COMP 422 Parallel Computing: An Introduction

John Mellor-Crummey

Department of Computer Science
Rice University
johnmc@rice.edu

Course Information

- **Time: TTh 1:00-2:15**
- **Place: DH 1075**
- **Instructor: John Mellor-Crummey**
  - **—Email: johnmc@rice.edu**
  - **—Office: DH 3082, x5179**
  - **—Office Hours: Wednesday 8:30am-9:30am or by appointment**
- **Teaching Assistants**
  - **—Sri Raj Paul email: srp7@rice.edu office: DH 3064**
  - **—Lai Wei email: lai.wei@rice.edu office: DH 3111**
  - **—Office Hours: TBA**
- **WWW site: http://www.clear.rice.edu/comp422**

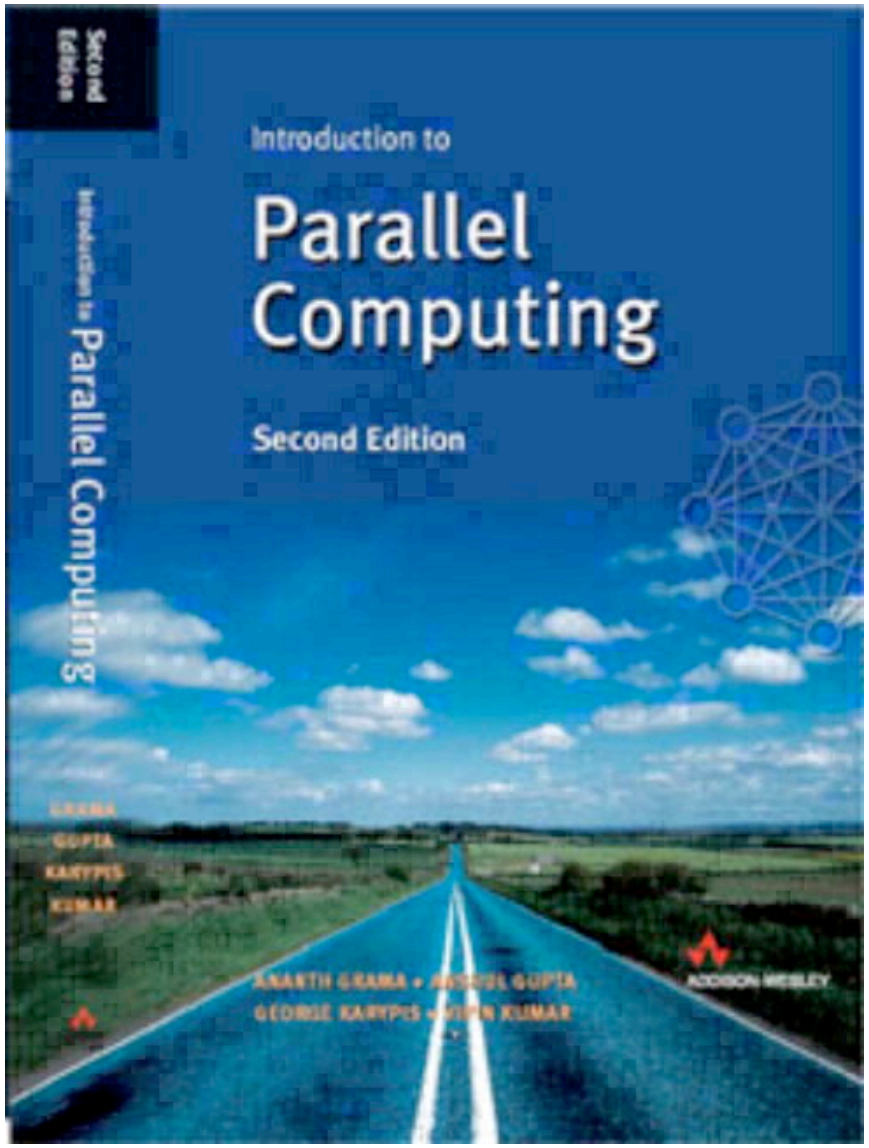
Parallelism

- Definition: ability to execute parts of a program concurrently
- Goal: solve large problems fast
  - —with more parallelism
    - solve larger problems in a fixed time
    - solve fixed size problems in shorter time
- Grain of parallelism: how big are the units?
  - —bits, instructions, statements, loop iterations, procedures, …
- COMP 422 focus: explicit thread-level parallelism
  - —thread = a flow of control executing a sequence of instructions

Course Objectives

- Learn fundamentals of parallel computing
  - —principles of parallel algorithm design
  - —programming models and methods
  - —parallel computer architectures
  - —parallel algorithms
  - —modeling and analysis of parallel programs and systems
- Develop skill writing parallel programs
  - —programming assignments
- Develop skill analyzing parallel computing problems
  - —solving problems posed in class

Recommended Text



Introduction to Parallel Computing, 2nd Edition

Ananth Grama,

Anshul Gupta,

George Karypis,

Vipin Kumar

Addison-Wesley

2003

Topics (Part 1)

- Introduction
- Principles of parallel algorithm design (Chapter 3)
  - —decomposition techniques
  - —mapping & scheduling computation
  - —templates
- Programming shared-address space systems (Chapter 7)
  - —Cilk/Cilk++
  - —OpenMP
  - —Pthreads
  - —synchronization
- Parallel computer architectures (Chapter 2)
  - —shared memory systems and cache coherence
  - —distributed-memory systems
  - —interconnection networks and routing

Topics (Part 2)

- Programming scalable systems (Chapter 6)
  - —message passing: MPI
  - —global address space languages
- Collective communication
- Analytical modeling of program performance (Chapter 5)
  - —speedup, efficiency, scalability, cost optimality, isoefficiency
- Parallel algorithms (Chapters 8 & 10)
  - —non-numerical algorithms: sorting, graphs, dynamic programming
  - —numerical algorithms: dense and sparse matrix algorithms
- Performance measurement and analysis of parallel programs
- GPU Programming with CUDA
- Problem solving on clusters using MapReduce

Prerequisites

- Programming in C, C++, or similar
- Basics of data structures
- Basics of machine architecture
- Prerequisites
  - —(COMP 211 or COMP 215) and Comp 221
  - —or equivalent
- See me if you have concerns

Motivations for Parallel Computing

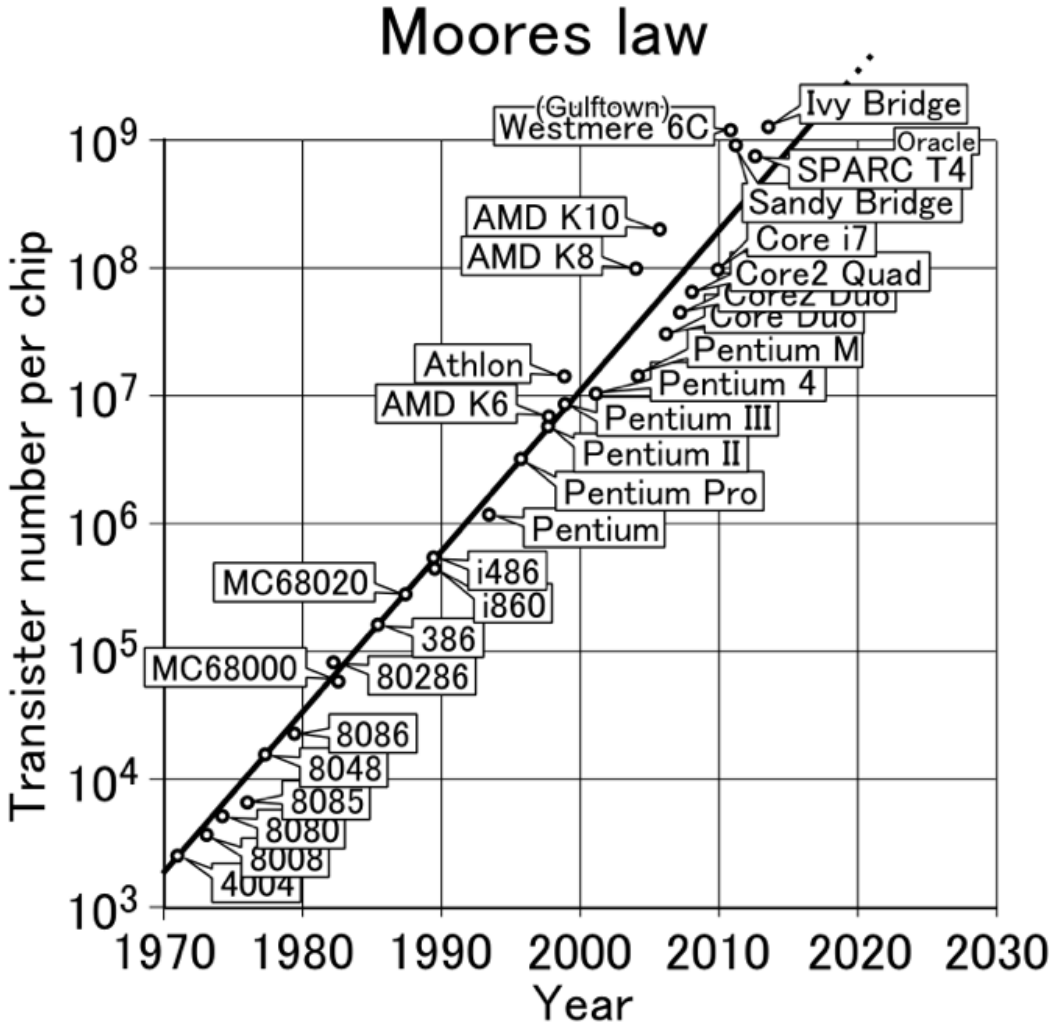
- **Technology push**
- **Application pull**

The Rise of Multicore Processors

Advance of Semiconductors: “Moore’s Law”

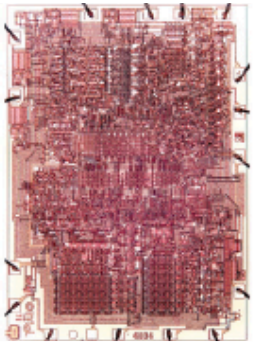
Gordon Moore,
Founder of Intel

- **1965: since the integrated circuit was invented, the number of transistors/inch^2 in these circuits roughly doubled every year; this trend would continue for the foreseeable future**
- **1975: revised - circuit complexity doubles every two years**

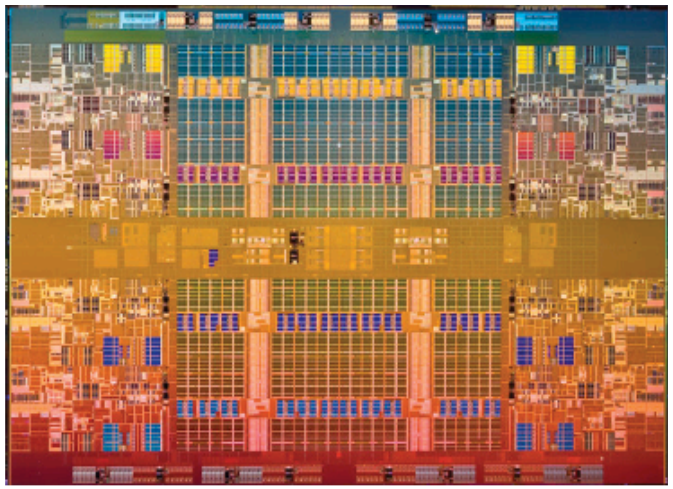


By shigeru23 CC BY-SA 3.0, via Wikimedia Commons

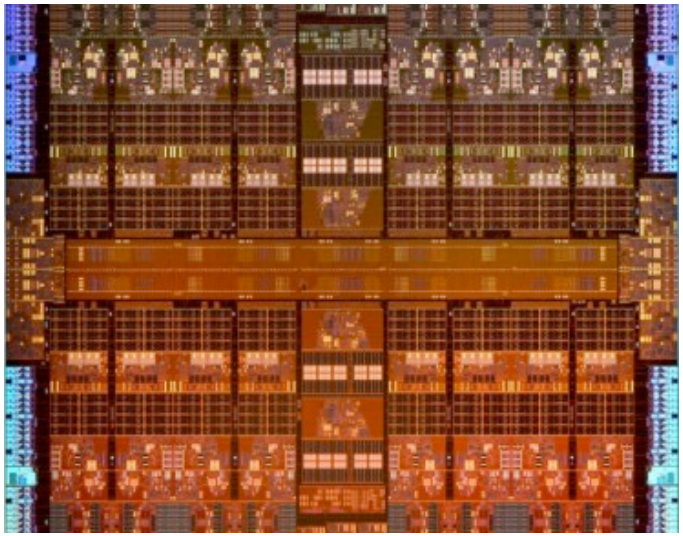
Evolution of Microprocessors 1971-2015



Intel 4004, 1971
1 core, no cache
23K transistors



Intel Nehalem-EX, 2009
8 cores, 24MB cache
2.3B transistors



Oracle M7, 2015
32 cores, 64MB cache
10B transistors

Figure credits:

Intel processors: Shekhar Borkar, Andrew A. Chien, The Future of Microprocessors. Communications of the ACM, Vol. 54 No. 5, Pages 67-77 10.1145/1941487.1941507.

Oracle M7: Timothy Prickett Morgan Oracle Cranks Up The Cores To 32 With Sparc M7 Chip, Enterprise Tech - Systems Edition, August 13, 2014.

(Chip pictures not to scale)

Leveraging Moore's Law Trends

From increasing transistor count to performance

- More transistors = ↑ opportunities for exploiting parallelism
- Parallelism in a CPU core
  - —implicit parallelism: invisible to the programmer
    - pipelined execution of instructions
    - multiple functional units for multiple independent pipelines
  - —explicit parallelism
    - SIMD processor extensions
      - operations on 1, 2, and 4 data items per instruction
      - integer, floating point, complex data
        - e.g. SSE, AVX
    - long instruction words (VLIW)
      - bundles of independent instructions that can be issued together
        - e.g., Itanium processor

Microprocessor Architecture (Mid 90's)

- **Superscalar (SS) designs were the state of the art**
  - —multiple functional units (e.g., int, floating point, branch, load/store)
  - —multiple instruction issue
  - —dynamic scheduling: HW tracks instruction dependencies
  - —speculative execution: look past predicted branches
  - —non-blocking caches: multiple outstanding memory operations
- **Apparent path to higher performance?**
  - —wider instruction issue
  - —support for more speculation

The End of the Free Lunch

Increasing issue width provides diminishing returns

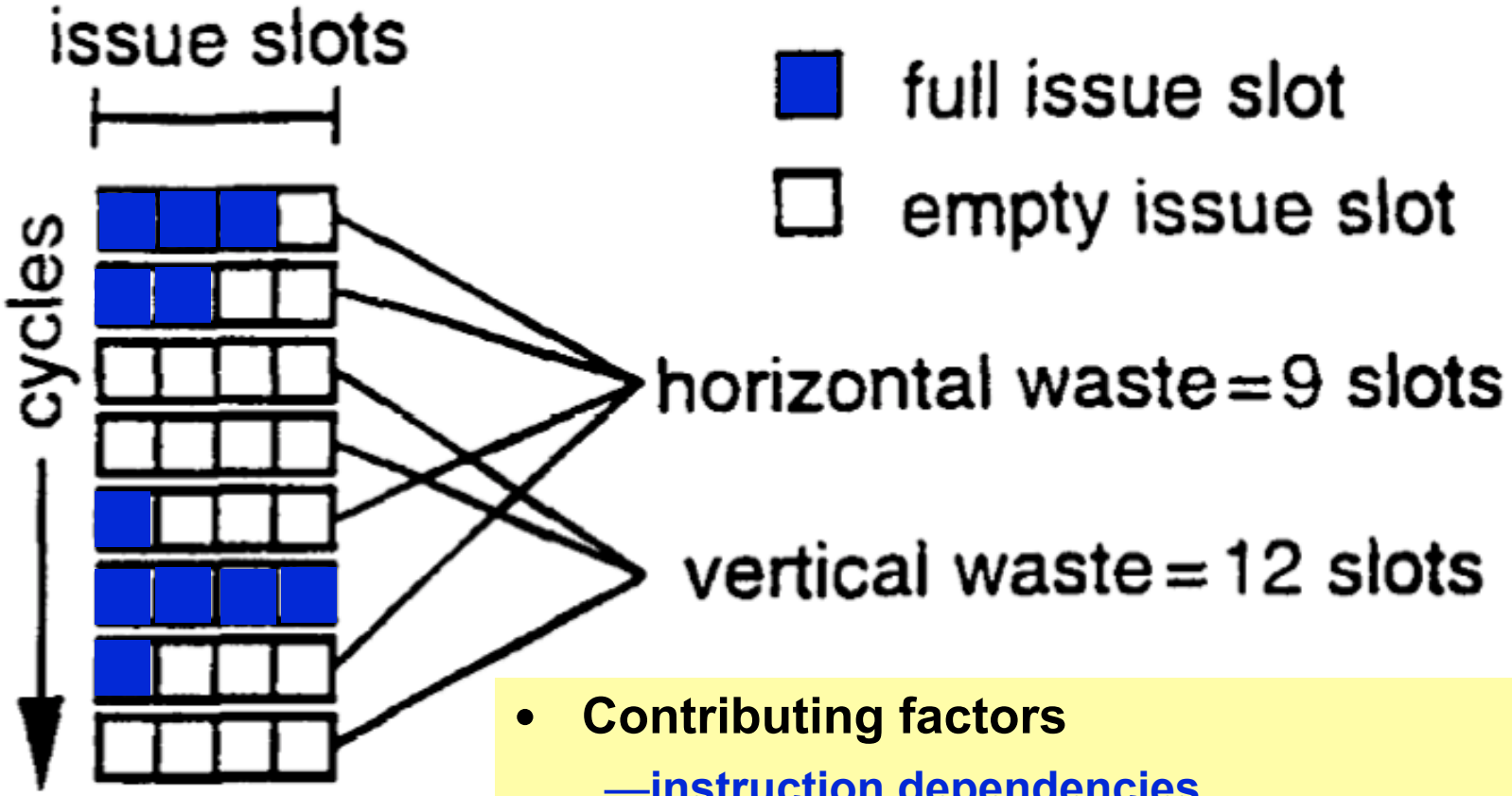
Two factors[1]

- **Fundamental circuit limitations**
  - **—delays ⇑ as issue queues ⇑ and multi-port register files ⇑**
  - **—increasing delays limit performance returns from wider issue**
- **Limited amount of instruction-level parallelism**
  - **—inefficient for codes with difficult-to-predict branches**

[1] **The case for a single-chip multiprocessor, K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K. Chang, ASPLOS-VII, 1996.**

Instruction-level Parallelism Concerns

Issue Waste



- **Contributing factors**
  - **—instruction dependencies**
  - **—long-latency operations within a thread**

Some Sources of Wasted Issue Slots

- TLB miss
- I cache miss
- D cache miss
- Load delays (L1 hits)

Memory Hierarchy

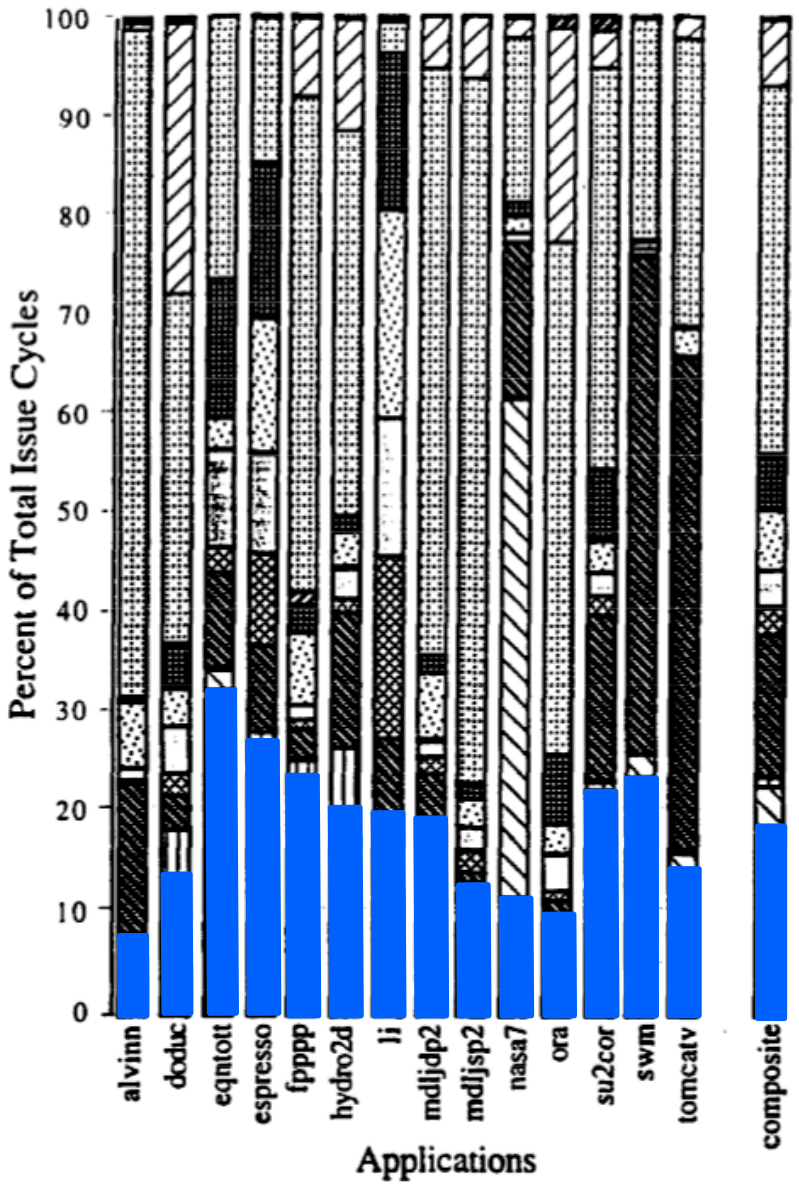
- Branch misprediction

Control Flow

- Instruction dependences
- Memory conflict

Instruction Stream

Simulations of 8-issue Superscalar



Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.

Summary:
Highly underutilized

Applications: most of SPEC92

- On average < 1.5 IPC (19%)
- Dominant waste differs by application
- Short FP dependences: 37%

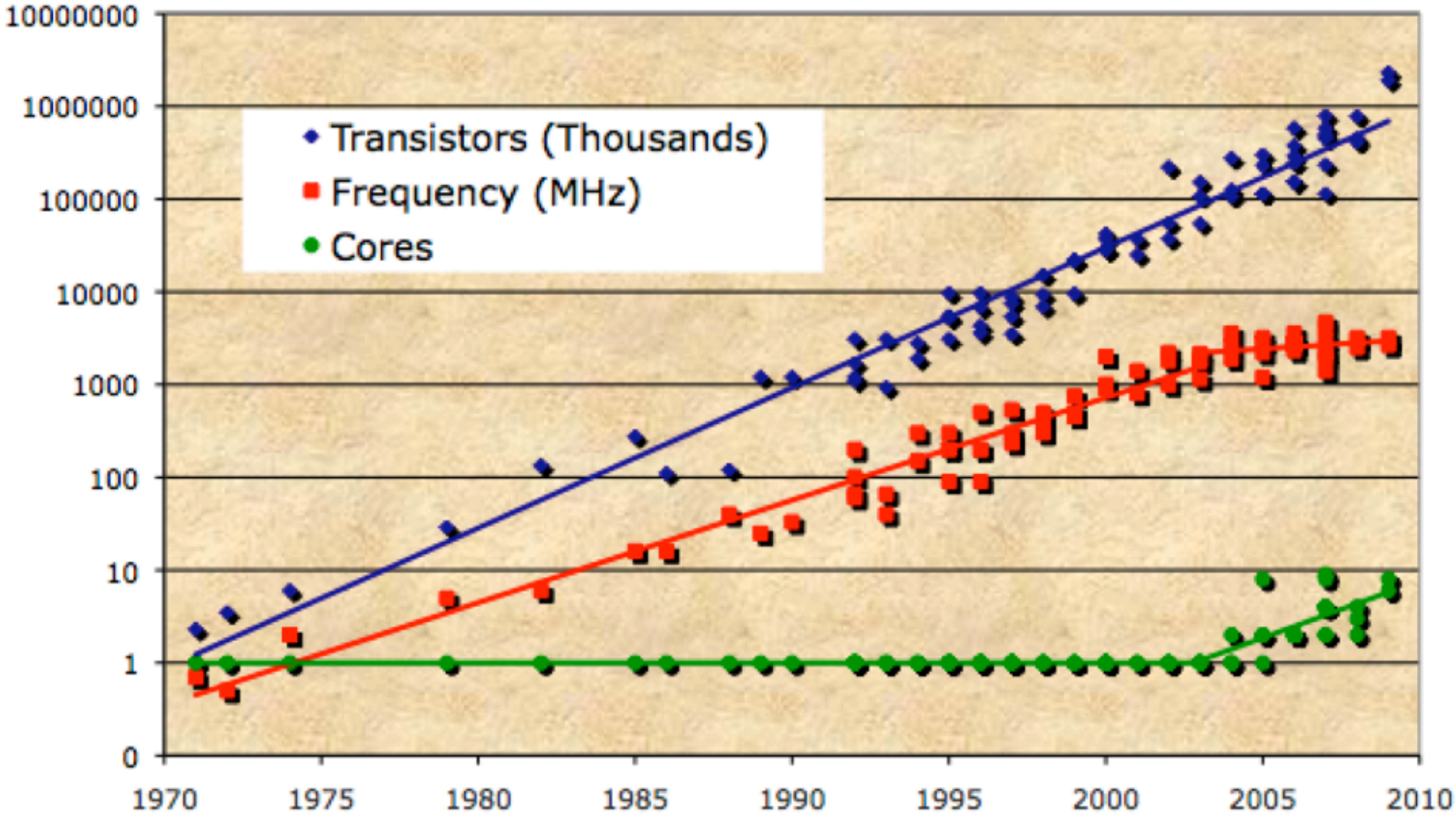
Power and Heat Stall Clock Frequencies

New York Times

May 17, 2004 … Intel, the world's largest chip maker, publicly acknowledged that it had hit a "thermal wall" on its microprocessor line. As a result, the company is changing its product strategy and disbanding one of its most advanced design groups. Intel also said that it would abandon two advanced chip development projects …

Now, Intel is embarked on a course already adopted by some of its major rivals: obtaining more computing power by stamping multiple processors on a single chip rather than straining to increase the speed of a single processor … Intel's decision to change course and embrace a "dual core" processor structure shows the challenge of overcoming the effects of heat generated by the constant on-off movement of tiny switches in modern computers … some analysts and former Intel designers said that *Intel was coming to terms with escalating heat problems so severe they threatened to cause its chips to fracture at extreme temperatures*…

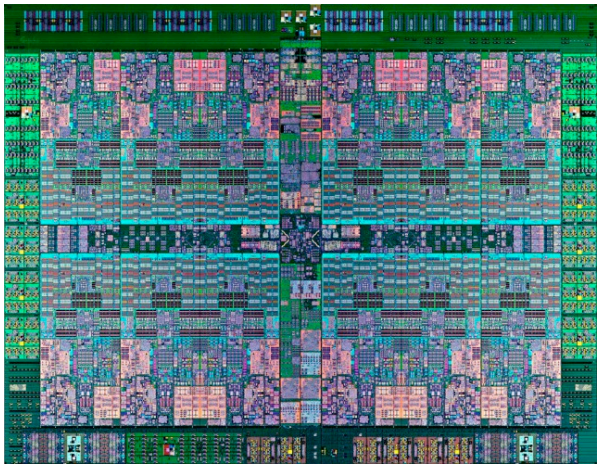
Technology Trends



The Future of Computing Performance: Game Over or Next Level? The National Academies Press, Washington, DC, 2011.

Recent Multicore Processors

- 2H 15: Intel Knight's Landing
  - 60+ cores; 4-way SMT; 16GB (on pkg)
- 2015: Oracle SPARC M7
  - 32 cores; 8-way fine-grain MT; 64MB cache
- Fall 14: Intel Haswell
  - 18 cores; 2-way SMT; 45MB cache
- June 14: IBM Power8
  - 12 cores; 8-way SMT; 96MB cache
- Sept 13: SPARC M6
  - 12 cores; 8-way fine-grain MT; 48MB cache
- May 12: AMD Trinity
  - 4 CPU cores; 384 graphics cores
- Q2 13: Intel Knight's Corner (coprocessor)
  - 61 cores; 2-way SMT; 16MB cache
- Feb 12: Blue Gene/Q
  - 16+1+1 cores; 4-way SMT; 32MB cache



IBM Power8

http://www.extremetech.com/wp-content/uploads/2014/04/ibm-power8-die-shot-640x496.jpg

Application Pull

- **Complex problems require computation on large-scale data**
- **Sufficient performance available only through massive parallelism**

Computing and Science

- "Computational modeling and simulation are among the most significant developments in the practice of scientific inquiry in the 20th century. Within the last two decades, scientific computing has become an important contributor to all scientific disciplines.
- It is particularly important for the solution of research problems that are insoluble by traditional scientific theoretical and experimental approaches, hazardous to study in the laboratory, or time consuming or expensive to solve by traditional means"

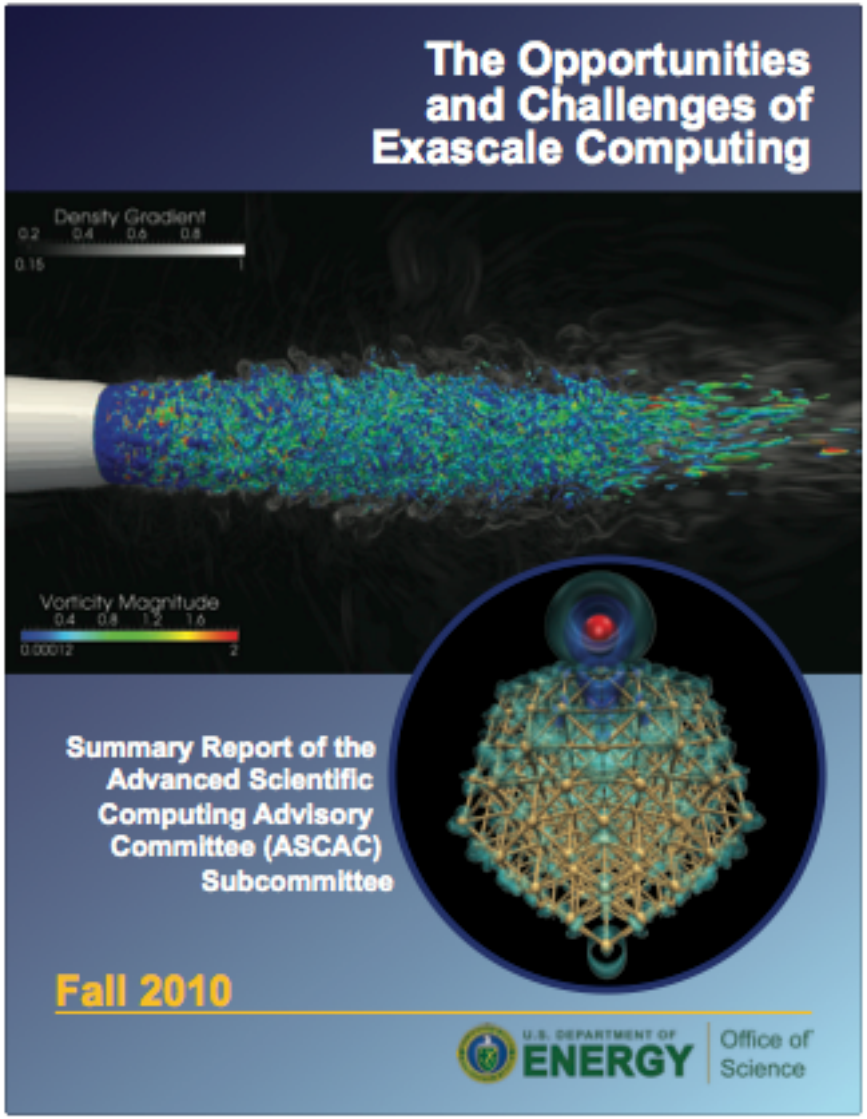
— "Scientific Discovery through Advanced Computing"
DOE Office of Science, 2000

The Need for Speed: Complex Problems

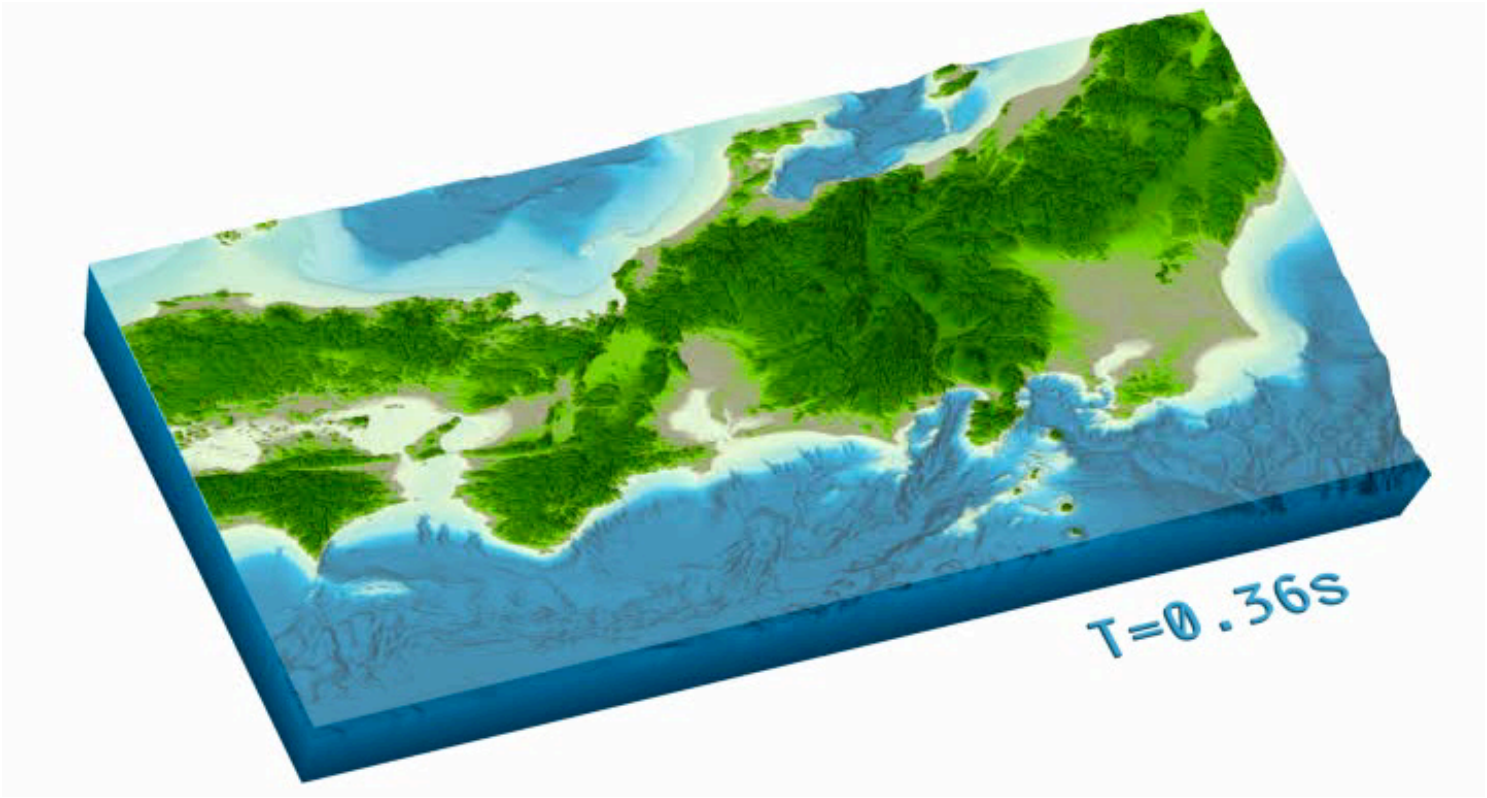
- Science
  - —understanding matter from elementary particles to cosmology
  - —storm forecasting and climate prediction
  - —understanding biochemical processes of living organisms
- Engineering
  - —combustion and engine design
  - —computational fluid dynamics and airplane design
  - —earthquake and structural modeling
  - —pollution modeling and remediation planning
  - —molecular nanotechnology
- Business
  - —computational finance - high frequency trading
  - —information retrieval
  - —data mining “big data”
- Defense
  - —nuclear weapons stewardship
  - —cryptology

The Scientific Case for Exascale Computing

- Predict regional climate changes: sea level rise, drought and flooding, and severe weather patterns
- Reduce carbon footprint of transportation
- Improve efficiency and safety of nuclear energy
- Improve design for cost-effective renewable energy resources such as batteries, catalysts, and biofuels
- Certify the U.S. nuclear stockpile
- Design advanced experimental facilities, such as accelerators, and magnetic and inertial confinement fusion
- Understand properties of fission and fusion reactions
- Reverse engineer the human brain
- Design advanced materials



Earthquake Simulation in Japan

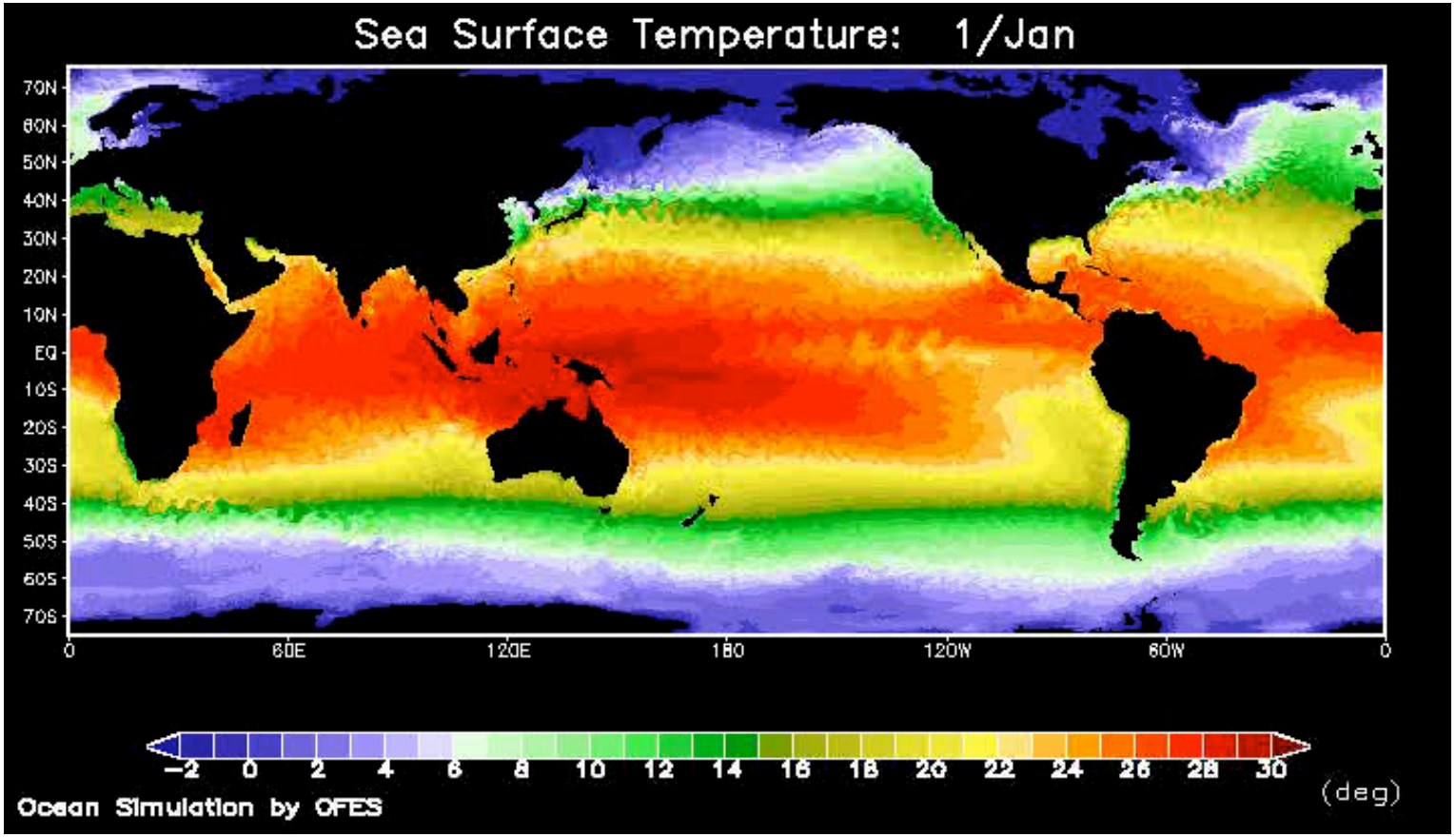


Earthquake Research Institute, University of Tokyo

Tonankai-Tokai Earthquake Scenario

Video Credit: The Earth Simulator Art Gallery, CD-ROM, March 2004

Ocean Circulation Simulation



Ocean Global Circulation Model for the Earth Simulator

Seasonal Variation of Ocean Temperature

Video Credit: The Earth Simulator Art Gallery, CD-ROM, March 2004

Community Earth System Model (CESM)

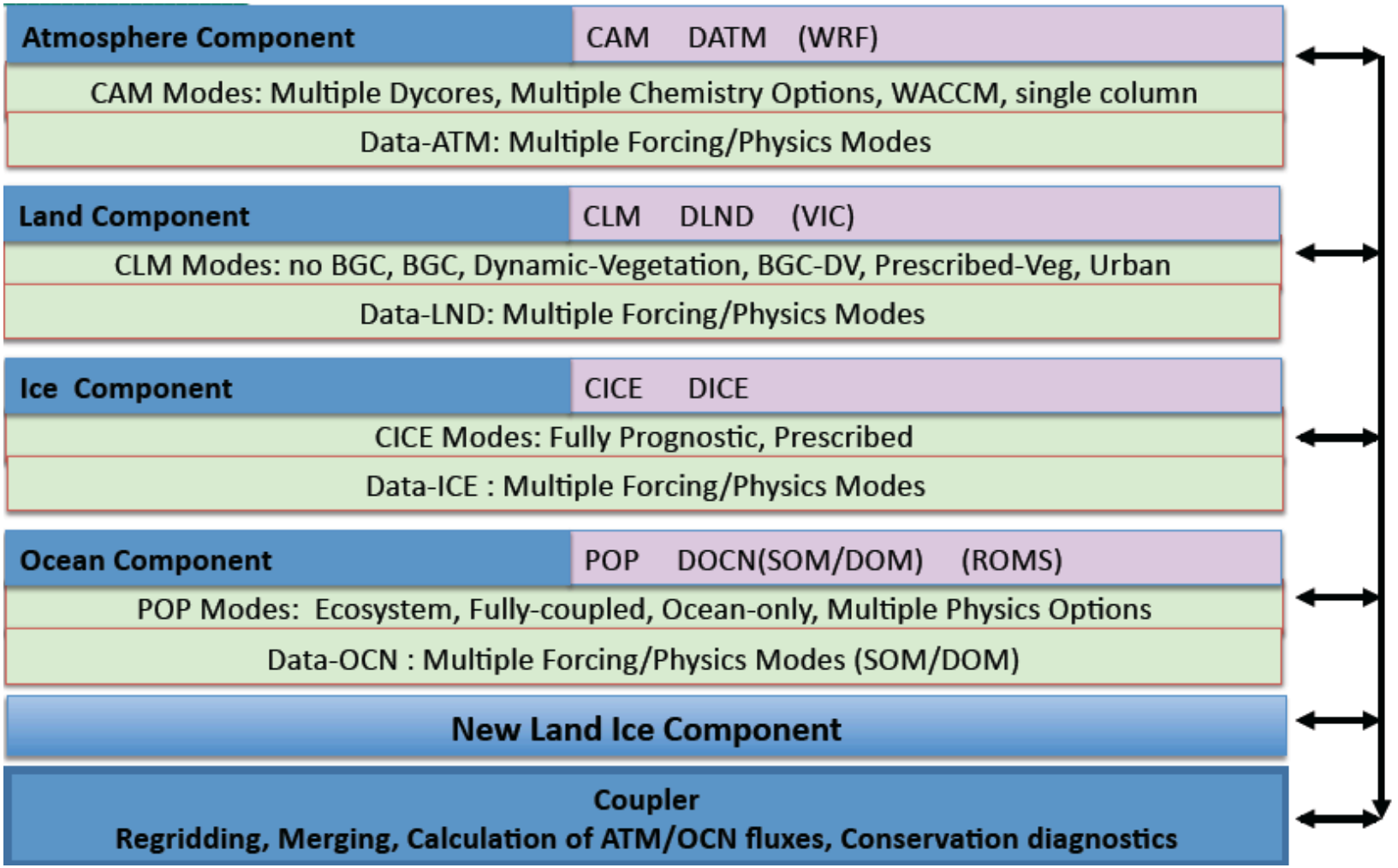


Figure courtesy of M. Vertenstein (NCAR)

CESM Execution Configurations

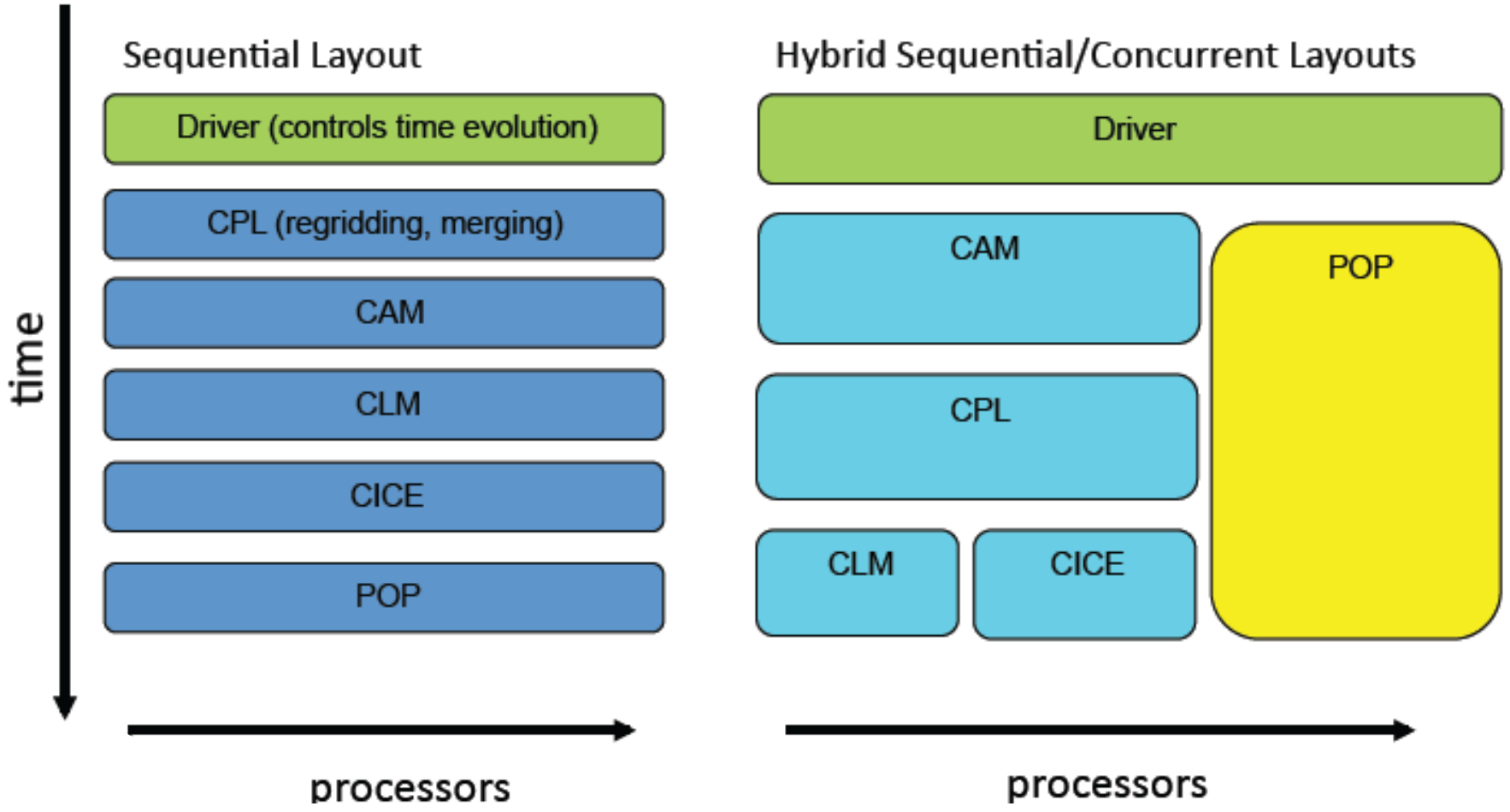
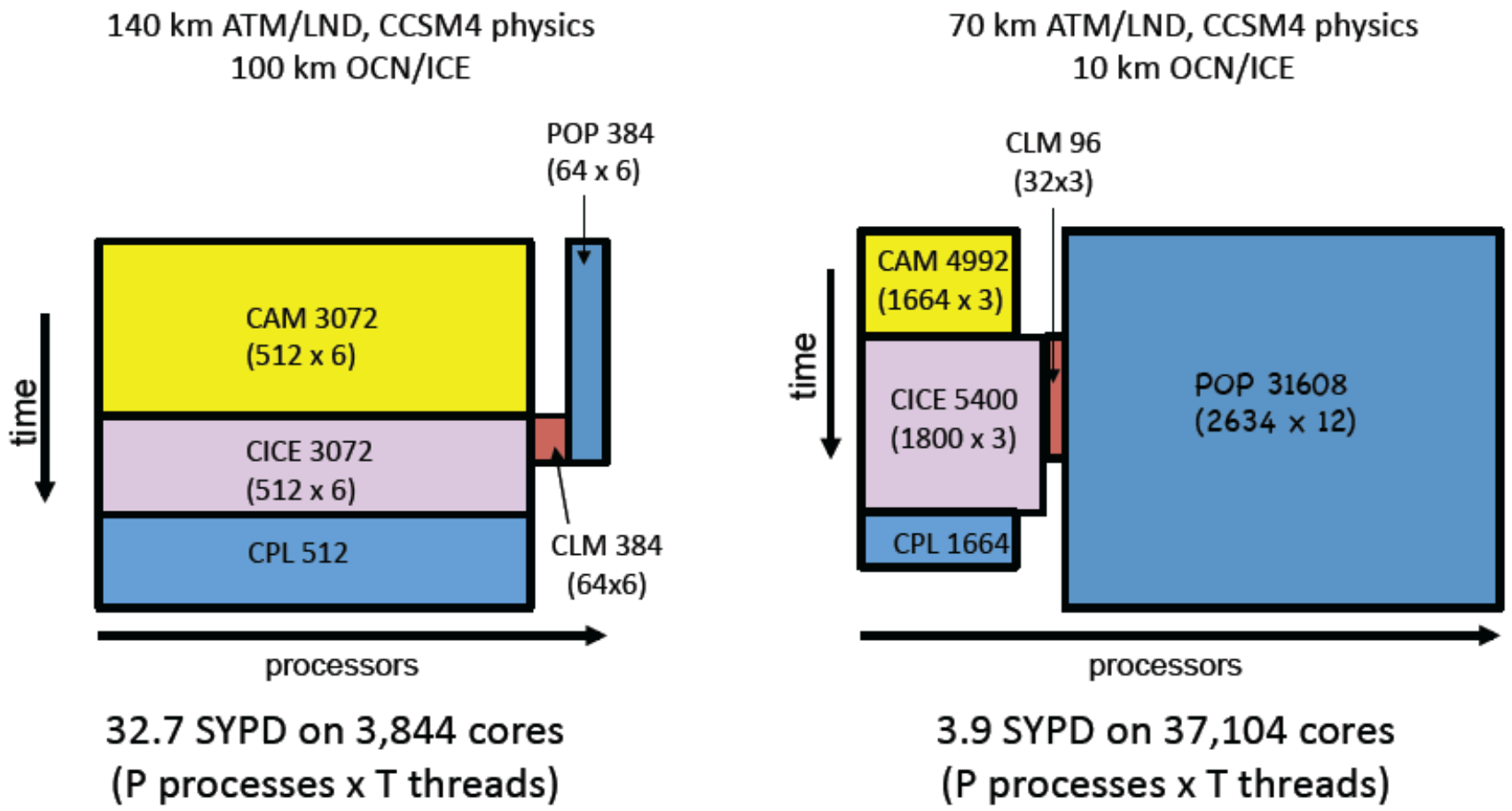


Figure courtesy of M. Vertenstein (NCAR)

CESM Simulations on a Cray Supercomputer



32.7 SYPD on 3,844 cores
(P processes x T threads)

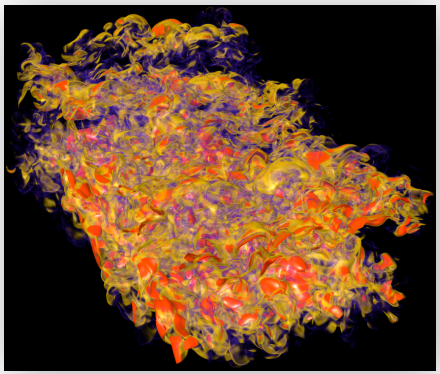
3.9 SYPD on 37,104 cores
(P processes x T threads)

Performance Limiters: Left is CAM; Right is POP.

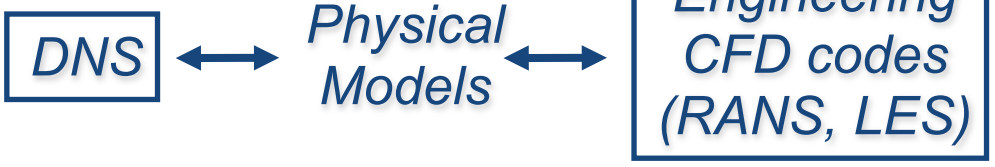
Figure courtesy of Pat Worley (ORNL)

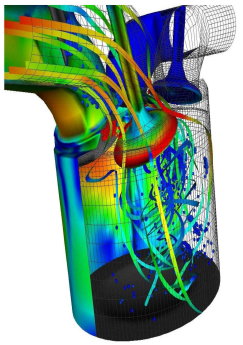
Simulating Turbulent Reacting Flows: S3D

- **Direct numerical simulation (DNS) of turbulent combustion**
  - —state-of-the-art code developed at CRF/Sandia
    - PI: Jaqueline H. Chen, SNL
  - —2014: Cray XK7 (106M hours)
  - —"DNS Turbulent Combustion Towards Flexible Fuel Gas Turbines and IC Engines"



- **Science**
  - —study micro-physics of turbulent reacting flows
    - physical insight into chemistry turbulence interactions
  - —simulate detailed chemistry; multi-physics (sprays, radiation, soot)
  - —develop and validate reduced model descriptions used in macro-scale simulations of engineering-level systems





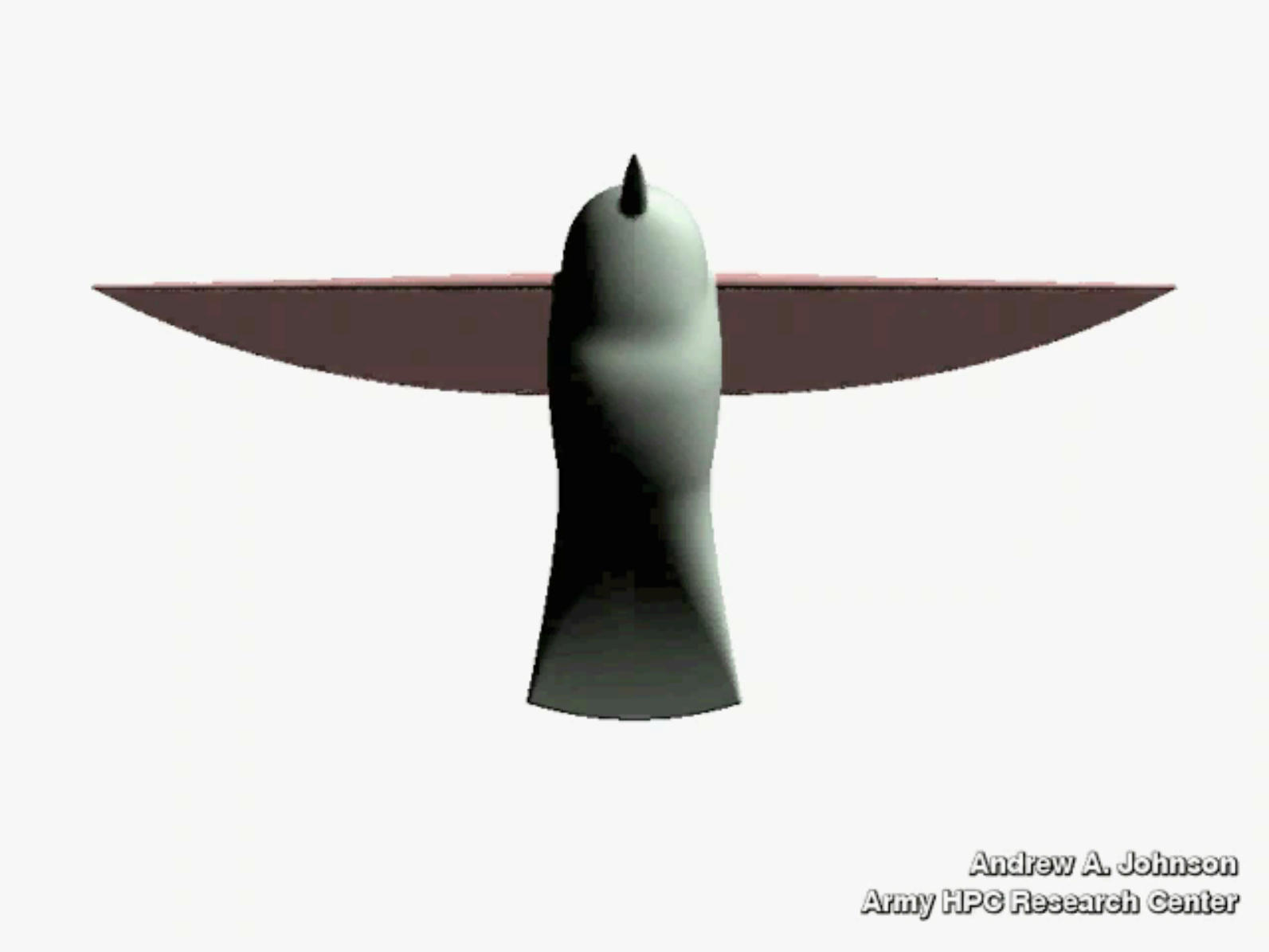
Text and figures courtesy of Jacqueline H. Chen, SNL

Fluid-Structure Interactions

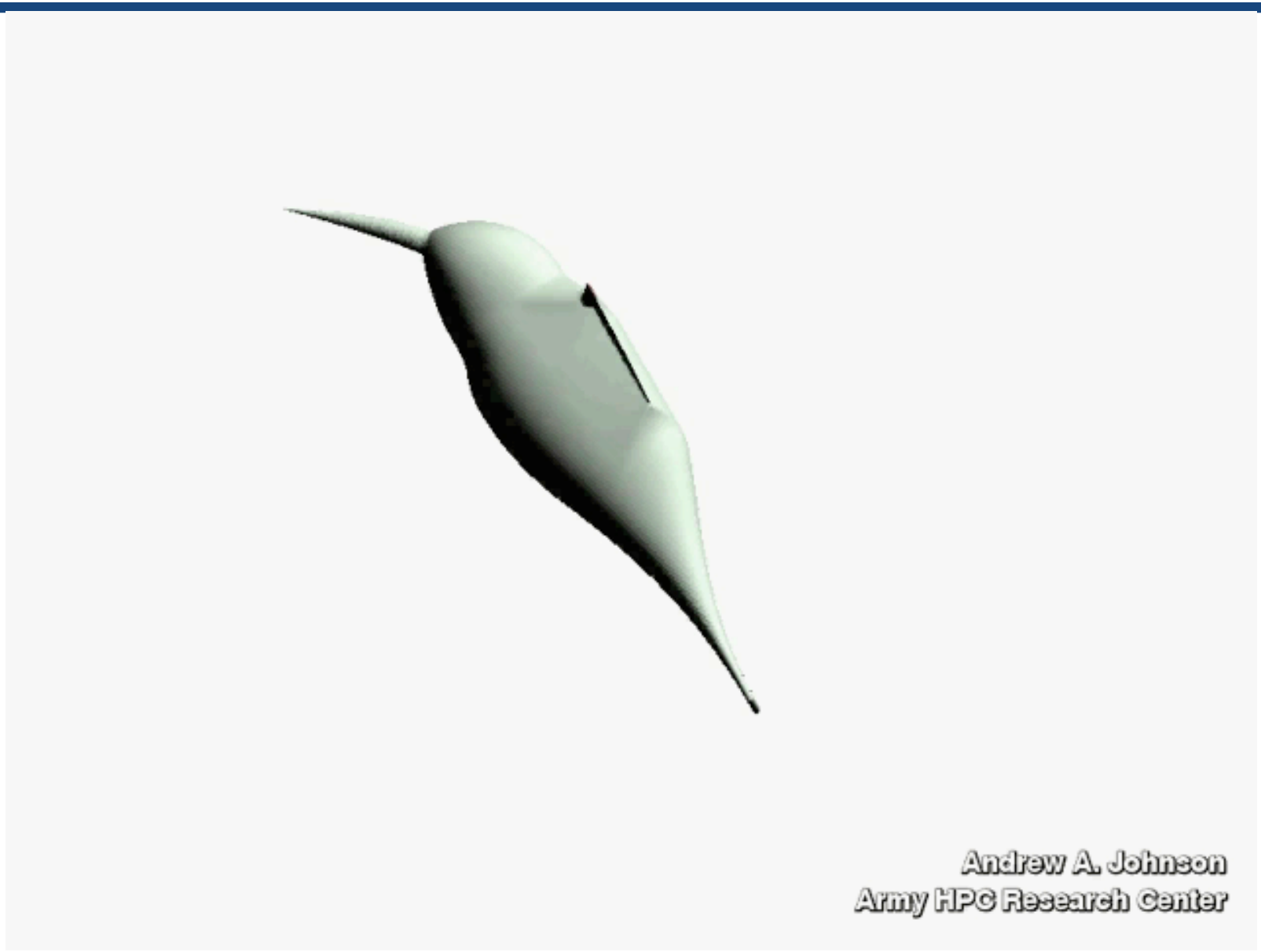
- **Simulate …**
  - —rotational geometries (e.g. engines, pumps), flapping wings
- **Traditionally, such simulations have used a fixed mesh**
  - —drawback: solution quality is only as good as initial mesh
- **Dynamic mesh computational fluid dynamics**
  - —integrate automatic mesh generation within parallel flow solver
    - nodes added in response to user-specified refinement criteria
    - nodes deleted when no longer needed
    - element connectivity changes to maintain minimum energy mesh
  - —mesh changes continuously as geometry + solution changes
- **Example: 3D simulation of a hummingbird's flight**

[Andrew Johnson, AHPCRC 2005]

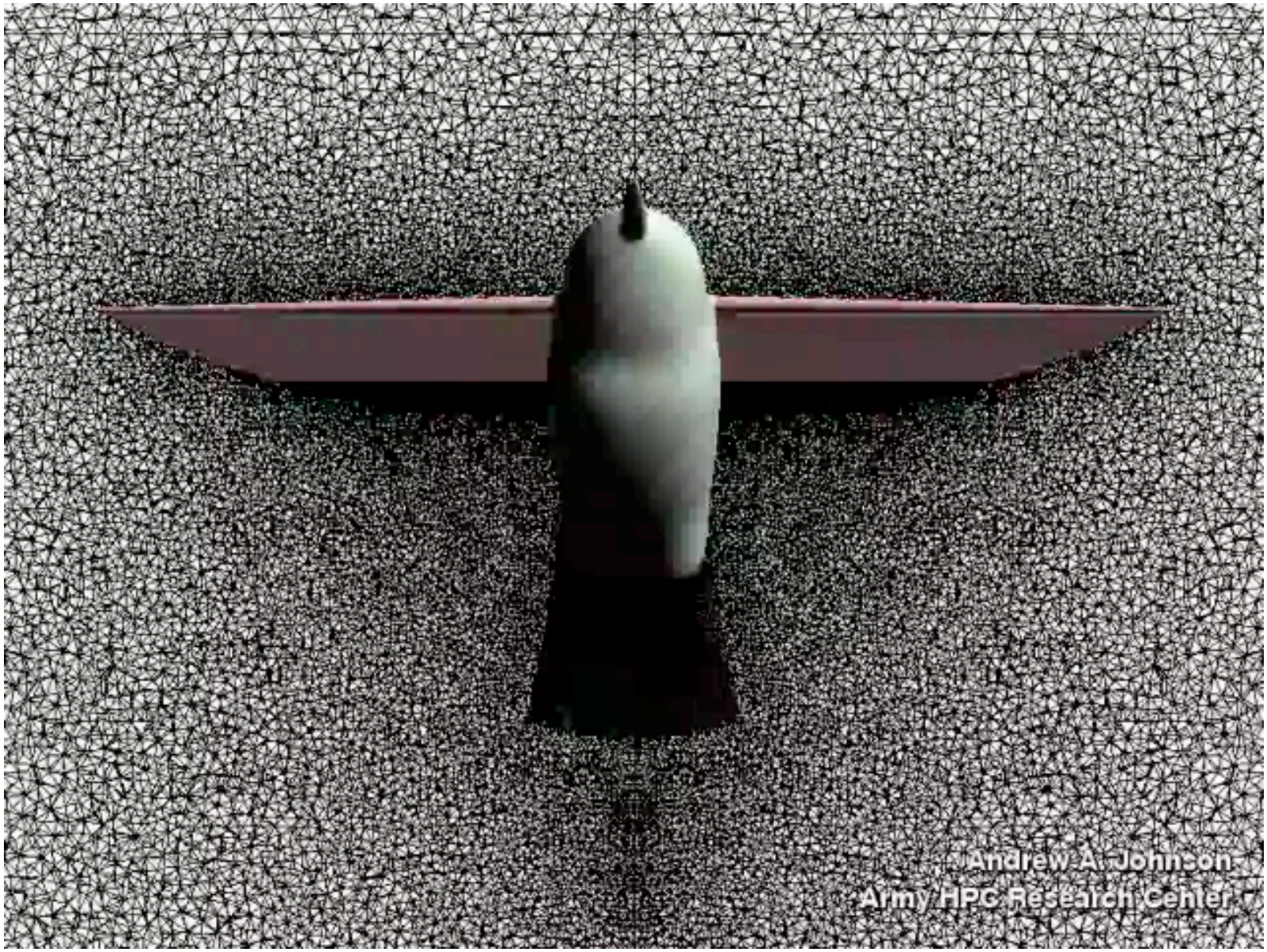
Air Velocity (Front)



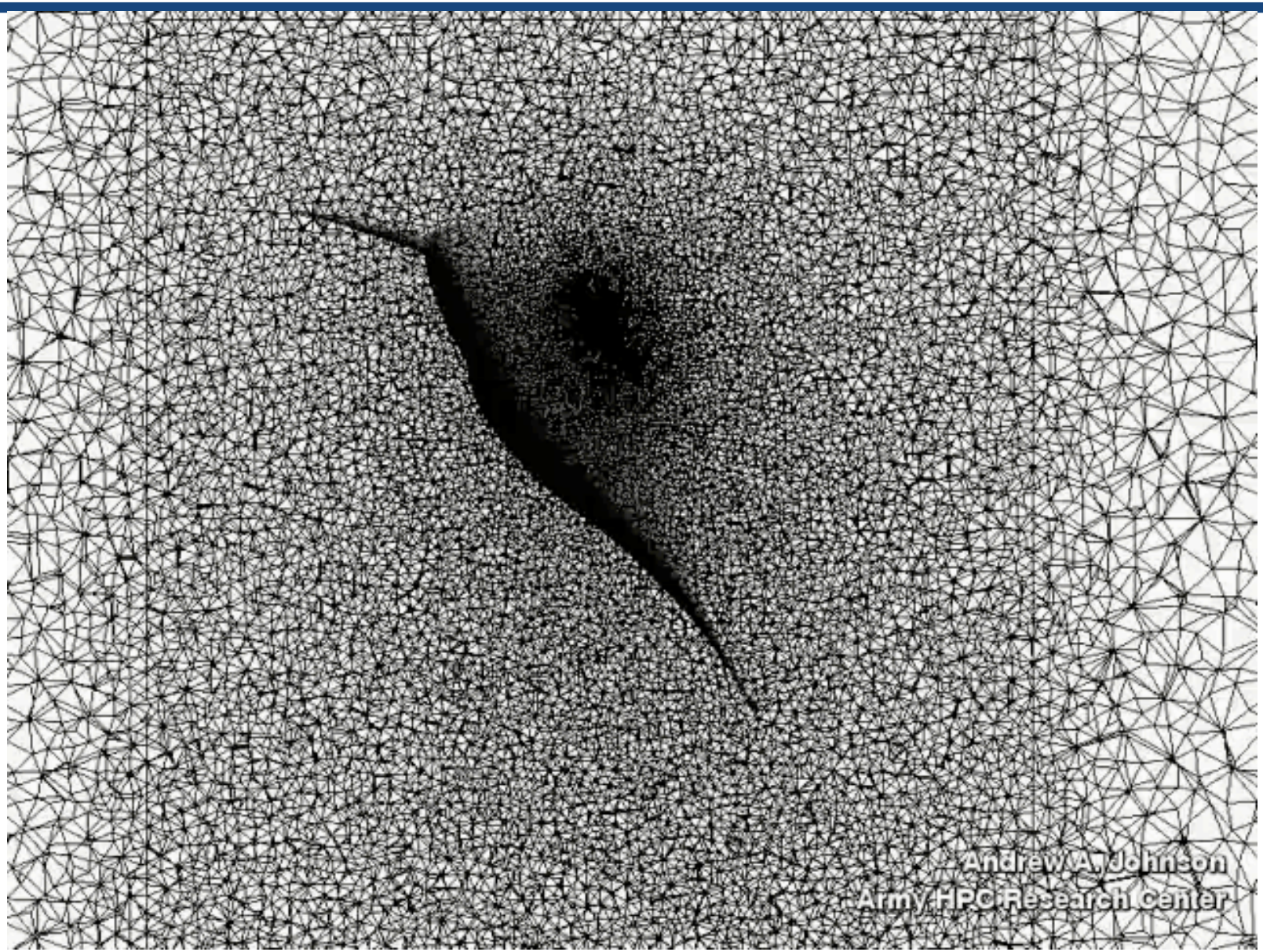
Air Velocity (Side)



Mesh Adaptation (front)



Mesh Adaptation (side)



Challenges of Explicit Parallelism

- **Algorithm development is harder**
  - —complexity of specifying and coordinating concurrent activities
- **Software development is much harder**
  - —lack of standardized & effective development tools and programming models
  - —subtle program errors: race conditions
- **Rapid pace of change in computer system architecture**
  - —a great parallel algorithm for one machine may not be suitable for another
    - – example: homogeneous multicore processors vs. GPUs

Hummingbird Simulation in UPC

- **UPC: PGAS language for scalable parallel systems**
- **Application overview**
  - —distribute mesh among the processors
  - —partition the mesh using recursive bisection
  - —each PE maintains and controls its piece of the mesh
    - has a list of nodes, faces, and elements
  - —communication and synchronization
    - read-from or write-to other processor's data elements as required
    - processors frequently synchronize using barriers
    - use "broadcast" and "reduction" patterns
  - —constraint
    - only 1 processor may change the mesh at a time

Algorithm Sketch

At each time step…

- **Test if re-partitioning is required**
- **Set up interprocessor communication if mesh changed**
- **Split elements into independent (vectorizable) groups**
- **Calculate the refinement value at each mesh node**
- **Move the mesh**
- **Solve the coupled fluid-flow equation system**
- **Update the mesh to ensure mesh quality**
  - **—swap element faces to obtain a “Delaunay” mesh**
  - **—add nodes to locations where there are not enough**
  - **—delete nodes from locations where there are too many**
  - **—swap element faces to obtain a “Delaunay” mesh**

Parallel Hardware in the Large

Blue Gene/Q Supercomputer

- **Cores with pipelining and short vectors**
- **Multicore processors**
- **Scalable parallel system**

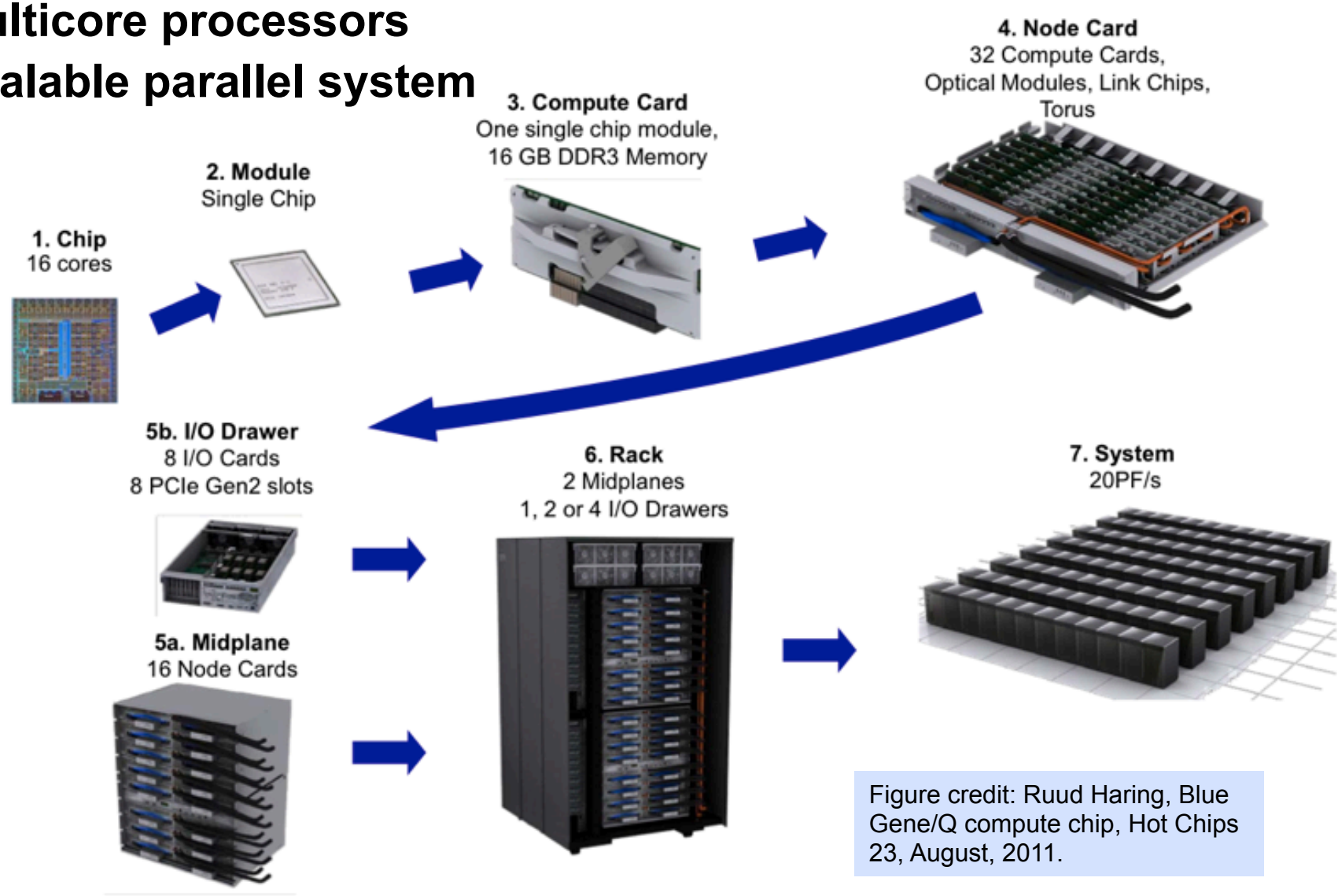


Figure credit: Ruud Haring, Blue Gene/Q compute chip, Hot Chips 23, August, 2011.

Historical Concurrency in Top 500 Systems

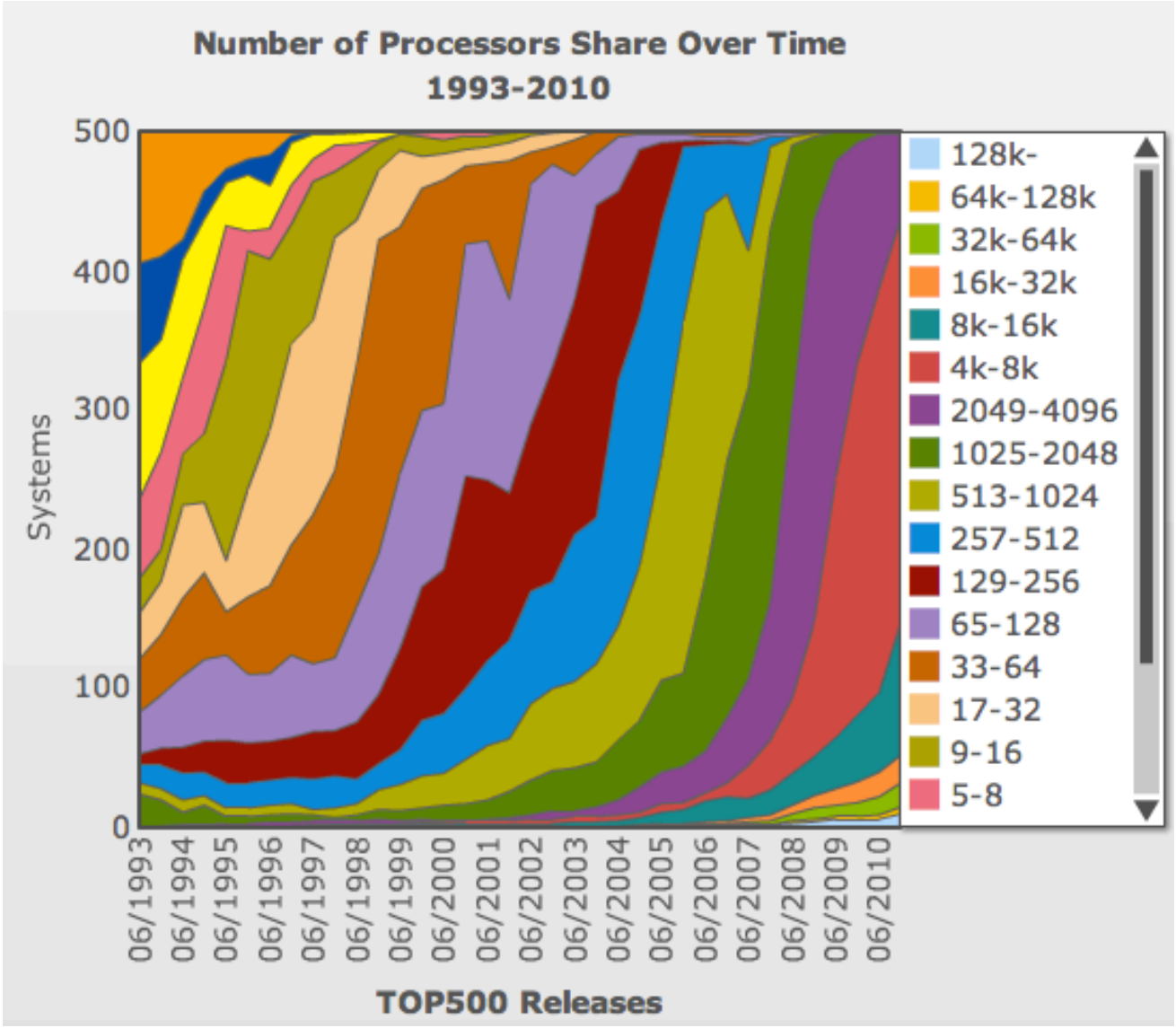


Image credit: http://www.top500.org/overtime/list/36/procclass

Scale of the Largest HPC Systems (Nov 2014)

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,659.9
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393,216	4,293.3	5,033.2	1,972
10	Government United States	Cray CS-Storm, Intel Xeon E5-2660v2 10C 2.2GHz, Infiniband FDR, Nvidia K40 Cray Inc.	72,800	3,577.0	6,131.8	1,498.9

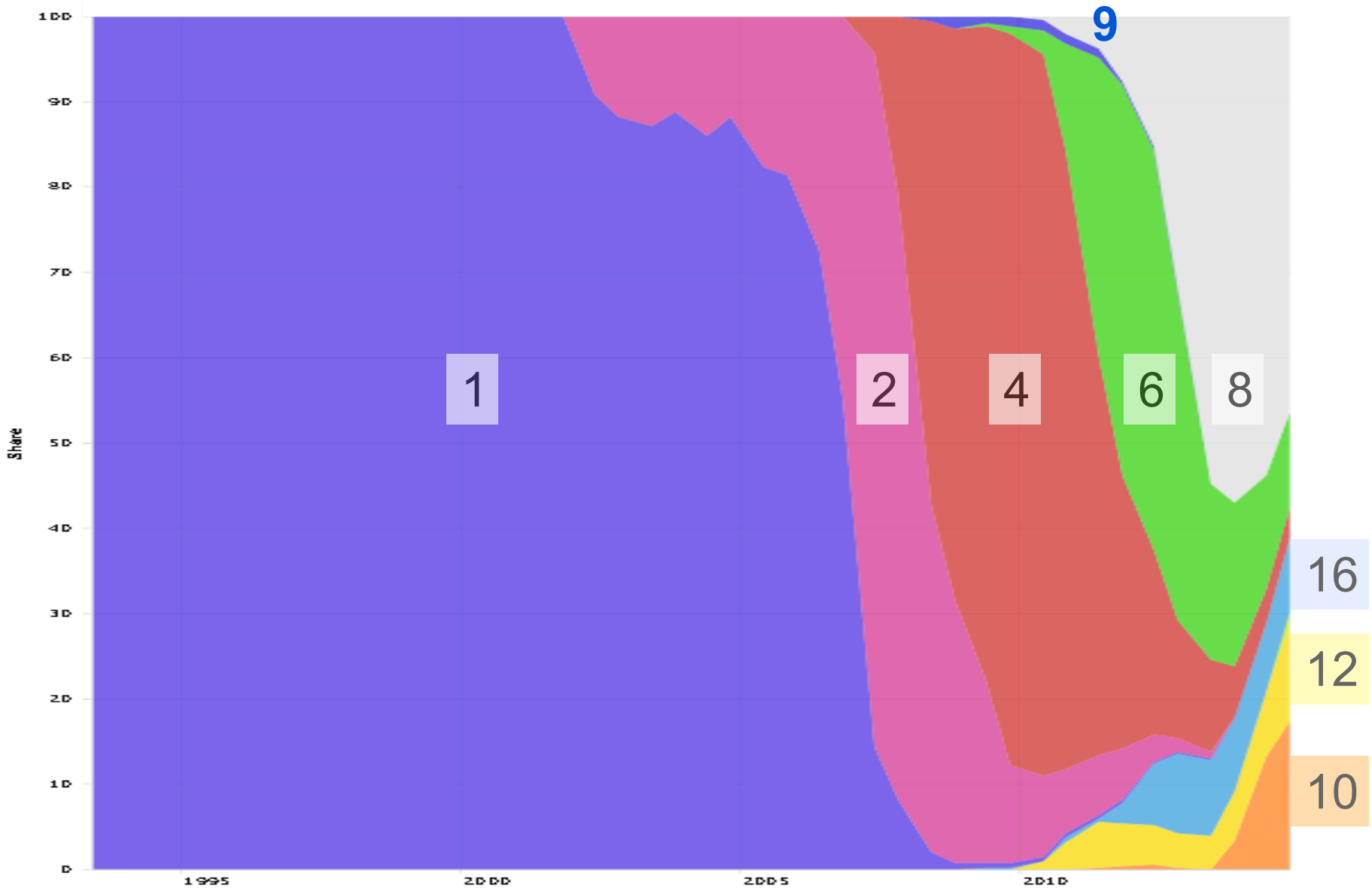
> 1.5M cores

top 9 > 100K cores

hybrid CPU+GPU

hybrid CPU+manycore

Cores per Socket (System Share, Nov 2014)



Achieving High Performance on Parallel Systems

Computation is only part of the picture

- **Memory latency and bandwidth**
  - — CPU rates have improved 4x as fast as memory over last decade
  - — bridge speed gap using memory hierarchy
  - — multicore exacerbates demand
- **Interprocessor communication**
- **Input/output**
  - — I/O bandwidth to disk typically needs to grow linearly with the # processors

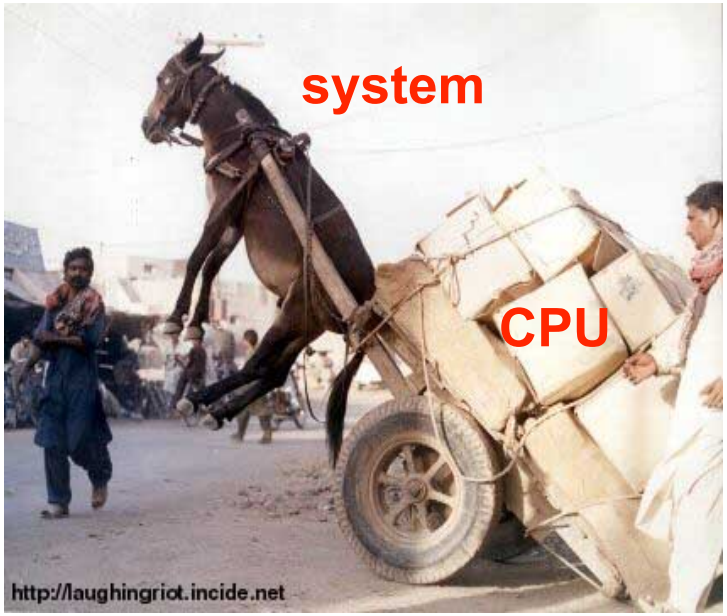


Image Credit: Bob Colwell, ISCA 1995

Challenges of Parallelism in the Large

- Parallel science applications are often very sophisticated
  - e.g. adaptive algorithms may require dynamic load balancing
- Multilevel parallelism is difficult to manage
- Extreme scale exacerbates inefficiencies
  - algorithmic scalability losses
  - serialization and load imbalance
  - communication or I/O bottlenecks
  - insufficient or inefficient parallelization
- Hard to achieve top performance even on individual nodes
  - contention for shared memory bandwidth
  - memory hierarchy utilization on multicore processors

Thursday's Class

- Introduction to parallel algorithms
  - —tasks and decomposition
  - —task dependences and critical path
  - —mapping tasks
- Decomposition techniques
  - —recursive decomposition
  - —data decomposition

Parallel Machines for the Course

- **STIC**
  - —170 nodes, each with two 4-core Intel Nehalem processors
  - —Infiniband interconnection network
  - —no global shared memory
- **Biou**
  - —48 nodes, each with four 8-core IBM Power7 processors
    - – 4-way SMT (4 hardware threads per processor core); 256GB/node
  - —Infiniband interconnection network
  - —no global shared memory
- **DAVinCI**
  - —192 nodes, each with two 6-core Intel Westmere processors
  - —16 nodes equipped with NVIDIA FERMI GPUs
  - —Infiniband interconnection network
  - —no global shared memory