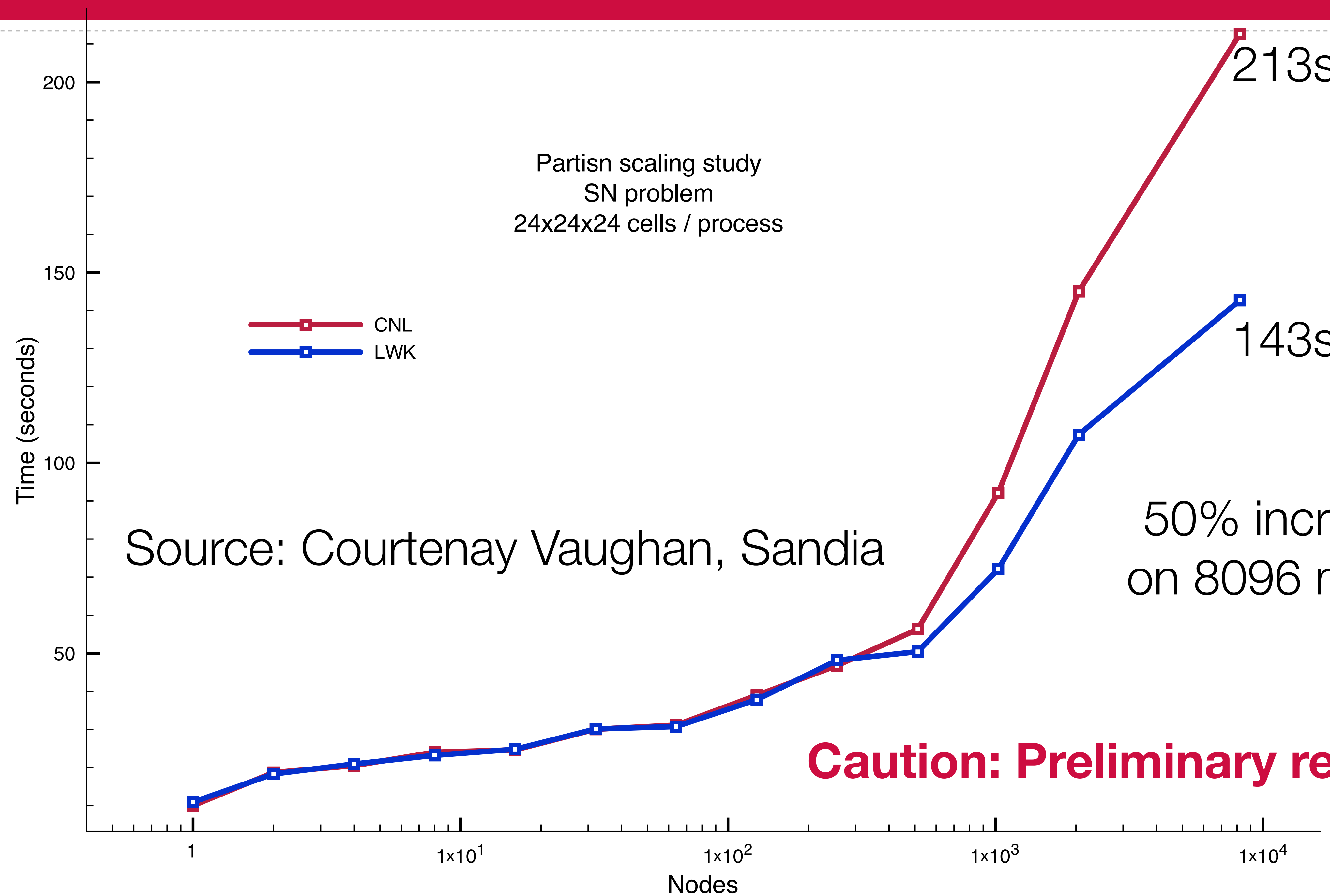
CTH on Catamount and CNL



Source: Courtenay Vaughan, Sandia

Caution: Preliminary results

Partisn on Catamount and CNL

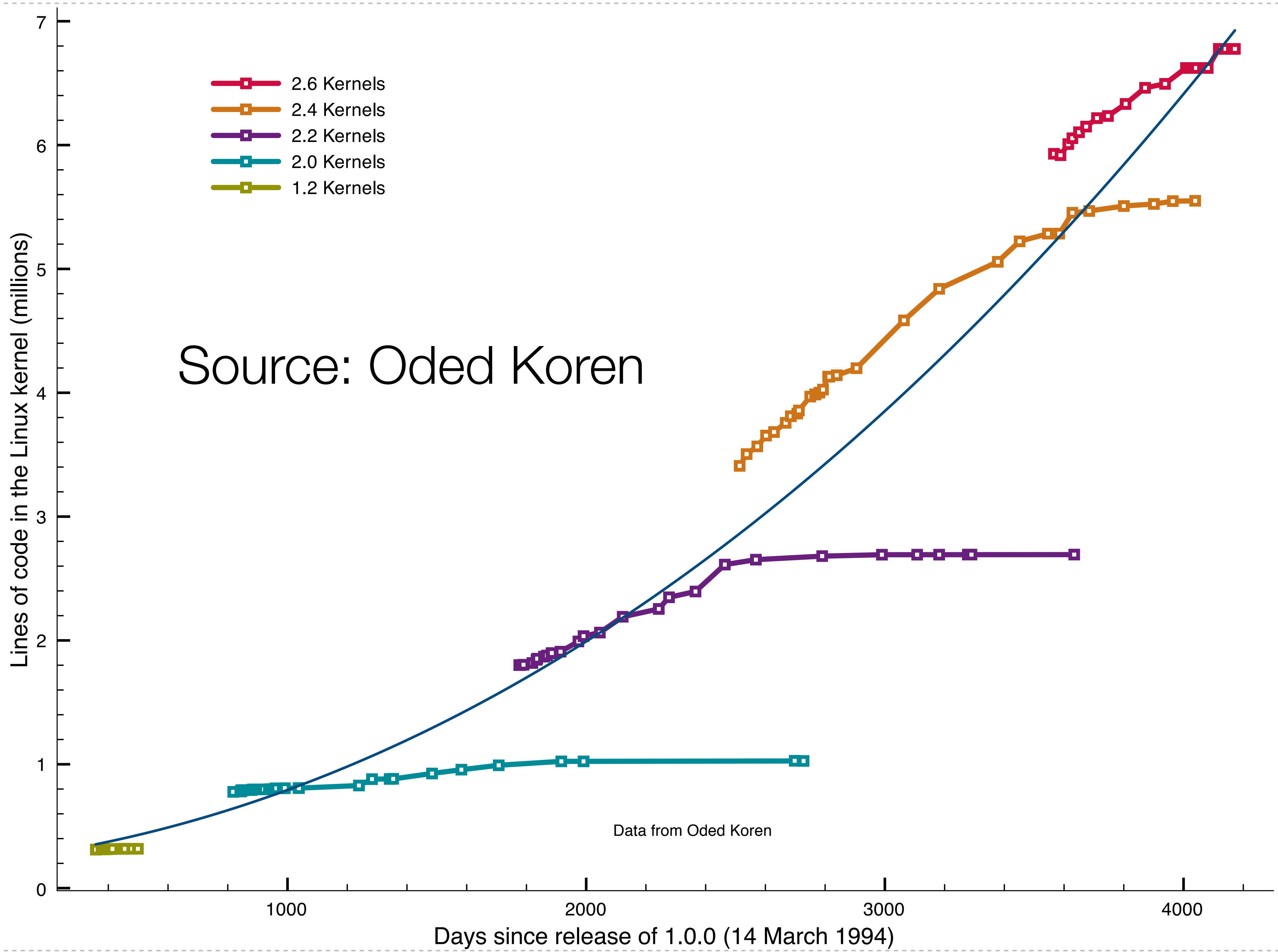


Source: Courtenay Vaughan, Sandia

50% increase on 8096 nodes

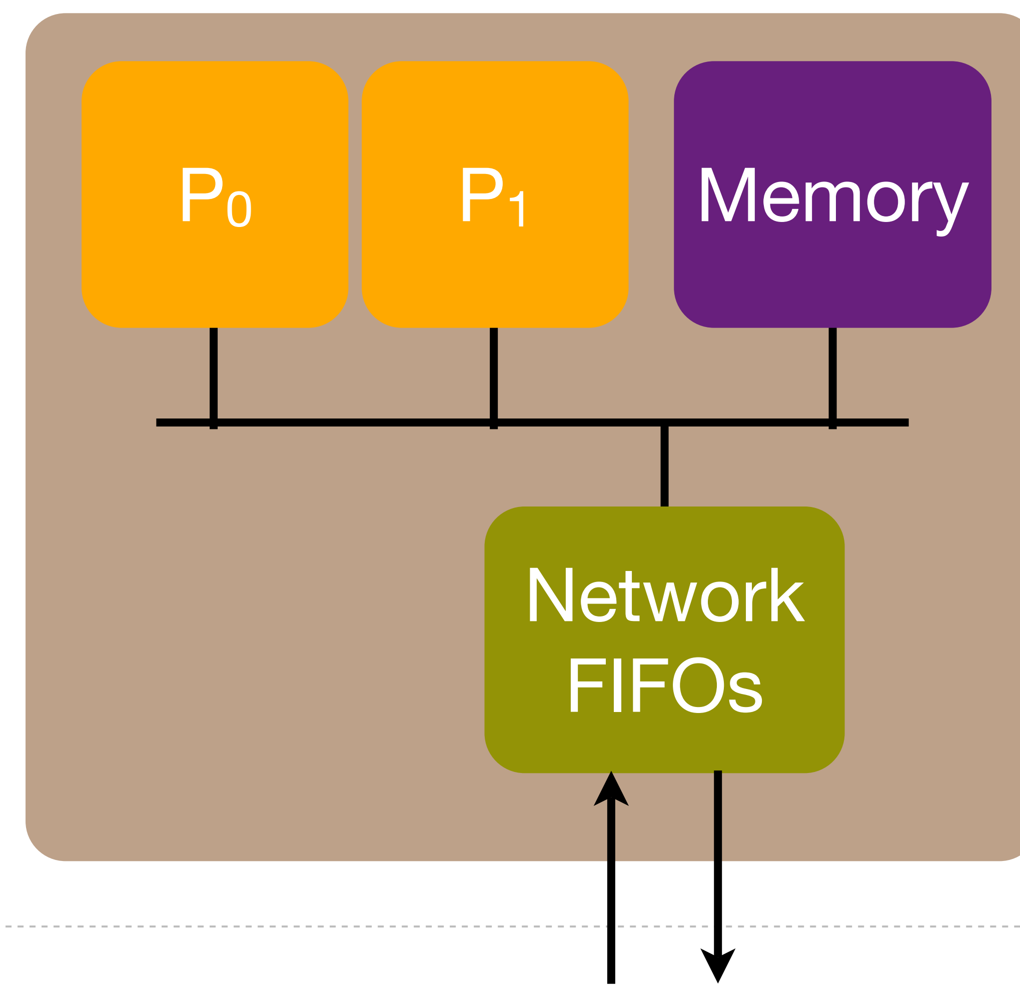
Caution: Preliminary results

Keeping up with Linux



Catamount is **Nimble**

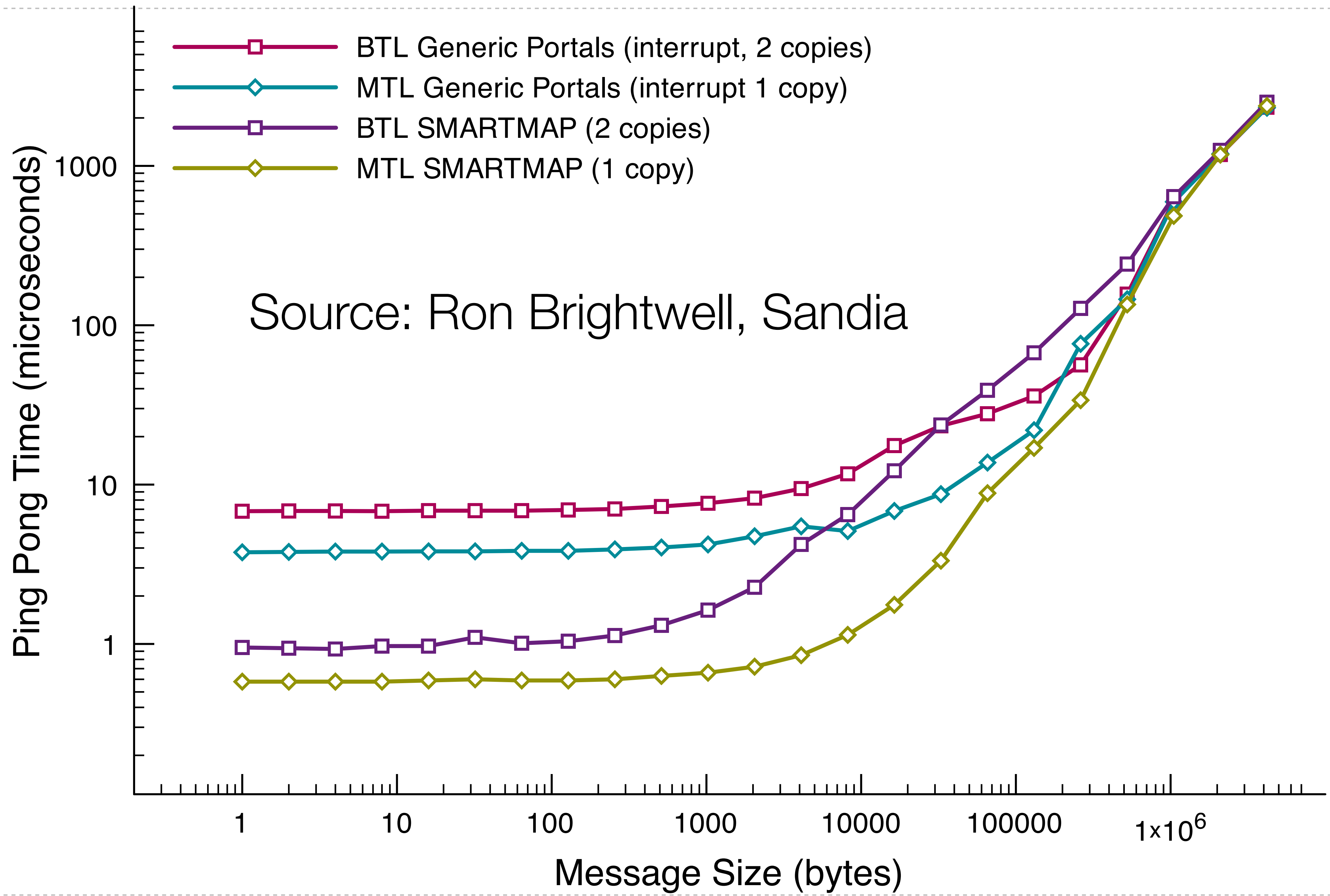
- Source code is small enough that developers can keep it in their head
  - Catmount is <100,000 lines of code
- Early example: dual processors on ASCI/Red
  - Heater mode
  - Message co-processor mode
    - designed/expected mode of use
  - Compute co-processor mode
    - aka “stunt mode”
  - Virtual node mode
    - 6 man-month effort to implement
    - became the standard mode



Adding **Multicore** Support

- SMARTMAP (Brightwell, Pedretti, and Hudson)
  - Map every core’s memory view into every other core’s memory map
  - Almost threads, almost processes
  - modified 20 lines of kernel code
  - in-line function (3 lines of code) to access another core’s memory
- Modified Open MPI
  - Byte Transport Layer (BTL), requires two copies
  - Message Transport Layer (MTL), message matching in Portals
- Less than a man-month to implement

SMARTMAP Performance



Why Linux .... Why not ....

- Community
  - Easier to hire Linux specialists
  - Lots of eyes to find solutions, and others care
- Environment
  - Performance tools
  - Development tools (compilers)
  - Libraries
- Highlander: there will be one

- One is the loneliest number... diversity is a good thing
- Linux is a moving target
  - hard to get changes into Linux
  - HPC is not the goal
- Shrinking Linux eliminates parts of the environment
  - when does it stop being Linux?

Catamount as a virtualization layer

Lightweight Storage Systems

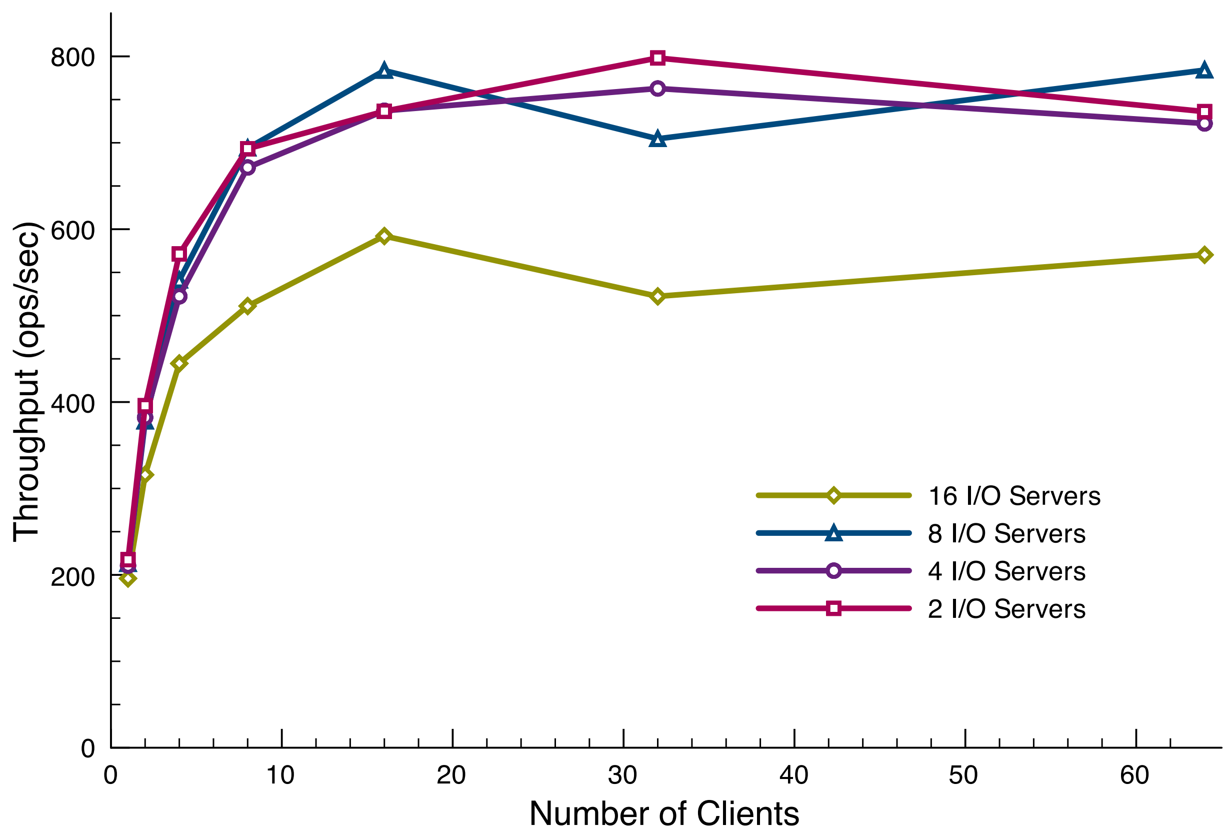
Basic Idea

- Apply lightweight design philosophy to storage systems
- **Enforce** access control: authentication, capabilities with revocation
- **Enable** consistency: lightweight transactions
- Expose full power of the storage resources to applications
  - Applications manage bandwidth to storage
- “Off line” meta data updates – “Meta bots”

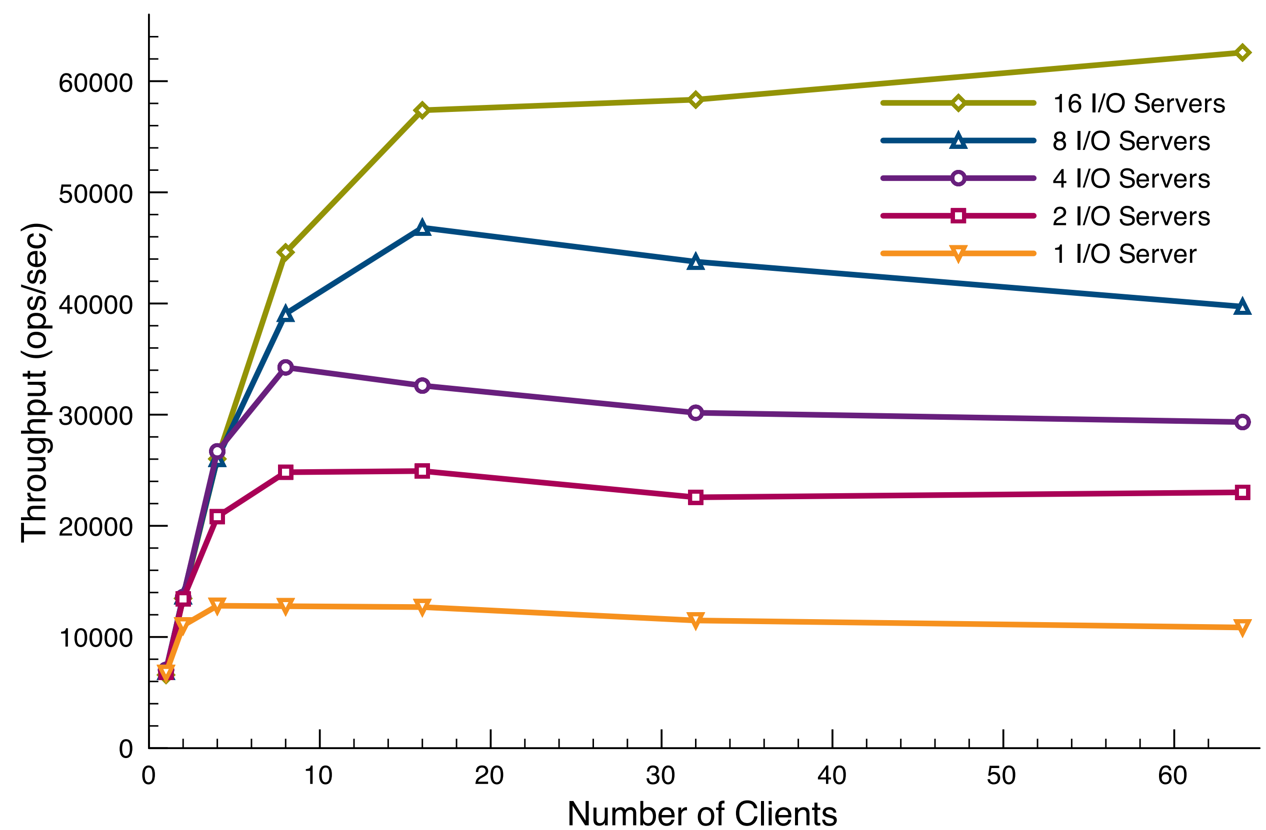
File/Object Creates

Source: Ron Oldfield, Sandia

Lustre File Creates



LWFS Object Creates



Note different scales for y axis

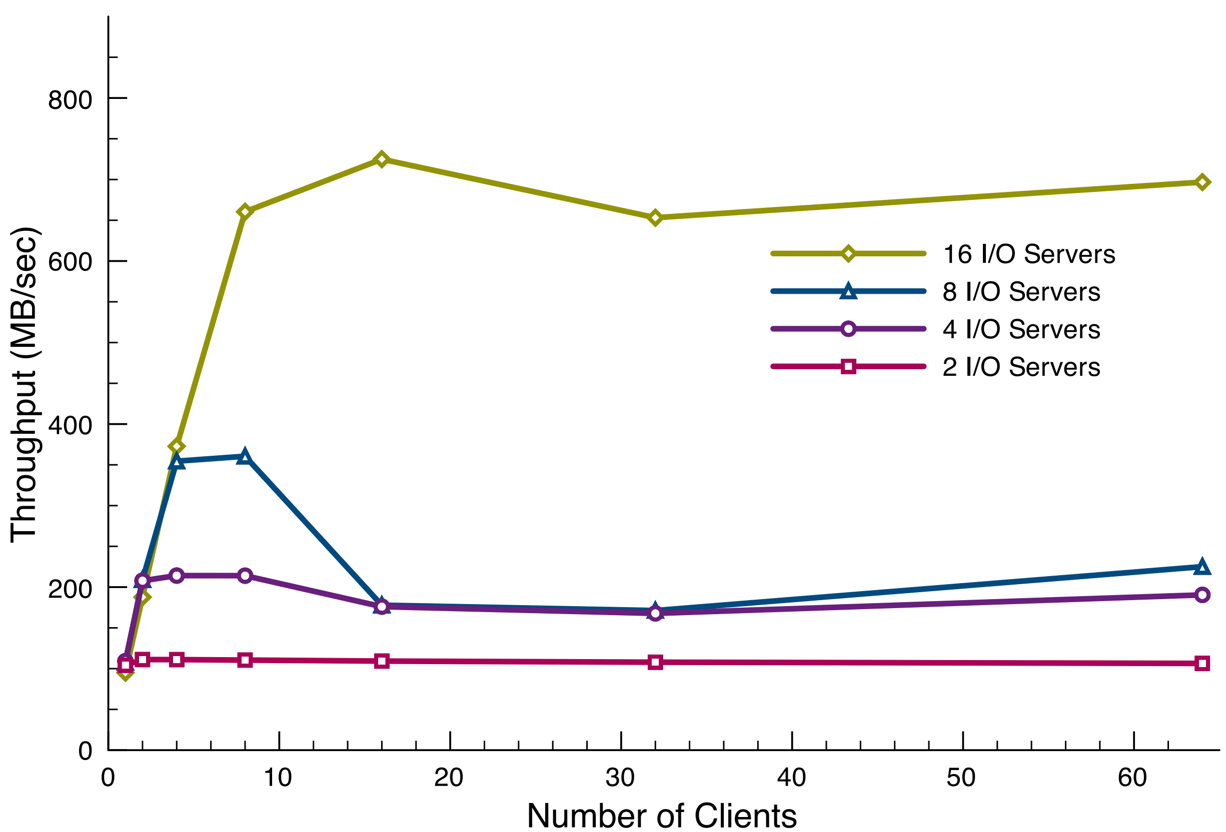
Checkpoints

1. Initiate a lightweight transaction on node 0
   - Broadcast transaction id to all nodes
2. Each node creates a unique data object & dumps local data
   - parallelism only limited by disks
   - no metadata, no consistency, no coherency
   - data objects are transient
3. All nodes send their data object id to node 0
4. Node zero builds an “index object” and commits the transaction
   - two phase commit with the storage servers
   - data objects and index object are permanent
   - could be done “off line” by a meta-bot

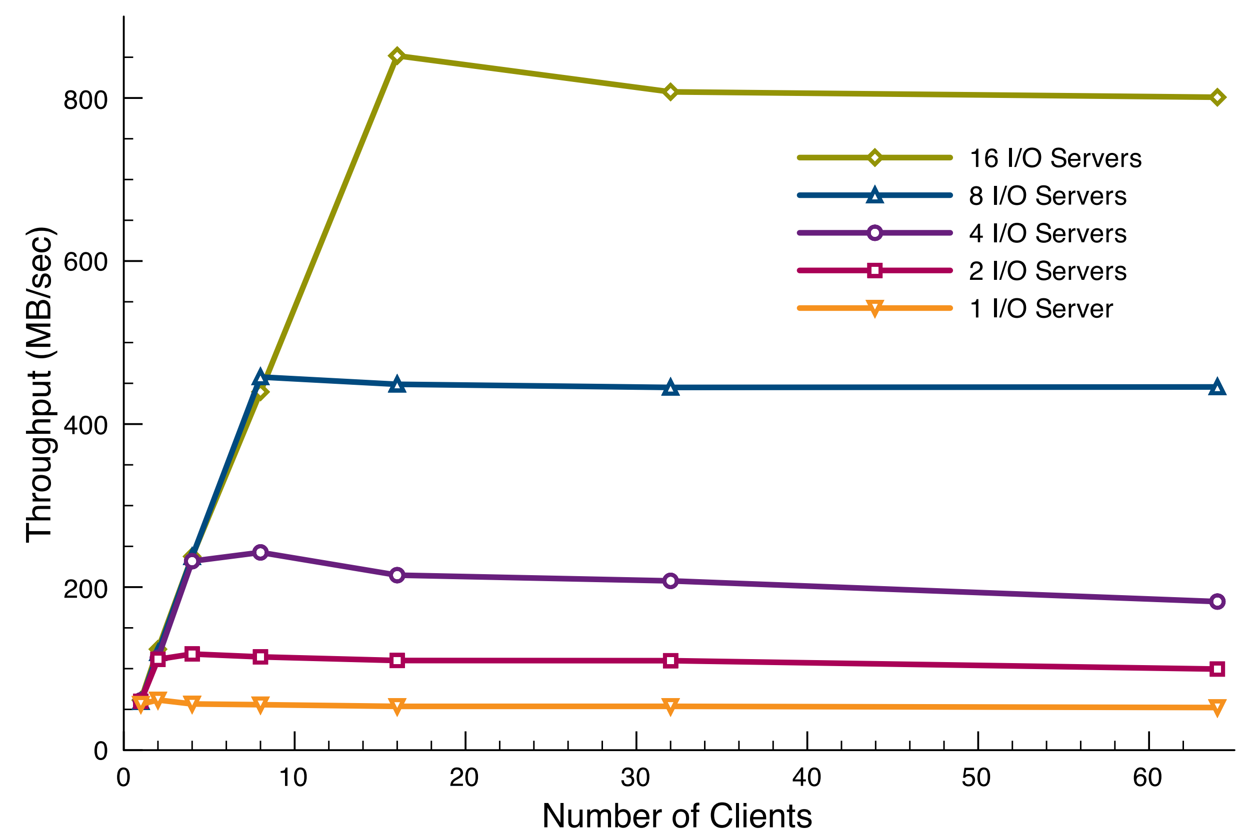
Write Throughput

Source: Ron Oldfield, Sandia

Lustre write throughput file/process



LWFS write throughput object/process



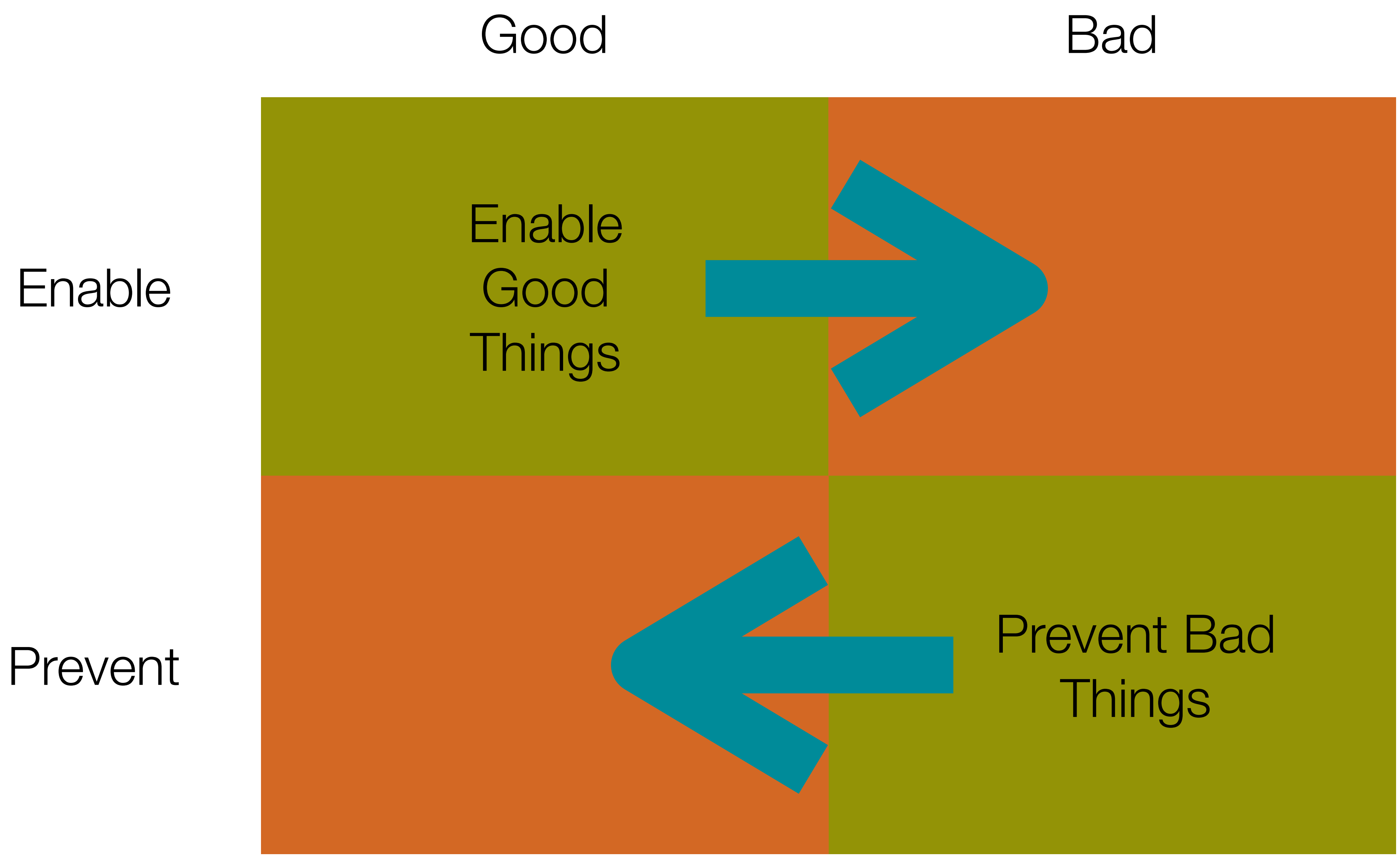
A final story

- Many-to-one operations are problematic at scale
- Cannot reserve buffer space on compute nodes for 10,000 to 1
- Catamount perspective–it's a protocol failure, fix the application!
  - Upper levels are responsible for flow control
  - Catamount happily drops messages–failing sooner rather than later is better
- BG/L–the customer is right
  - Protect applications from themselves
  - Flow control is fundamental, even if it handicaps well written applications

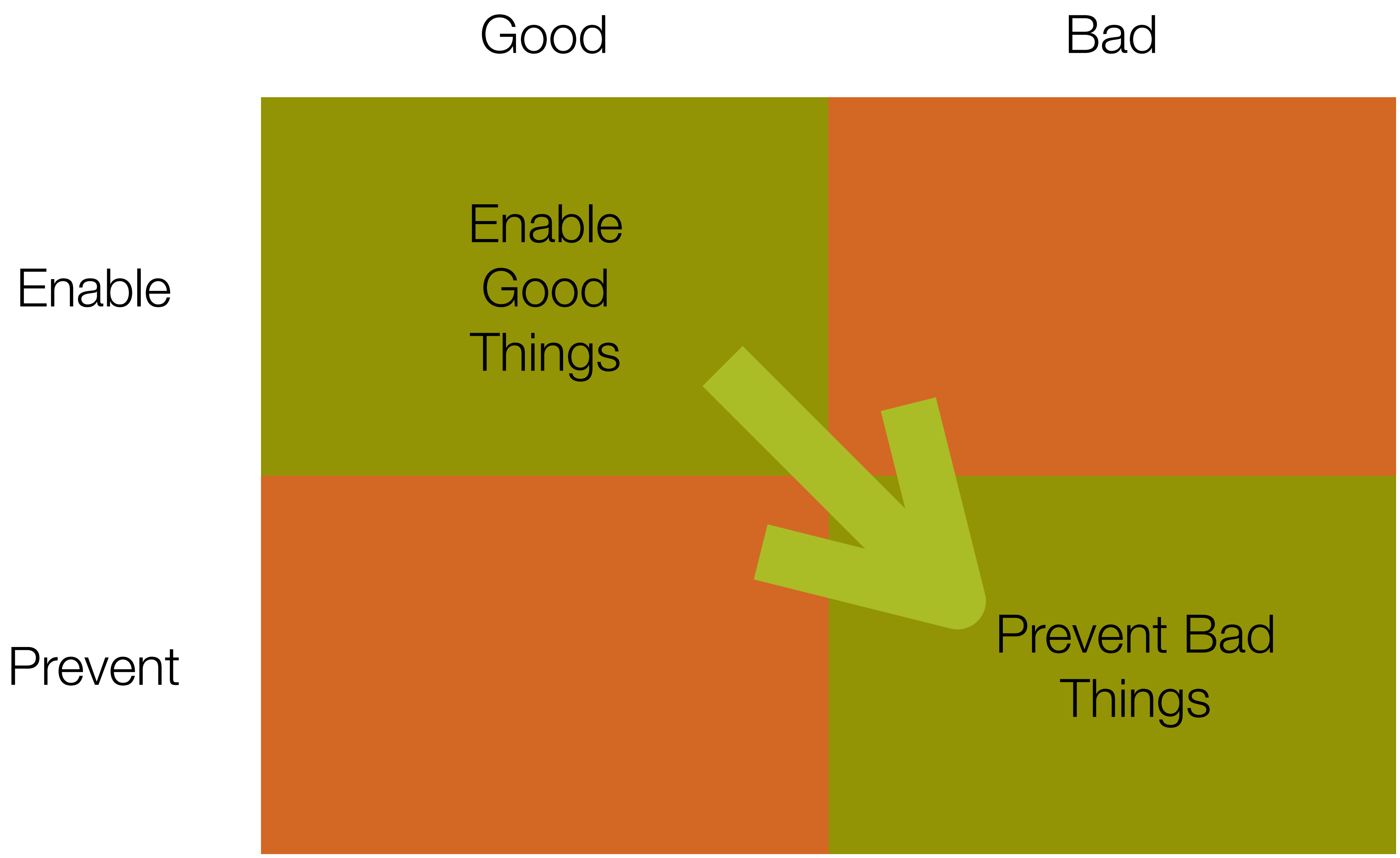
The Design Space

	Good	Bad
Enable	Enable Good Things	
Prevent		Prevent Bad Things

The Design Space



The Design Space



Thanks

- UNM Scalable Systems Lab
  - Patrick Bridges, Patrick Widener, Kurt Ferreira
- Sandia National Labs
  - Ron Brightwell, Ron Oldfield, Rolf Riesen, Lee Ward, Sue Kelly

"Fools ignore complexity; pragmatists suffer it; experts avoid it; geniuses remove it."
Alan J. Perlis