Scientific Innovation Through Integration

Data-Driven Challenges in Scientific Computing at PNNL

PNNL-SA-72116

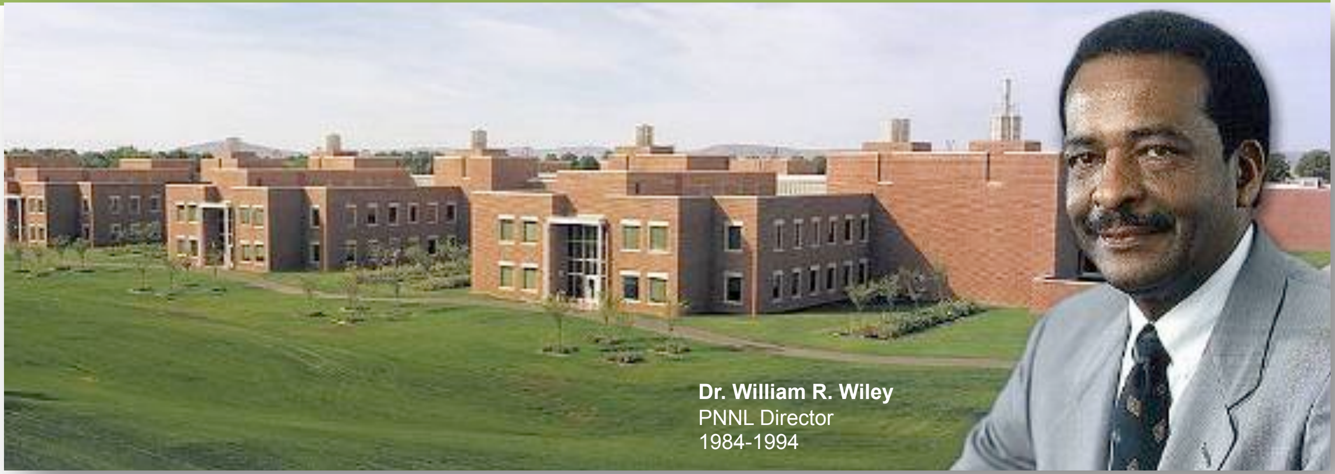
David Cowley

Senior Research Scientist

Pacific Northwest National Laboratory

EMSL is a national scientific user facility

William R. Wiley's Vision:
An innovative multipurpose user facility providing *"synergism between the physical, mathematical, and life sciences."*

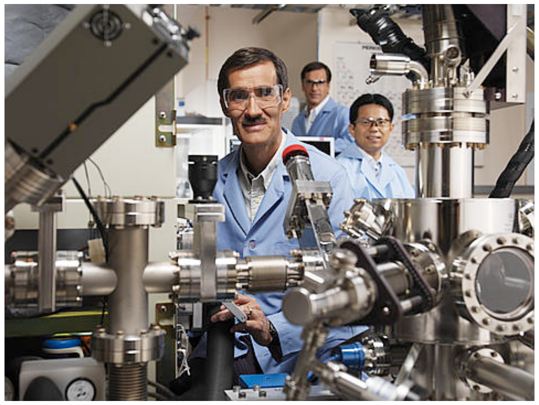


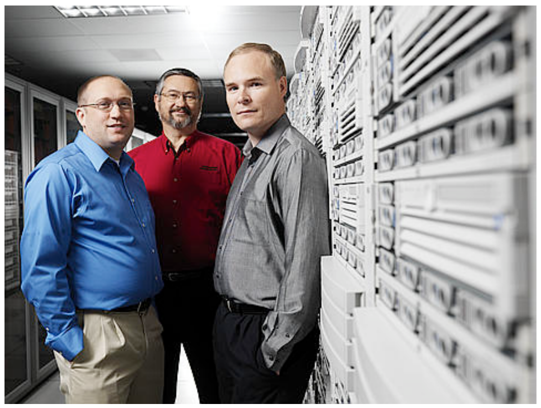
Mission

EMSL, a national scientific user facility at Pacific Northwest National Laboratory, provides ***integrated experimental and computational resources*** for ***discovery and technological innovation*** in the environmental molecular sciences to ***support the needs of DOE and the nation***.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965
U.S. DEPARTMENT OF ENERGY

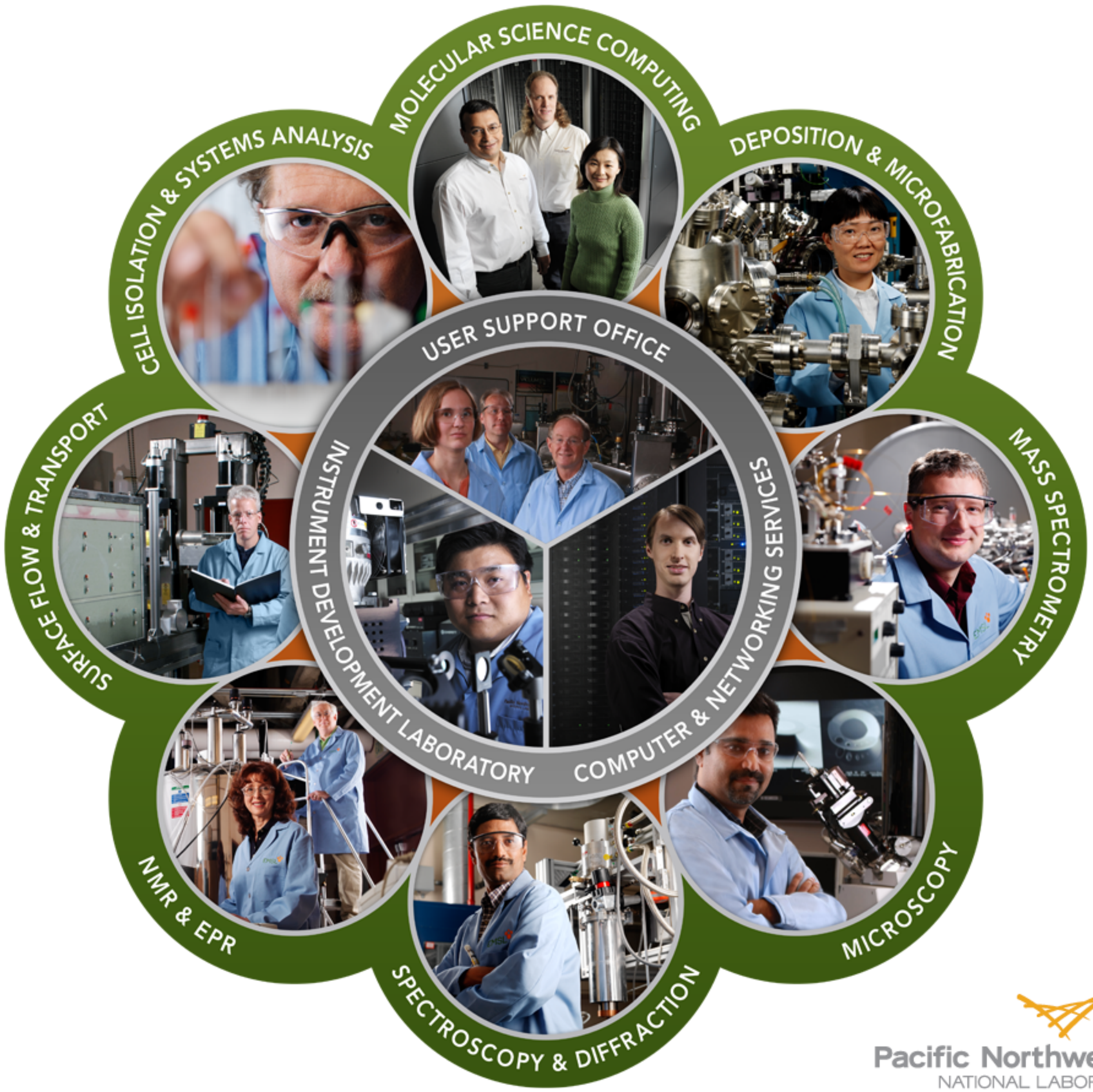
Characteristics of EMSL





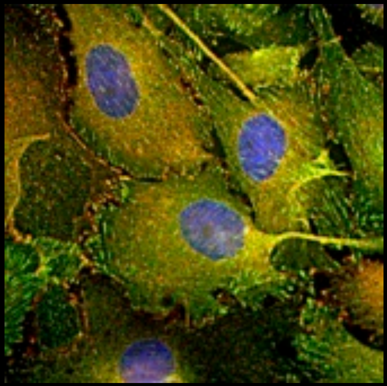
- Scientific **expertise** that enables scientific discovery and innovation.
- Distinctive focus on **integrating** computational and experimental capabilities and collaborating among disciplines.
- A **unique collaborative environment** that fosters synergy between disciplines and a complimentary suite of tools to address the science of our users.
- An impressive suite of **state-of-the-art instrumentation** that pushes the boundaries of resolution and sensitivity.
- An **economical** venue for conducting non-proprietary research.

EMSL capabilities


MOLECULAR SCIENCE COMPUTING
DEPOSITION & MICROFABRICATION
MASS SPECTROMETRY
MICROSCOPY
SPECTROSCOPY & DIFFRACTION
NMR & EPR
SURFACE FLOW & TRANSPORT
CELL ISOLATION & SYSTEMS ANALYSIS
USER SUPPORT OFFICE
COMPUTER & NETWORKING SERVICES
INSTRUMENT DEVELOPMENT LABORATORY

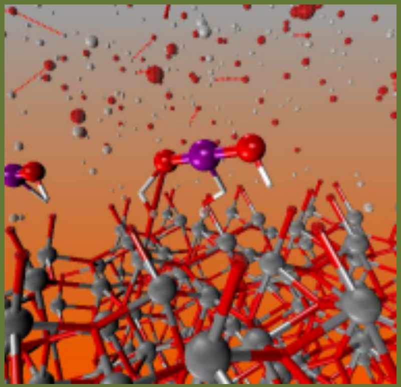
The user program is focused on three science themes

Biological Interactions and Dynamics



Understanding and optimizing the response of biological systems to their environment.

Geochemistry/Biogeochemistry & Subsurface Science



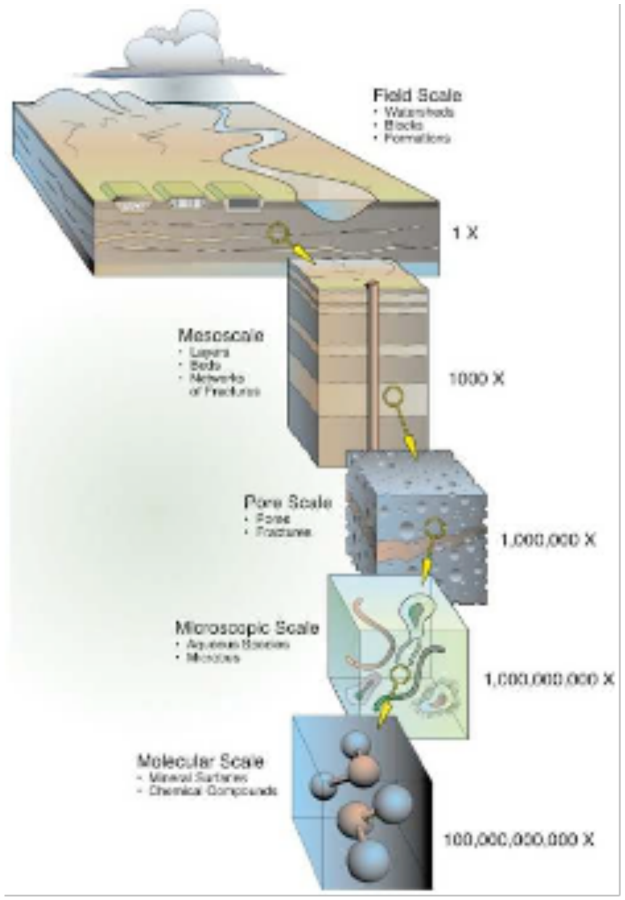
Unraveling molecular-level phenomena to determine their impact on contaminant migration and transformation.

Science of Interfacial Phenomena

Developing & verifying predictive models for interfacial processes and advancing understanding of structure-function relationships in complex systems.

Sample application: Carbon Dioxide (CO_2) Sequestration

- It's a simple idea
  - Problem: burning fossil fuels has increased the amount of CO_2 in the atmosphere
  - Solution: capture it and put it back in the ground!
- Would it work?
- Computer modeling is the best way to figure that out
- But to really model this, we need to understand a lot!
  - How CO_2 will react chemically with minerals and fluids in the ground
  - How CO_2 will interact with biology in the ground
  - Heat, temperature, and physical stresses
  - Changes in permeability of rock, clay, and soil
  - Geological effects such as cracking, fracturing, and sealing
- This illustrates the multi-scale nature of this type of science problem

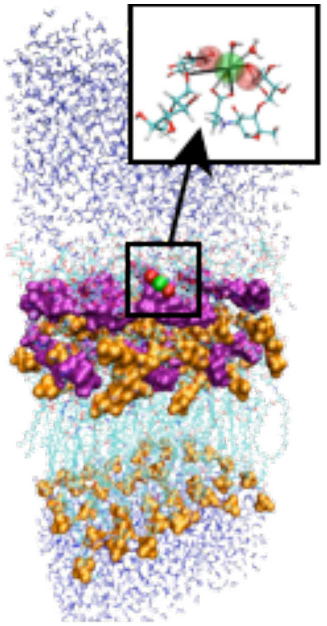


Sample application: Biofuels from cellulose

- Another simple idea
- We need chemical fuels for their energy capacity, and fossil fuels have a lot of problems with them
- We can get around this by making ethanol from plants
- Problem: It's proving to be a really bad idea to use potential food crops as a fuel source
- Solution: Use cellulose!
  - It's Renewable and plentiful (10x the world's energy needs)
  - It's not food and doesn't release fossil carbon
  - There's a lot of it already in waste products
- We know certain enzymes can be used to break cellulose down into ethanol (certain fungi and bacteria make these enzymes)
- To make this ethanol at useful scales, we must understand at the molecular level how these enzymes work, which is a major computational biology challenge
- We are going to need far more computational power than we currently have in order to do the simulation that's needed!

A major new challenge: Computational Biology

- Computational biology is coming on strong
- In some ways it's "big chemistry": more molecules, more complexity, larger systems
  - Chemists can just now do high-quality Molecular Dynamics (MD) simulations on a hundred atoms
  - Biologists want to do this on at least many thousands!
  - Biologists want to keep and compare MD trajectory data sets (we've always treated these as "too big to keep")
- In other ways it's nothing like chemistry
  - Many applications involve searching databases of proteins & peptides for matches
  - Experiments are now highly automated, creating a flood of large, growing, complex data sets
  - Challenges calling for exascale systems:
    - analysis of mass spectrometry data for whole genome scale proteomics measurements
    - structure and dynamics of molecular machines
    - discovery of cellular networks and their reconstruction from high-throughput data
    - simulation of the dynamic properties of cellular networks

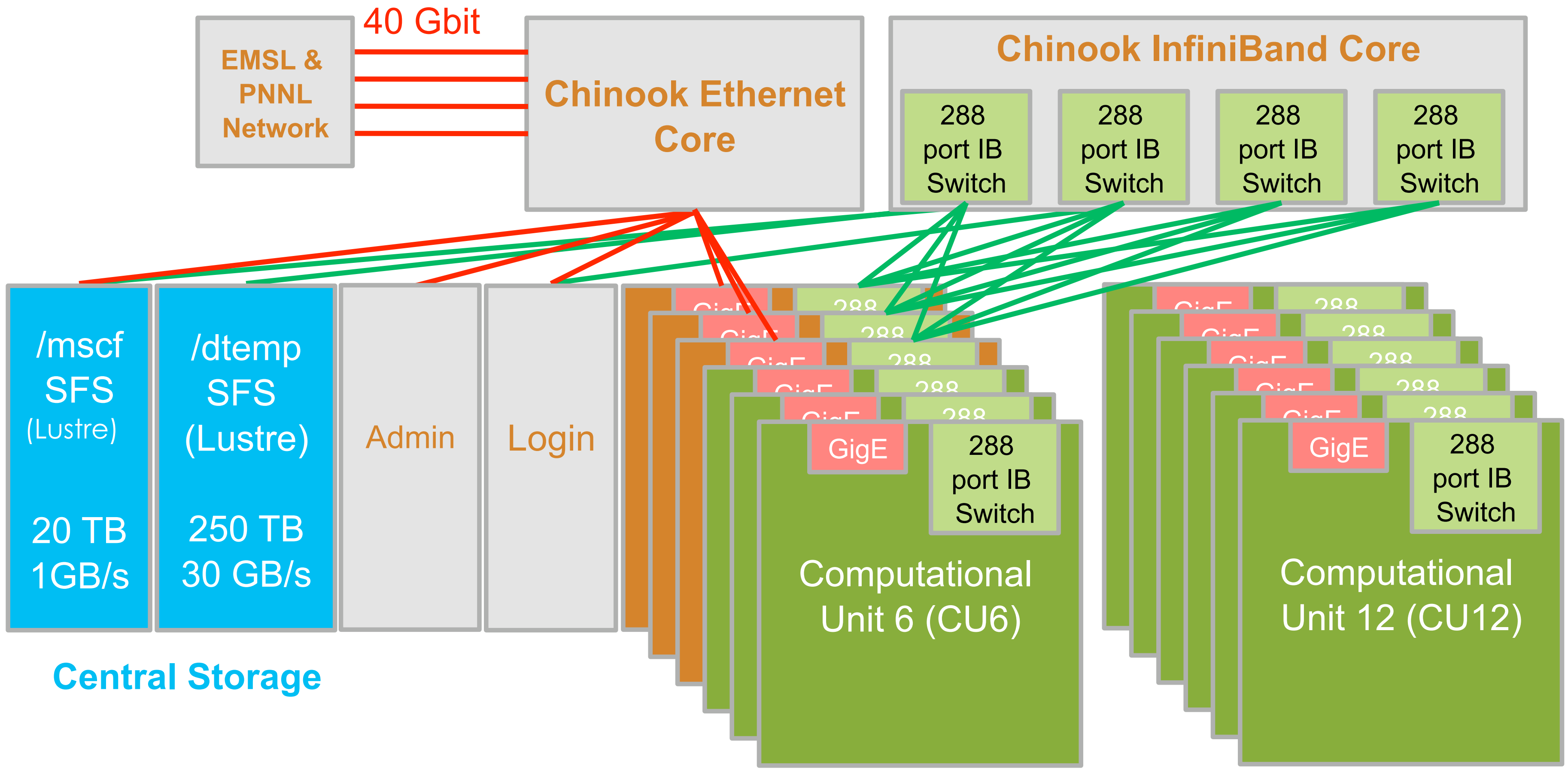


EMSL's flagship HPC system: Chinook

2323 node HP cluster

Feature	Detail
Interconnect	DDR InfiniBand (Voltaire, Mellanox)
Node	Dual Quad-core AMD Opteron 32 GB memory
Local scratch filesystems	440 MB/s, 1 TB/s aggregate 440 GB per node. 1 PB aggregate
Global scratch filesystem	30 GB/s 250 TB total
User home filesystem	1 GB/s 20 TB total

Chinook cluster architecture



2323 nodes, ~192 per CU

Experiences with Chinook

EMSL

- Full system in production for just over a year now
- Lots of good science is getting done on it
- Strengths:
  - 4 GB RAM/core
  - Lots of local disk bandwidth/core
- Weaknesses:
  - Version lag in the software stack
  - Our Infiniband topology is a little weird, which limits our options for optimizing its performance
  - We'd still like more RAM per processor
- Use of large pages is still a feature we'd like to use
  - other work has drained staff time away
  - reinforces the idea that it's more cumbersome than it ought to be

peds
Biopolymers
THE JOURNAL OF PHYSICAL CHEMISTRY C
Journal of Physics Condensed Matter
Science
THE JOURNAL OF PHYSICAL CHEMISTRY B
THE JOURNAL OF PHYSICAL CHEMISTRY B
PCCP
THE JOURNAL OF PHYSICAL CHEMISTRY A
PCCP
CHEMICAL PHYSICS LETTERS
THE JOURNAL OF PHYSICAL CHEMISTRY A
PROTEINS
THE JOURNAL OF PHYSICAL CHEMISTRY C

EMSL's new test bed system: Barracuda

- Barracuda was procured with Recovery Act funds via DOE's Office of Science
- It's got all the latest hardware toys to serve as a testbed for scaling parallel codes towards exascale:
  - Multicore Nehalem processors
  - QDR Infiniband
  - Nvidia Tesla GPGPUs
  - Fusion IO PCE-Express SSDs
- This new system will be crucial to develop and test software that will allow NWChem and other software to exascale

Next stop: Exascale computing

- The previous examples (and many others!) call for exascale computing
- Today's fastest computers have just beaten the petaflop barrier
  - Petaflop = 10^{15} operations/second
  - Exaflop = 10^{18} operations/second
- Current hardware trends will take us to exascale in the 2018 time frame
- Current software and computational techniques will not get us there!
  - There's too much complexity in the programming models for today's tools
  - We have no way of even supporting enough processes

A note about recent evolution of HPC software

- Terascale to petascale transition:
  - Hardware got dramatically faster
  - System architecture stayed about the same
  - degree of parallelism increased ~10x
- Petascale to exascale transition:
  - Physics put the brakes on ever-increasing clock speeds
  - Node architecture changes dramatically – we get *more* parts instead of faster ones
  - Degree of parallelism must increase by ~1000x

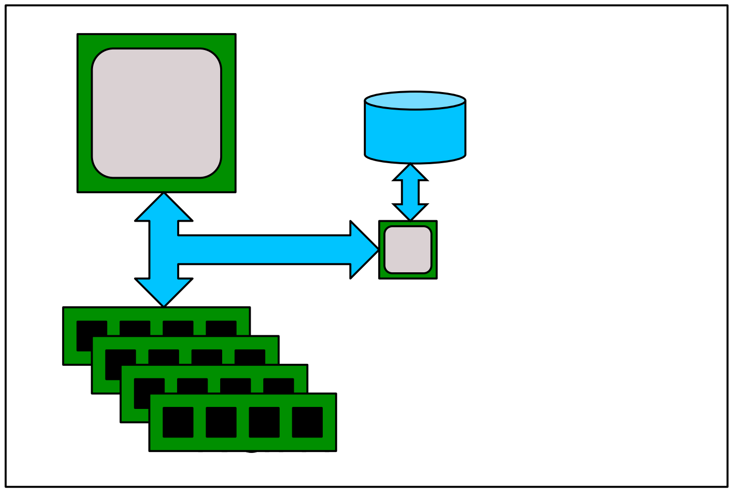
Computation is likely to become data-starved

- A core needs a constant stream of data in and out to do anything efficiently
  - Memory speeds are not keeping pace with processor speeds
  - If core counts double as memory speed increases by 50%, each core has less and less effective RAM bandwidth in newer and newer machines
- Data rates will throttle performance
  - Do we move the data to where the work is done?
  - Do we move the work to where the data is?
  - Can we even keep all the values we compute?
  - How is a filesystem going to behave when it's got a million processes banging on it?

Internet Exascale Software Project (IESP)

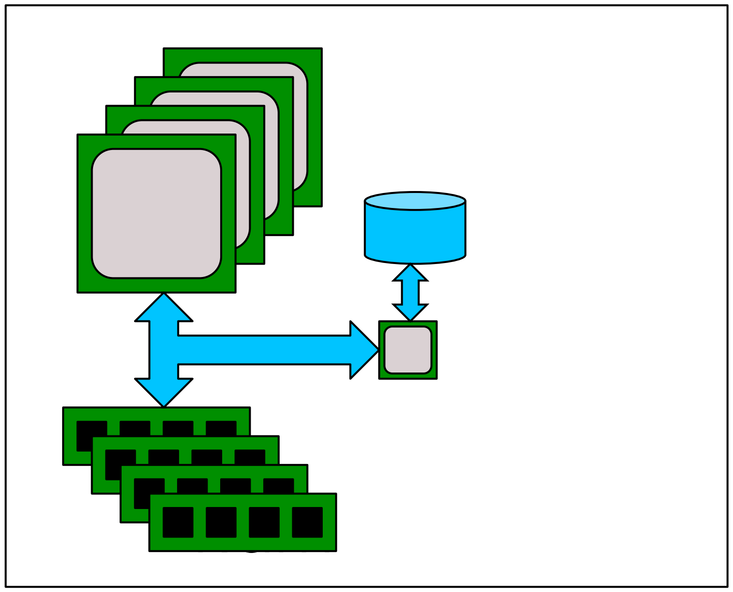
- http://www.exascale.org
- Studying issues associated with exascale software and developing a roadmap for getting there
- They call out 3 areas where system architectures will need to be improved:
  - Concurrency – increasing the number of processes/threads doing work
  - Reliability – mitigating the probabilistic error rates that are inevitable in a system containing billions of transistors
  - Power Consumption – Reducing power consumption of exascale system from projected 65 – 100 megawatts
- Concurrency will give us plenty to talk about today!
- Open source has a role to play here
  - It's a good model for the level of collaboration that's needed
  - They do feel it's going to need to be a very coordinated kind of collaboration

A compute node in the "good old days"



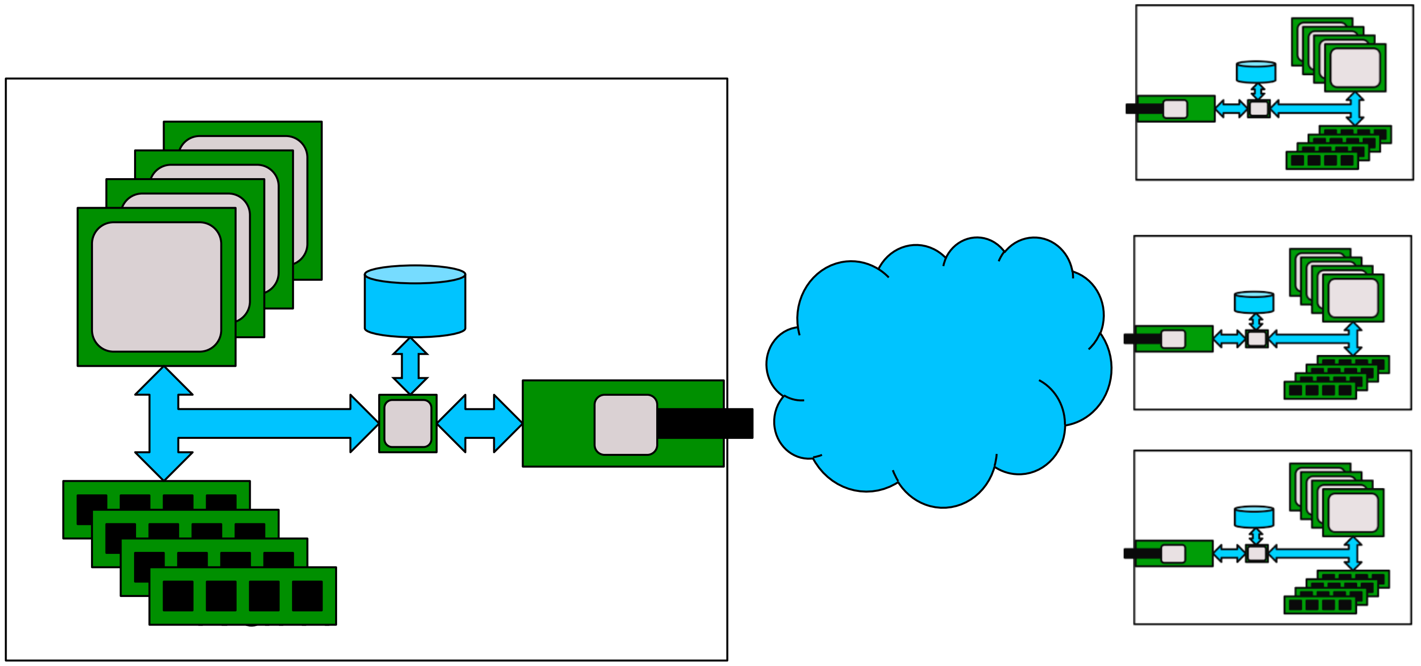
- Processor, RAM, Disk storage, maybe instruction or data caches.
- Simple!
- But we needed more

Multiprocessing



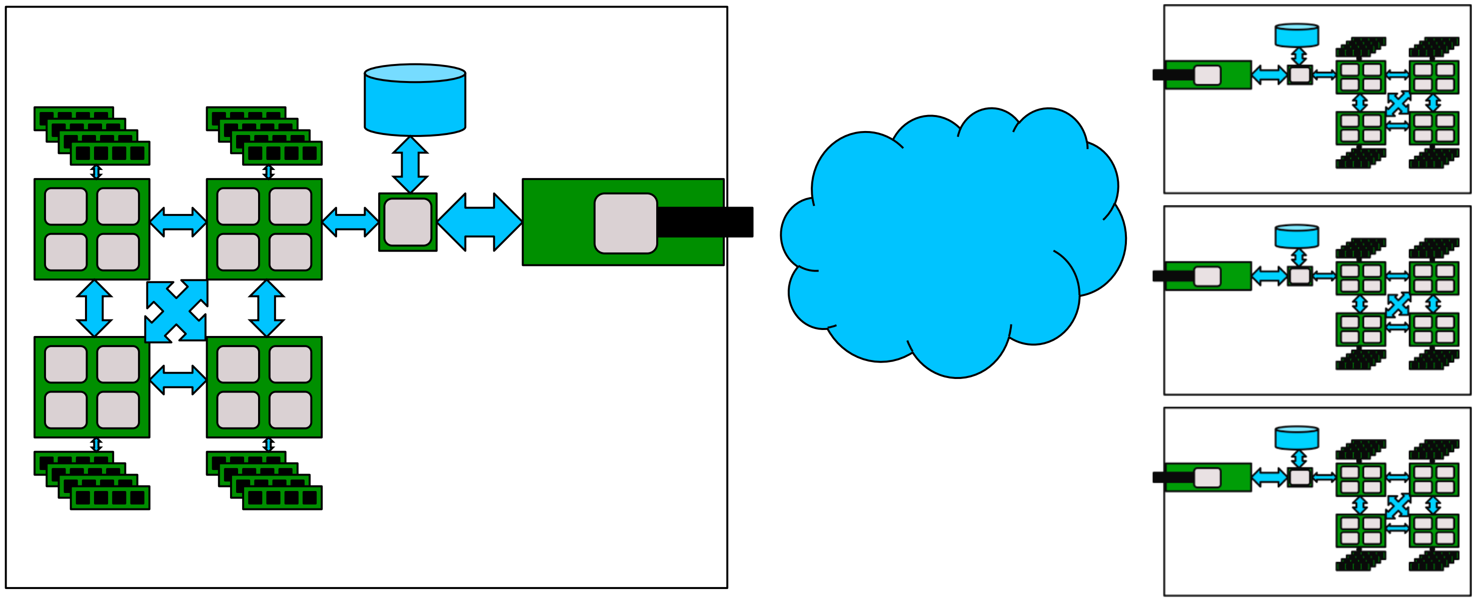
- Add more processors, get more work done!
- Processors have level 1 level 2 caches, with hardware cache coherency
- This works – to a point. But hooking more processors to the same memory gets expensive fast!!
- Systems were built up to about 64-way like this, but they were very pricey

Distributed Memory



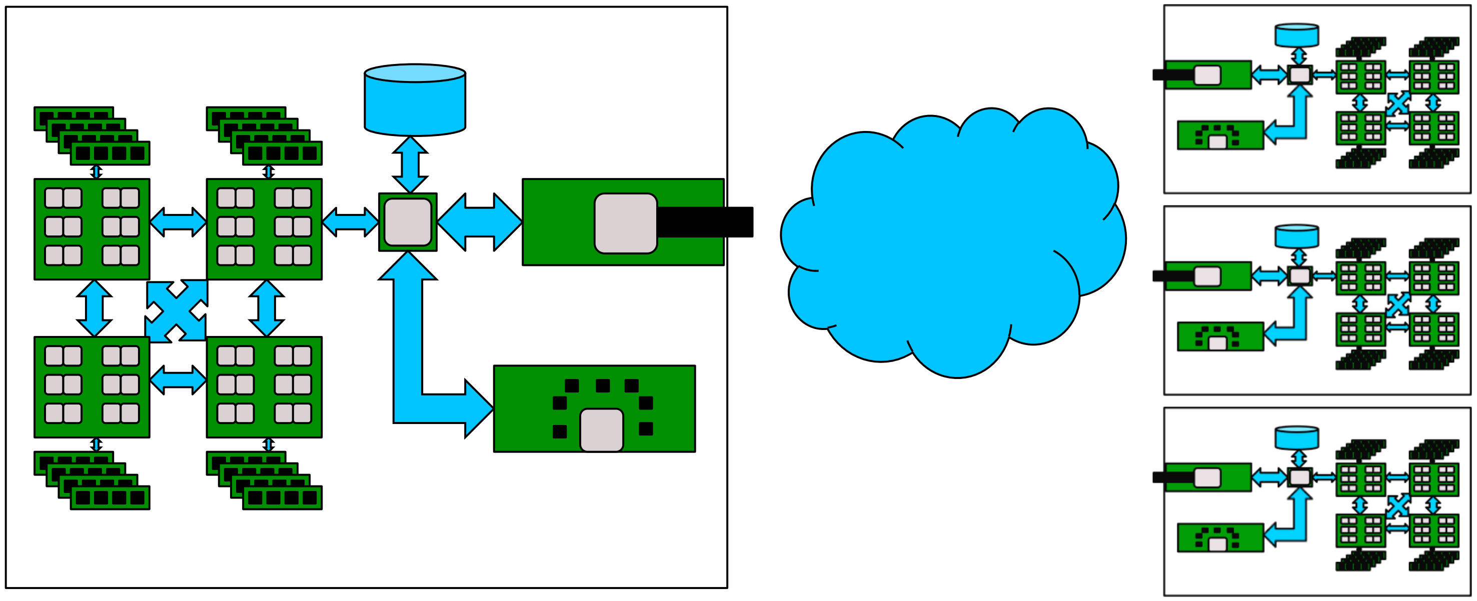
- OK, big multiprocessors are expensive, so lash together economical smaller machines with a network
- Run parallel code on sets of machines
- Use message passing to move data between the machines when needed
- This is where we were a couple of years ago, and it gets us just into teraflops territory
- This is hard to program, because it puts all the job of correctly managing data on the programmer's shoulders!

Multicore!



- Four cores/socket is common today, but 8- and 12-way chips are in stores now!
- Note the added complexity and what happens with memory:
  - It's now connected to the processor socket
  - A given memory location may be on your socket, or on a remote one
  - Cores tend to have their own level 1 and level 2 caches, with level 3 cache shared socket-wide

Coming soon: even *more* multicore and GPGPU



- Between all those cores and the GPU, there is tremendous raw processor power
- The price to be paid is:
  - Tremendous complexity in programming (multiprocessing, message passing, GPU programming and dispatch)
  - Making sure hundreds to thousands of processes can be scheduled and serviced effciently
  - Memory and storage speed per core is 1 / #cores: approaches zero as core count increases!

The process model problem

- Silicon is not getting faster anymore
- Since the manufacturers can't give us faster cores, they give us more
- Parallelism is the only way to go significantly faster, and it's going to have to be massive parallelism
- Apply Moore's law to the number of processor cores!
- Today we can do thousands or tens of thousands of processes on the most scalable applications
- That's just not enough
- Consider: if a chip has 100 cores on it, each core gets 1/100th the memory (or cache speed) that we enjoy today!
- Exaflop applications will require million-way parallelism
- We don't have ways to do that yet!

Time	Core count
Now	8
+18 months	16
+36 months	32
+54 months	64
+72 months	128
+90 months	256
+108 months (2018)	512

Proudly Operated by Battelle Since 1965

Computation on GPGPU

- Nvidia “Fermi” 400 series offers ECC, improved double-precision floating point math
- So now it looks really good for HPC
- 512 concurrent hardware threads, peak performance of 600 GF/sec per chip
- 200 GB/sec inside GPU, but the bus it's in maxes out at 8 GB/sec
- “application kernel” based programming model – you write GPU code , then push it into the GPU to be run, along with the data it's supposed to operate on
- Need to balance kernel runtime against startup time against CPU performance & workload

So in this picture, what's going on with the data?

- All these flops and so little bandwidth!
  - We're on track to have 10^{-2} **less** effective memory bandwidth per flop than we do now
  - We need to maximize concurrency (i.e. keep as many things as busy as possible at the same time)
  - Consider that it may make more sense to recalculate than to fetch from memory!
- So where's the data? The *programmer* now has to care!
  - Recently:
    - It's in local memory somewhere
    - It's on a remote node
  - Today and tomorrow:
    - Maybe it's in this socket's memory
    - Maybe it's in another socket's memory
    - Maybe it's down in the GPU
    - Maybe it's in one of those places on another node

Kernel Issues

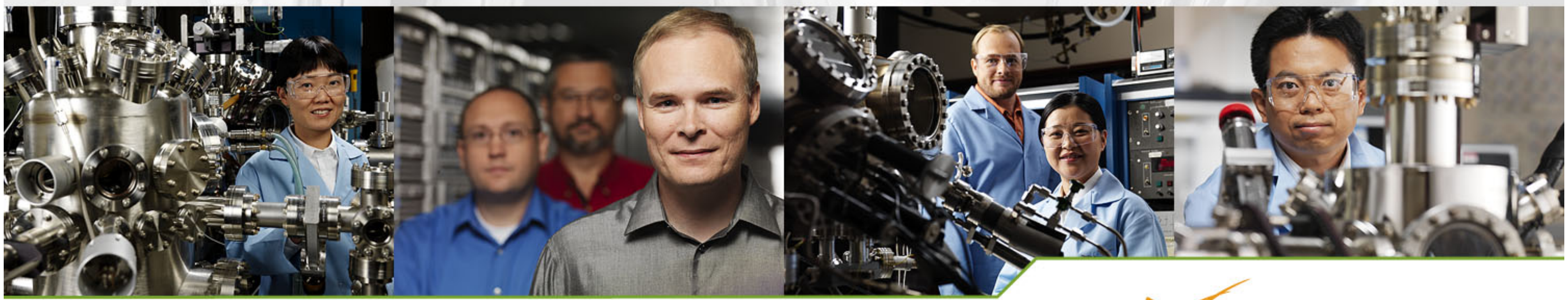
- Can the kernel support 10^2 or 10^3 processors with at least that many busy threads?
- All of those processes will want to do things like:
  - Get scheduled and run
  - Allocate memory
  - Use shared memory
  - Do I/O to block devices and network interfaces
  - Dispatch work to accelerators
- The kernel will need to use as much asynchrony as it possibly can as it goes about its business
- Oh, and hardware cache coherency?
  - It's probably gone once we have multiple dozens of cores in a socket
  - So expect that not all processors will see the same value in the same memory address unless you fix it yourself

"...the scalability requirements for even a single socket of an extreme scale system will be two orders of magnitude higher than what can be supported by Linux today. It is clear that future scalability improvements in Linux are expected to be harder to achieve, as evidenced by the RCU [(read-copy update API)] experience and the complexities uncovered by ongoing efforts to reduce the scope of the Linux big kernel lock." – Software Challenges at Extreme Scale, Dongarra, et al, SciDAC review #16 Special Issue 2010

Takeaway messages

- Energy and environmental problems (and many others) are crying out for exascale computing
- The hardware will take us there within the next decade
- But that hardware will demand that we have a software stack that:
    - Runs 1000x more processes than we can now
    - Does sensible things with insensible quantities of data
    - Relieves the programmers and users from having to know the details of a system's memory architecture, general purpose processors, accelerators, and interconnect
- Further reading:
    - http://www.emsl.pnl.gov/capabilities/computing
    - http://www.exascale.org
    - http://www.scidacreview.org/1001/index.html

Questions? Comments?



www.emsl.pnl.gov