Introduction to Parallel Processing

ACF Spring HPC Training Workshop
Match 15-16, 2016
Kwai Wong

Joint Institute for Computational Sciences
THE UNIVERSITY OF TENNESSEE KNOXVILLE
THE UNIVERSITY OF TENNESSEE HEALTH SCIENCE CENTER

Acknowledgements:

- Support from NSF, UTK, JICS, NICS
- Efforts from many colleagues, collaborators, and students
- Credits to many researchers and industrial practitioners for a lot of materials that I use in this talk, plenty of new science, new technologies and applications in HPC
- www.olcf.ornl.gov, www.xsede.org, www.nics.utk.edu, www.jics.utk.edu/recsem-reu

University of Tennessee – GO VOLS

Contents

- Landscape of Supercomputers
- Performance Ranking : Top500, HPL
- Supercomputers => Big Science + Big Data
- Programming Model on High Performance Computers
- General Practices

JICS Joint Institute for Computational Sciences UT ORNL Computational Sciences

THE UNIVERSITY OF TENNESSEE KNOXVILLE

THE UNIVERSITY OF TENNESSEE HEALTH SCIENCE CENTER

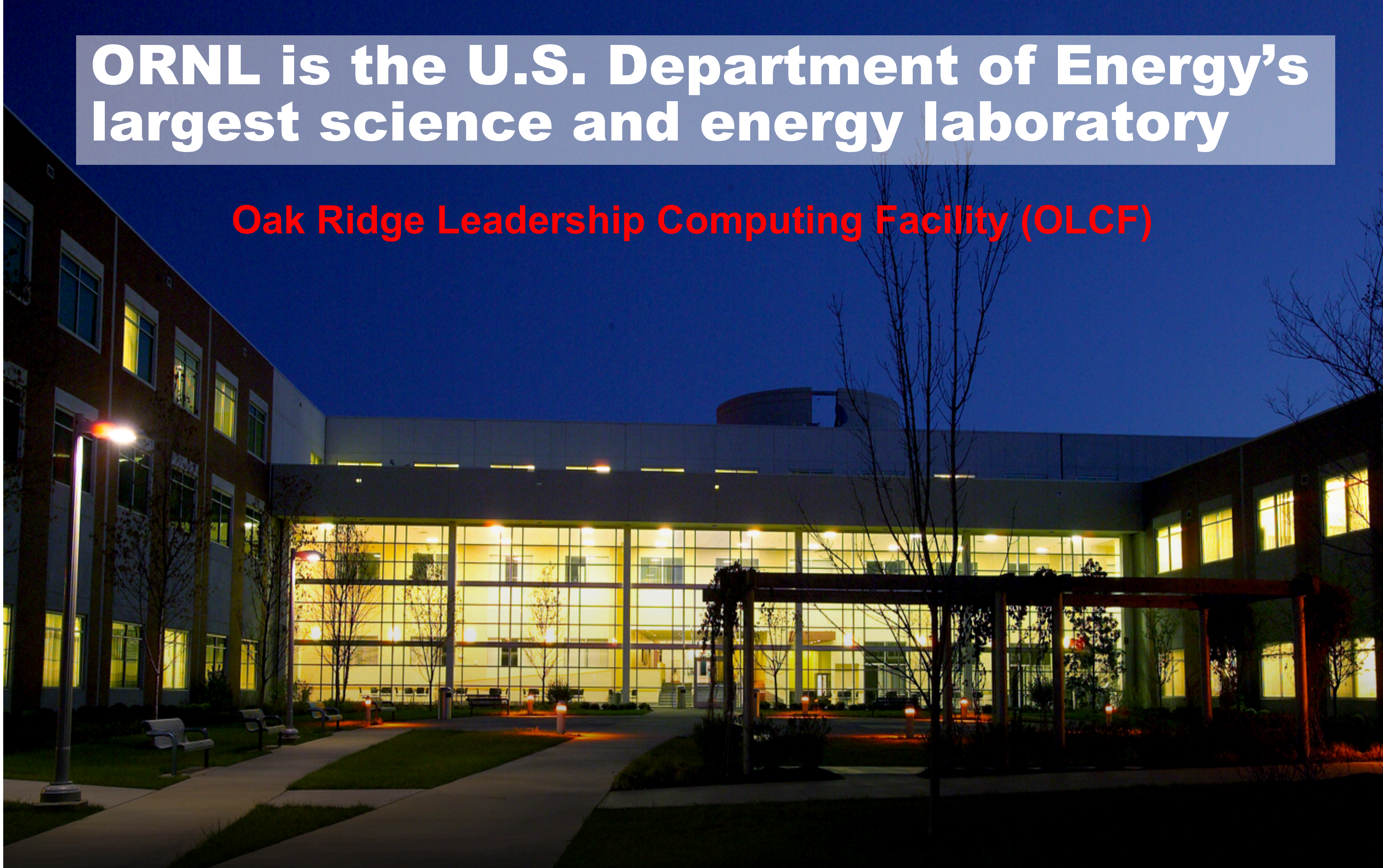
Joint Institute for Computational Sciences

- JICS is a joint research center between UTK and ORNL since 1991 to advance computational sciences activities
- Joint Faculty, research staff, National Institutes for Computational Sciences
- Projects : Kraken, RDAV, Keeneland, Beacon, XSEDE, ACF
- Total JICS funding > $100M

NICS – beacon (Xeon Phi), darter (XC30, kraken-E)

kraken: 1st Academic PetaFLOPS Computer (3rd 2009), 100 Cabinets, 112896 cores

ORNL is the U.S. Department of Energy's largest science and energy laboratory

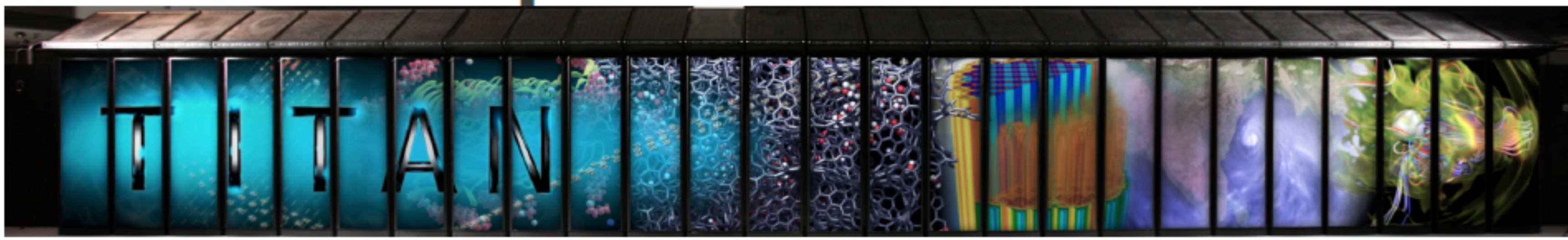


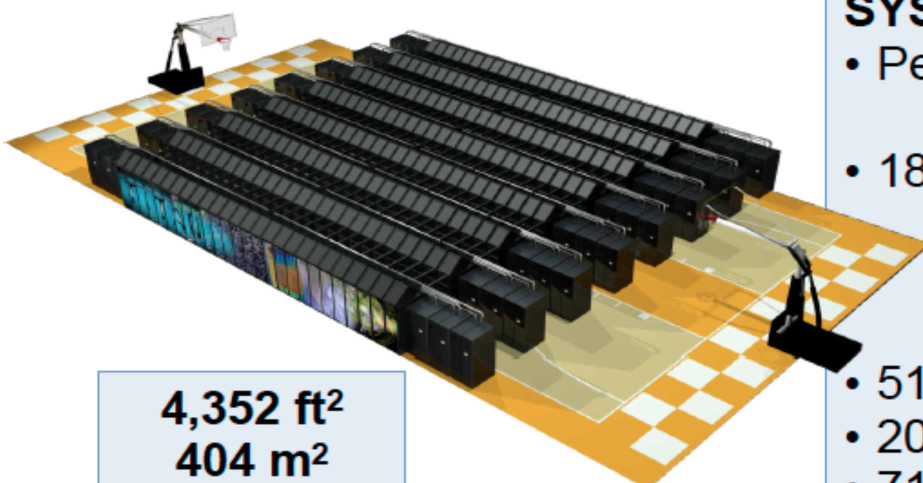
Oak Ridge Leadership Computing Facility (OLCF)

- World's premier computing facility
- Nation's largest concentration of open source materials research

- Nation's most diverse energy portfolio
- $1B+ Spallation Neutron Source project
- Managing the $1B+ U.S. ITER project

ORNL's "Titan" Hybrid System: Cray XK7 with AMD Opteron and NVIDIA Tesla processors





4,352 ft²
404 m²

SYSTEM SPECIFICATIONS:

- Peak performance of 27 PF
  - 24.5 Pflop/s GPU + 2.6 Pflop/s AMD
- 18,688 Compute Nodes each with:
  - 16-Core AMD Opteron CPU
  - 14-Core NVIDIA Tesla "K20x" GPU
  - 32 GB + 6 GB memory
- 512 Service and I/O nodes
- 200 Cabinets
- 710 TB total system memory
- Cray Gemini 3D Torus Interconnect
- 9 MW peak power

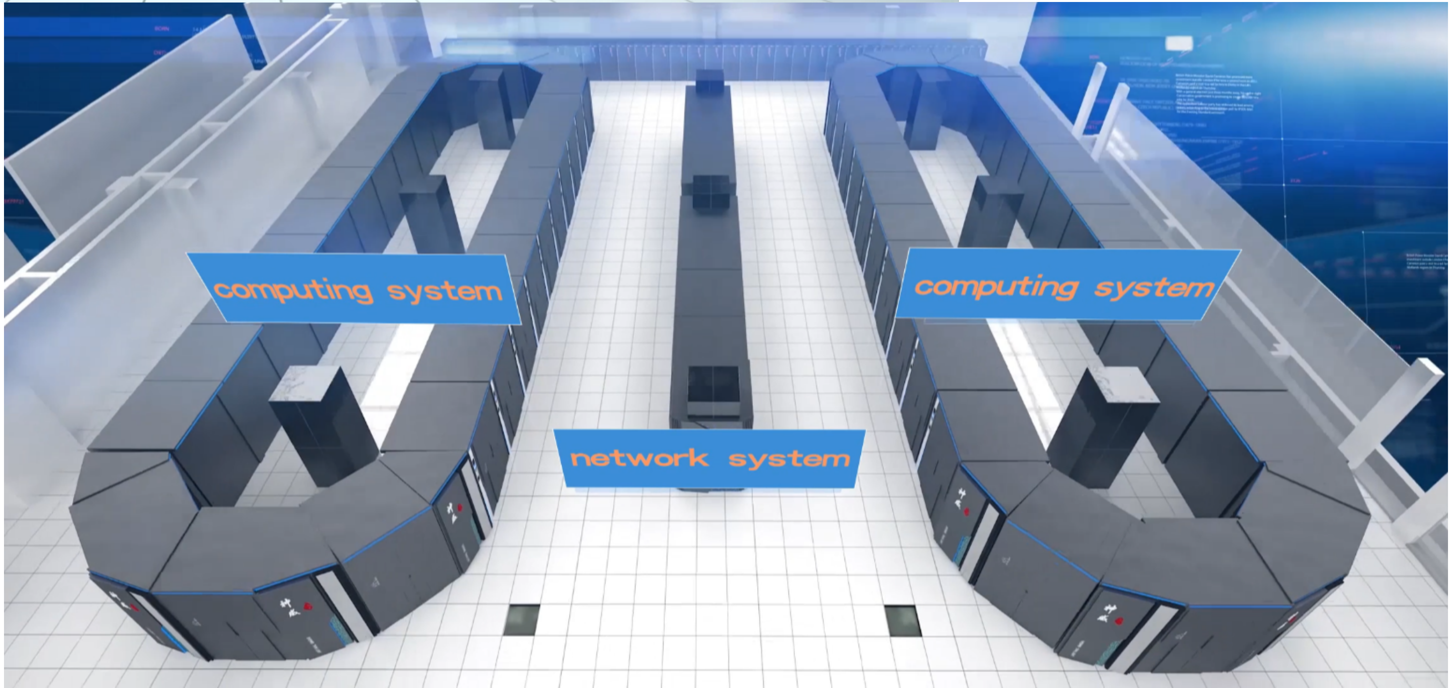
✓ (CPU) 2.26 x 4 x 18688 = 2.392 ; (GPU) 1.31 x 18688 x 14 = 24.27 PF, Peak=26.67

✓ 17.5 PFLOPS (HPL) 64.8% ; ~ 10 times faster than jaguar; 9 Megawatt,

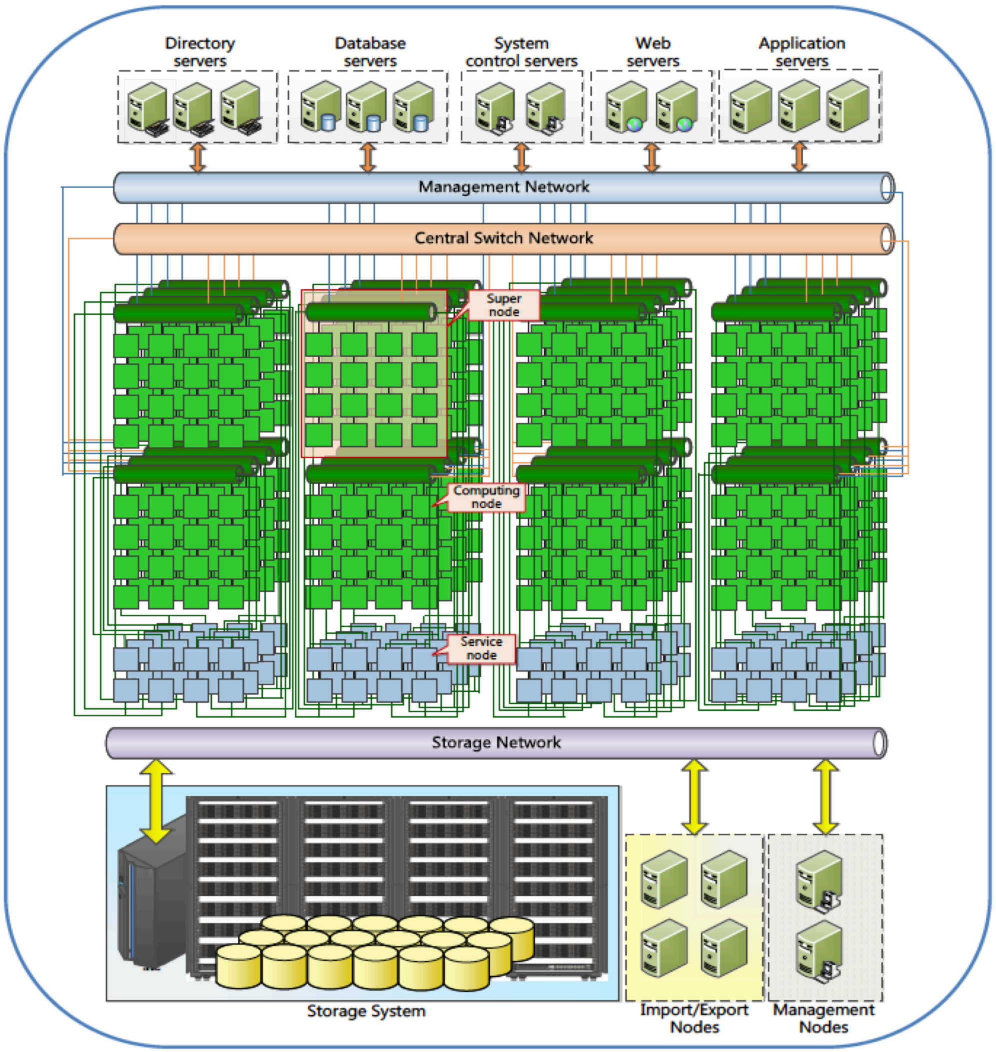
✓ 900 W/apartment – 10000 apartments !! --- Currently No. 5 in the world

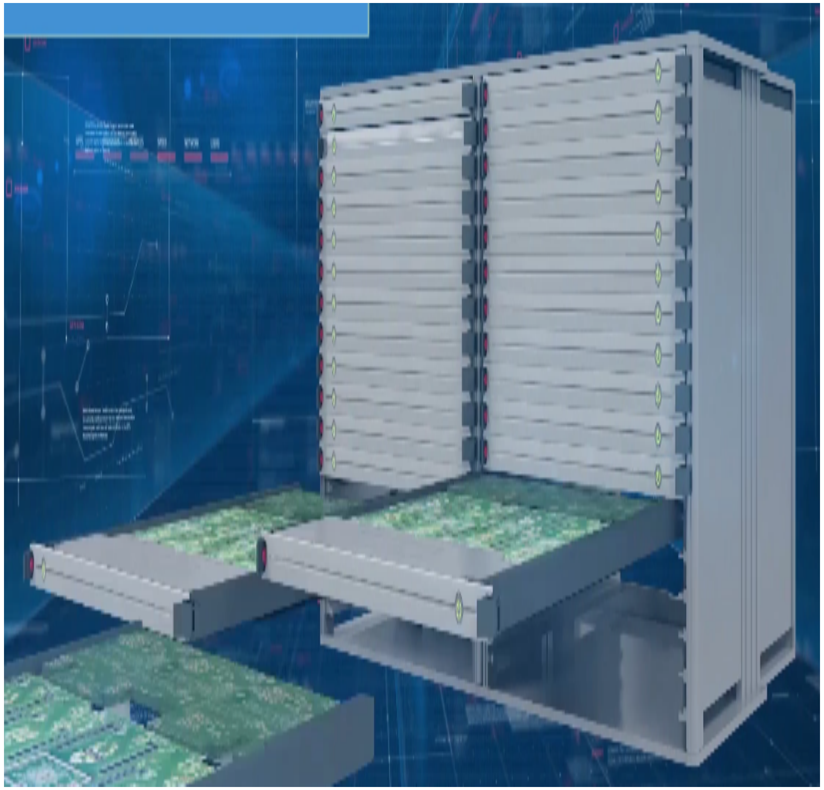
Sumway : Fastest Computer : TOP500

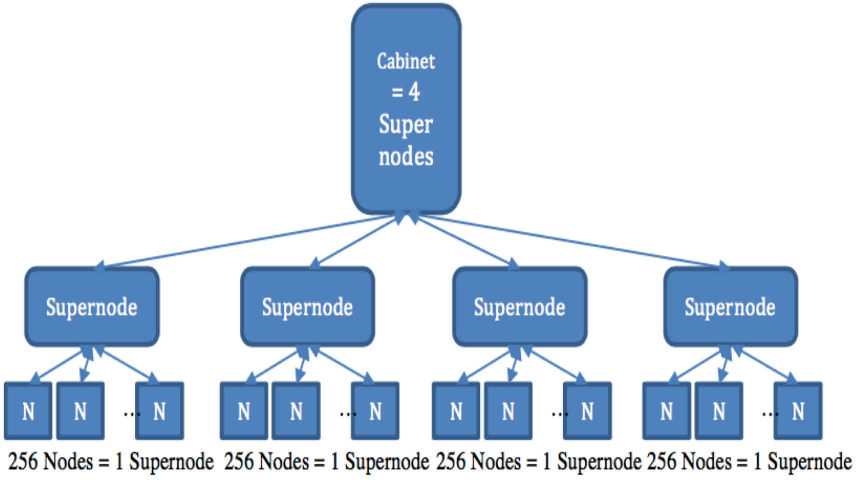
- Wuxi
- June 2016
- 15.3 MW
- 93 PF



Sunway - Wuxi - China





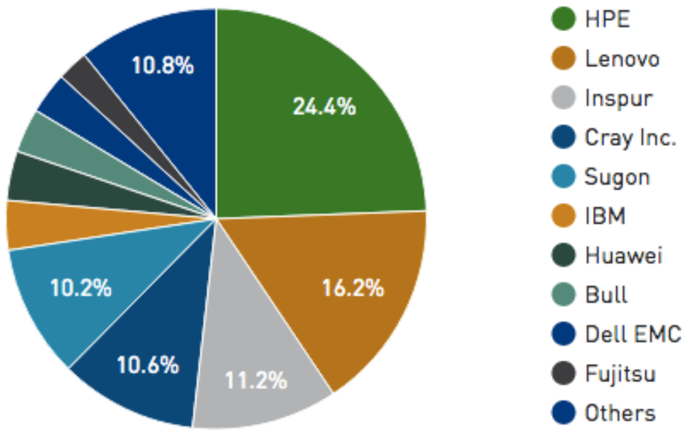


Top500 – Nov. 2017 – top500 list every 6 months
Solving a Ax=b : A is dense NxN Matrix ; MM

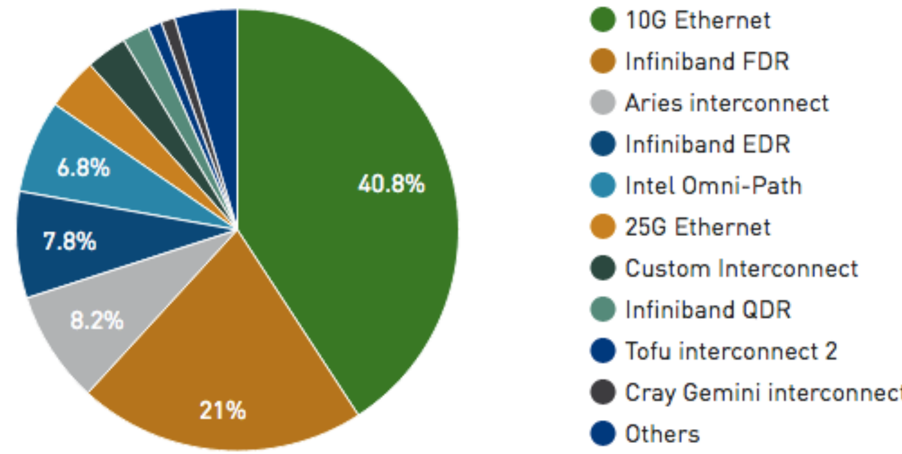
#	Site	Manufacturer	Computer	Country	Cores	Rmax [Pflops]	Power [MW]
1	National Supercomputing Center in Wuxi	NRCPC	Sunway TaihuLight NRCPC Sunway SW26010, 260C 1.45GHz	China	10,649,600	93.0	15.4
2	National University of Defense Technology	NUDT	Tianhe-2 NUDT TH-IVB-FEP, Xeon 12C 2.2GHz, IntelXeon Phi	China	3,120,000	33.9	17.8
3	Swiss National Supercomputing Centre (CSCS)	Cray	Piz Daint Cray XC50, Xeon E5 12C 2.6GHz, Aries, NVIDIA Tesla P100	Switzerland	361,760	19.6	2.27
4	Japan Agency for Marine-Earth Science and Technology	ExaScaler	Gyoukou ZettaScaler-2.2 HPC System, Xeon 16C 1.3GHz, IB-EDR, PEZY-SC2 700Mhz	Japan	19,860,000	19.1	1.35
5	Oak Ridge National Laboratory	Cray	Titan Cray XK7, Opteron 16C 2.2GHz, Gemini, NVIDIA K20x	USA	560,640	17.6	8.21
6	Lawrence Livermore National Laboratory	IBM	Sequoia BlueGene/Q, Power BQC 16C 1.6GHz, Custom	USA	1,572,864	17.2	7.89
7	Los Alamos NL / Sandia NL	Cray	Trinity Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries	USA	979,968	14.1	3.84
8	Lawrence Berkeley National Laboratory	Cray	Cori Cray XC40, Intel Xeons Phi 7250 68C 1.4 GHz, Aries	USA	622,336	14.0	3.94
9	JCAHPC Joint Center for Advanced HPC	Fujitsu	Oakforest-PACS PRIMERGY CX1640 M1, Intel Xeons Phi 7250 68C 1.4 GHz, OmniPath	Japan	556,104	13.6	2.72
10	RIKEN Advanced Institute for Computational Science	Fujitsu	K Computer SPARC64 VIIIfx 2.0GHz, Tofu Interconnect	Japan	795,024	10.5	12.7

STATISTICS

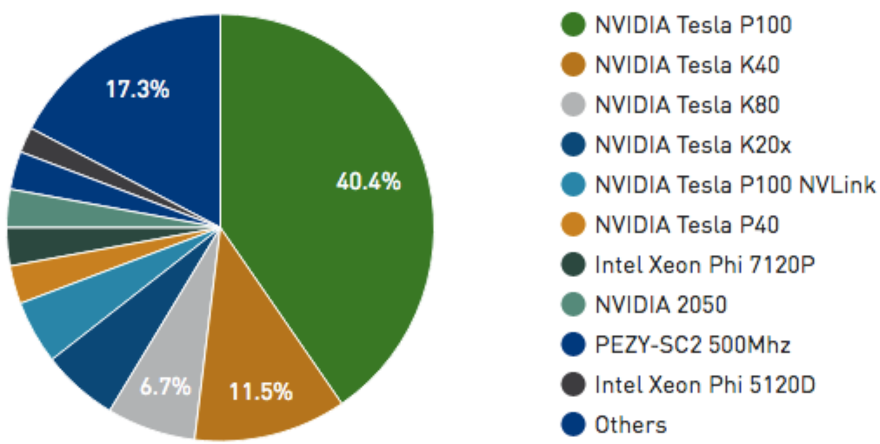
Vendors System Share

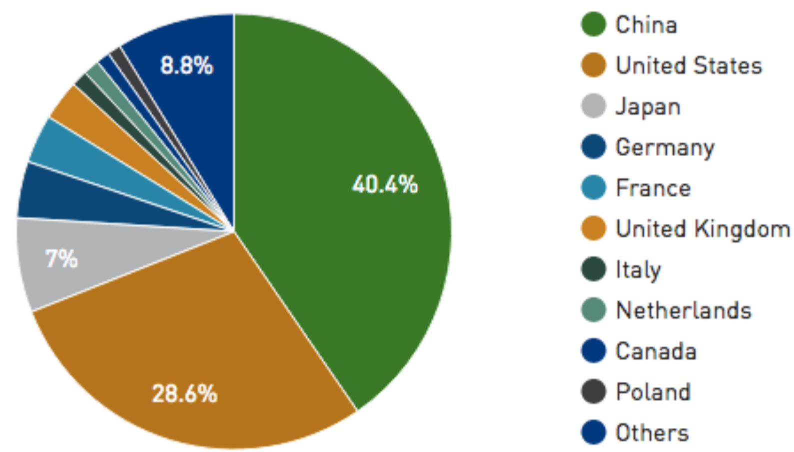


Interconnect System Share



Accelerator/Co-Processor System Share





Countries	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
China	202	40.4	298,876,659	524,584,484	22,797,764
United States	143	28.6	249,829,543	391,614,117	12,078,694
Japan	35	7	90,874,702	136,440,166	26,331,160
Germany	21	4.2	38,424,229	51,507,986	1,656,870
France	18	3.6	30,818,432	42,250,454	1,370,664
United Kingdom	15	3	32,268,888	41,186,451	1,296,368

Jaguar: 2009 World's Most Powerful Computer

www.olcf.ornl.gov

	jaguar XT4	jaguarpf XT5
Peak Performance	263.16 TFLOPS	2.33 PFLOPS
System Memory	61 TB	292 TB
Disk Space	750 TB	10,000 TB
Disk Bandwidth	44 GB/s	240 GB/s
Interconnect Bandwidth	157 TB/s	374 TB/s

Menu

TOP 500 – www.top500.org

TOP500 List - November 2009 (1-100)

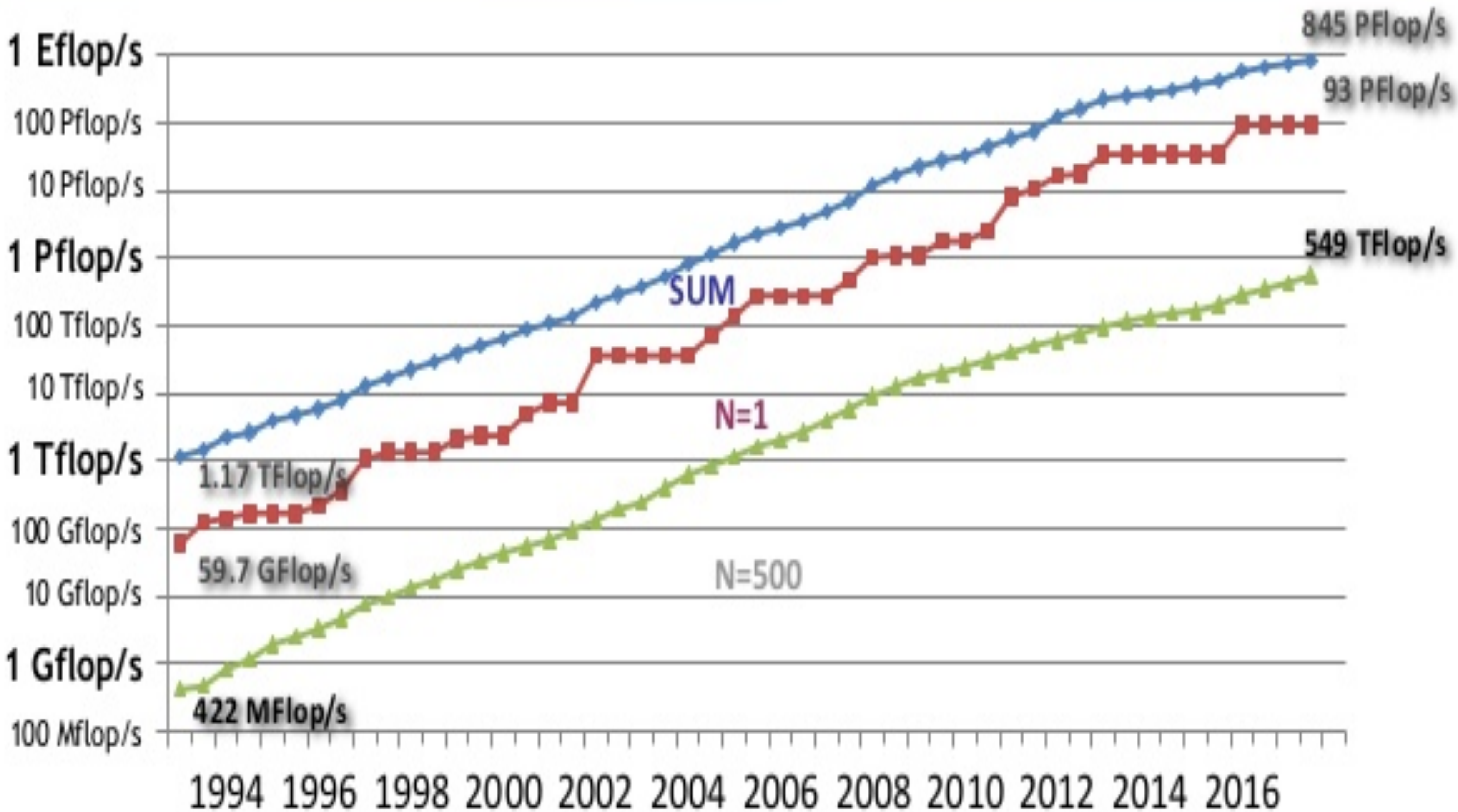
R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

nex

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
2	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM	122400	1042.00	1375.78	2345.50
3	National Institute for Computational Sciences/University of Tennessee United States	Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc.	98928	831.70	1028.85	
4	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
5	National SuperComputer Center in Tianjin/NUDT China	Tianhe-1 - NUDT TH-1 Cluster, Xeon E5540/E5450, ATI Radeon HD 4870 2, Infiniband / 2009 NUDT	71680	563.10	1206.19	

Performance Development



Hear more about this and the latest data at our BoF following at 5:15pm.

Numbers : Lots of Them: bit, byte, FLOP (S)

- Core : computing unit : processor
- Dual core machine (Intel or AMD CPU) : a CPU with 2 cores, each core is a 2.4 GHz computing unit with 2GB of RAM (memory in the processor not disk space)
- Binary bits (b) : “0” or “1” , 1 Byte (B) = 8 bits
- Binary number : 11111111= $(2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^{0)} = (2^8 -1) = 255$!!
- 32 bits machine or operating system => largest integer (all positive) = $(2^{32} -1)$ = (4,294,967,296 -1) or range of integer = $-(2^{31})$ to $(2^{31} -1)$
- 64 bits machine or operating system => range of integer = $-(2^{63})$ to $(2^{63} -1)$
- Kilo (K) = 10^3 (or 2^{10}) ; Mega (M) = 10^6 (or 2^{20}); Giga (G) = 10^9 (or 2^{30}); Tera (T billion) = 10^{12} (or 2^{40}) ; Peta (P) = 10^{15} (or 2^{50})
- FLoating Point Operation (+, -, / , *) : (10.1 + 0.1) * 1.0 / 2.0 = 5.1 => 3 FLOP
- FLOPS = FLOP per second :: 1 PetaFLOPS (kraken) = 10^{15} FLOP in one second
- **FLOPS in a core = (clock rate) x (floating point operation in one clock cycle)**
- **Peak Rate = (FLOPS in one compute unit, core) x (no. of core)**

HPL (High Performance Linpack): Solving Ax = b
http://www.netlib.org/benchmark/hpl/

$$\begin{aligned} 2x_1 + 2x_2 + 2x_3 &= 1 \\ 3x_1 + 4x_2 + 5x_3 &= 2 \\ 4x_1 + 6x_2 + 7x_3 &= 3. \end{aligned}$$

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 4 & 6 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1/2 \\ 3 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 3 & 4 & 5 \\ 4 & 6 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1/2 \\ 1 \end{bmatrix}.$$

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 0 & \star & \star \\ 0 & \star & \star \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ \star \\ \star \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1/2 \\ 0 \end{bmatrix}.$$

$$\boxed{x_3 = 0, \quad x_2 = 1/2 - x_3 = 1/2, \quad 2x_1 + 2_x 2 + 2x_3 = 1 \implies x_1 = 0.}$$

Total operation count for Gaussian elimination with backward substitution

$$\boxed{\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n.}$$

http://wiki.math.msu.edu/index.php/Gaussian_Elimination

Jaguar (ORNL) : World Fastest Computer, 1.759 PF (2009)

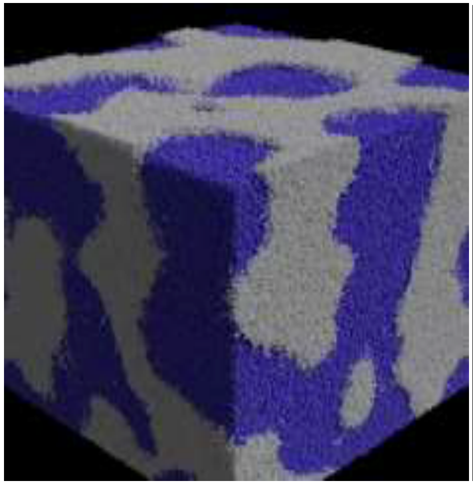
- FLOPS – FLoating Point Operation Per Second
- GFLOPS = 10^9 FLOPS ; TFLOPS = 10^12 ; PFLOPS = 10^15
- FLOPS = (clock rate) x (floating point operation in one clock cycle)
- Peak Rate = (FLOPS in one CPU) x (no. of CPU)
- Cray XT5 one core AMD Opteron :
  - Rpeak : (2.6 GHz) x (4) x (224162 cores) = 2331284 GFLOPS
  - Rmax : 1759000 GFLOPS → 75.4% of peak

jaguar: What does it do?

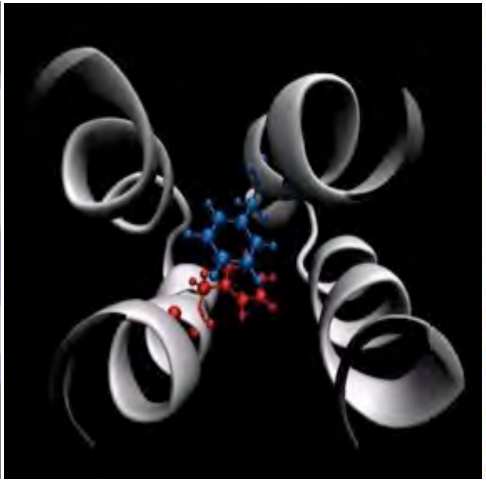
- Solve a very big system of equations : Ax = b using a standard benchmark C program (HPL)
- Nmax : Size of A for HPL (Solve Ax=b) = 5474272
- Total Memory needed = (Nmax) x (Nmax) x (8 Bytes) = 239741 GB
- Memory needed per core = 1.07 GB
- Elapse Time : 2(Nmax)(Nmax)(Nmax)/3/Rmax ~ = 13 hrs

Computer Benchmark (HPL) - Big Science, Big Memory Storage

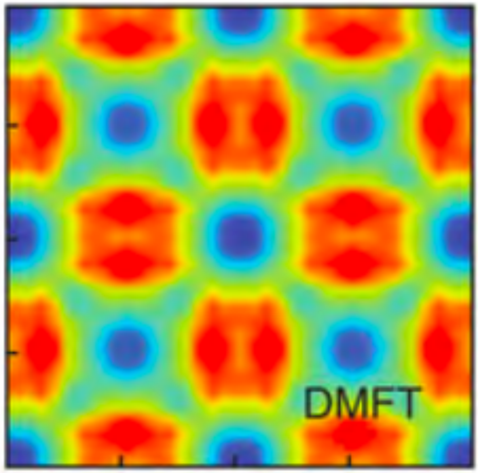
- HPL - Solve a system of equations : Ax = b, a standard benchmark C program to rank the top500 computers
- Size of matrix A = Memory used on a computer
- A = (Nmax) x (Nmax) x (8 Bytes) = 239741 GB (on jaguar)
- Jaguar : Nmax=5474272, Memory = 240 TB, ~ 1.07 GB/core
- Elapse Time : 2(Nmax)(Nmax)(Nmax)/3/Rmax ~ 13 hrs (jaguar)
- Titan ~ 10 times faster : Nmax ~ 8000000 : 1.7 GB / core; titan ~20 hrs, 65% of peak performance



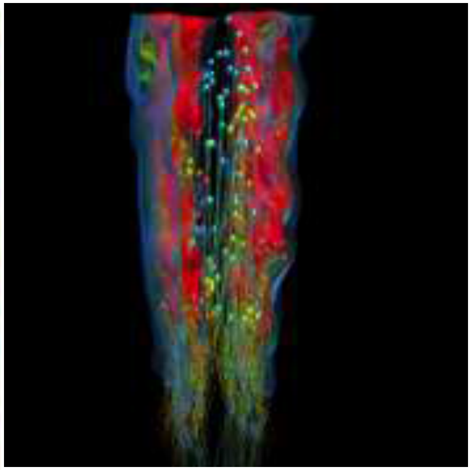
Organic Polymer (MD, LAMMPS)



Molecular Biology

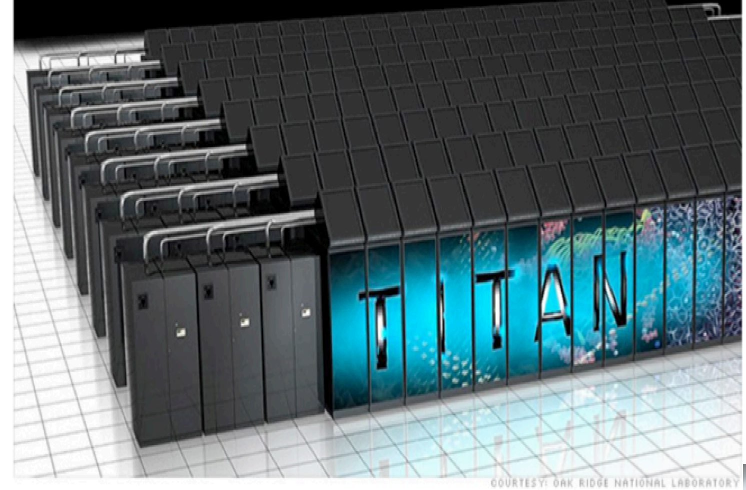


Superconductivity



Turbulent Combustion (DNS, S3D)

	ORNL Titan	NUDT Tianhe-2	Sunway TaihuLight
Theoretical Peak	27 Pflop/s = (2.6 CPU + 24.5 GPU) Pflop/s	54.9 Pflop/s = (6.75 CPU + 48.14 Coprocessor) Pflop/s	125.4 Pflop/s = CPEs +MPEs Cores per Node = 256 CPEs + 4 MPEs Supernode = 256 Nodes System = 160 Supernodes Cores = 260 * 256 * 160 = 10.6M
HPL Benchmark Flop/s	17.6 Pflop/s	30.65 Pflop/s	93 Pflop/s
HPL % Peak	65.19%	55.83%	74.16%
HPCG Benchmark	0.322 Pflop/s	0.580 Pflop/s	.371 Pflop/s
HPCG % Peak	1.2%	1.1%	0.30%
Compute Nodes	18,688	16,000	40,960
Node	AMD Optron Interlagos (16 cores, 2.2 GHz) plus Nvidia Tesla K20x (14 cores, .732 GHz)	2 – Intel Ivy Bridge (12 cores, 2.2 GHz) plus 3 - Intel Xeon Phi (57 cores, 1.1 GHz)	256 CPEs + 4 MPEs
Sockets	18,688 Interlagos + 18,688 Nvidia boards	32,000 Ivy Bridge + 48,000 Xeon Phi boards	40,960 nodes with 256 CPEs and 4 MPEs per node
Node peak performance	1.4508 Tflop/s = (.1408 CPU + 1.31 GPU) Tflop/s	3.431 Tflop/s = (2*.2112 CPU + 3*1.003 Coprocessor) Tflop/s	3.06 Tflop/s CPE: 8 flops/core/cycle (1.45 GHz*8*256 = 2.969 Tflop/s) MPE (2 pipelines) 2*4*8 flops/core/cycle (1.45 GHz*1= 0.0928Tflop/s)
Node Memory	32 GB CPU + 6 GB GPU	64 GB CPU + 3*8 GB Coprocessor	32 GB per node
System Memory	.710 PB = (.598 PB CPU and .112 PB GPU)	1.4 PB = (1.024 PB CPU and .384 PB Coprocessor)	1.31 PB (32 GB*40,960 nodes)
Configuration	4 nodes per blade, 24 blades	2 nodes per blade, 16 blades per	Node peak performance is 3.06 Tflop/s, or 11.7 Gflop/s per core.

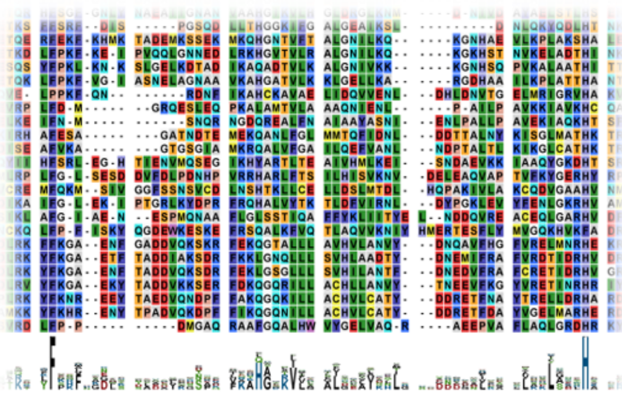


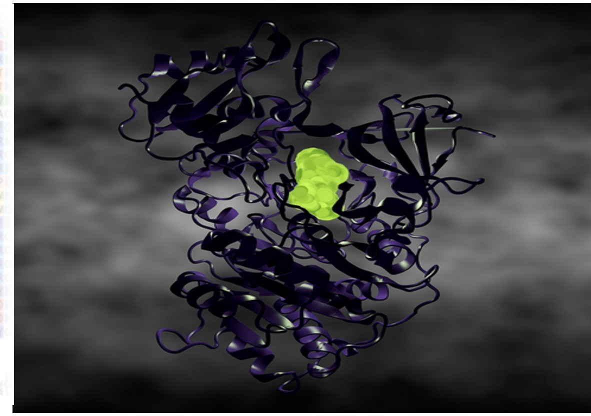
HPCG : Conjugate Gradient solver : MV

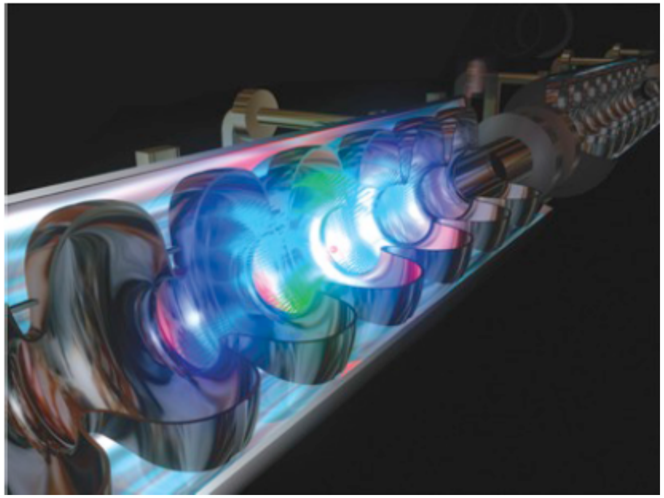
1-10

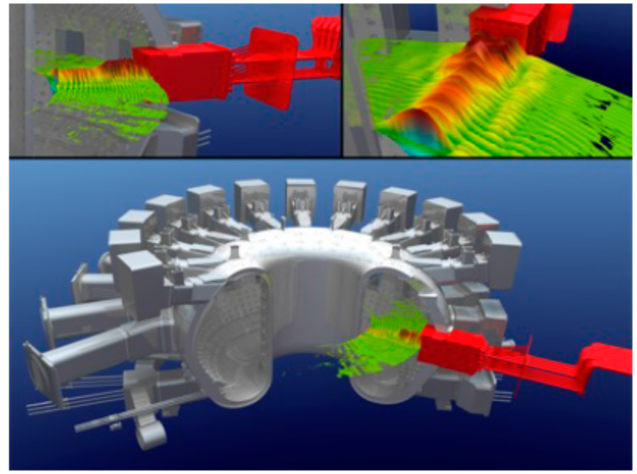
Rank (HPL)	Site	Computer	Cores	Rmax	HPCG	HPCG/HPL	% of Peak
1 (2)	NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.863	0.5800	1.7%	1.1%
2 (5)	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	705,024	10.510	0.5544	5.3%	4.9%
3 (1)	National Supercomputing Center in Wuxi	Sunway TaihuLight -- SW26010, Sunway	10,649,600	93.015	0.3712	0.4%	0.3%
4 (4)	DOE/NNSA/LLNL	Sequoia - IBM BlueGene/Q	1,572,864	17.173	0.3304	1.9%	1.6%
5 (3)	DOE/SC/Oak Ridge Nat Lab	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	560,640	17.590	0.3223	1.8%	1.2%
6 (7)	DOE/NNSA/LANL/SNL	Trinity - Cray XC40, Intel E5-2698v3, Aries custom	301,056	8.101	0.1826	2.3%	1.6%
7 (6)	DOE/SC/Argonne National Laboratory	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	786,432	8.587	0.1670	1.9%	1.7%
8 (11)	TOTAL	Pangea -- Intel Xeon E5-2670, Infiniband FDR	218592	5.283	0.1627	3.1%	2.4%
9 (15)	NASA / Mountain View	Pleiades - SGI ICE X, Intel E5-2680, E5-2680V2, E5-2680V3, Infiniband FDR	185,344	4.089	0.1555	3.8%	3.1%
10 (9)	HLRS/University of Stuttgart	Hazel Hen - Cray XC40, Intel E5-2680v3, Cray Aries	185,088	5.640	0.1380	2.4%	1.9%

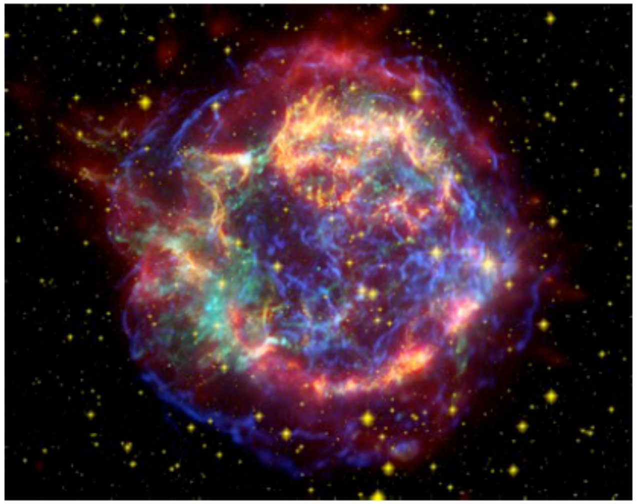
Big Computer
Big Science
Model base
Compute intensive

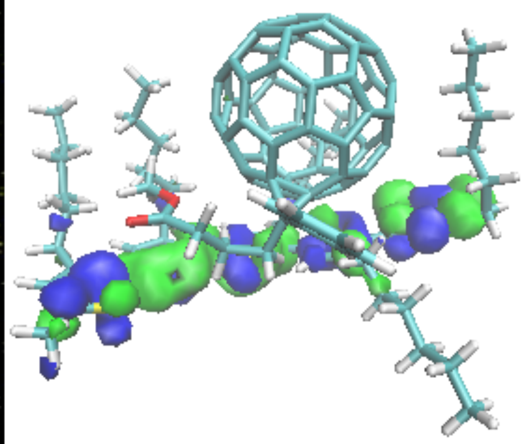


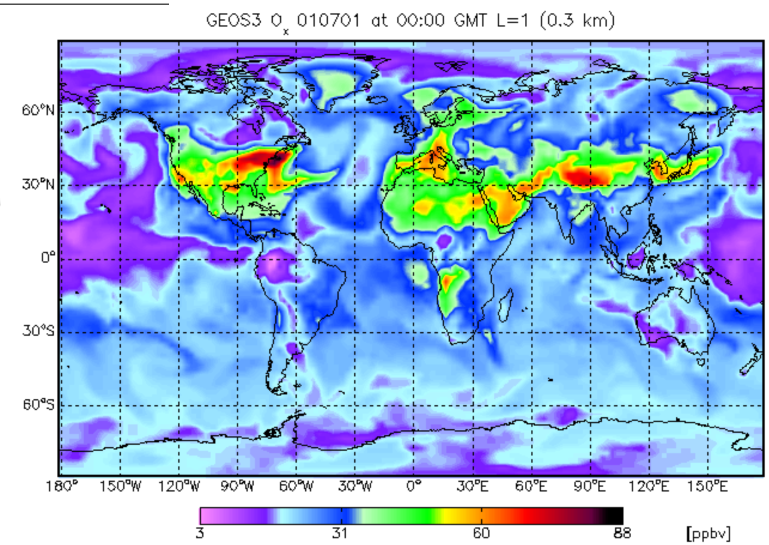




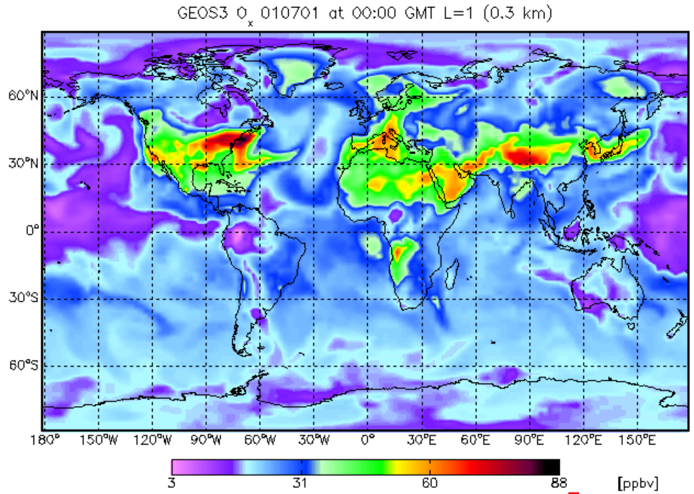


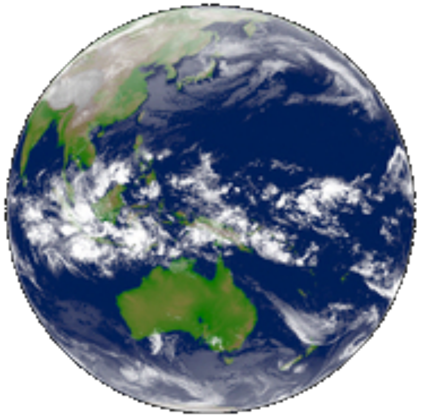


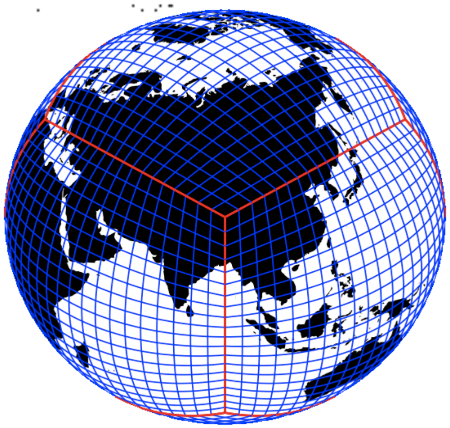




Climate Simulations and Weather (Storms) forecast



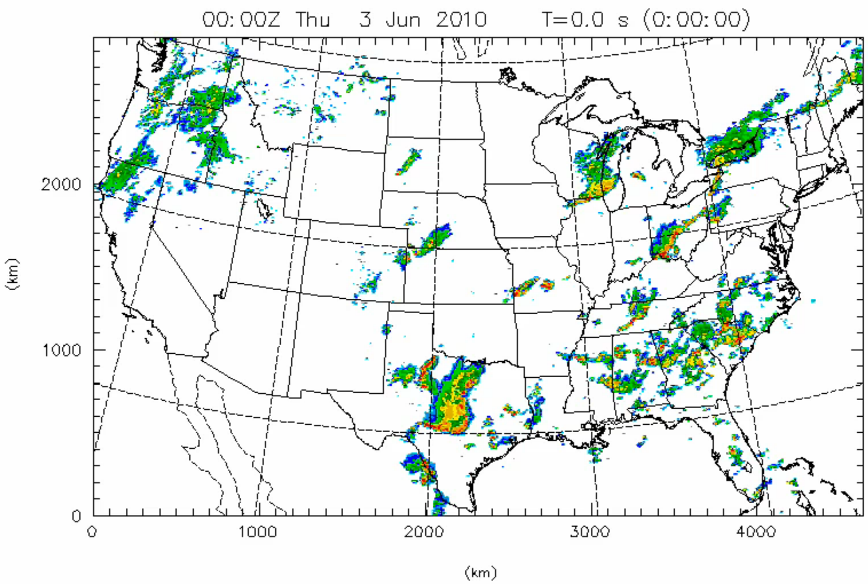


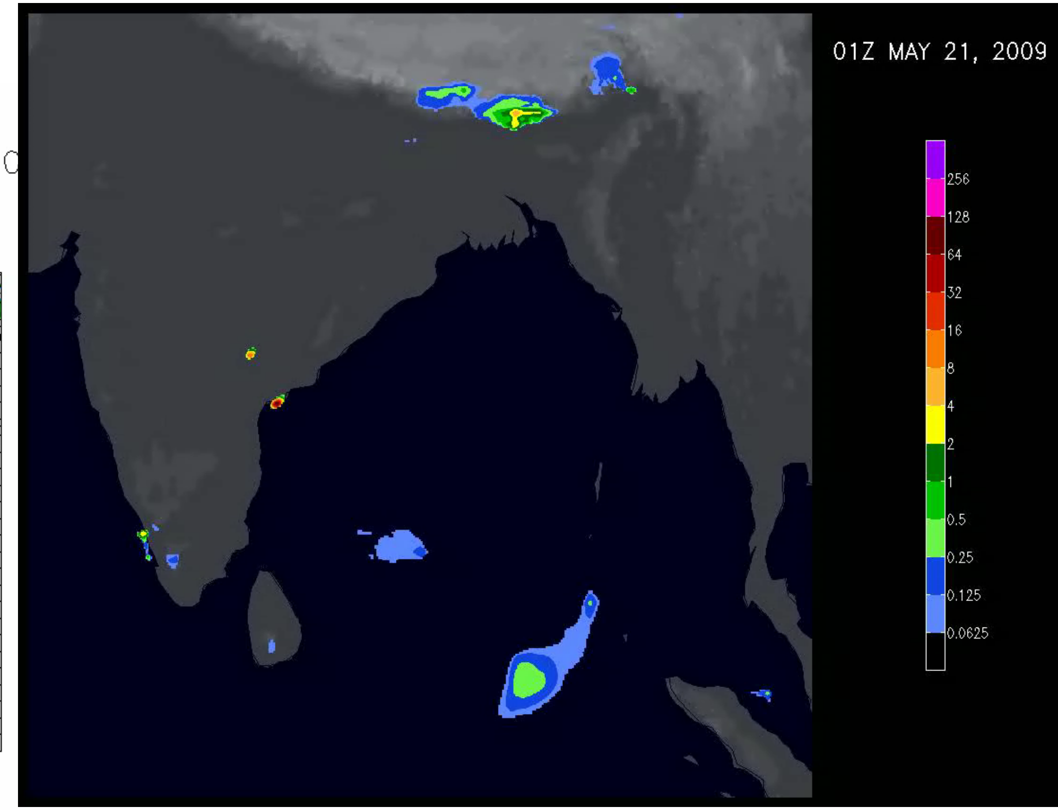


www.caps.ou.edu

SPC1 (4640x2880x50, dx=1 km)

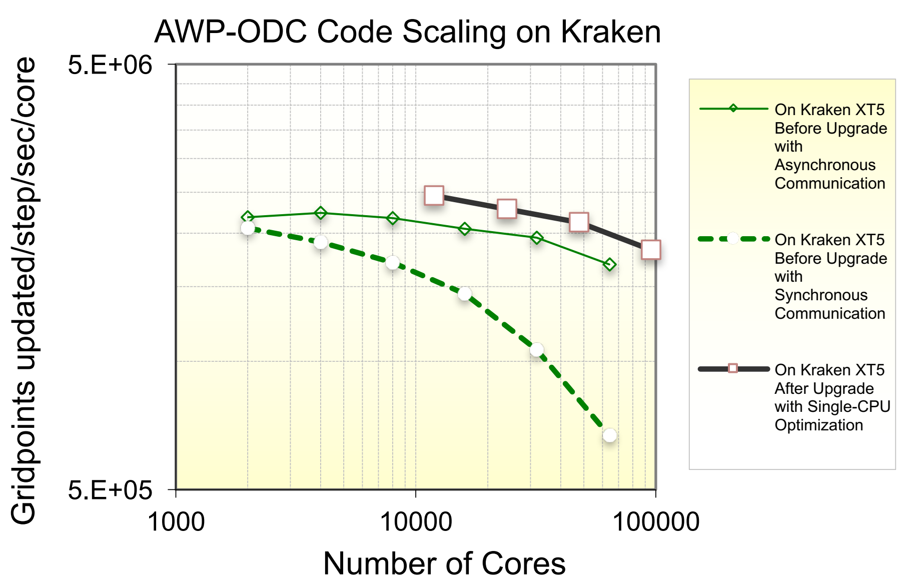
WRF Forecast starting at 00Z Thu 03 Jun 2010

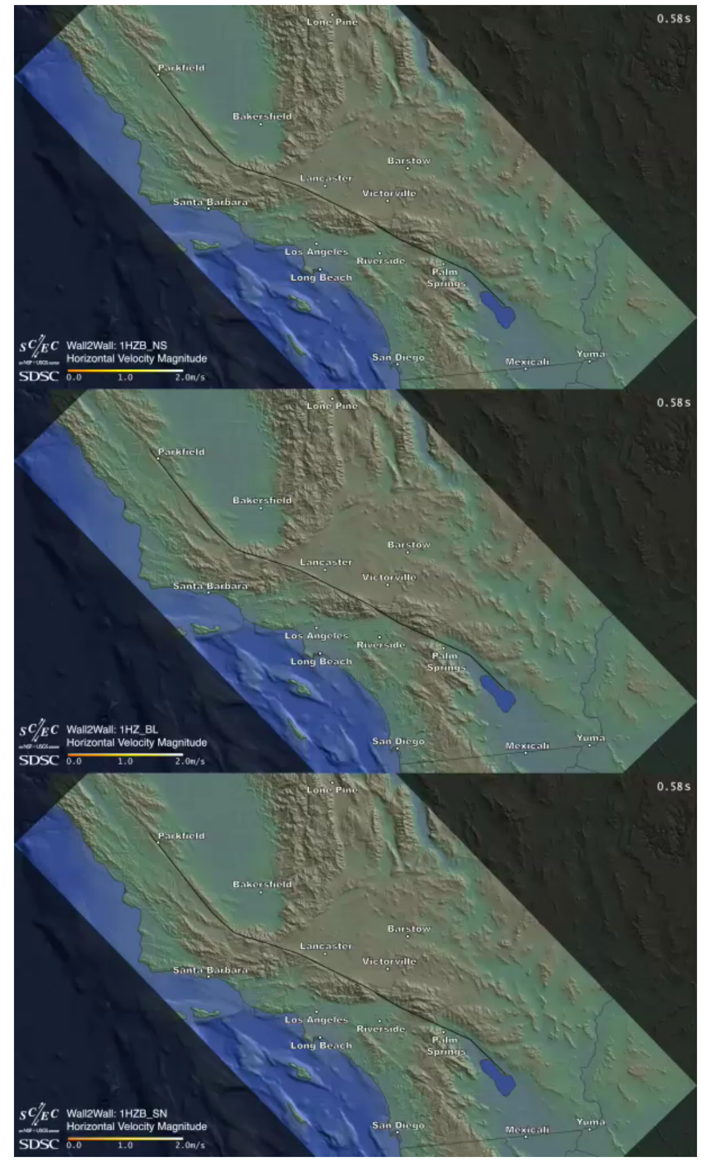




Simulating the Big One on Kraken
Southern California Earthquake Center

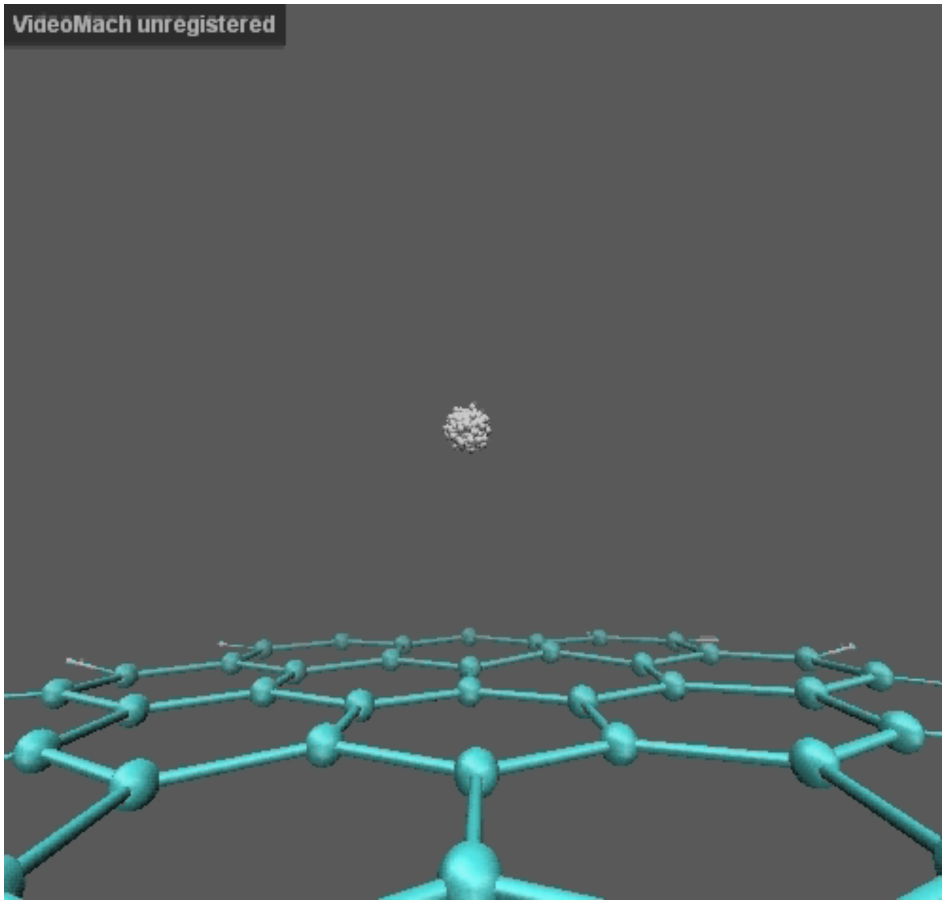
- Biggest Earthquake Simulation on San Andreas Fault, the Big One
- Simulated in a 32 billion grid point subset of the SCEC Community Velocity Model (CVM) V4 with a minimum shear-wave velocity of 500 m/s up to a maximum frequency of 1 Hz.
- 96,000 processor cores used for production runs on Kraken, 2.6 hrs WCT, 53 sustained TeraFlop/s

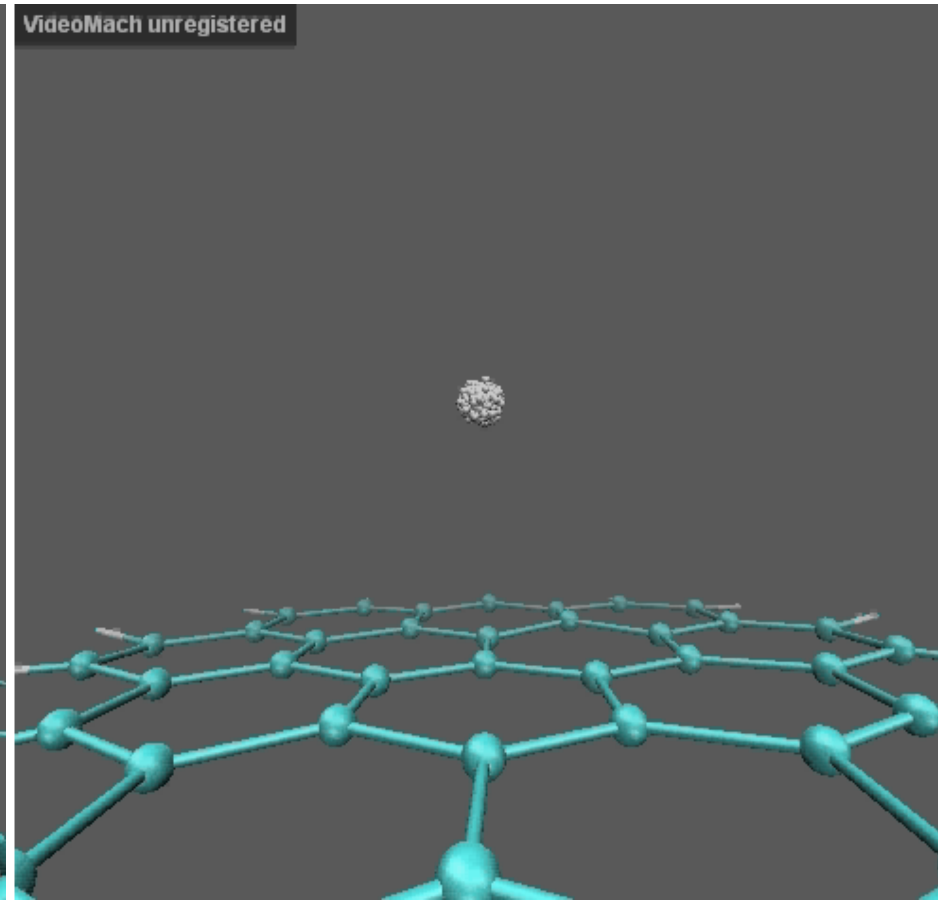




Materials Science Modeling
Bohmian Dynamics: graphene hydrogenation using DFTB

Separation of quantum and classical degrees of freedom

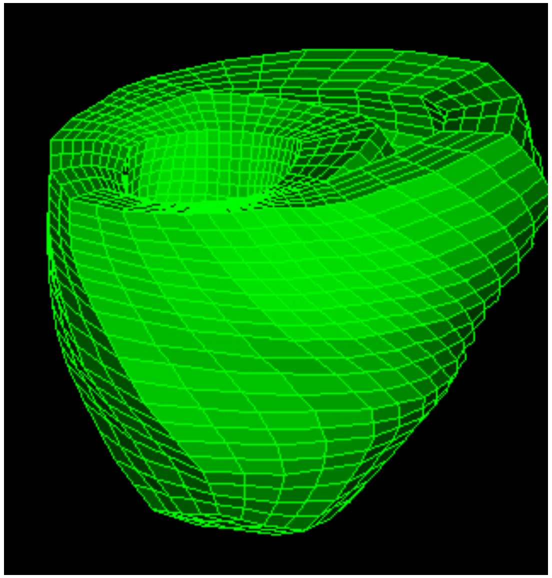


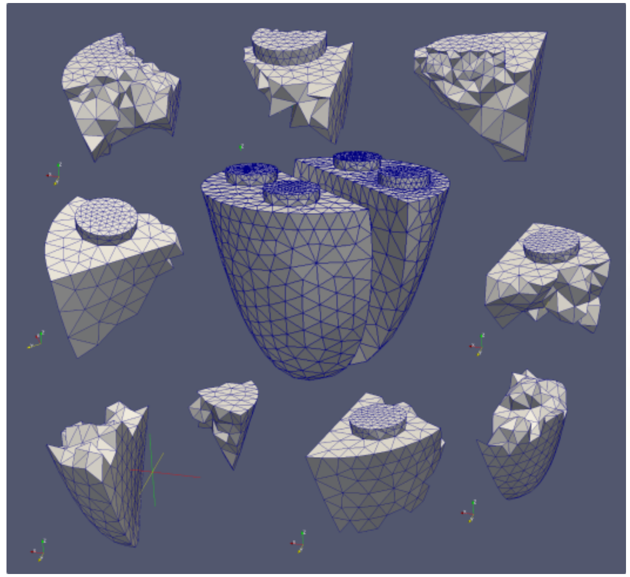


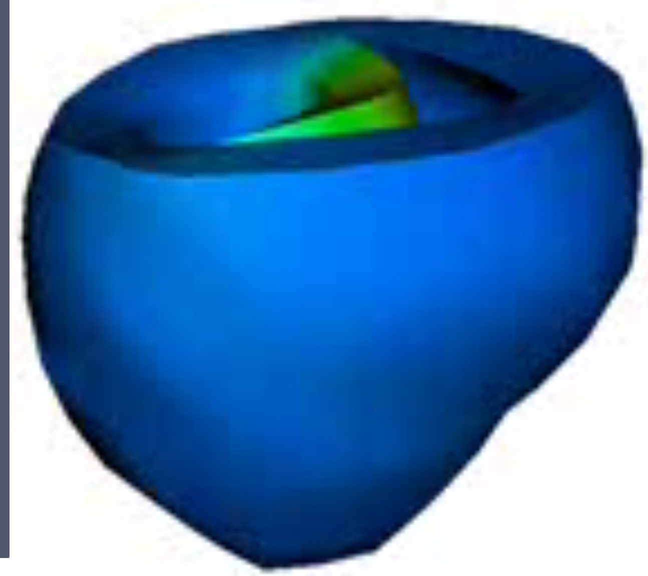
Quantum (U is on!)

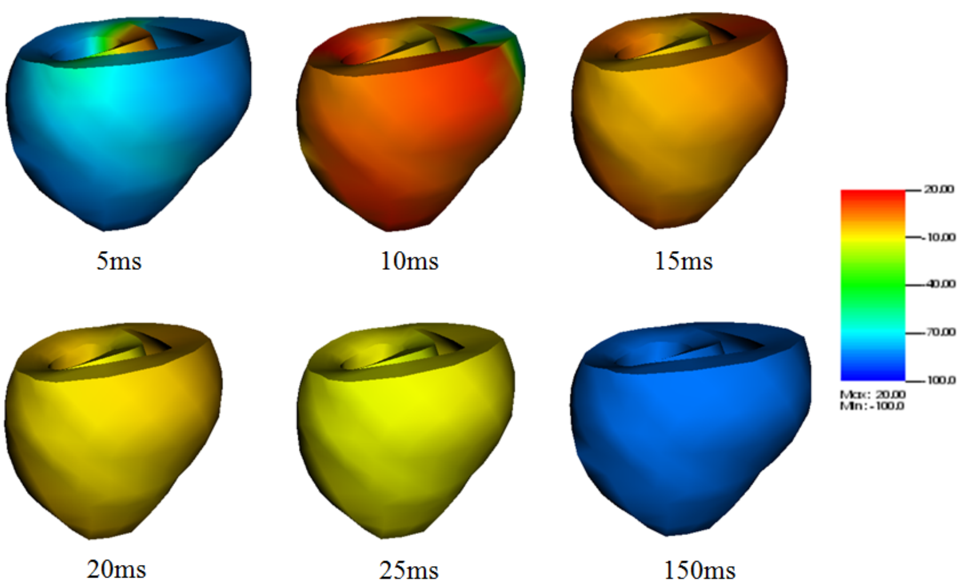
Classical (U is off)

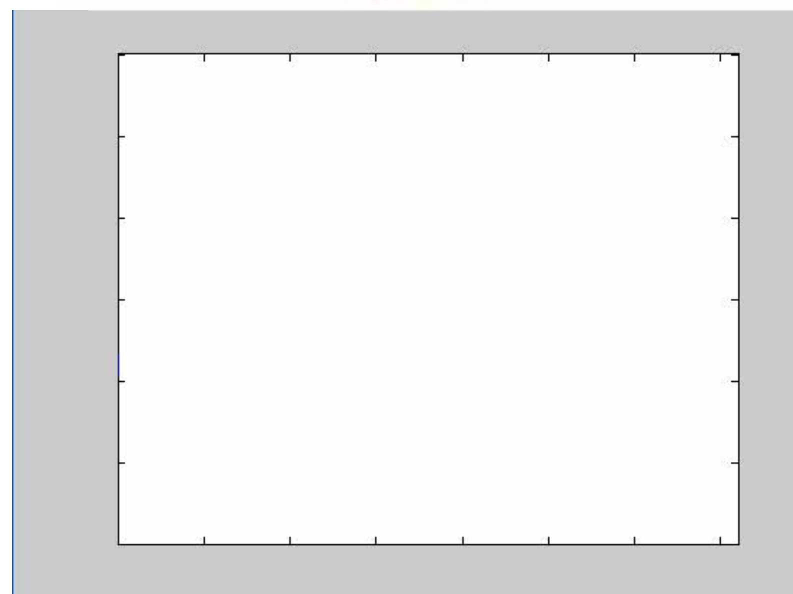
Modeling of Heart and Lung



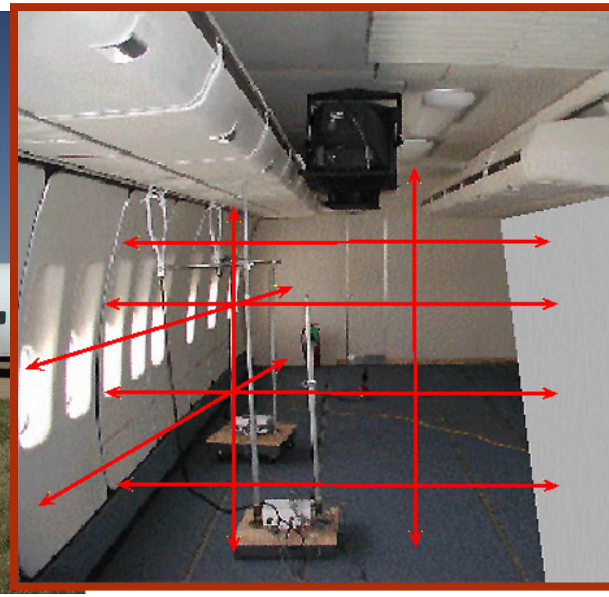


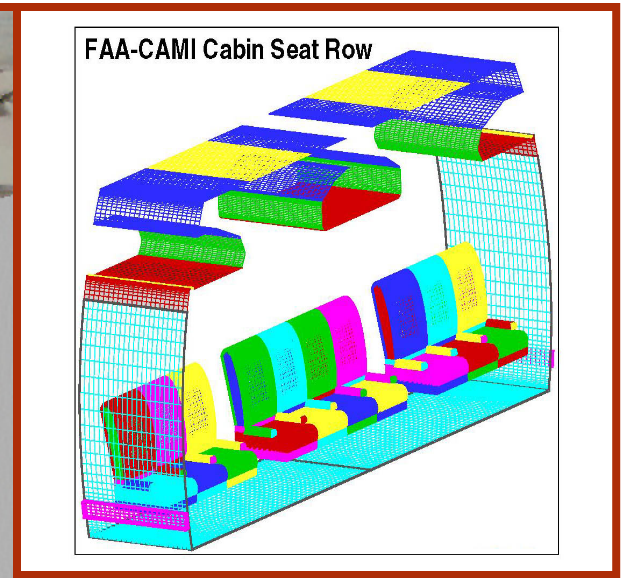


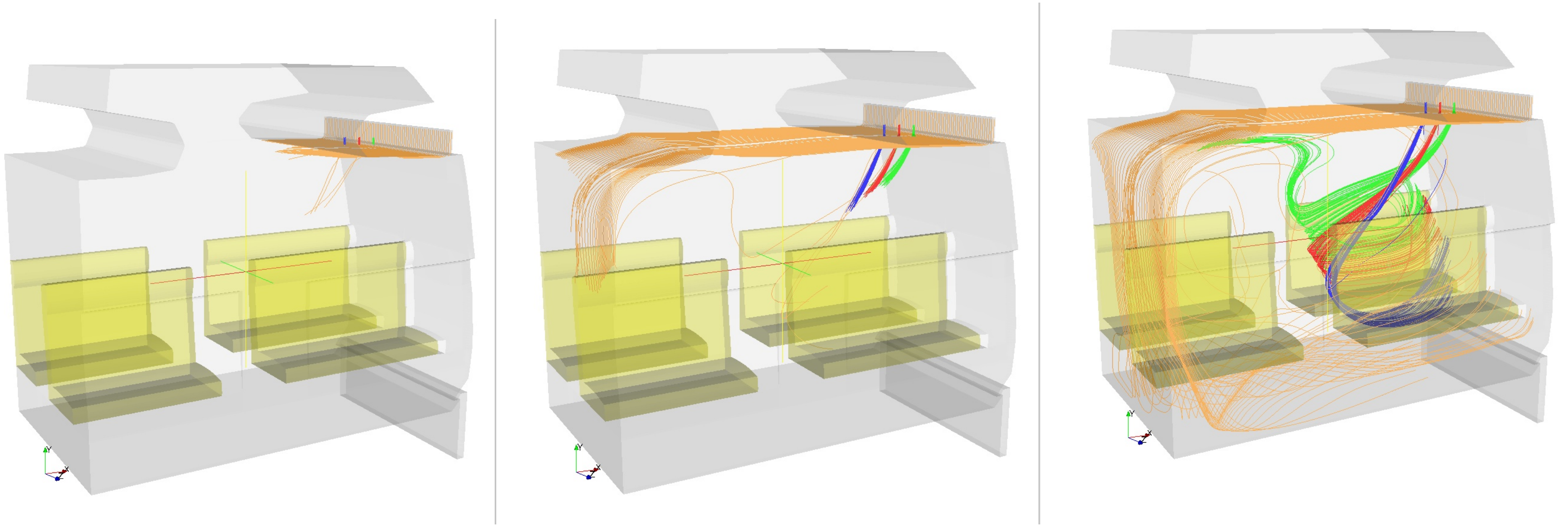




Air Flow Simulation B747 - Validation







- DreamWorks has a "render farm" of servers made up of about 20,000 processors (HP BladeSystem c-Class server blades).
- The image rendering jobs are broken up into small pieces, distributed out to the server farm, and are later recompiled to create the final images for a film.
- Required a whopping 80 million compute hours to render, 15 million more hours than DreamWorks' last record holder, "The Rise of the Guardians."
- Between 300 and 400 animators worked on "The Croods" over the past three years.
- After completing a film, about 70TB worth of data (things like background art or plants) is stored for future usage in future productions.

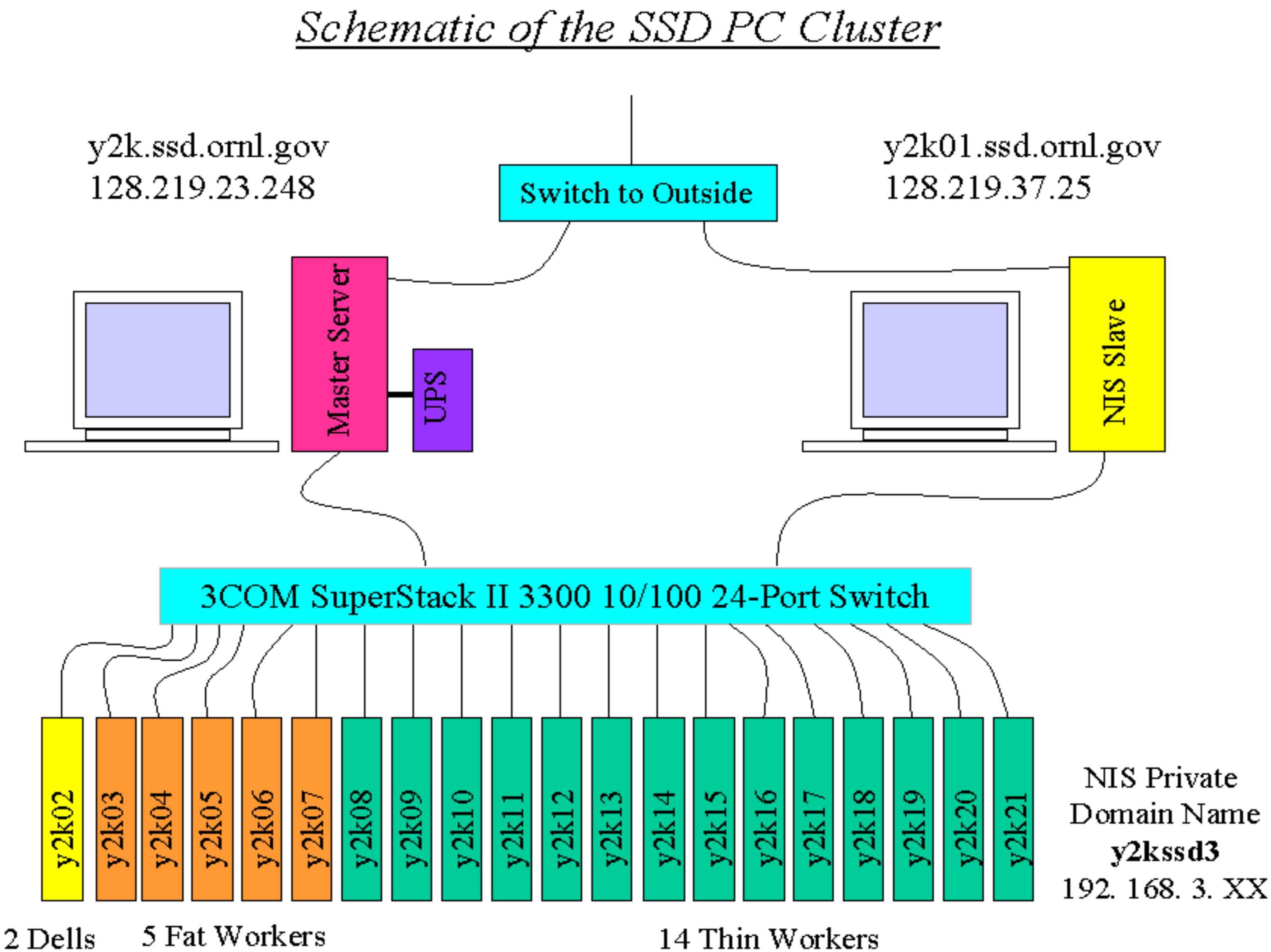
Road Map to Exascale Computing

- 1962 (CDC 1604), 1976 (Cray 1), 1982 (XMP), 1988 (YMP), 1994 (T90)
- 1992 – DOE HPCC - High Performance Computing and Communication 3T Initiative – 1 teraflops, 1 terabytes of memory, 1 terabytes/s bandwidth
- 1993 – launch of top500 list, CM5, Intel Paragon, ~100GFLOPS
- 1995 – ASCI – DOE Accelerated Strategic Computing Initiative, intended to do nuclear stockpile simulation
- 1996 – first Terascale computer, ASCII RED SNL
- 1998 – Boewulf, PC cluster – Commodity Components
- DOE – supercomputers, projects –, SciDAC, Human Genome project, HER, Climate, INCITE – terascale to petascale
- NSF Track I, II Teragrid, XSEDE -1st petascale
- DOE Leadership Computing Program – CORAL program, Exascale
- National Strategic Computing Initiative NSCI

Do It Yourself : A Typical PC Cluster (1999)

- One server node with dual CPU & SCSI Drive
- 5 Fat worker node with 1 GB RAM
- 16 Worker nodes with 512 MB RAM
- one 24 Port 100Mb Switch, total cost ~$40000

Simple Hardware Schematic



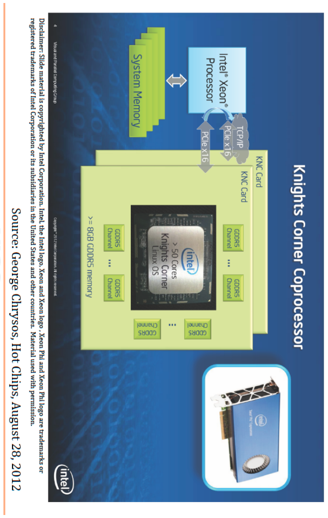
Simple Parallel Computer

Many commodity units connected by a COS interconnect

Modern Supercomputers

Commodity plus Accelerator Today

Accelerator (MIC)



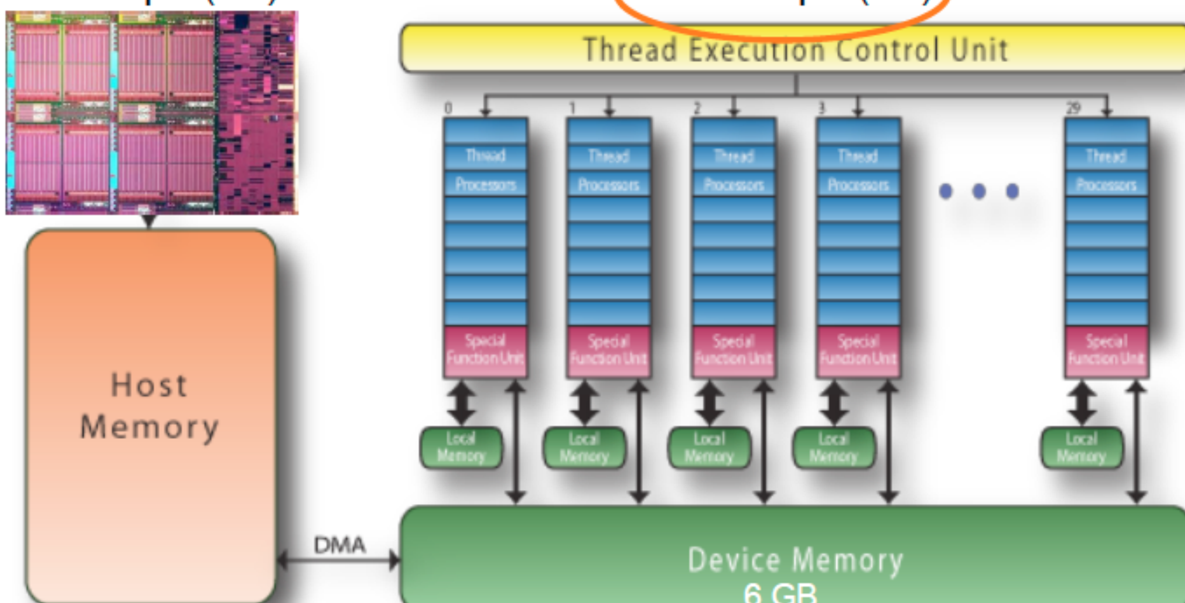
Commodity

Intel Xeon
8 cores
3 GHz
8*4 ops/cycle
96 Gflop/s (DP)

Accelerator (GPU)

Nvidia K20X "Kepler"
2688 "Cuda cores"
.732 GHz
2688*2/3 ops/cycle
1.31 Tflop/s (DP)

192 Cuda cores/SMX
2688 "Cuda cores"

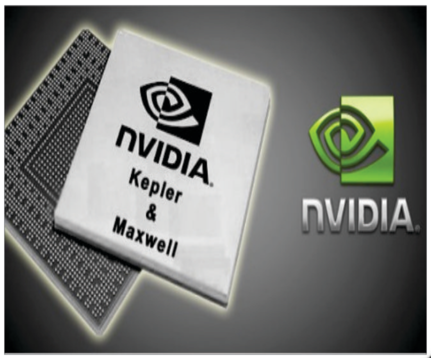


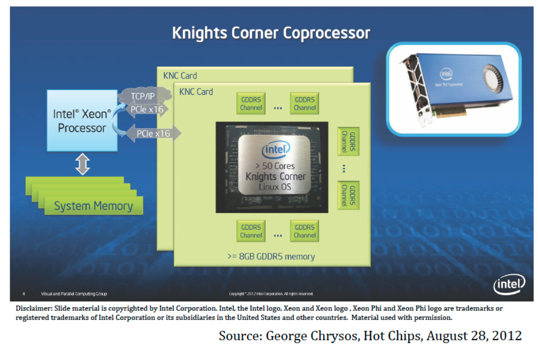
Interconnect
PCI-X 16 lane
64 Gb/s (8 GB/s)
1 GW/s

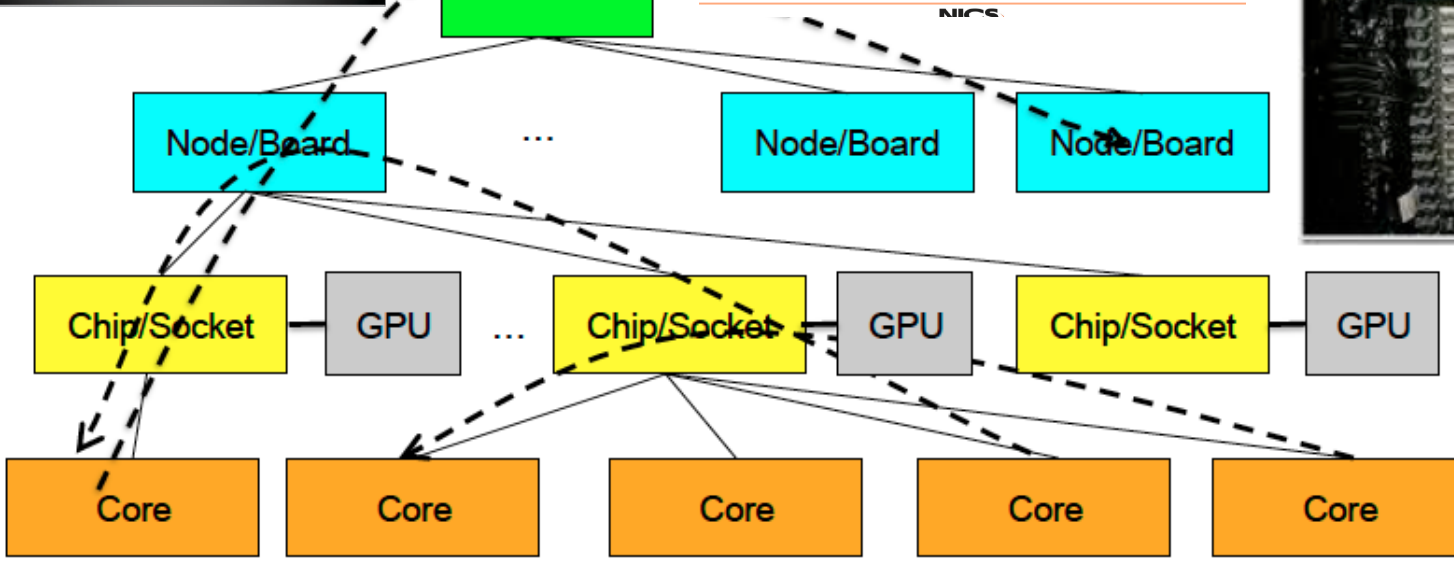
From ICL Dr. Jack Dongarra : icl.cs.utk.edu

Example of typical parallel machine

Shared memory programming between processes on a board and
a combination of shared memory and distributed memory programming
between nodes and cabinets



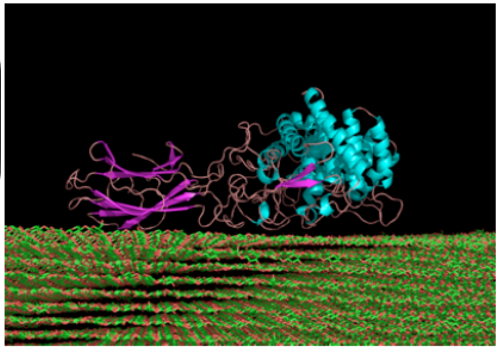


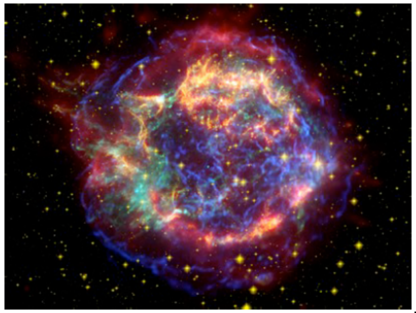


From ICL Dr. Jack Dongarra : icl.cs.utk.edu

Scale to the Future

Tesla





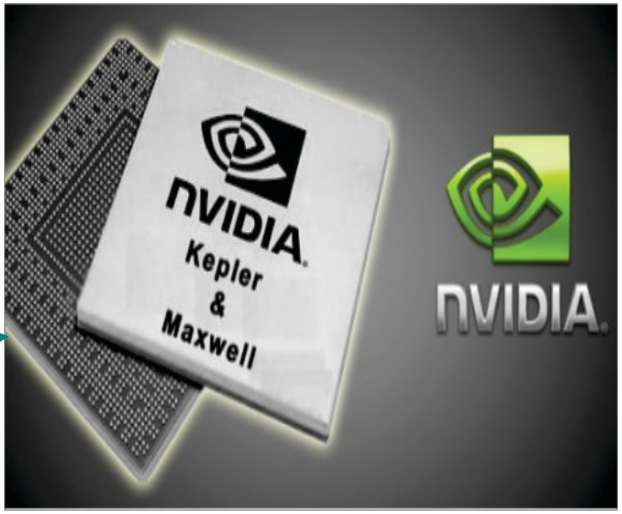
Kepler will implement
Virtual Memory Space
→Will allow larger problems
On GPU/CPU “shared” space

Over 100% increase in Flop/s for K2M2 Tests

Ride the technology curve

Fermi

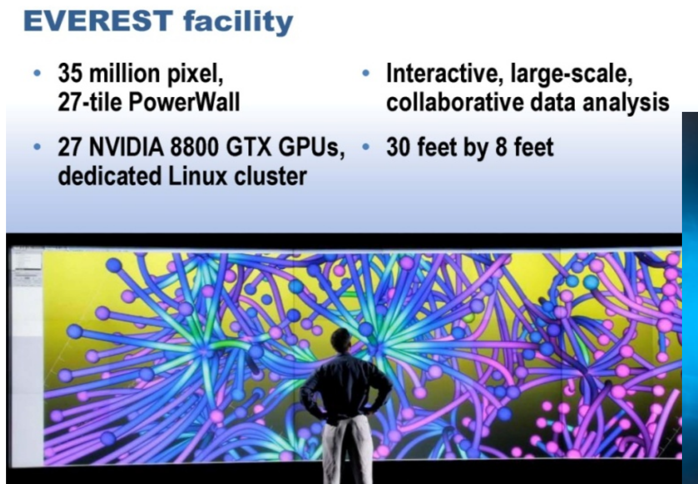
Predicted 80%+ Increase in Flop/s



Kepler (to P100, to V100)
Intel MIC (Landing)

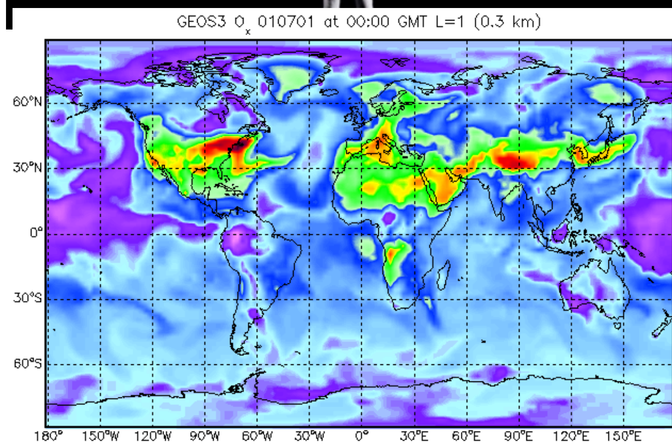
Ride on the Hardware Technology Curve

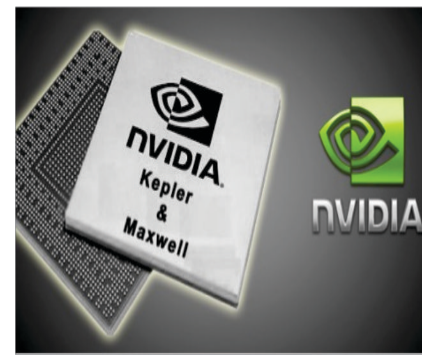
TACC – Stampede
10 PFLOPS



ORNL – TITAN
20 PFLOPS

64 cores, ~TFLOPS





Kepler (2012), ~TFLOPS

Transformational Science : RT Simulation

Volta..

Summit: Next Generation Supercomputer at ORNL (Exascale)

Challenges : Power limitation, Scaling application performance

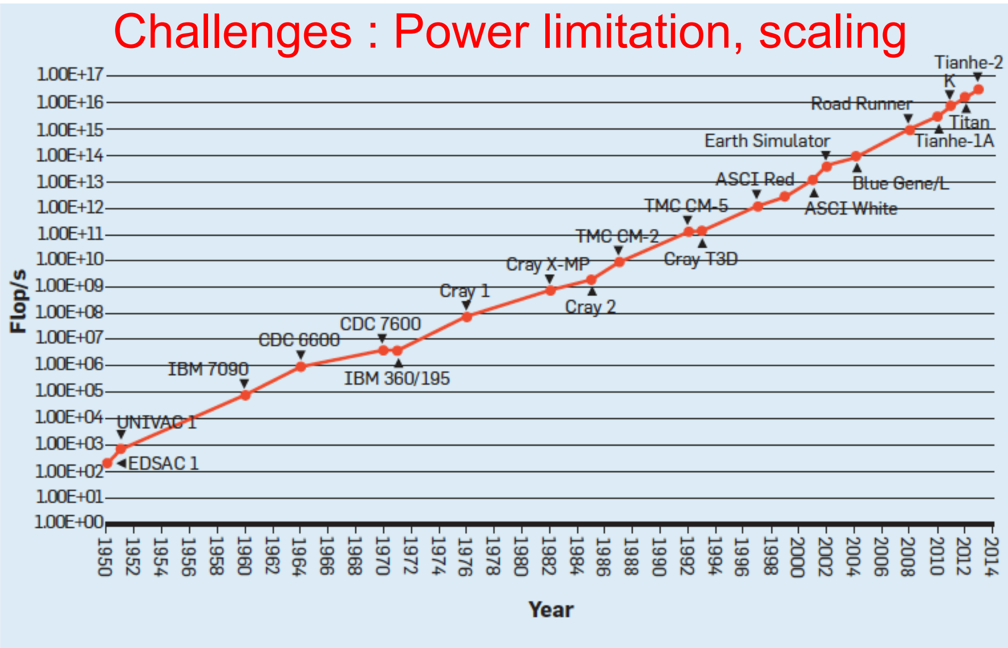
TITAN VS SUMMIT

Compute System Comparison

ATTRIBUTE	TITAN	SUMMIT
Compute Nodes	18,688	~3,400
Processor	(1) 16-core AMD Opteron per node	(Multiple) IBM POWER 9s per node
Accelerator	(1) NVIDIA Kepler K20x per node	(Multiple) NVIDIA Volta GPUs per node
Memory per node	32GB (DDR3)	>512GB (HBM+DDR4)
CPU-GPU Interconnect	PCI Gen2	NVLINK (5-12x PCIe3)
System Interconnect	Gemini	Dual Rail EDR-IB (23 GB/s)
Peak Power Consumption	9 MW	10 MW

Source - "Oak Ridge Leadership Computing Facility by Jack Wells : SciDAC PI Meeting 23 July 2015

Capability vs Capacity Computing



Capability Computing, Single extreme scale, problem, shortest Time

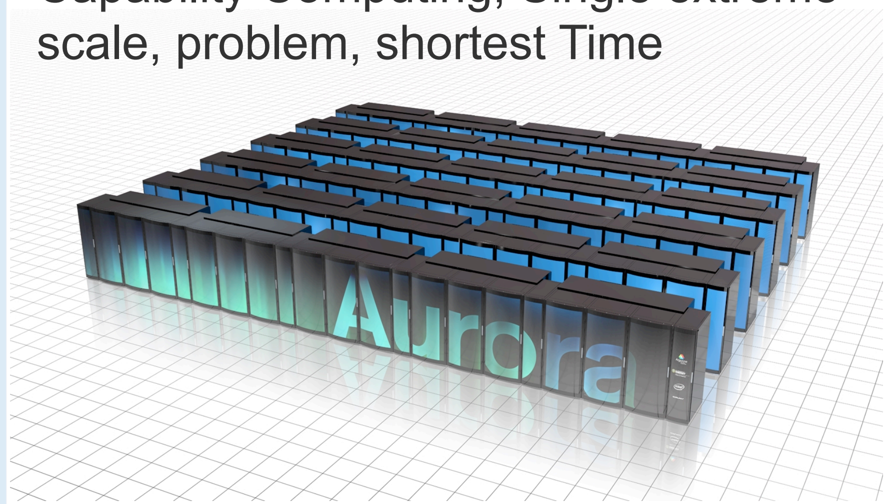
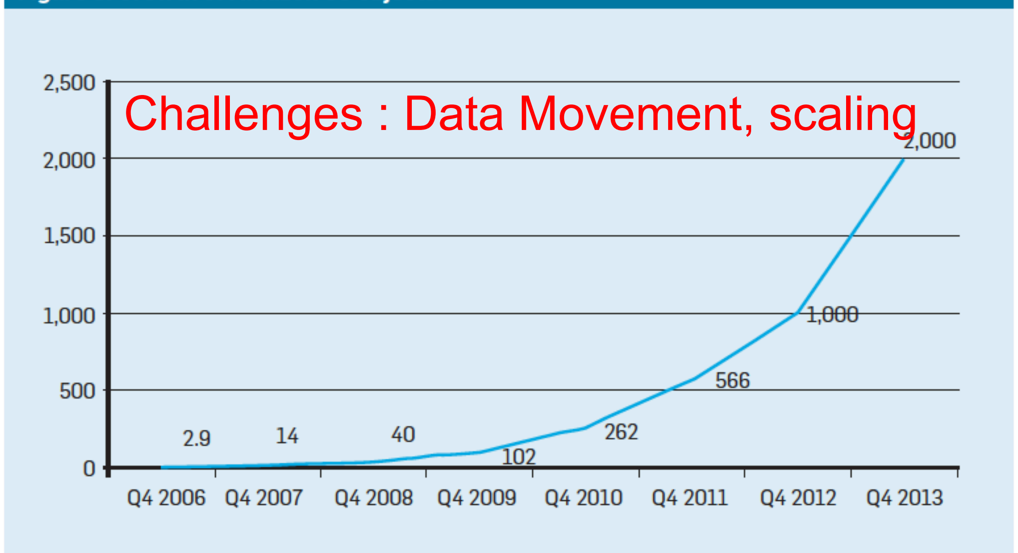
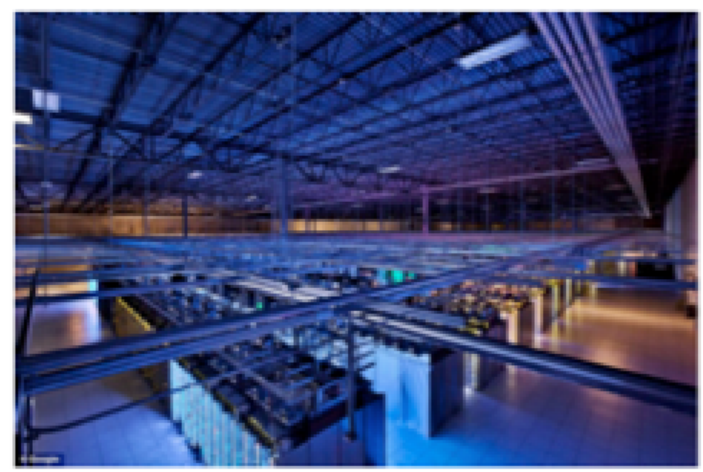


Figure 3. Growth of Amazon S3 objects.



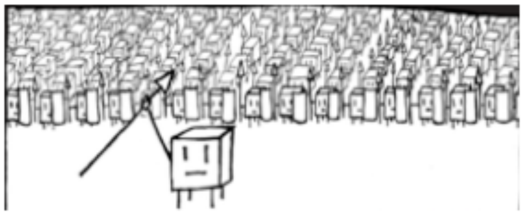
Capacity Computing, medium scale problems, data engine, analysis

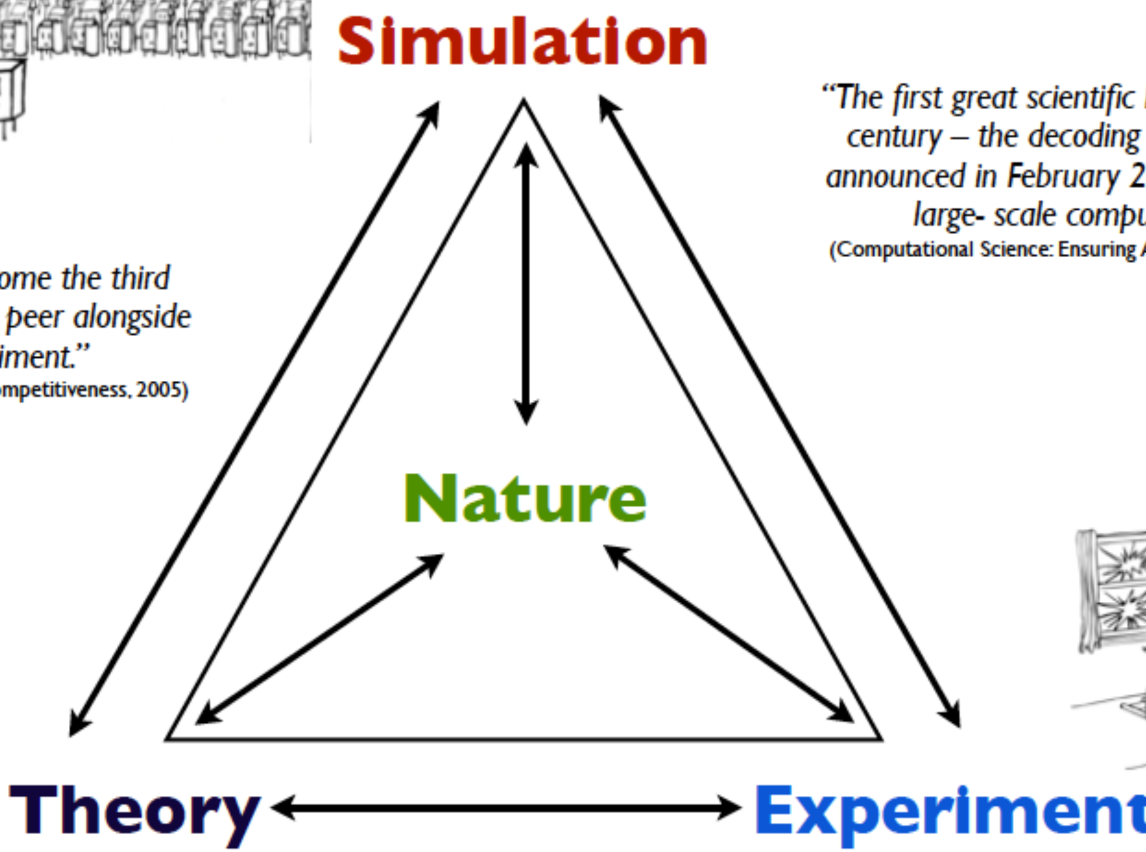


Google's Datacenter

Ienu

Big Science, Big Data, Big Iron





"The first great scientific breakthrough of the new century – the decoding of the human genome announced in February 2001 – was a triumph of large- scale computational science."
(Computational Science: Ensuring America's Competitiveness, 2005)

"Computational science has become the third pillar of the scientific enterprise, a peer alongside theory and physical experiment."
(Computational Science: Ensuring America's Competitiveness, 2005)

"A Guide to Monte Carlo Simulations in Statistical Physics", David Landau, Kurt Binder

Dealing with the Knowns and Unknowns
Uncertainty Quantification – Data Analytics

“As we know there are known knowns.

There are things we know we know.

We also know there are known unknowns.

That is to say, we know there are some things we do not know.

But there are also unknown unknowns.

The ones we don’t know we don’t know,” D. Rumsfeld

Given enough data, can we find the unknowns and predict the knowns?

Four Tiers – Computational Ecosystem

Advanced Computing Architectures

- Emergent Architectures
- Tactical Computing
- Next Generation Computing Systems
- High Performance Networking and Memory

Computing Sciences

- Programming Environments
- Programming Languages
- Software Integration

MAIN STREET

BIG

High Performance **Compute Ecosystem** Computing

DATA

Predictive Simulation Sciences

- Computational Math & Algorithms
- Scientific Computing
- Verification, Validation & Uncertainty Quantification
- Applied Computer Modeling and Analysis

Data Intensive Sciences

- Sciences of Large Data
- Computational Math for Data Analytics
- Real-time Data Access & Analytics

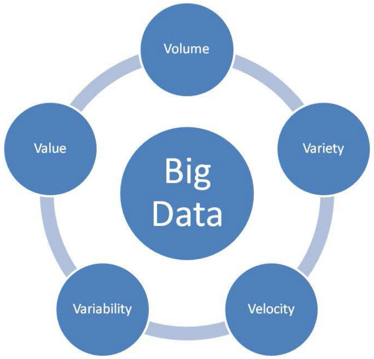
Menu

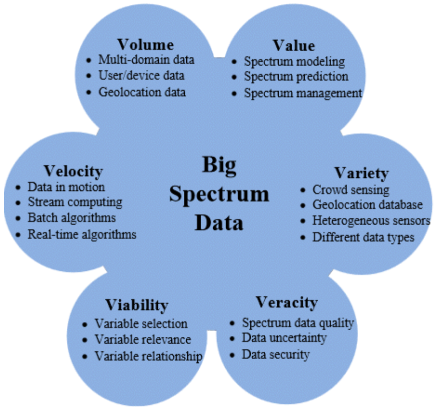
Big Data Predictive Model

- ✓ A collection of large data sets that are asymmetric or too large to be processed by traditional tools. Often the data sets are noisy and heterogeneous but in general could be co-related to some significant events.

Big Data Characterized by

- *Volume*
  - How much data
- *Velocity*
  - The speed at which data arrives and the speed with which decisions based on it must be made
- *Variety*
  - Heterogeneity of storage platforms, data types, representation, semantic interpretation, and security classification or other distribution limitations
- *Veracity*
  - How trustworthy is the data, what is its uncertainty, and what is the error associated with it
- *Value*
  - What is the data worth





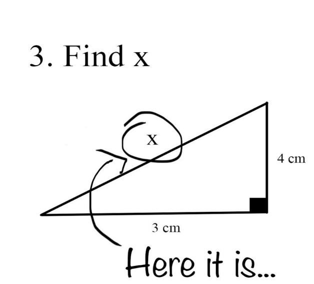
- ✓ Challenges include storage, classification, mining, sharing, visualization..
- ✓ Need capacity, infrastructure, domain knowledge + compute , CS, Math..

Programming Models & Tools Ecosystem:

Big Data is inter-disciplinary,
Need community effort to coordinate
creation of tools

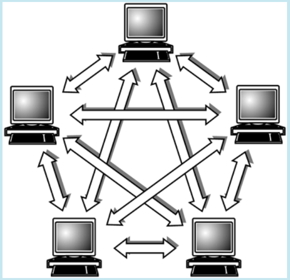
- Flat file, Excel, CVS
- Database, SQL,
- Distributed DD, HDFS
- Large graph, matrix, SVD
- Storage, I/O, network
- Sensors, big instruments

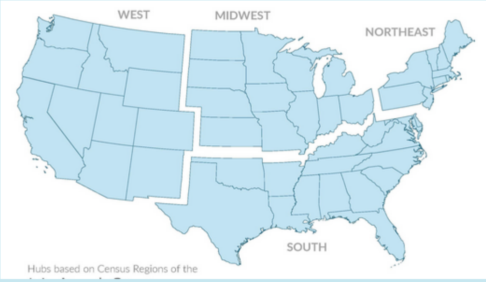
- Data Mining, searching, compression, neural network, deep learning, smart detection, predictive models, visaulization
- Images (picture, neutron, thermal, x-ray…), spatial temporal data, noise, signal, voice, smell, ….
- Healthcare, social, politics, science, finance, agriculture, entertainment, geographic, transportation ….
- Perhaps layman sense?!



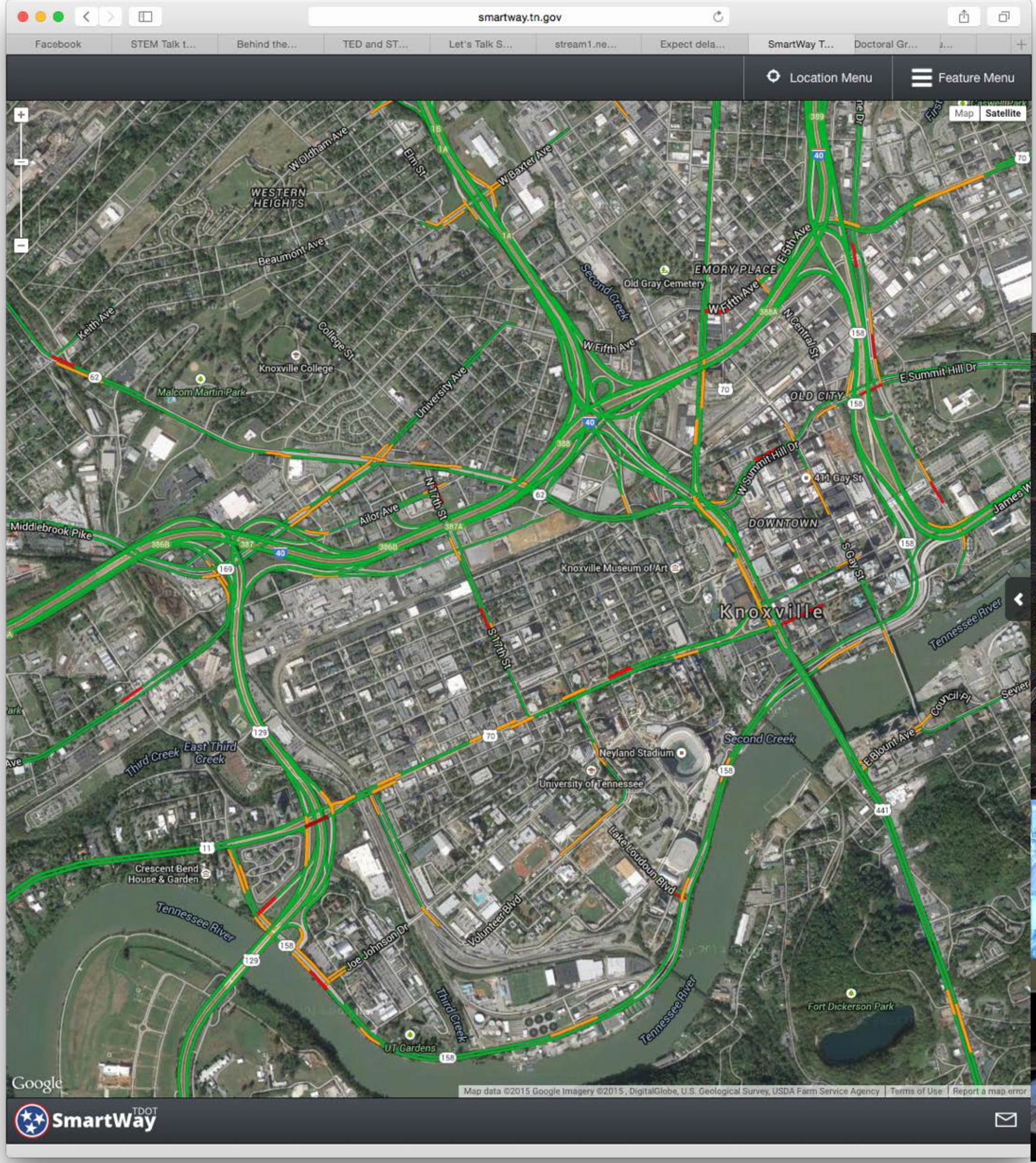
Milestones –Capacity (Big data)

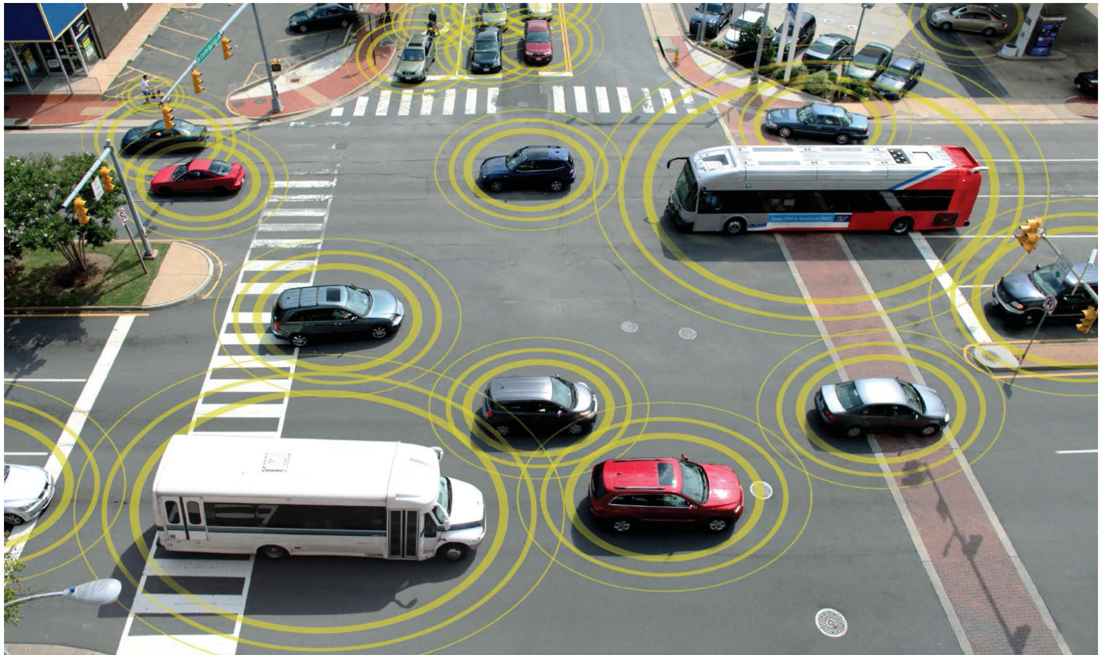
- 1973 – Internet was "officially" named
- 1990s Internet widely used
- 1993 Mosaic (NCSA), web browser.. netscape, IE, Mozzilla, Firefox..
- 1995- Google, Amazon
- 1996 – IBM Deep Blue Chess machine, first Terascale, ASCII RED
- 1999 – Grid Computing
- 2000 – Baidu
- 2004 – Facebook, MapReduce
- 2005 – Hadoop
- 2006 deep learning, Geoffrey Hinton, Neural Computing
- Clouds, machine learning framework, GPU
- 2015 – NSCI
- 2015 – NSF - Big Data Hub

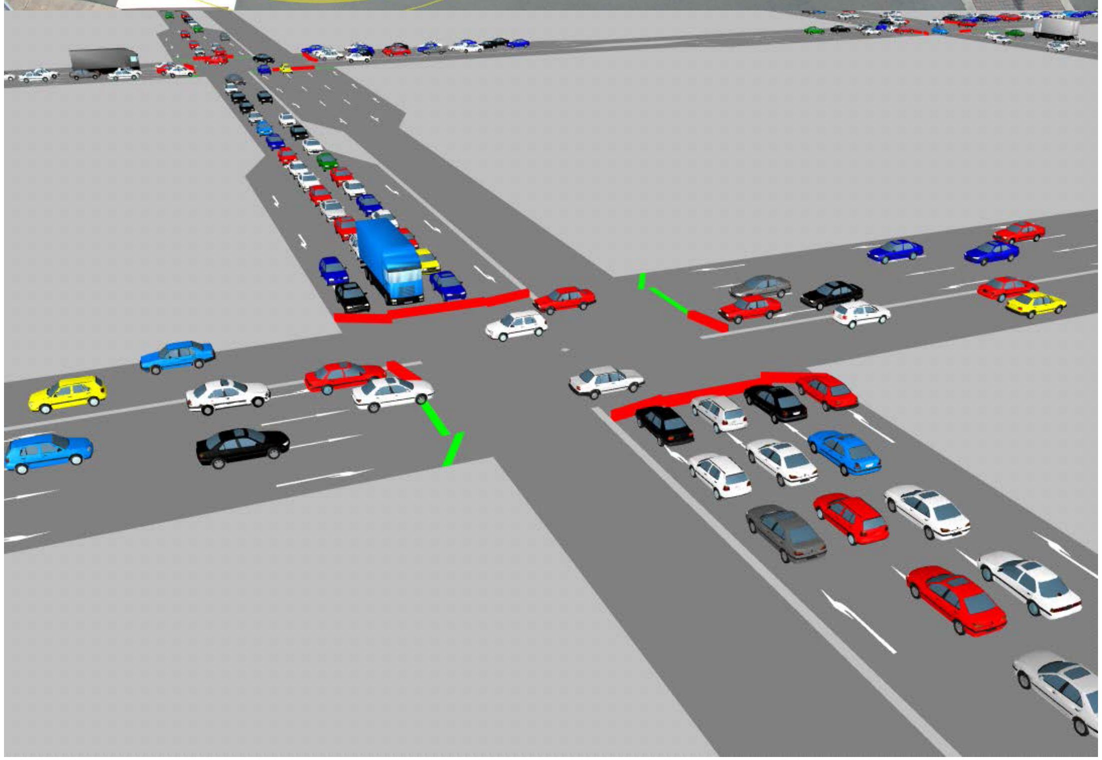




Big Data – Transportation
Ph.D Students Needed- (Dr. Han, UTK)

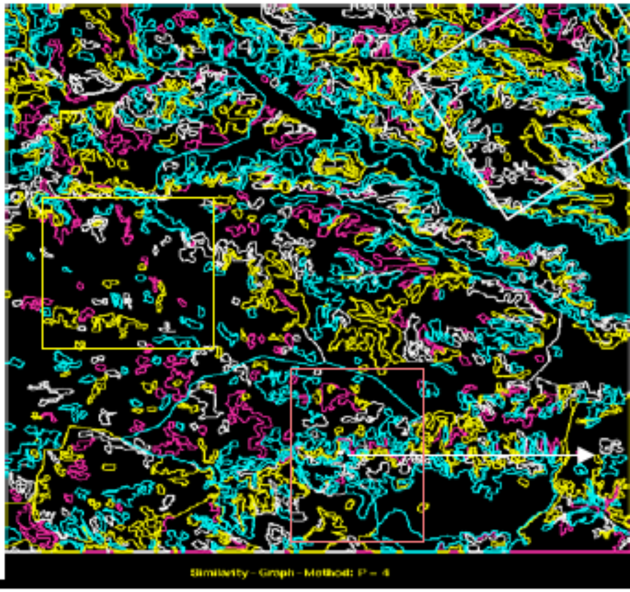






Big Data – Modeling
Auto Pilot, GPS

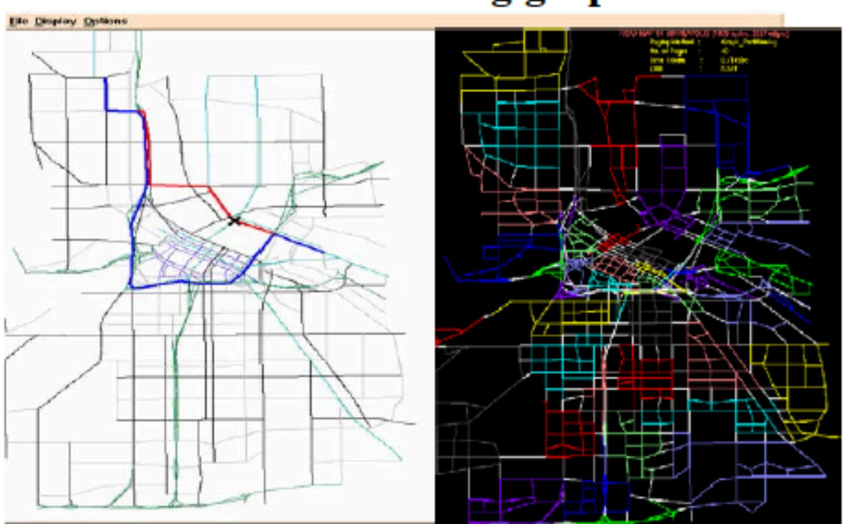
Spatial Database



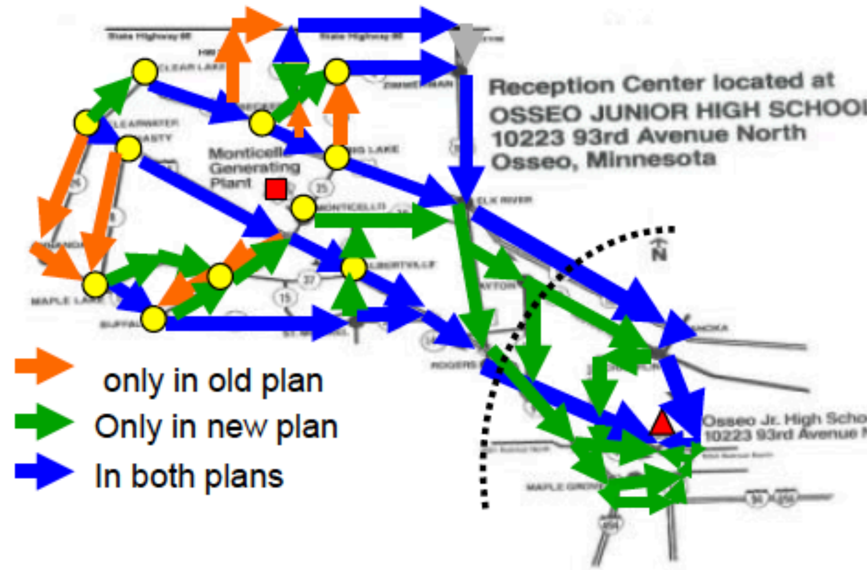
Parallelize Range Queries

Shortest Paths

Storing graphs in disk blocks



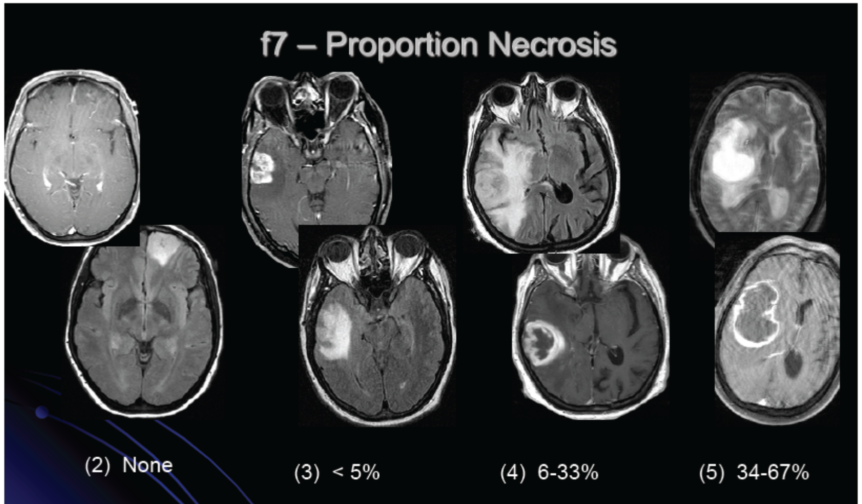
Evacuation Route Planning



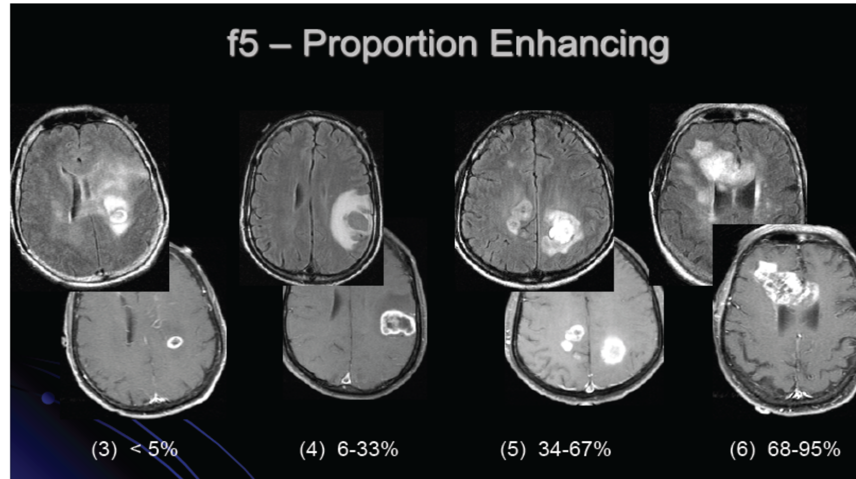
Source - "From GPS and Virtual Globes to Spatial Computing," Shashi Shekhar. IEEE Big Data Conference 2015

Menu

Big Data Applications: Healthcare



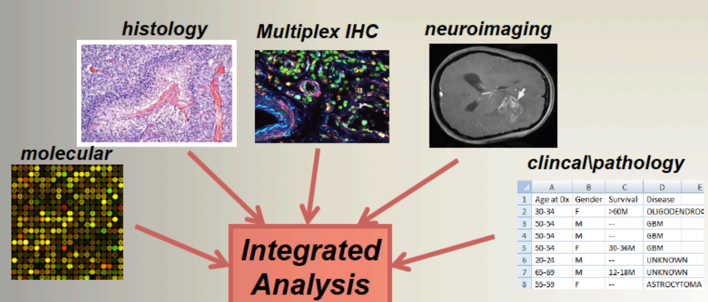
Visually, when scanning through the entire tumor volume, what proportion of the tumor is estimated to represent necrosis. Necrosis is defined as a region within the tumor that does not enhance or shows markedly diminished enhancement, is high on T2W and proton density images, is low on T1W images, and has an irregular border). (Assuming that the entire abnormality may be comprised of: (1) an enhancing component, (2) a non-enhancing component, (3) a necrotic component and (4) a edema component.)



Visually, when scanning through the entire tumor volume, what proportion of the entire tumor would you estimate is **enhancing.** (Assuming that the entire abnormality may be comprised of: (1) an enhancing component, (2) a non-enhancing component, (3) a necrotic component and (4) a edema component.)

Integrative Cancer Research with Digital Pathology

High-resolution whole-slide microscopy



Integrative Analysis: OSU BISTI NBIB Center Big Data (2005)

- Associate genotype with phenotype
- Big science experiments on cancer, heart disease, pathogen host response
  - Tissue specimen -- 1 cm^3
  - 0.1 µ resolution – roughly 10^{15} bytes
  - Molecular data (spatial location) can add additional significant factor; e.g. 10^2
    - Multispectral imaging, laser captured microdissection, Imaging Mass Spec, Multiplex QD
  - Multiple tissue specimens; another factor of 10^3
- Total: 10^{20} bytes -- 100 ***exabytes*** per big science experiment

Source - IEEE Big Data Conference 2015

Big Data vs HPC

	Big Data	HPC
Applications	Data analytics: Social networks, industry	Large-scale scientific simulation: government, industry
Characterized by	Typically, independent file operations, database queries	Typically map to 3-D grid to represent physical space
Prevalent data abstractions	Graphs (sparse), databases, text files	Arrays (dense and sparse), objects
Programming Models	Map-Reduce/HIVE/Giraph etc.	MPI/OpenMP/CUDA widely used
Failure Model	Assume failures common, need to be tolerated	Assume failures infrequent (spend $)
System Cost	Use the technology with the best price-performance ratio	Use the fastest possible processors/network

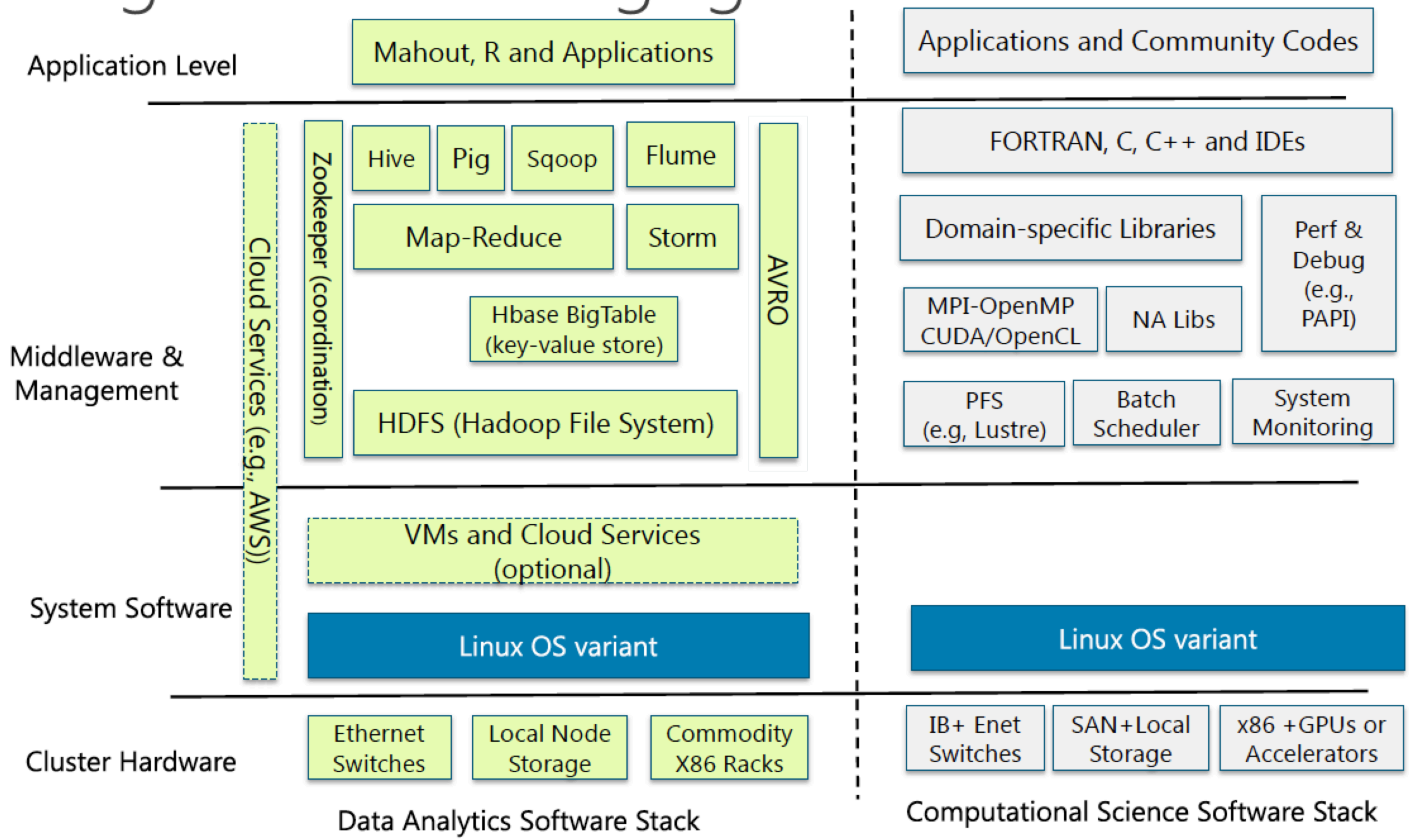
Source - “ Adaptive Large Scale Computing Systems : Need vs Want,” Dan Reed, IEEE Big Data Conference

Challenges (Exascale/Big Data)

Exascale	Big Data
Energy budget limitation	Energy Consumption
Interconnect tightly couple	Interconnect wide and open
Memory, hierarchical	Memory, flat and big
Scalable system software	Scalable storage system
Programming systems	Programming tools
Data management	Data management
Network, Workflow engine	Network, Workflow engine
Exascale Algorithms	Exabyte Data Algorithms
Algorithm for recovery, fault tolerance, hard crashing	Algorithm for recovery, fault tolerance, soft landing
Correctness, reproductively	Stochastic convergent, reproductively
Science productivity	Conclusive guidance and predictive conclusion
Real time simulation	

Challenges - Big Data/Exascale

Integration is challenging in both directions



Source - " Adaptive Large Scale Computing Systems : Need vs Want," Dan Reed, IEEE Big Data Conference

Big Data in Machine Learning – GPU acceleration

Source captured frin - Julie Bernauer – HPC Advisory Council Stanford Tutorial – 2017/02/07

GOOGLE DATACENTER

1,000 CPU Servers
2,000 CPUs • 16,000 cores

600 kWatts
$5,000,000

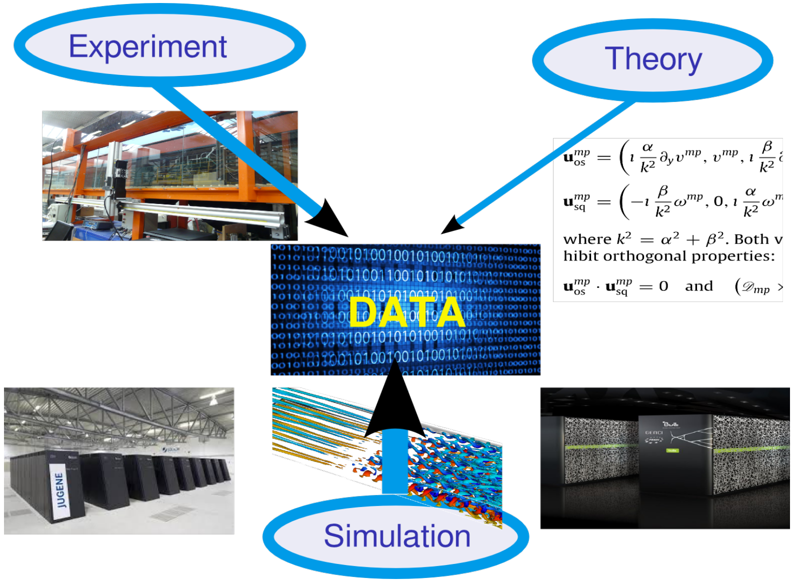
STANFORD AI LAB

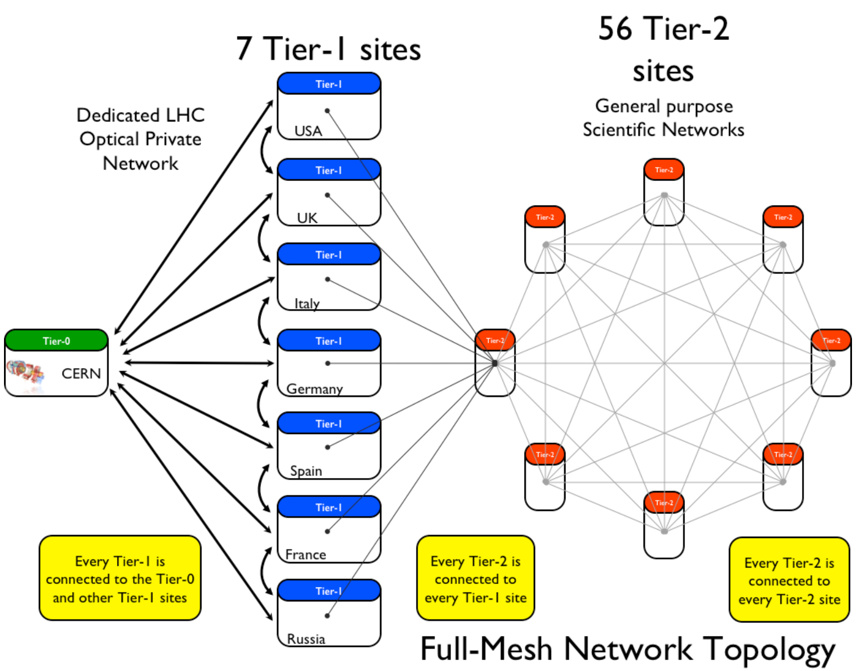
3 GPU-Accelerated Servers
12 GPUs • 18,432 cores

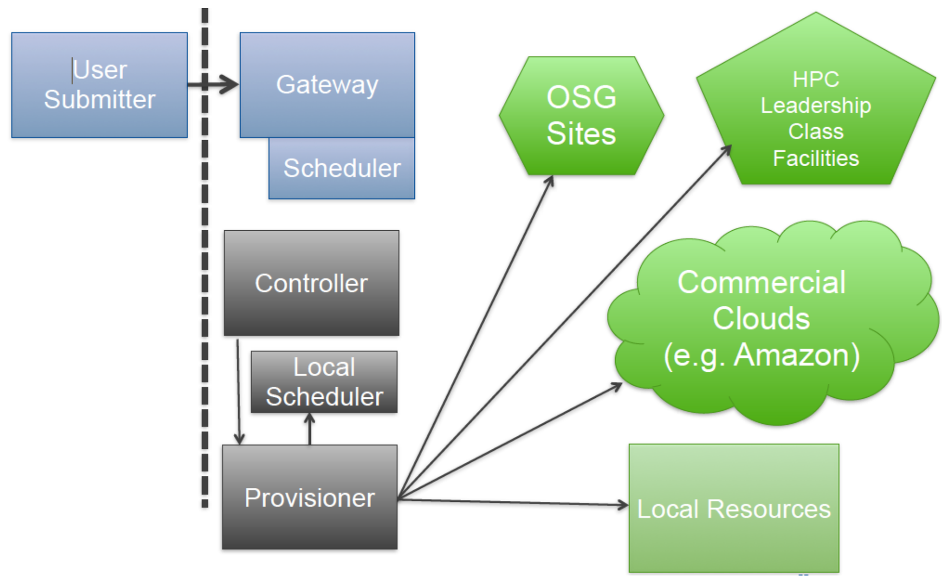
4 kWatts
$33,000

Source captured frin - Julie Bernauer – HPC Advisory Council Stanford Tutorial – 2017/02/07

Gateway, Workflow, Unified Tools, Instrumentation





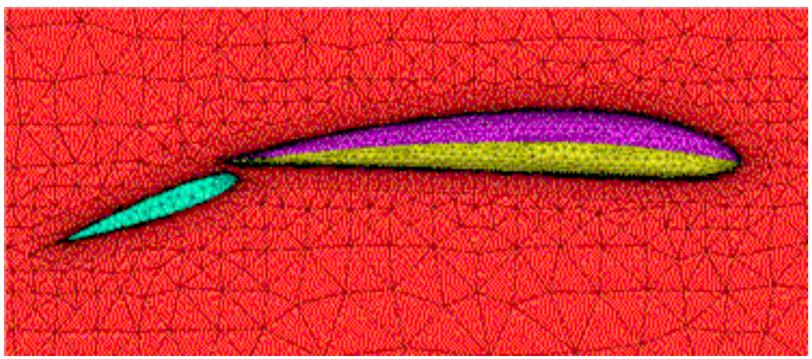


Source - IEEE Big Data Conference 2015

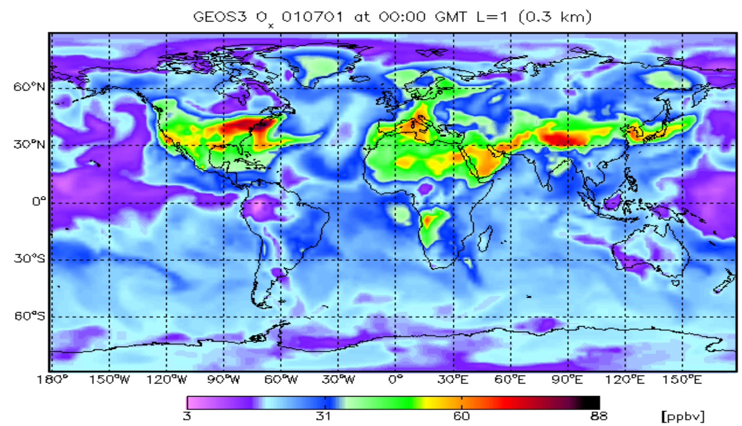
Menu

Need of Parallel Computer

- Requirement of computational capacity depends on applications and formulations and what you want to achieve
- Length Scale (memory) - resolution of the dimension, e.g. number of grid points
- Time Scale (fast) - resolution of duration, e.g. number of time step



- 2D problem :
- grid points 100x100 = 10000 pts
- a vector of 10000 elements ~ 80 KB
- need 10 such vectors ~ 800KB
- Steady State in seconds



- 3D problem :
- grid points 10000x10000 x100 = 10e10 pts
- 10e10 unknowns ~ 80 GB
- need 10 such vectors ~ 800 GB MEMORY
- 100 years simulation !!

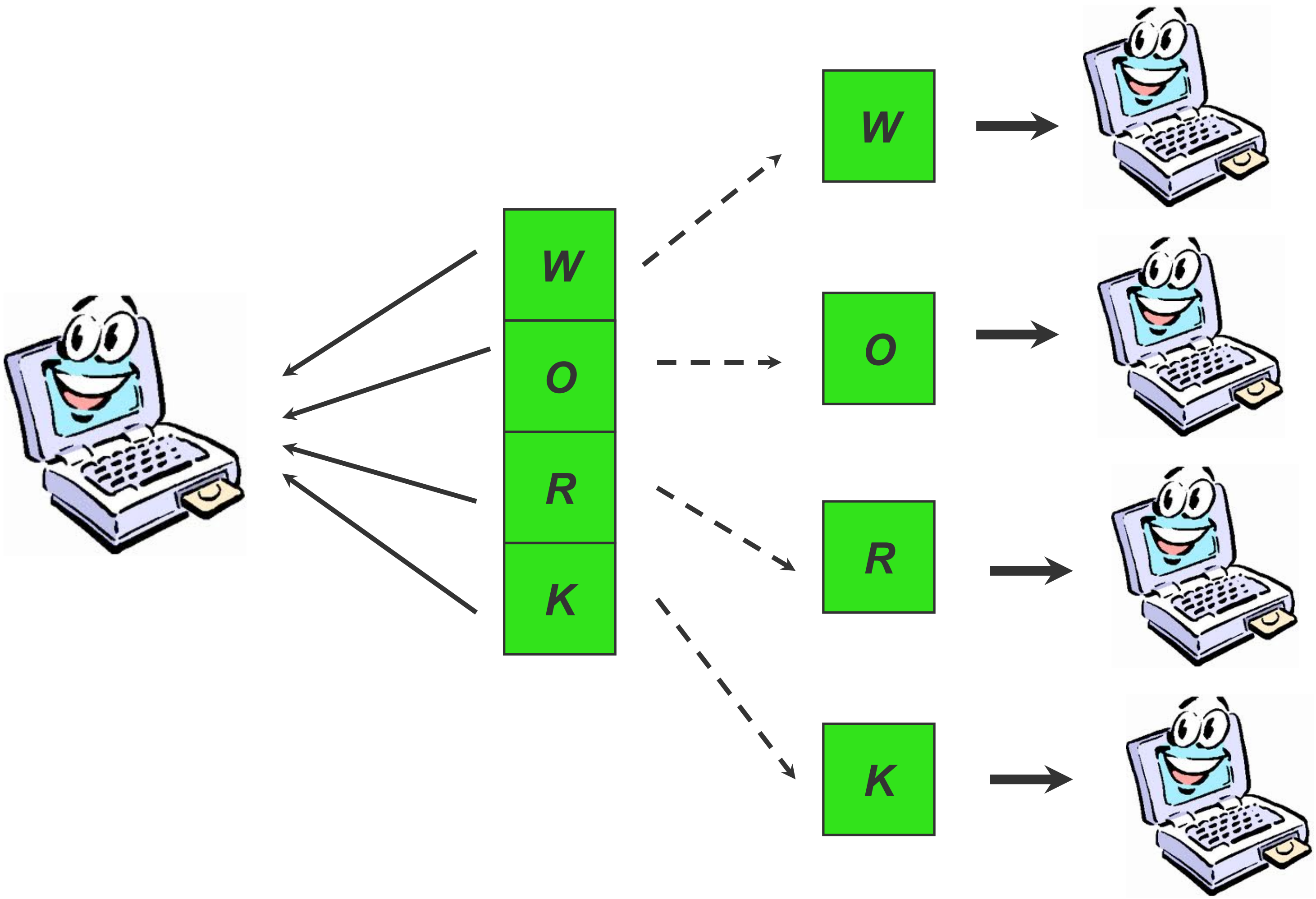
NEED MULTIPLE WORKERS and MEMORY – PARALLEL COMPUTER

Parallel Computing

Division of work into smaller tasks

Multiple computers work on smaller tasks simultaneously

>> Reduce Wall Clock Time <<



Issues of Parallel Computing

- Pros :
  - decrease wallclock time
  - deliver huge amount of memory
  - Allow realistic simulation
- Cons :
  - Difficult to construct
  - Efficient parallel algorithm may need some thoughts
  - Cost of program development

KEYS:

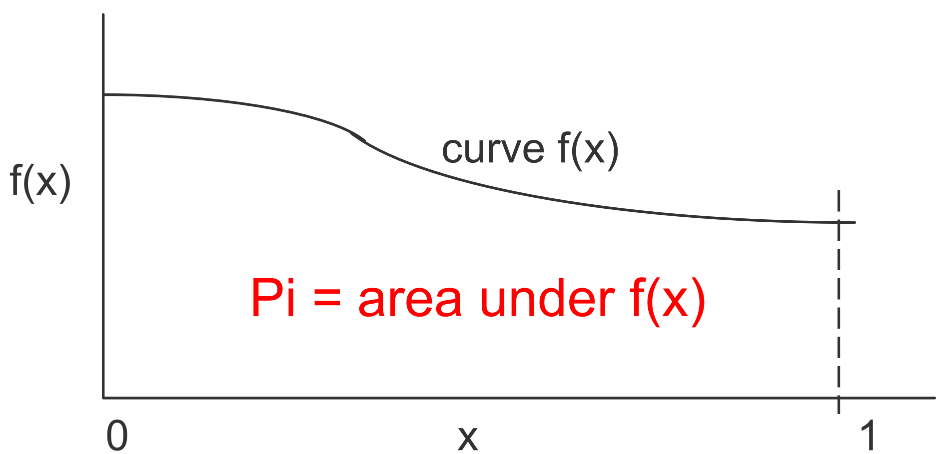
1) <u>LOAD BALANCE</u> - same amount of work for every processor
2) <u>LOCALITY</u> - minimize communications among processors
3) <u>PORTABILITY</u> - work well on different platforms of computers
4) <u>SCALABILITY</u> - can solve larger problem efficiently

Parallel Programming Example: Calculating Pi

- **Use numerical integration to compute Pi**
- **Let $f(x) = 4 / (1+x^2)$ then integrate f(x) from x = 0 to 1**
- **Using the rectangle rule**

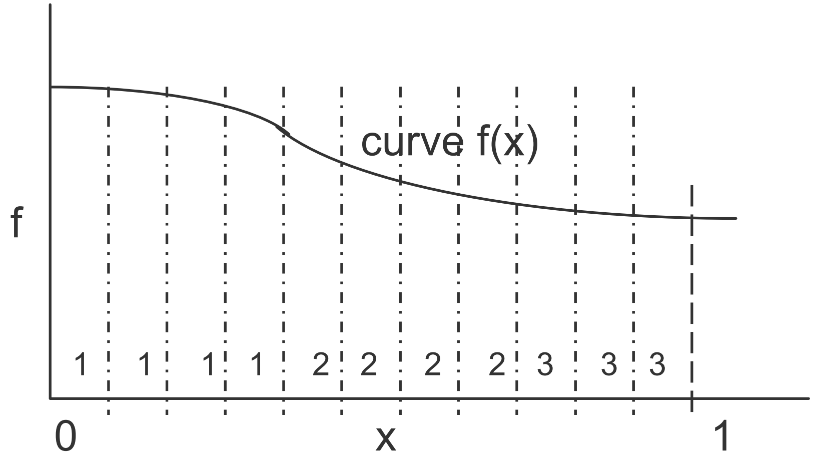
$$R_n(f) = h\sum_{i=1}^{n} f(x_i)$$

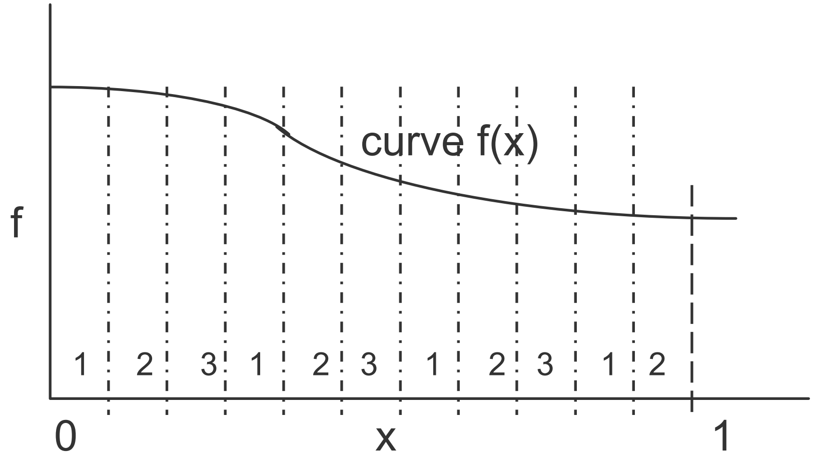
where n = the number of intervals, h = 1/n is the rectangle width and $x_i = h.(n-0.5)$ is the midpoint of each rectangle



Pi Using Rectangles

- Method: Divide area under curve into rectangles and distribute the rectangles to the processors
- Suppose there are 3 processors, how should the distribution be done?





Parallel Performance Measure

- Using multiple processors you hope your program will go faster
- Observed Speedup using N processors to accomplish a task

$$\boxed{\text{Speedup} = \frac{T(1)}{T(N)} \quad \frac{\text{Time taken using 1 processor}}{\text{Time taken using N processors}}}$$

- To be fair, should use the "best" serial algorithm on 1 processor, not the parallel algorithm, simply restricted to 1 processor
- Linear speedup:
  - Two processors take 1/2 the time of 1 processor, so speedup =2
  - N processors take 1/N the time of 1 processor, so speedup =N
- Superlinear speedup
  - May be obtained occasionally, usually due to cache and memory improvements

Amdahl's Law

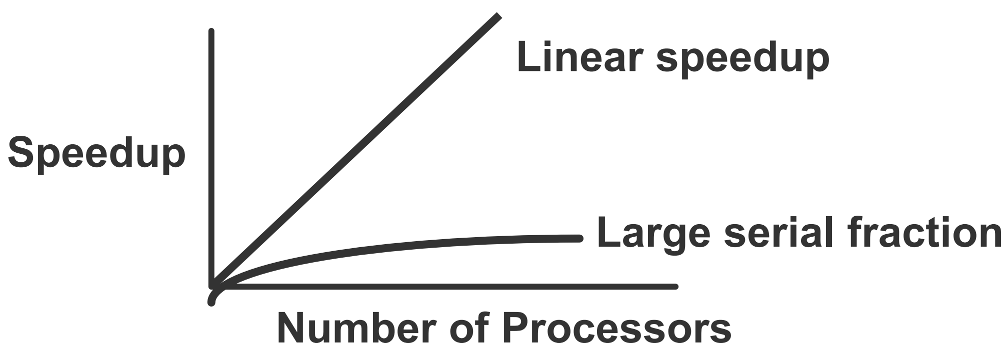
- Maximum speedup is limited by the serial fraction of a program
- Serial code

parallelizable (90) **Serial (10)**

  - Time taken: 100
- Parallel code (using num procs P >> 10)

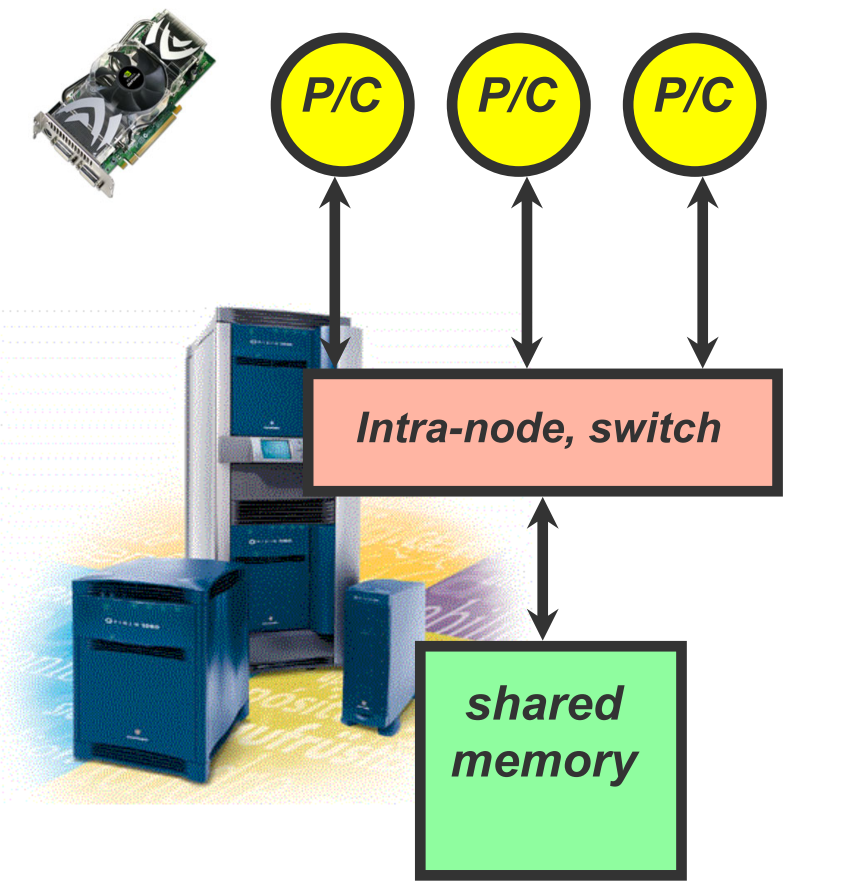
parallel (0) **serial(10)**

  - Time taken =10, maximum speedup=10, regardless of P

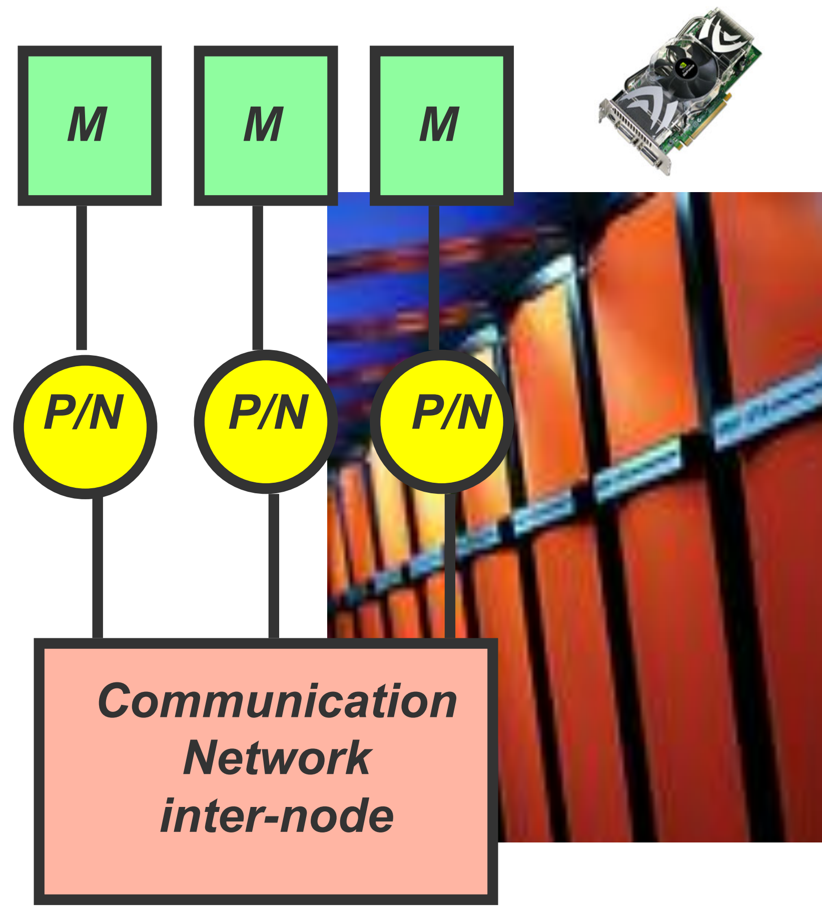


Parallel Computers (simple story)

Shared Memory Systems (SMP)
(Multicore Node)
(Thread-base, OpenMP,)



Distributed Memory Systems (MPP)
(IBM SP, Cray XT or PC Cluster)
(USE MESSAGE PASSING)



Modern Supercomputers

Commodity plus Accelerator Today

Commodity

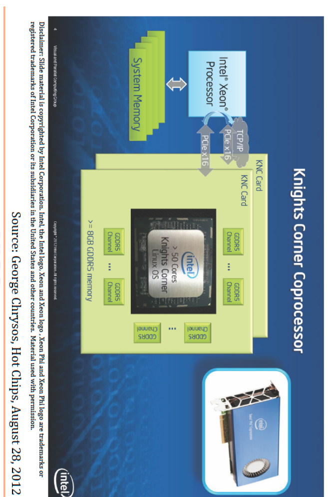
Intel Xeon
8 cores
3 GHz
8*4 ops/cycle
96 Gflop/s (DP)

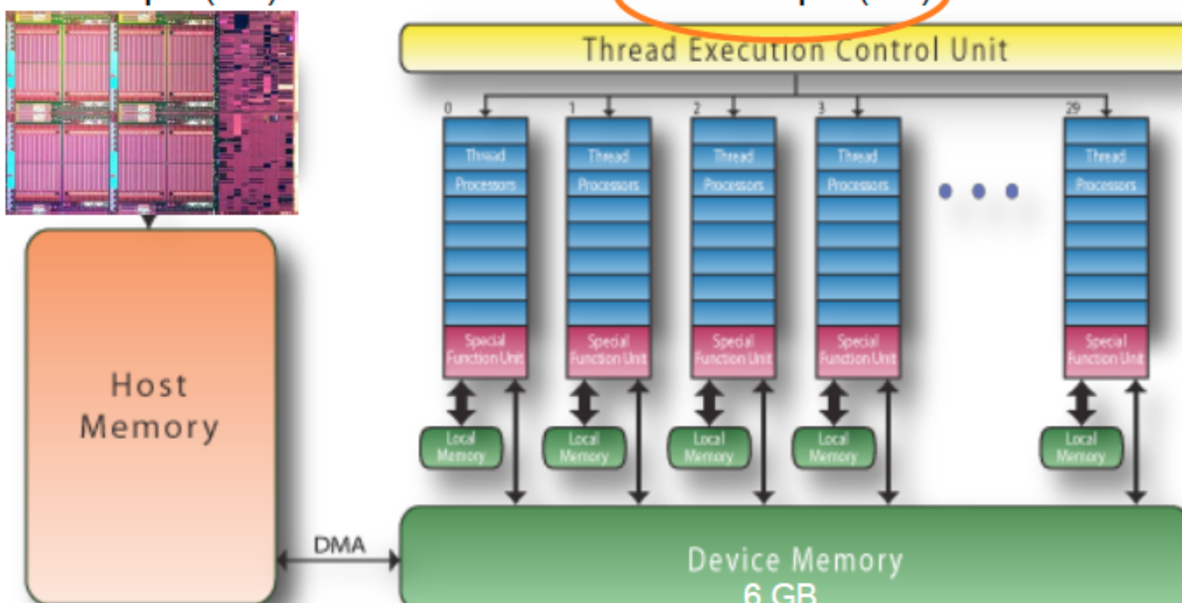
Accelerator (GPU)

Nvidia K20X "Kepler"
2688 "Cuda cores"
.732 GHz
2688*2/3 ops/cycle
1.31 Tflop/s (DP)

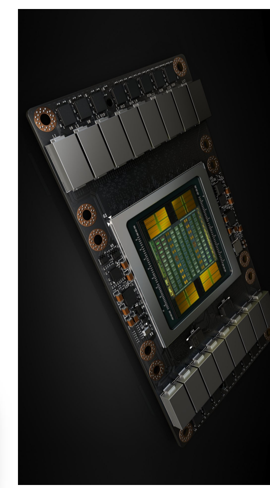
192 Cuda cores/SMX
2688 "Cuda cores"

Accelerator
(Intel PHI)





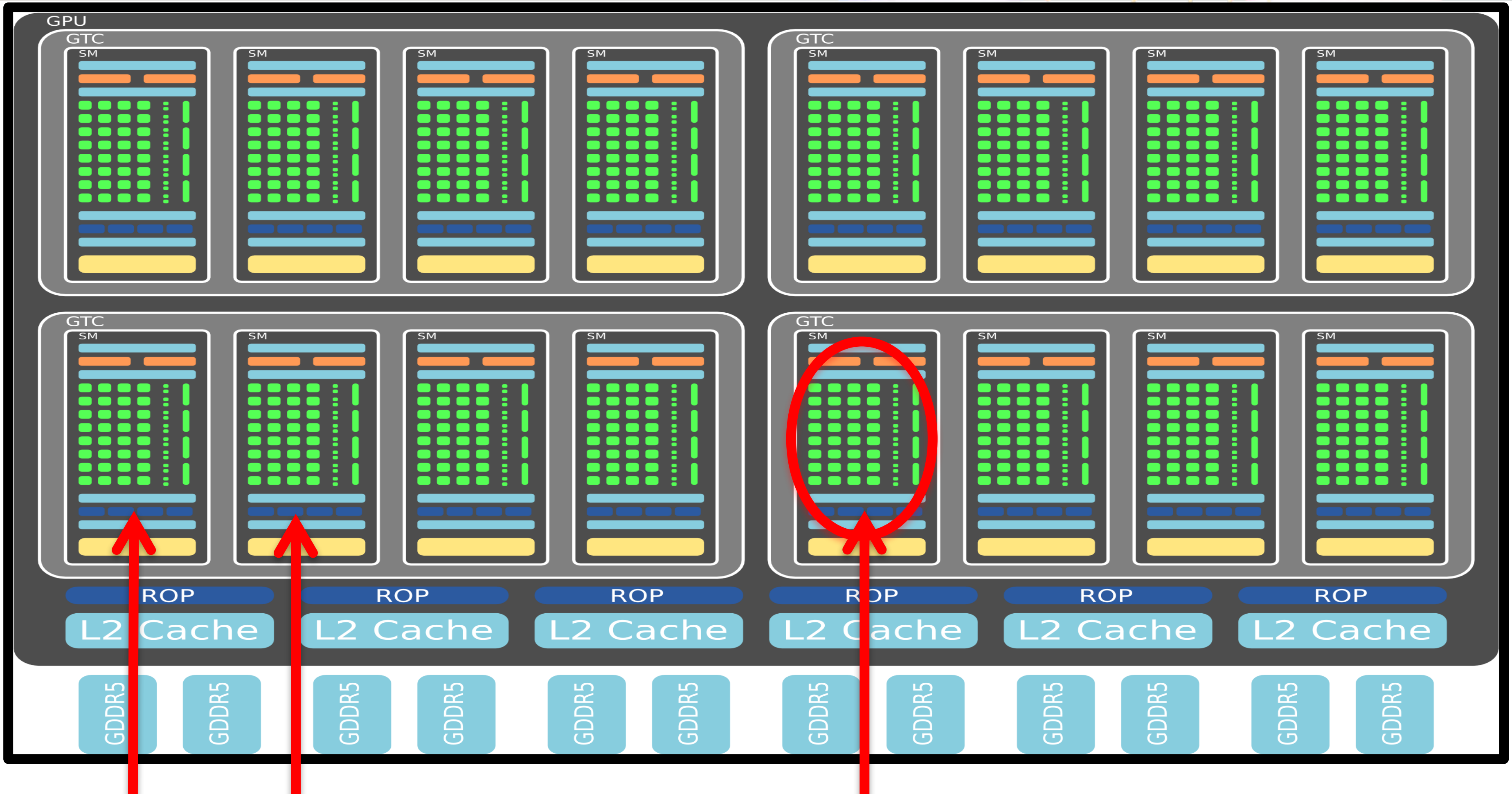
Accelerator
(GPU Volta)



Interconnect
PCI-X 16 lane
64 Gb/s (8 GB/s)
1 GW/s

From ICL Dr. Jack Dongarra : icl.cs.utk.edu

GPU architecture :



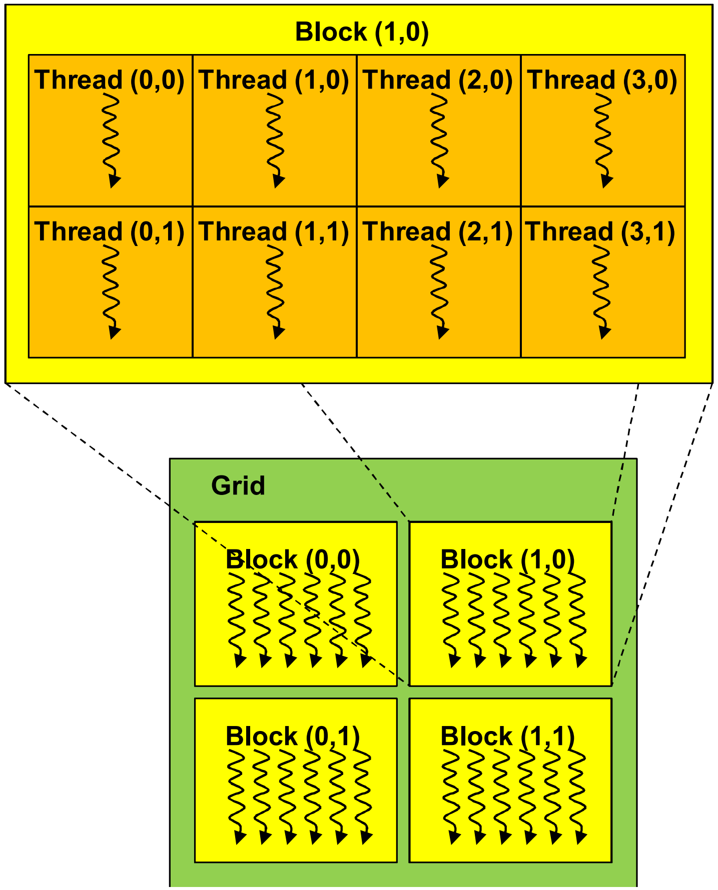
Streaming Multiprocessors (SMs)

32 cores

Reference: http://nvidia.com

GPU programming model:

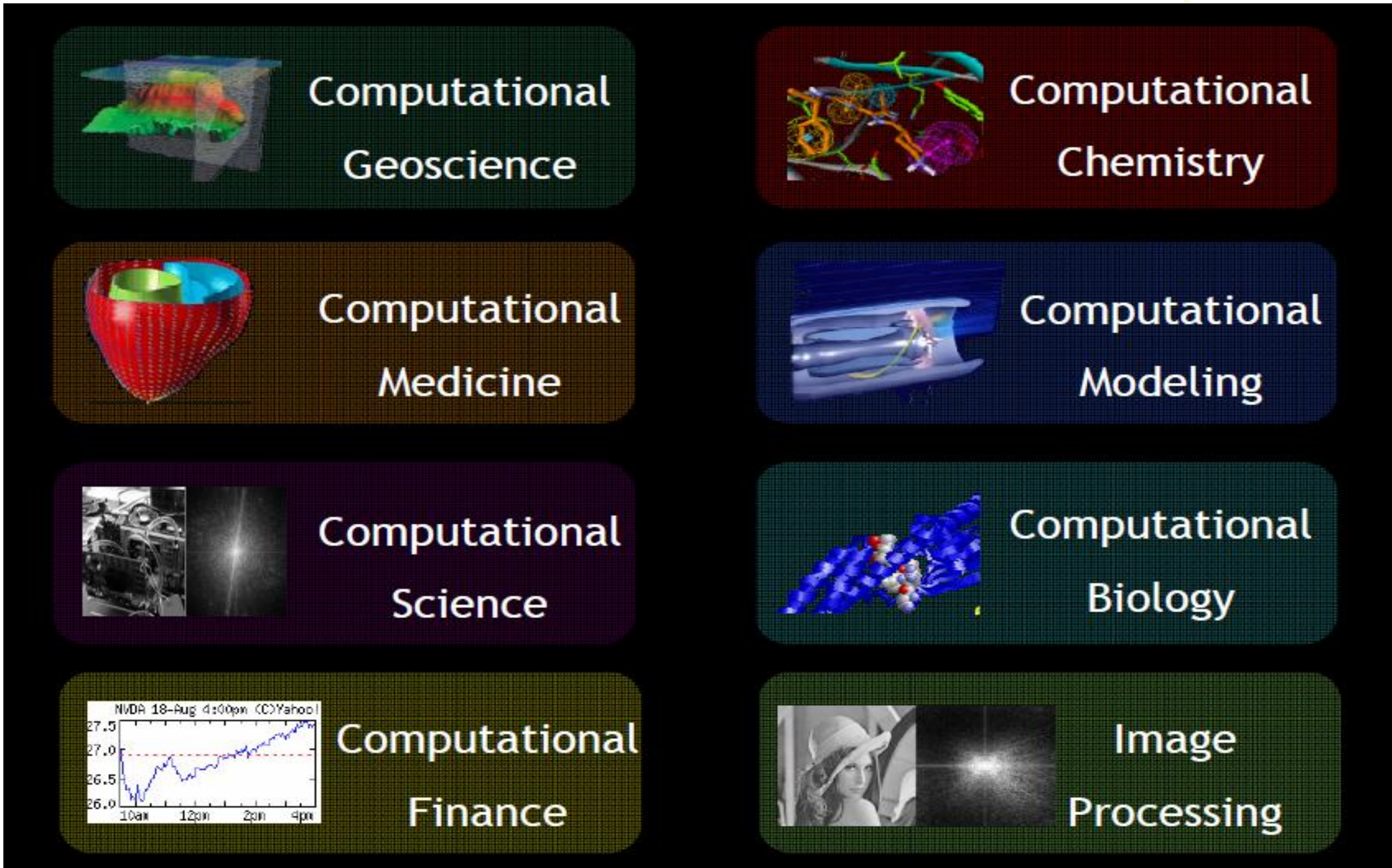
- **GPU accelerator is called device, CPU is host.**
- **GPU code (kernel) is launched and executed on the device by several threads.**
- **Threads grouped into thread blocks.**
- **Program code is written from single thread's point of view.**
  - **Each thread can diverge and execute a unique code path (can cause performance issues)**
- **Compute Unified Device Architecture (CUDA)**



Introduction to CUDA:

- **Compute Unified Device Architecture**
- **CUDA is a C/C++ language extension for GPU programming.**
  - **PGI has developed similar FORTRAN 2003 extension.**
- **Two APIs: Runtime and Driver**

CUDA applications:



Good Practices

Use existing libraries
Understand the issues
Does it worth it to start from scratch
Ask the experts

Improving Scientific Computing: the process

- 1.Write the program, or build it from previous codes, etc.
- 2. Debug your code (with optimization switches off)
- 3. Ensure mathematical correctness of the program!
- 4. Profile your code – determine where most of the computing time is spent
- 5. Optimize the algorithm, the data mapping, the communication, the I/O
- 6. Try out different combinations of compiler flags and/or compiler directives
- 7. Profile your code again
- 8. Re-examine blocks of code that consume the most execution time
- 9. Repeatedly apply various optimizations to such blocks
- 10. Rerun optimized code, compare performance, and start again until "satisfied".

Final thoughts: Strategies for Improved Performance

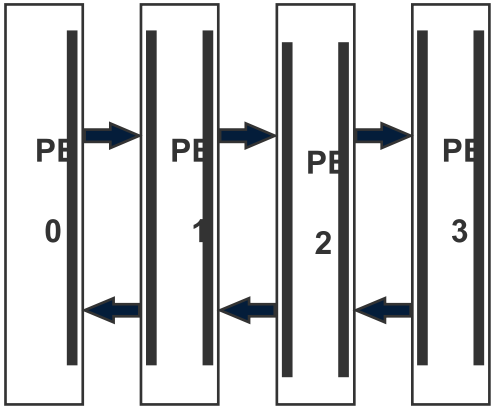
- Improving performance is a complex task, and the amount of time and effort put into it might not always be worth it.
- A certain trade-off must be reached between the developmental time and the "final" production run time.
- If you need to work on a previously existing code, then take the time to learn the details of its logic (if possible). Sometimes you might be better off rewriting the whole code directly in parallel!
- If you write the program from scratch, take some time to think about the different performance issues presented here and/or elsewhere.
- Examine benchmark results and know the limits of the computing platform

Finally: What else can be done?

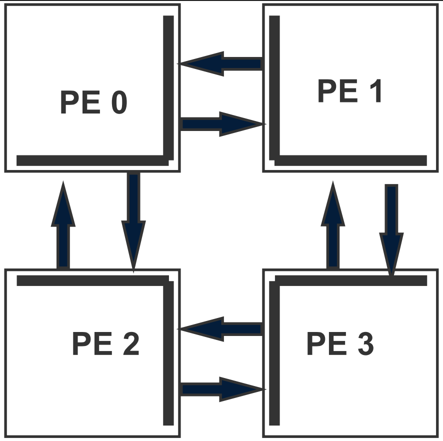
- Practice, try new approaches, innovate, ask others
- Remember to concentrate only on subroutines worth improving
- Rethink the whole algorithm from scratch !?
- Remember to re-check the results for “correctness” (whenever possible!)
- Change parallel method (?), or change parallel machine (?)
- (ask someone else to do the calculations! ;-)

Mapping Problem : Decomposition

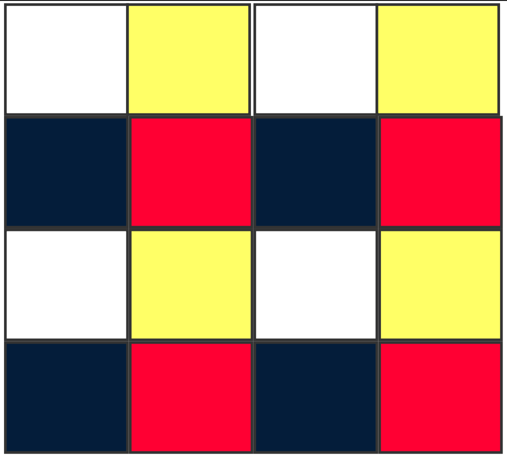
- Each processor should have a similar amount of work
- Expensive communications should be minimized.
- Communications should be:
  - eliminated where feasible
  - localized otherwise (i.e. communicate between close CPU neighbors) (not crucial anymore)
- Concurrency should be maximized
- NOTE: finding the best mapping is an NP-complete problem!   :-(



1D Decomposition



2D Decomposition



2D Block Cyclic

Load Balancing

- **Static**
  - **Data or tasks are partitioned initially among the existing node processors**
  - **Problem: finding a good initial mapping of data or tasks to the processors**

- **Dynamic**
  - **Assumes there is a pool of tasks which can be selected and distributed at runtime (e.g. a task queue or bag_of_tasks)**
  - **Next available task is assigned to a free processor**
  - **Or, it implies that the data can be redistributed appropriately during execution of program**
  - **Problem: Synchronization issues**

Communication Characteristics

- Relatively slow communication vs. computation
  - Peak bandwidths: ~1 MB/sec w/ethernet connections
  - 12.5 MB/sec with a 100 Mbit/sec switch network
  - 150 MB/sec on the SP2
  - 9.6 GB/sec On the Cray XT5 between nodes
  - Implies advantage of using either coarse -or medium- grained parallelism
- The bigger communication cost is in the "startup" or latency
- overhead - 40 usec (software) latency on the SP2
  - sending separate 1-byte messages --> 1s/40us = 25 KBytes/sec !!
  - better sending few large messages rather than many small ones
  - Cray XT5 – latency : a few us
- Bottom line: try to minimize the ratio of
  - (# messages) / (# computations)

Communication Issues

- **Contentions, or traffic jams**
  - **Have good distribution of messages. Circular or round-robin methods in one or two dimensions are fairly efficient for certain problems.**
  - **Avoid as much as possible the use of indirect addressing.**
  - **Use threads on multicore**
- **Ready mode in MPI or post receive before send**
  - **use MPI_Rsend when you are *sure* that a matching receive (MPI_Recv) has been posted appropriately**
  - **this allows faster transfer protocols**
  - **-HOWEVER! behavior is undefined if receive was not posted in time!**
  - **Post receive before send on Cray**
- **Mask communication with computation**
  - **Use asynchronous mode,**
  - **Avoid barrier**

I/O and Parallel I/O

- I/O can be a serious bottleneck for certain applications. The time to read/write data to disks could be an issue. But sometimes the shear size of the data file is a problem.
- Parallel I/O systems allow (in theory) the efficient manipulation of huge files
- Unfortunately, parallel I/O is only available on some architectures, and software is not always good. (MPI-2 has parallel MPI-IO on ROMIO implementation)
- They are restricted to few (around 4 or so) parallel disk drives, through designated I/O nodes.
- On the IBM with GPFS
- Lustre on the ACF System
- One single files vs file/process
- Using local /tmp for input output
- Progress is still needed in this area!

Strategies for Improved Performance

- **Improving performance is a complex task, and the amount of time and effort put into it might not always be worth it.**
- **A certain trade-off must be reached between the developmental time and the "final" production run time.**
- **If you need to work on a previously existing code, then take the time to learn the details of its logic (if possible). Sometimes you might be better off rewriting the whole code directly in parallel!**
- **If you write the program from scratch, take some time to think about the different performance issues that we have been presenting here.**
- **Examine benchmark results and know the limit of the computing platform**
- **profilers "prof" give information on:**
  - **how much time (seconds) is spent in each subroutine**
  - **what percentage of time each subroutine is consuming**
  - **the cummulative time**
  - **the # of calls to subroutines made**
  - **the time (msecs) per call**
  - **Use available system tools**

Performance Tuning Process

- 1. Debug your code (with optimization switches off)
- 2. Ensure mathematical correctness of the program!
- 3. Profile your code
- 4. Optimize the algorithm
- 5. Compile with optimization switches on
- 6. Profile your code
- 7. Examine blocks of code that consume the most execution time
- 8. Repeatedly apply various optimizations to such blocks
- 9. Ensure again the numerical correctness of the program!

- **Finally: What else can be done?**
  - Practice, try new approaches, innovate, ask others
  - Concentrate only on subroutines worth improving
  - Rethink the whole algorithm from scratch !?
  - Re-check the results for correctness (whenever possible!)
  - Change parallel method (?)
  - Change parallel machine (?)
  - (ask someone else to do it! ;-)

Writing Parallel Programs

- Use prewritten programs
  - There are parallel database codes, genetic algorithms, neural networks, linear algebra, etc available
- Writing code to take advantage of parallel libraries
  - Use libraries like ScaLAPACK (Scalable Linear Algebra Package), and other optimized parallel libraries in your code
  - Usually much faster and more robust than code you could easily write
- Writing your own code from scratch
  - The hardest choice… but used by many because of its flexibility

The End

Quote: “I think there is a world market for maybe five computers”
Thomas Watson, chairman of IBM, 1943