Scaling System Software to Exascale

Ron Brightwell

Scalable System Software Department

Sandia National Laboratories

rbbrigh@sandia.gov

SOS-14

March 10, 2010

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Exascale Challenge for Operating Systems

Programming/Execution Model

MPI+OpenMP

MPI+PGAS

MPI+CUDA

ParalleX

MPI

PGAS

Chapel

MPI+OpenCL

Operating/Runtime System

Architecture

Non-Cache-Coherent Many-Core

Hybrid Multi-Core

Distributed Memory

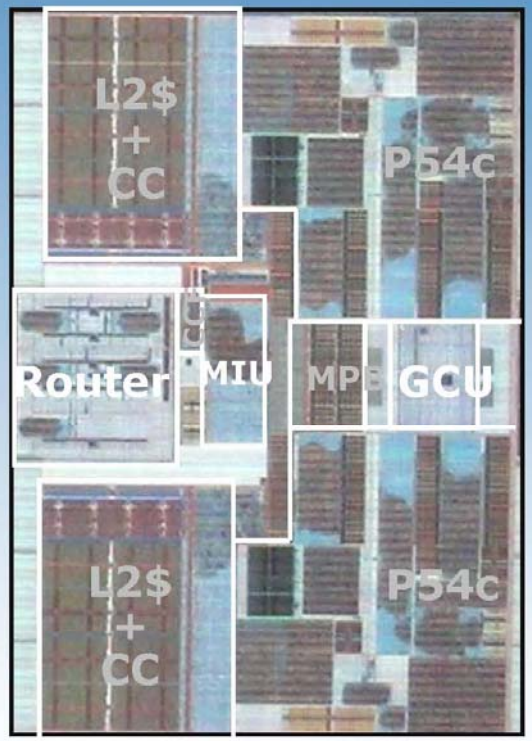
Global Address Space

Homogeneous Multithreaded

What’s the node architecture?

Intel's SCC

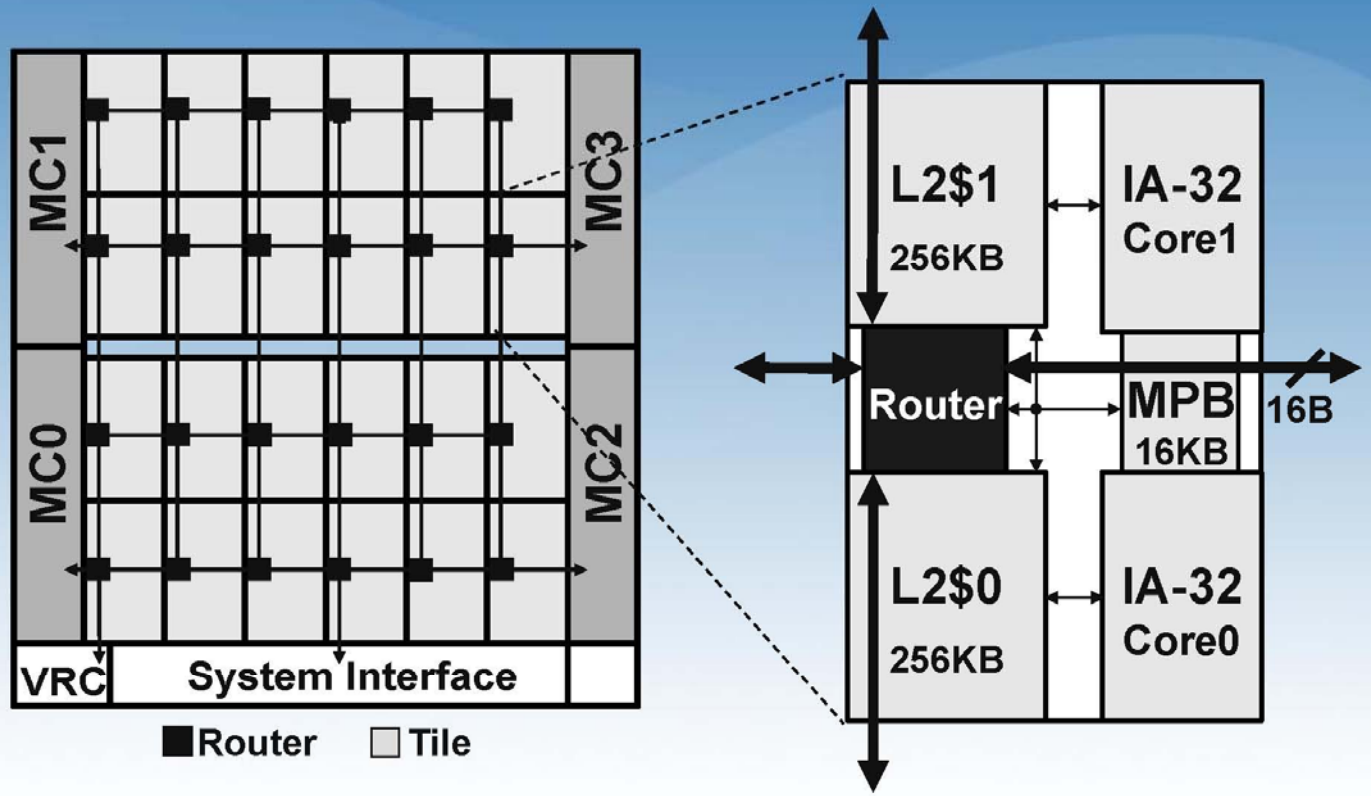
SCC Dual-core Tile



- 2 P54C cores (16K L1$/core)
- 256K L2$ per core
- 8K Message passing buffer
- Clock Crossing FIFOs b/w Mesh interface unit and Router

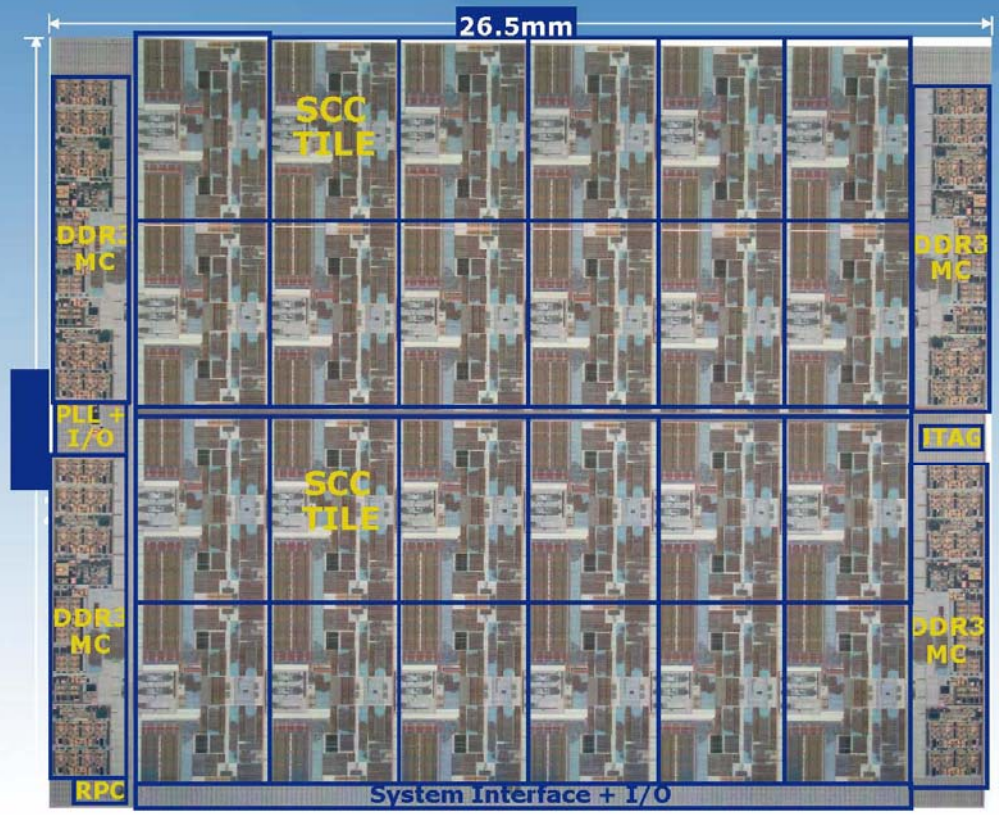
- Tile area 18.7mm^2
- Core are 3.9mm^2
- Cores and uncore units @1GHz
- Router @2GHz

Die Architecture



2 core clusters in 6x4 2-D mesh

SCC full chip

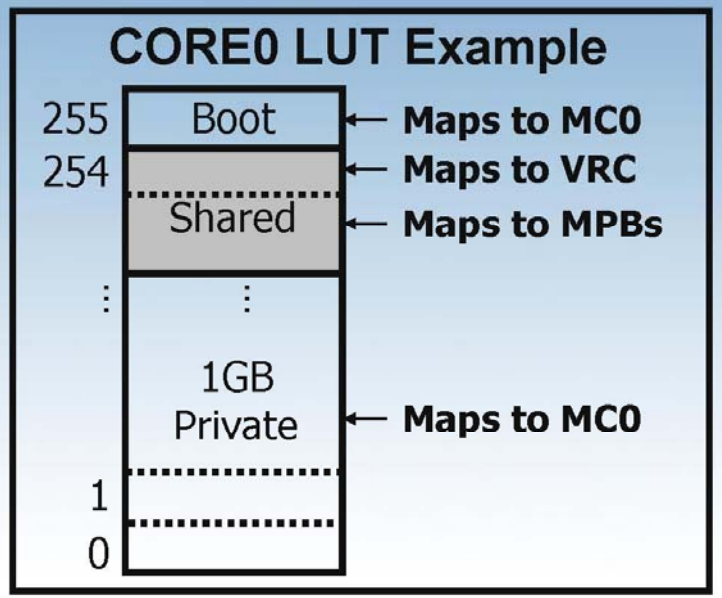


Technology	45nm Process
Interconnect	1 Poly, 9 Metal (Cu)
Transistors	Die: 1.3B, Tile: 48M
Tile Area	18.7mm^2
Die Area	567.1mm^2

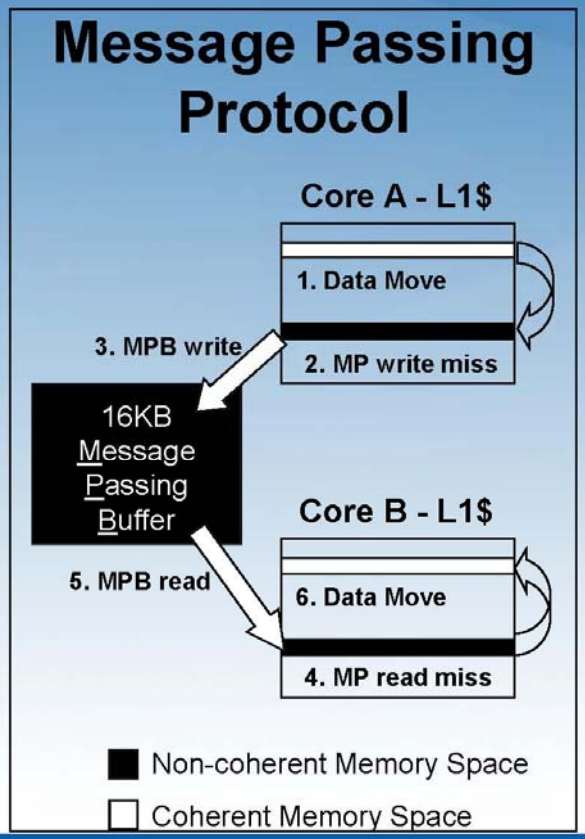
Core Memory Management

- Core cache coherency is restricted to private memory space
  - Maintaining cache coherency for shared memory space is under software control

- Each core has an address Look Up Table (LUT) extension
  - Provides address translation and routing information
- LUT must fit within the core and memory controller constraints
- LUT boundaries are dynamically programmed

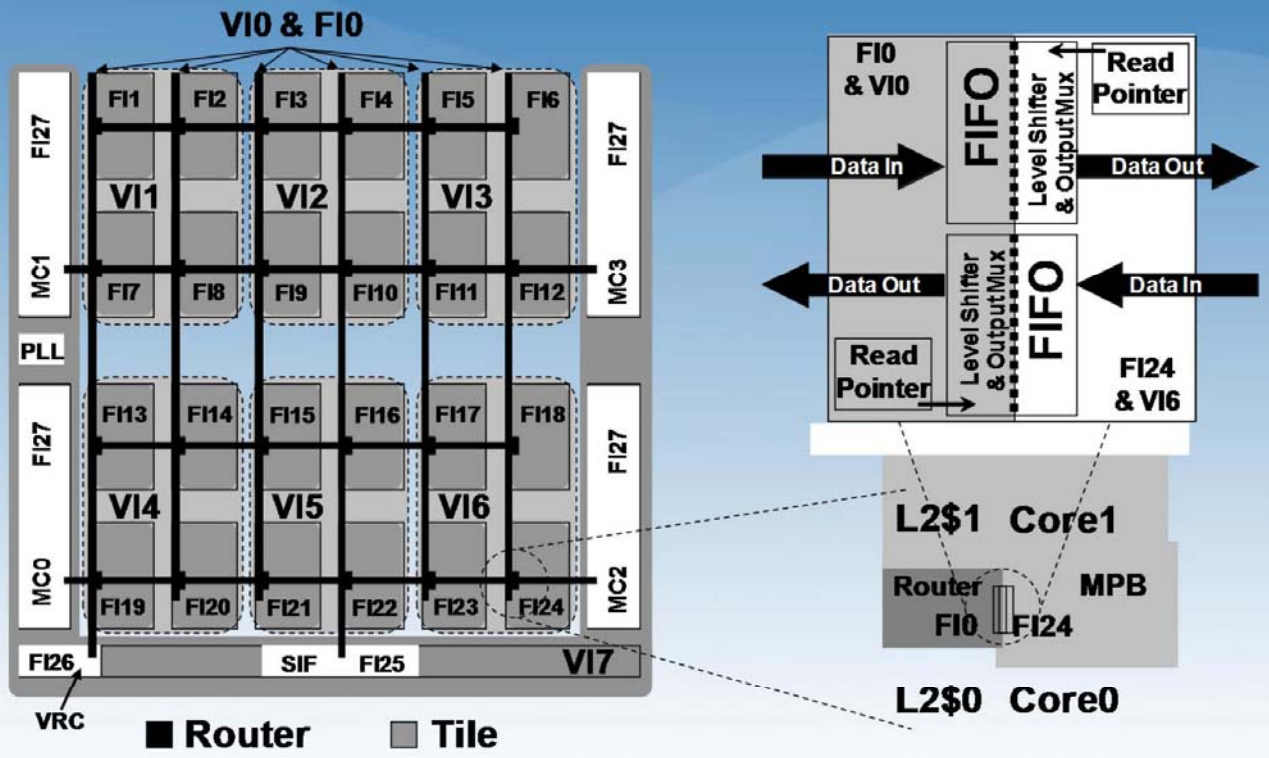


Message Passing Protocol



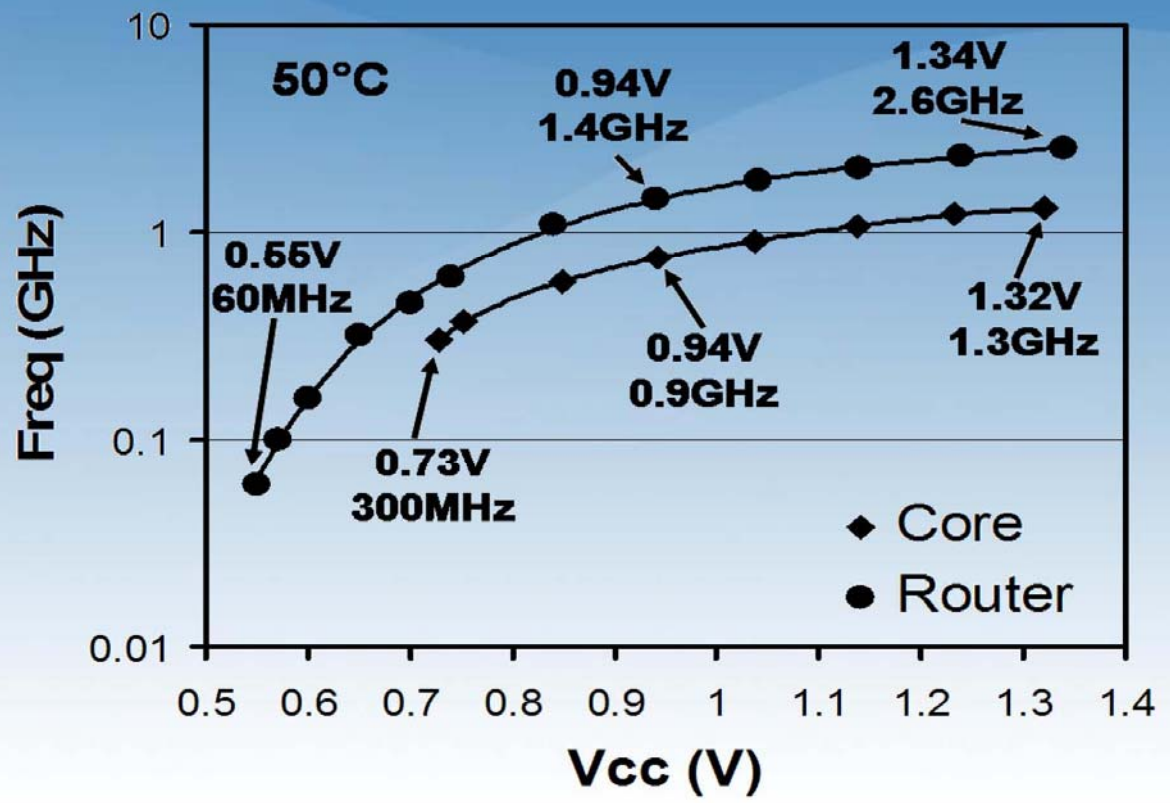
- Cores communicate through small fast messages
  - L1 to L1 data transfers
  - New Message Passing Data Type (MPDT)
- Message passing Buffer (MPB) – 16KB
  - 1 MPB per tile for 384KB of on-die shared memory
  - MPB size coincides with L1 caches

Voltage and Frequency islands

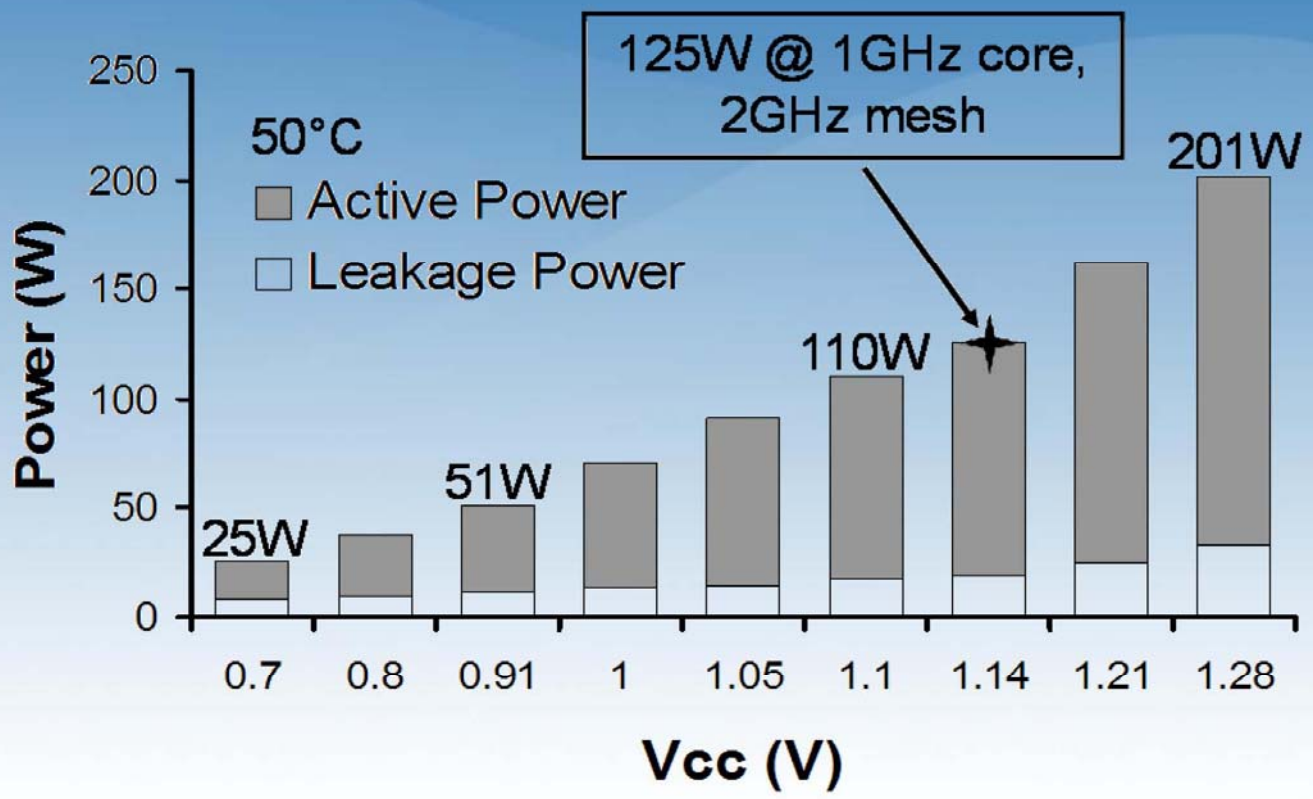


27 Frequency Islands (FI) 8 Voltage Islands (VI)

Core & Router Fmax

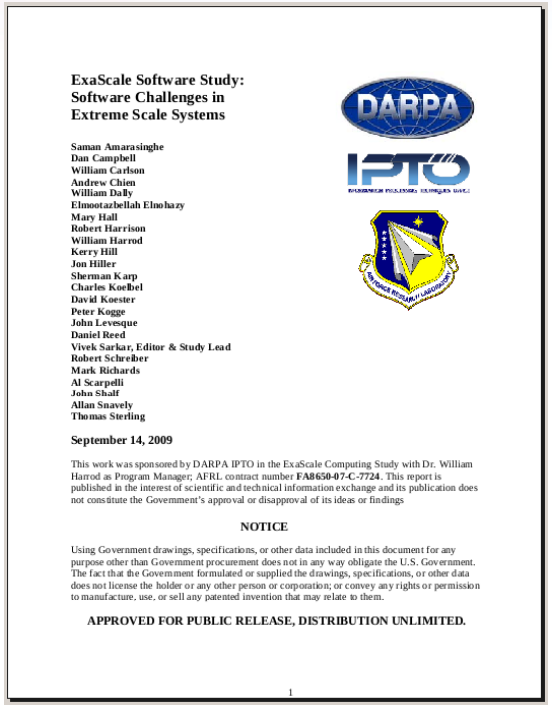


Measured full chip power



What are the system software challenges?

Plenty of Material to Choose From



ExaScale Software Study: Software Challenges in Extreme Scale Systems

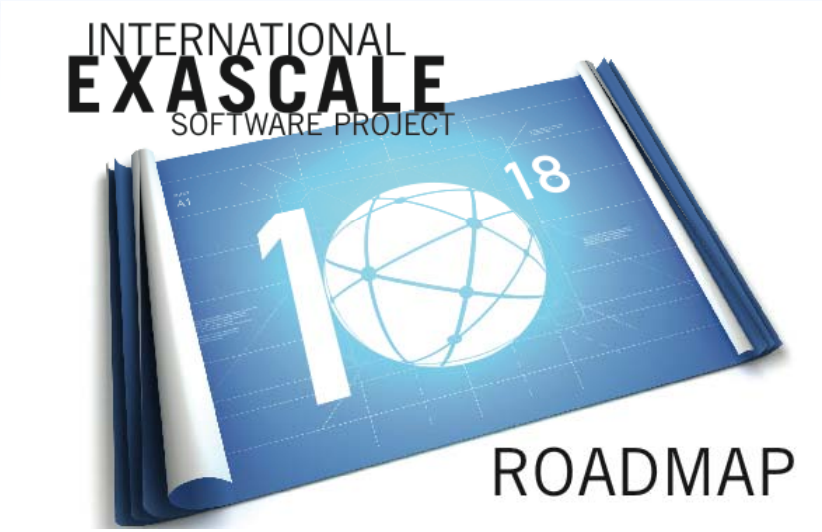
Saman Amarasinghe
Dan Campbell
William Carlson
Andrew Chien
William Dally
Elmootazbellah Elnohazy
Mary Hall
Robert Harrison
William Harrod
Kerry Hill
Jon Hiller
Sherman Karp
Charles Koelbel
David Koester
Peter Kogge
John Levesque
Daniel Reed
Vivek Sarkar, Editor & Study Lead
Robert Schreiber
Mark Richards
Al Scarpelli
John Shalf
Allan Snavely
Thomas Sterling

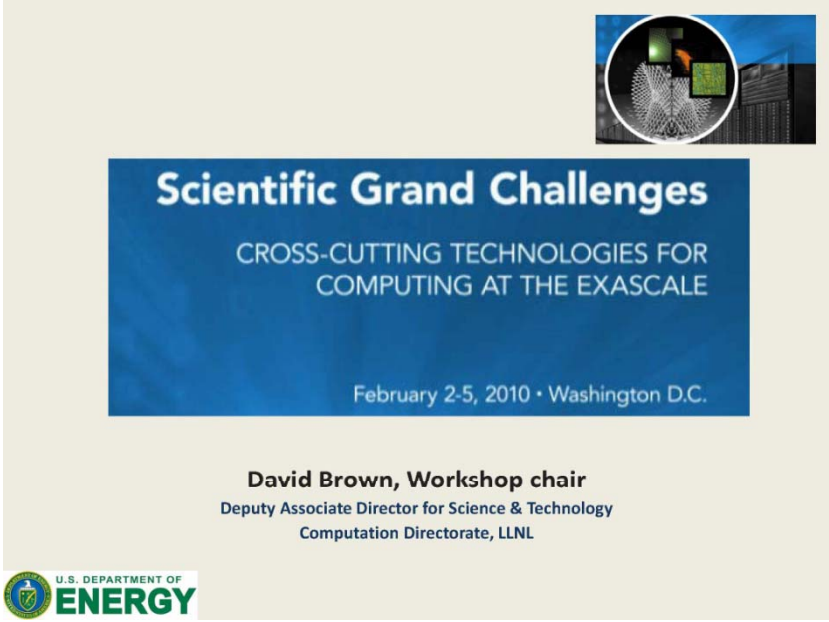
September 14, 2009

This work was sponsored by DARPA IPTO in the ExaScale Computing Study with Dr. William Harrod as Program Manager; AFRL contract number **FA8650-07-C-7724**. This report is published in the interest of scientific and technical information exchange and its publication does not constitute the Government's approval or disapproval of its ideas or findings

NOTICE

1





Extreme Scale OS

(DARPA Exascale Software Study Group)
SC2008 Meeting

Why we need a new research platform for OS development

National Nuclear Security Administration

Sandia National Laboratories

Role of OS at Extreme Scale

- Started discussion in "Runtimes" group
  - Why?
- Nearly every important policy decision that would be made by the *runtime* is mediated by the OS
  - The OS is slow to respond (privilege changes are slow)
  - Doesn't even allow us to control the key policies required for new execution models
  - The OS wasn't designed for manycore

Old OS Assumptions are Bogus

- **Assumes limited number of CPUs that must be shared**
  - ***Old CW**: time-multiplexing (context switching and cache pollution!)*
  - ***New CW:** spatial partitioning*
- **Greedy allocation of finite I/O device interfaces (eg. 100 cores go after the network interface simultaneously)**
  - ***Old CW:** First process to acquire lock gets device (resource/lock contention! Nondeterm delay!)*
  - ***New CW:** QoS management for symmetric device access*
- **Background task handling via threads and signals**
  - ***Old CW**: Interrupts and threads (time-multiplexing) (inefficient!)*
  - ***New CW**: side-cores dedicated to DMA and async I/O*

Old OS Assumptions are Bogus

- **Fault Isolation**
  - ***Old CW:*** *CPU failure --> Kernel Panic (will happen with increasing frequency in future silicon!)*
  - ***New CW:*** *CPU failure --> Partition Restart (partitioned device drivers)*
- **Inter-Processor Communication**
  - ***Old CW:*** *invoked for ANY interprocessor communication or scheduling*
  - ***New CW:*** *direct HW access mediated by hypervisor*
- **Parallel Interrupt Dispatch and Interrupt Routing**
  - ***Old CW:*** *invoked for ANY interprocessor communication or scheduling. Interrupts routed by programming APIC.*
  - ***New CW:*** *direct HW access mediated by hypervisor*
  - *dedicated to DMA and async I/O*
  - *Channel-based delivery of interrupts as "urgent messages"*

Intersection with Execution Models

- **OS prevents emergence of novel/scalable execution models**
  - Example: Scheduler is privileged (outside of control of the Application)
  - OS also controls locality management policy (use laborious process pinning to fix it)
- **Need to give more control to application or runtime environment to manage scheduling**
  - However, need to grant that control without compromising security and fault isolation
  - With thousands of cores, there are new opportunities to maintain security and fault isolation via physical (spatial) partitions
- **We cannot do research into novel execution models without at least a provisional thin-OS as a research platform**
  - Impractical to hack directly on the Linux OS scheduler (experiments often are inconclusive)
  - In-effect ceding more policy control over to the OS and

Inhibitors to Change

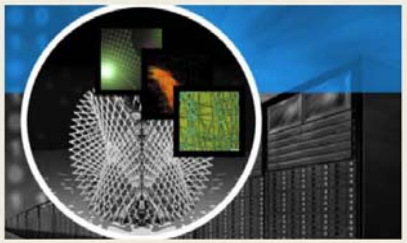
- Replacing the OS tends to be an all-or-nothing proposition
  - Modern OS's have accreted years of capabilities
  - Bad track record (remember PINK?)
- However, we will make zero progress on alternative execution models and associated runtimes if we don't have a minimal research substrate
  - Impractical, resource intensive, and inconclusive to keep hacking on the Linux kernel (scheduler etc..) to perform these experiments

Restricting Role of OS

- Need to get the OS out of our way
  - Need bare-metal hardware access
  - Give the runtime system more control over resource allocation and policy (with QoS limits)
  - Cannot even do experiments with advanced execution models because OS pollutes the result!
  - Also, OS includes many features we don't always use (bad if memory is shrinking per-core)
- But we need to protect the hardware state from errant or malicious programs
  - How can we get both?

Deconstructing the OS

- Separate out hardware protection layer (e.g., hypervisor, micro-kernel) from other optional/customizable services that can be accessed directly in user mode
  - Spatial partitioning instead of time-sharing
  - Virtualized device interfaces
  - Dedicated cores for handling interrupts and asynchronous I/O
- Fault isolation of partition containing failure
- Permit access to communication HW in user mode (with hypervisor monitoring for potential state corruption)



Scientific Grand Challenges

CROSS-CUTTING TECHNOLOGIES FOR COMPUTING AT THE EXASCALE

February 2-5, 2010 • Washington D.C.

David Brown, Workshop chair

Deputy Associate Director for Science & Technology

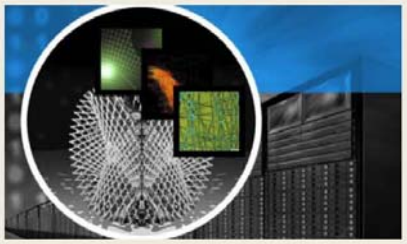
Computation Directorate, LLNL

U.S. DEPARTMENT OF ENERGY

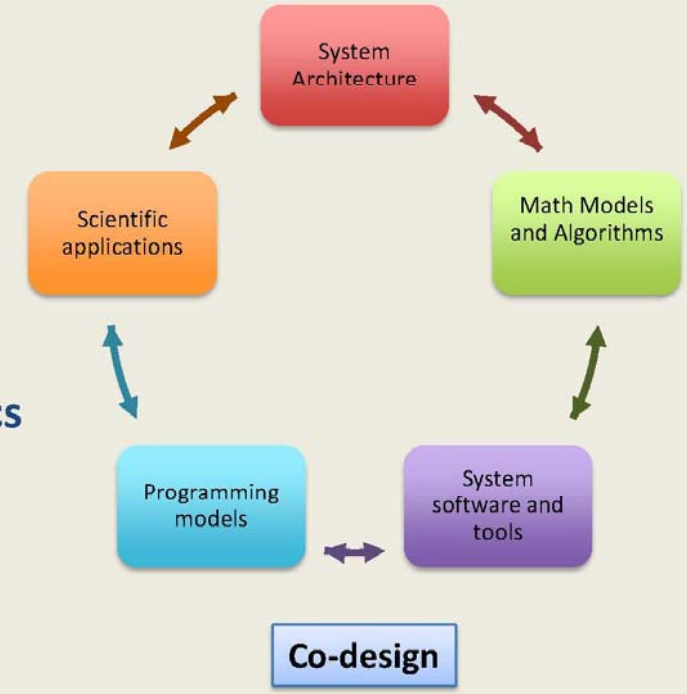
NNSA National Nuclear Security Administration

Sandia National Laboratories

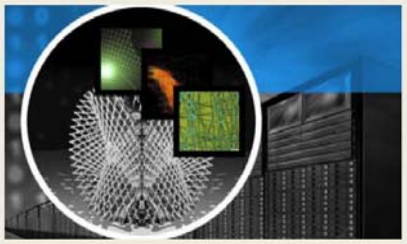
Workshop Objectives

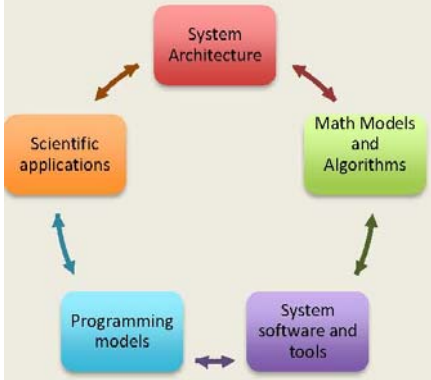


- Outline the R&D needed for co-design of exascale computational science environment
- Identify opportunities for "disruptive" computational approaches for future scientific discovery
- Produce a first cut at characteristics of hardware/software system roadmap that will meet science application needs over the next decade
  - Initial systems 2015 @ 100-300TF
  - Exascale systems in 2018



Multiple breakout sessions will address the three workshop themes





- **Theme 1: Math models and algorithms**
  - Impact of application needs and architectural developments on math models, algorithms and programming models
  - Impact of application, math model and algorithms needs on architectural development
- **Theme 2: System Software Functionality**
  - System software functionality
  - What tasks traditionally handled by systems software will need to be addressed elsewhere, e.g. resiliency handling?
- **Theme 3: Programming models and environment**
  - What programming models and environments are needed?
  - Will programming models provide suitable abstractions and tools for applications/algorithm needs?

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories

PDE's and System Software

John Bell and Ron Brightwell

Cross-Cutting Technologies for Computing at the Exascale

February 2-5, 2010

Network Topology

- PDE characteristic
  - Domain decomposition with nearest neighbor communication
  - Elliptic requires nonlocal small communication
- Topology can be dynamic
  - System issues
  - Algorithm issues
- Two approaches
  - Expose network topology to the application
  - Expose problem connectivity to the OS

Memory hierarchies and cache coherence

- Lots of parallelism within a node
  - Easily identify 1000 way fine grained parallelism
- Hierarchical model often a win
  - AMR
  - Linear solvers
- Application needs to express
  - Spatial locality
  - Temporal locality
  - Liveliness
- Scratchpad can be effective for managing locality

Asynchronous messaging and threads

- Low cost mechanism for thread creation / destruction
- Global name space?
- What is the appropriate threading model?
  - Over committing
  - Dynamic threads
  - How to manage memory coherence
- Threads should be first class objects

Debugging and fault tolerance

- How do you tell the difference from between a bug and a fault?
  - Bugs and system faults may will look identical
- How hard is it to develop criteria for correctness in a function / subroutine call
- Need a tool to localize impact of errors

Heterogeneity

- Moving to simplified homogenous cores appears to be ok
  - At least need to support conditionals
- Can leverage multiprecision in some parts of applications
  - Solvers

Outline Co-design Opportunities
Between Discrete Math and System Software

- Memory Management
  - Apps and System software researchers want the OS to handle it
  - Algorithm and Compiler researchers want OS hooks to manage memory themselves. Co-design needed to define API and scope of control
- Resource Management
  - Dynamic load balancing
  - Scheduling work on heterogeneous nodes
  - Dynamically changing #processors (energy efficiency, faults, app needs)
  - Co-design opportunity is use of graph algorithms to schedule tasks

    memory efficient, fast, incremental update, verses overhead of imbalance
- Simulation
  - Co-design requires discrete event simulation
  - Discrete math can help with the development of the simulator
  - And needs the simulator for memory management and resource management studies

What system software functionality is required by applications at the Exascale?

- Need Adaptive Runtime
  - Adapt to the problem evolution
  - Adapt to faults and in general changing system configuration
- Resource Management
  - Ability to schedule resources for power consumption
  - Allocate the whole 20MW for data movement
  - Request to be able to allocate system by power (J/wk) vs. (hrs/yr)
- I/O
  - MPI-IO efficient handling of sparse segments (zero length segments)
  - Irregular datastreams in HDF or equivalent
- Memory management
  - Ability for user to define data that is guaranteed uncorrupted
- Archive data path guarantee
  - Send data to archive and read it back unchanged

Outline Co-design Opportunities
Between PDE II and System Software

- **Fault tolerance**
  - is the biggest issue from the application user, would like systems software to deal with this as much as possible
  - Need to define standard interfaces to get information (notification)
  - Local recovery (algorithms have small checkpoints)
  - API of Publish/subscribe component (automatically run in user space with app)
- **Memory management/hierarchies**
  - are a second priority – automatic load balancing is a lower priority since we already do much of that ourselves
- Autotuning
  - Discussion of swapping algorithms for different situations
  - Automatic code generation optimized for particular system configuration (dynamic)

What system software functionality is required by applications at the Exascale?

- Debug Tool
  - On node, modest size system, full size (high level msg only)
  - How find large scale bugs? Long run bugs?
  - Sometimes used to debug science
- Performance Tool
  - Holistic performance measure (node-system) w/ integrated perf model
- File system and I/O
  - Currently spending 1/3 of our allocation doing I/O, currently use 10% of the nodes just for check pointing, the rest of the nodes send data to them and get back to work
  - File system scalability and robustness (concern)
  - I/O will be different than we do now – more in situ analysis to reduce data
- System dynamic (configuration/performance/faults) information space
  - Can be queried by tools, apps, system layers

Not-in-same-computation (simulation) processing

- In-situ analysis (OS issue: multitasking). What is the programming model and what is the OS mechanism that separates the analysis from the simulation in a way that makes sense? Multiple possibilities:
  - As part of the same of computation.
  - Shared address space mapping.
- Computation near the I/O. The analysis does not operate necessarily in the same cluster but it operates on the same cluster storage. There are many issues:
  - Performance isolation: (what the visualization does cannot cause simulation to slow down).
  - Partitioning visualization to execute where the data is.

NVRAM

- For data analysis there will be computations that are not sequential on huge data sets.
- For those, if the NVRAM is there for checkpointing, we would like the system software to expose NVRAM.
- Attraction: NVRAM has 2 orders of magnitude faster access compared to disk.
- We need to identify an appropriate programming model and services provided by the OS, while also ensuring checkpoint success.

Seamless Memory/Storage usage?

- Depending on the size of data, we can either
  - Do the analysis in RAM of visualization node if it fits in memory,
  - Or, move the computation to storage.
- How do we manage this?
  - This is a programming model challenge and system software challenge.
  - It could be easier for the OS to explore solutions if the data fits in memory

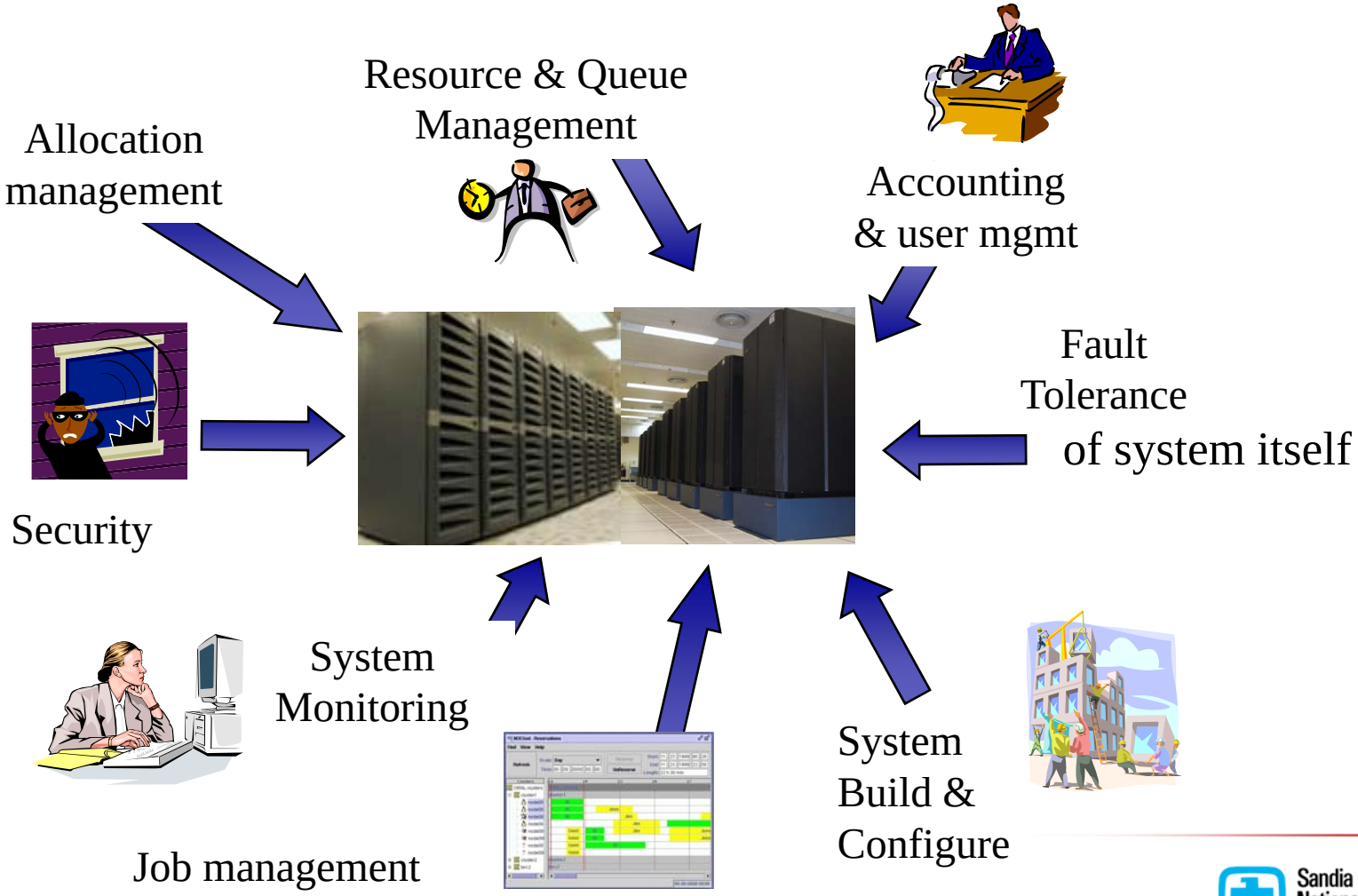
System Software for Applications

Software Stack Components for Developing and Running Jobs

- Execution
  - Operating system
  - Runtime system
  - File system and parallel I/O
  - Communication (MPI, SHMEM, etc.)
  - Fault detection, notification, recovery
- Development
  - Compilers
  - Debuggers
  - Performance tools
  - Math libraries
- Knowledge discovery
  - Data analysis
  - Visualization
  - Workflows

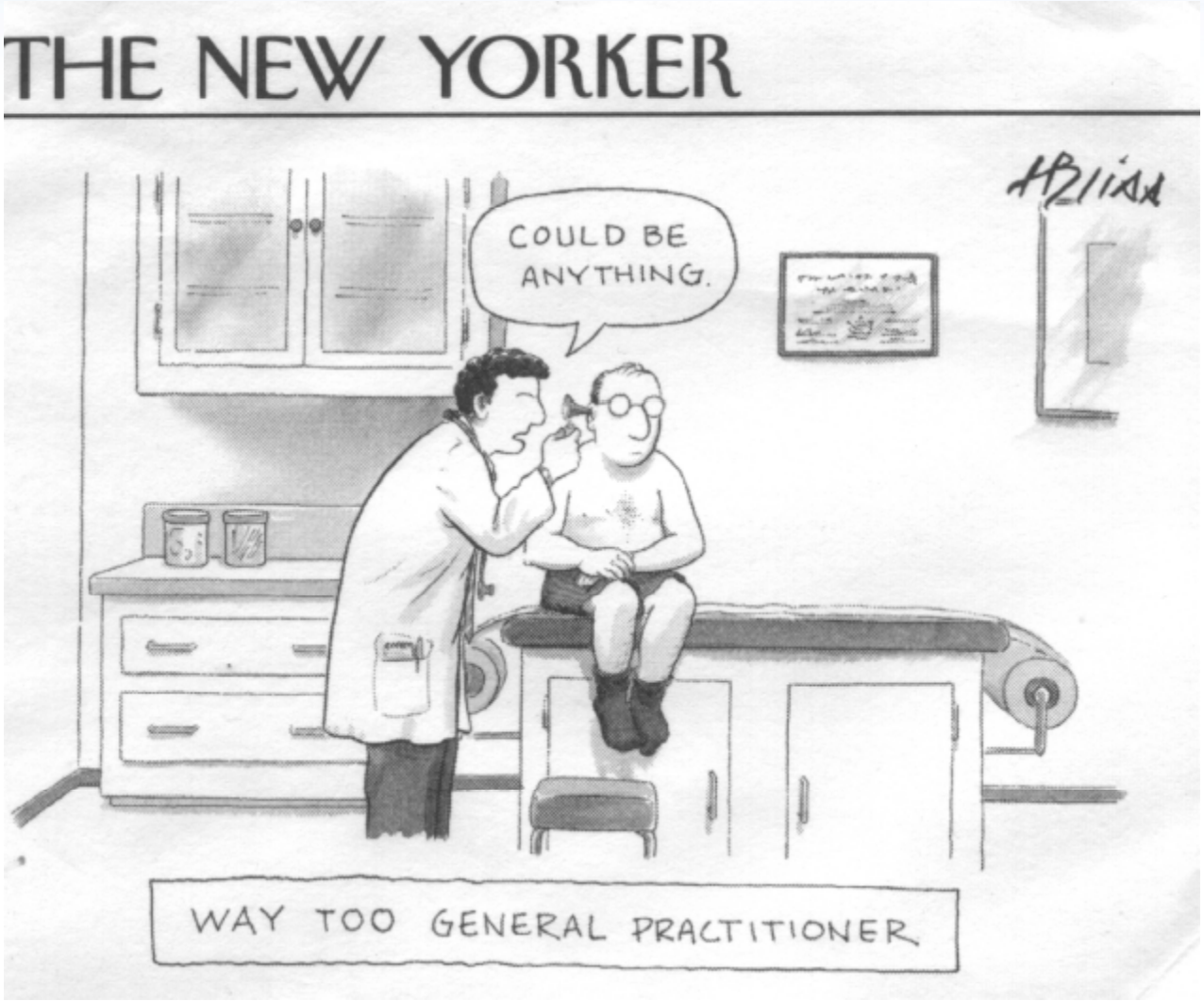
Many Aspects of System Software

There is managing users, and maintaining the system



People From Whom I Stole Slides

- Jason Howard (Intel)
- John Shalf (LBL)
- David Brown (LLNL)
- John Bell (LBL)
- Al Geist (ORNL)
- Mihai Anitescu (ANL)


THE NEW YORKER
COULD BE ANYTHING.
WAY TOO GENERAL PRACTITIONER