COMP 422/534 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, 713-348-5179**
  - **—Office Hours: Thursday 9am-10am or by appointment**
- **WWW site: http://www.clear.rice.edu/comp422**

Parallelism

- Definition: ability to execute parts of a computation concurrently
- Goal: solve large problems fast
  - —with more parallelism
    - solve larger problems in the same time
    - solve a fixed size problem in shorter time
- Grain of parallelism: how big are the units?
  - —bits, instructions, blocks, loop iterations, procedures, …
- COMP 422/534 focus: explicit thread-level parallelism
  - —thread = a unit of execution consisting of a sequence of instructions that is managed by either the operating system or a runtime system

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

Difference Between 422 and 534

- COMP 422 assignments
  - —produce well-written parallel programs
  - —examine their scalability and performance
  - —write a report about how each program works and your observations about its scalability and performance
- COMP 534 assignments
  - —same assignments as 422 students with an added component
    - use a performance tool to analyze where a program spends its time and how each program component scales
  - —performance and scalability of submitted programs counts for a larger fraction of the grade

Recommended Books

- **Introduction to Parallel Computing, 2nd Ed, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar (2003)**
- **Using OpenMP: Portable Shared Memory Parallel Programming - Barbara Chapman, Gabriele Jost, Ruud van der Pas (2008)**
- **Using MPI: Portable Parallel Programming with the Message-Passing Interface, 3rd Ed - William Gropp, Ewing Lusk, Anthony Skjellum (2014)**
- **Programming Massively Parallel Processors: A Hands-on Approach, 3rd Ed. - David B. Kirk, Wen-mei W. Hwu (2016)**

Topics (Part 1)

- Introduction
- Principles of parallel algorithm design
  - —decomposition techniques
  - —mapping & scheduling computation
  - —templates
- Programming shared-address space systems
  - —Cilk Plus
  - —OpenMP
  - —Pthreads
  - —synchronization
- Parallel computer architectures
  - —shared memory systems and cache coherence
  - —distributed-memory systems
  - —interconnection networks and routing

Topics (Part 2)

- Programming scalable systems
  - —message passing: MPI
  - —global address space languages
- Collective communication
- Analytical modeling of program performance
  - —speedup, efficiency, scalability, cost optimality, isoefficiency
- Parallel algorithms
  - —non-numerical algorithms: sorting, graphs
  - —numerical algorithms: dense and sparse matrix algorithms
- Performance measurement and analysis of parallel programs
- GPU Programming with CUDA
- Problem solving on clusters using MapReduce
- Warehouse-scale computing

Prerequisites

- Programming in C, C++, or similar
- Basics of data structures
- Basics of machine architecture
- Prerequisites
  - —COMP 321 (formerly 221) INTRO TO COMPUTER SYSTEMS
  - —or equivalent
- See me if you have concerns

Rules

- Generally, use Piazza for class communication (see syllabus)
- If you send me email
  - —subject line must include COMP 422/534
  - —send it from your Rice email address

- Don't share code for the assignments
  - —you may not share code with classmates
  - —you may not collaborate with people who are not your classmates or instructor in any way
    - e.g., don't post questions to programming forums
  - —you may not take more than two lines of code from an external resource and include it in one of your assignments
  - —we use automated tools to identify violations
    - don't underestimate their power!
  - —violations will be reported to the honor council
- See syllabus for full definition of misconduct

Motivations for Parallel Computing

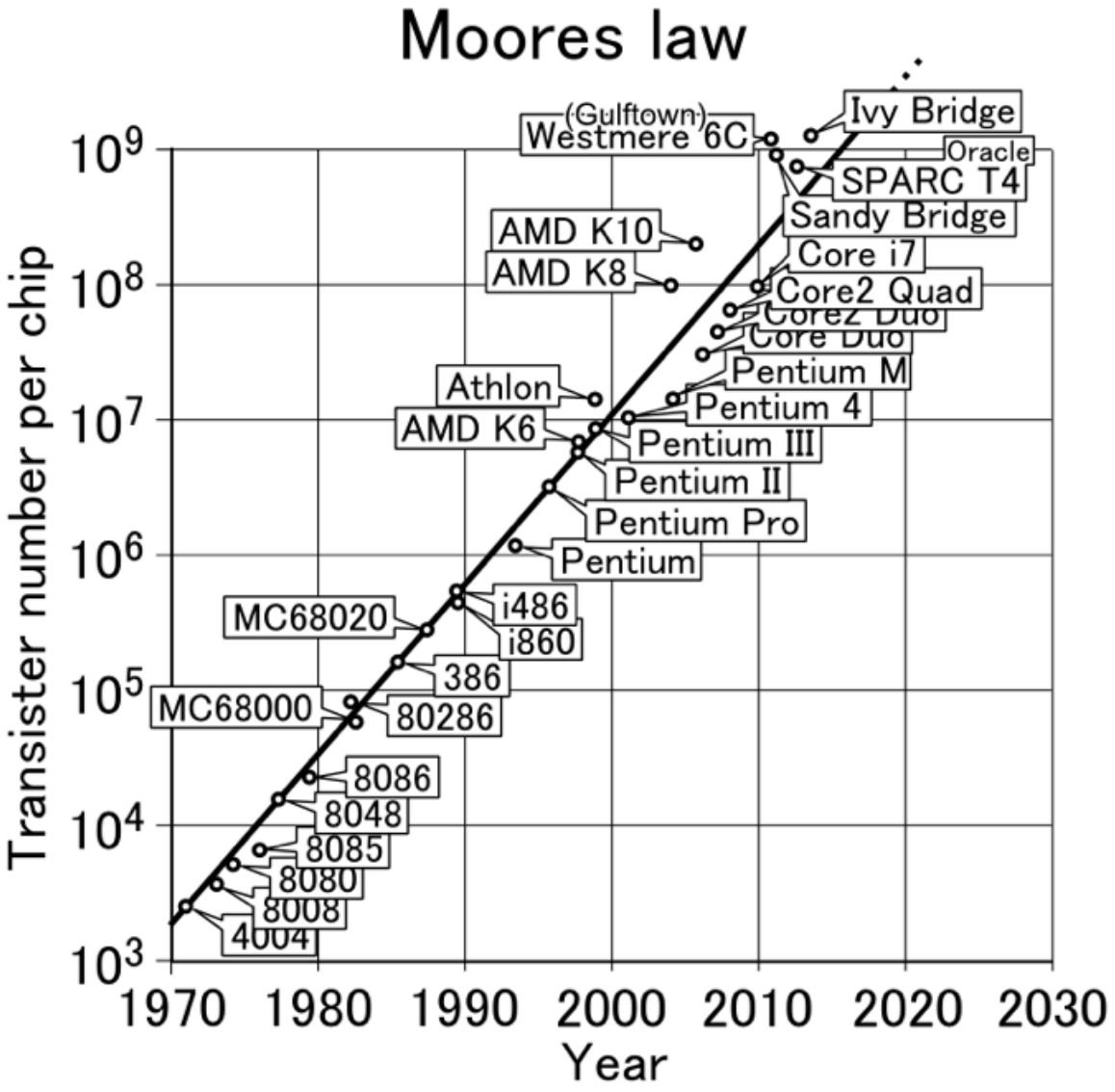
- **Technology push**
- **Application pull**

The Rise of Multicore Processors

Advance of Semiconductors: “Moore’s Law”

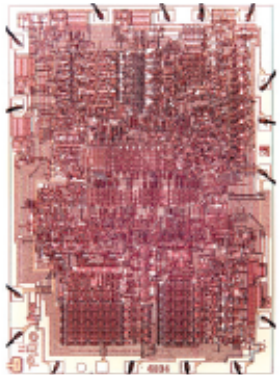
Gordon Moore, Founder of Intel

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

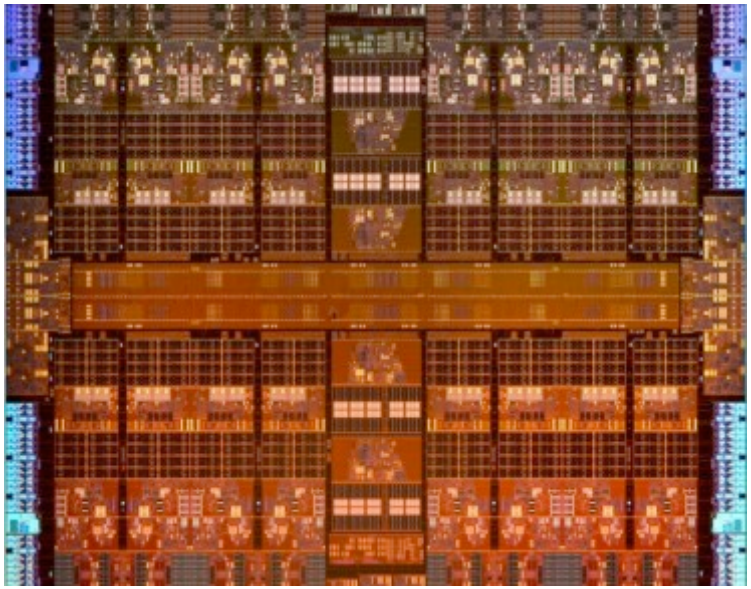


By shigeru23 CC BY-SA 3.0, via Wikimedia Commons

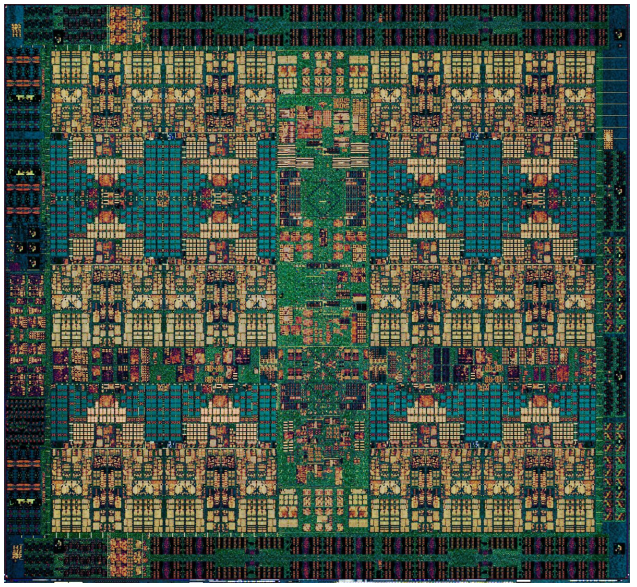
Evolution of Microprocessors 1971-2017



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



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



IBM Power9, 2017
24 cores, 120MB cache
8B 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.

https://en.wikichip.org/wiki/ibm/microarchitectures/power9

(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
    - long instruction words (VLIW)
      - bundles of independent instructions that can be issued together
        - e.g., Intel Itanium processor 2000-2017
    - SIMD processor extensions up to 512 bits wide (AVX512)
      - integer, floating point, complex data
      - operations on up to 16 32-bit data items per instruction

Microprocessor Architecture (Mid 90's)

- **Superscalar (SS) designs were the state of the art**
  - —multiple functional units (e.g., int, float, 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

Trouble on the Horizon

Increasing issue width provides diminishing returns

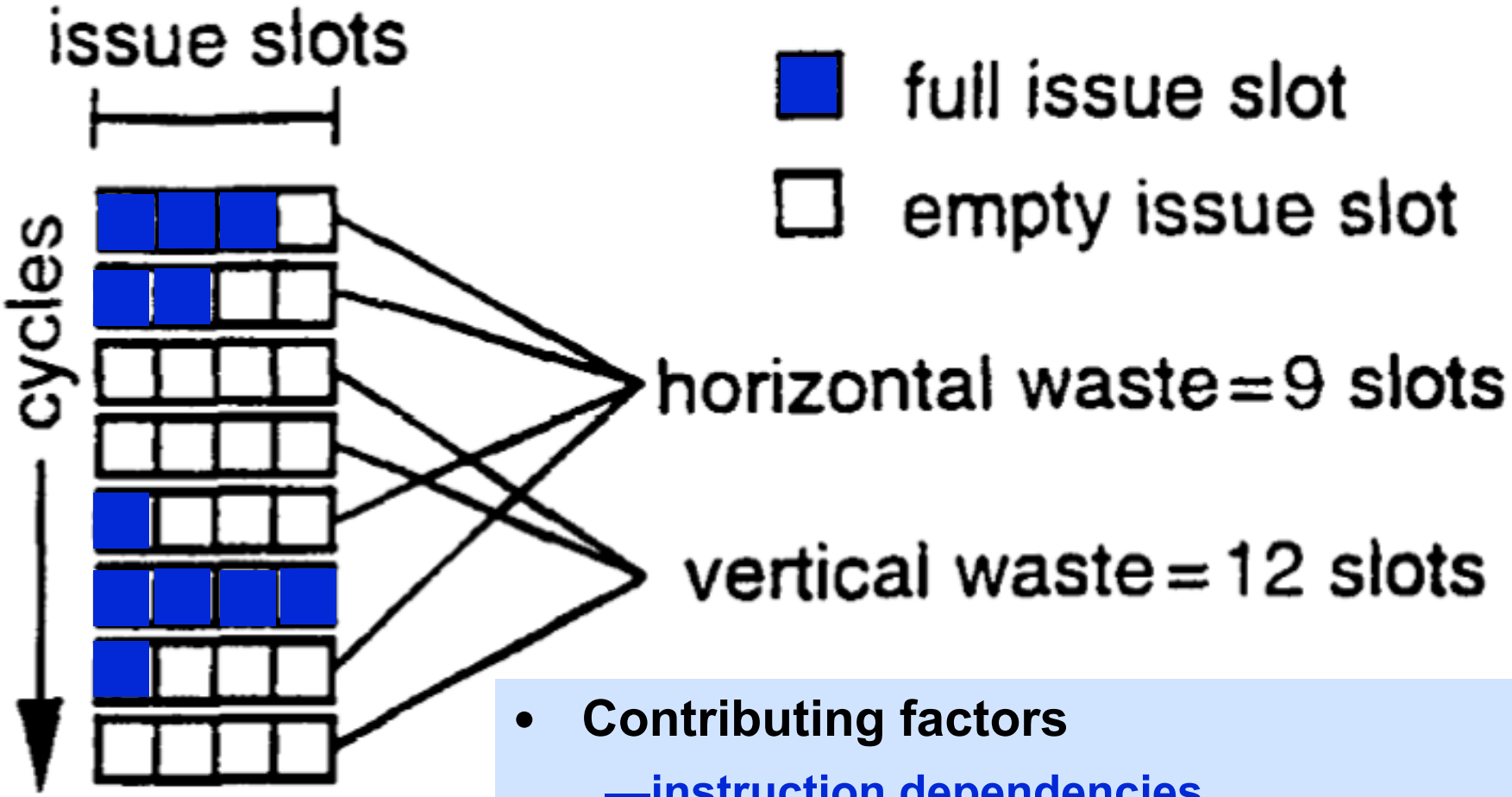
Two factors[1]

- **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 programs 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

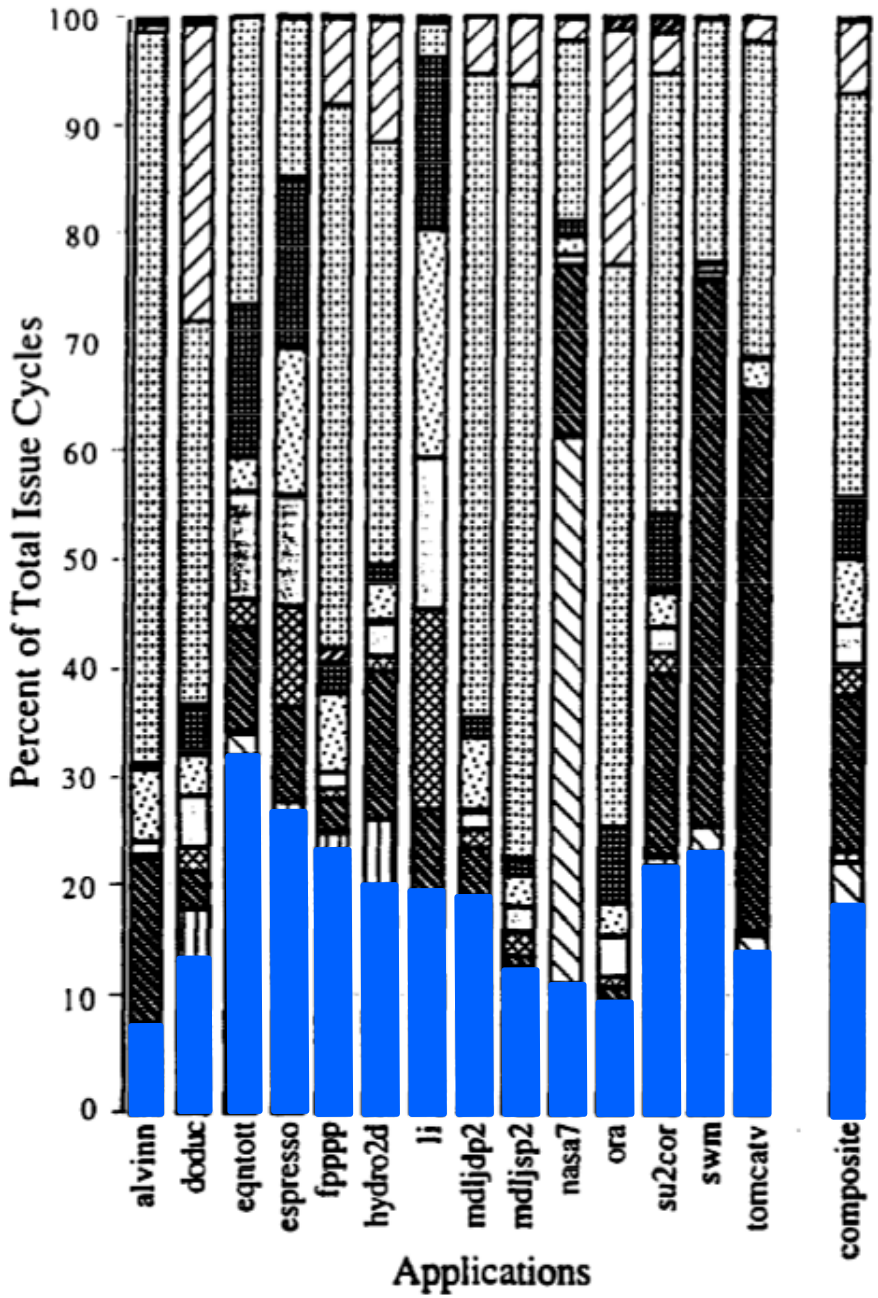
- Branch misprediction

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%

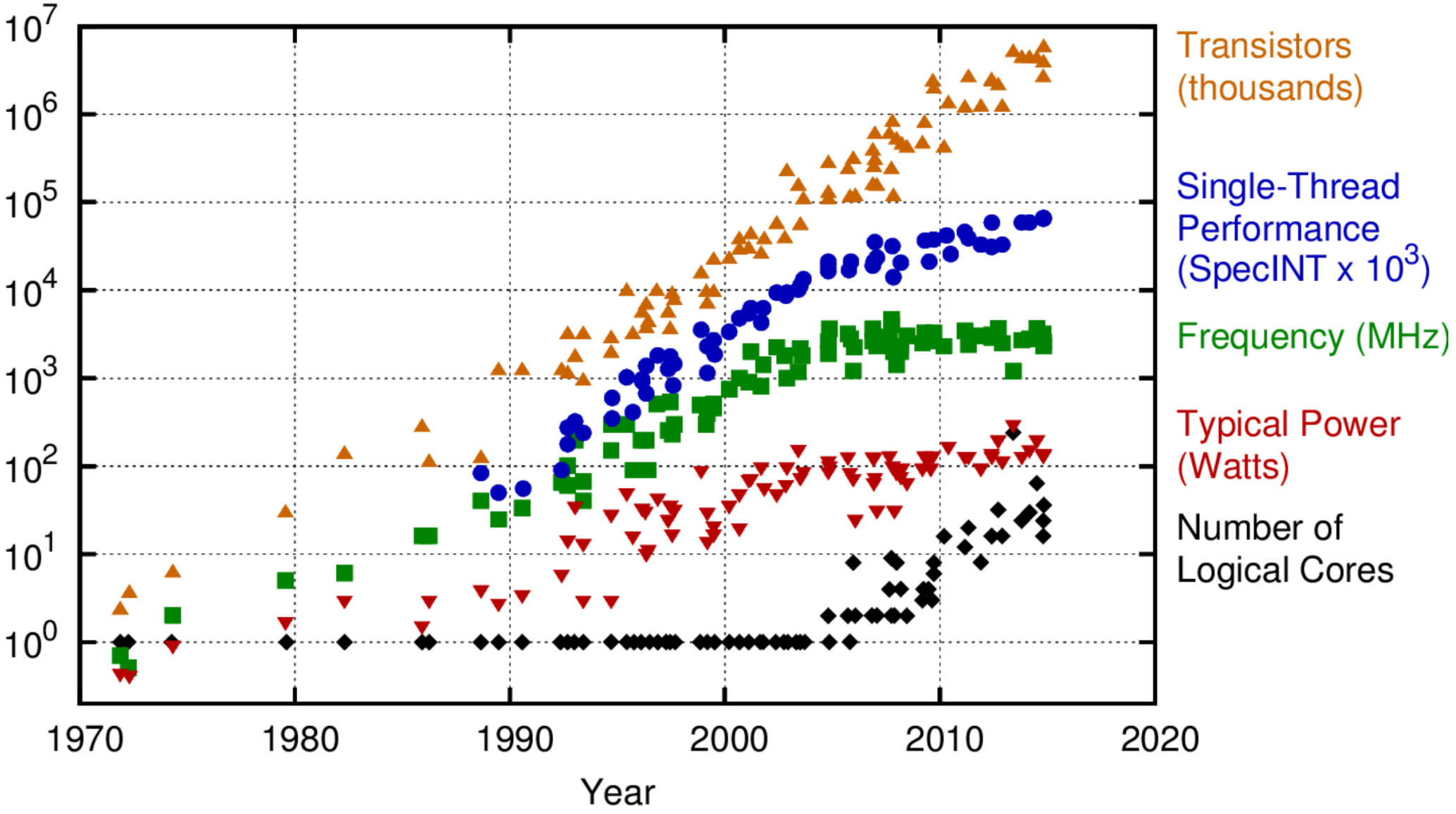
Power and Heat Stall Clock Frequencies

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*…

New York Times

Technology Trends

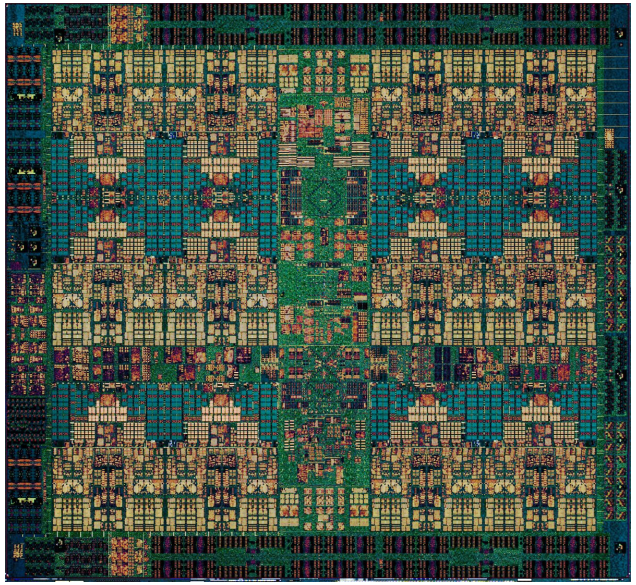


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Figure credit: Karl Rupp. https://www.karlrupp.net/wp-content/uploads/2015/06/40-years-processor-trend.png.

Recent Multicore Processors

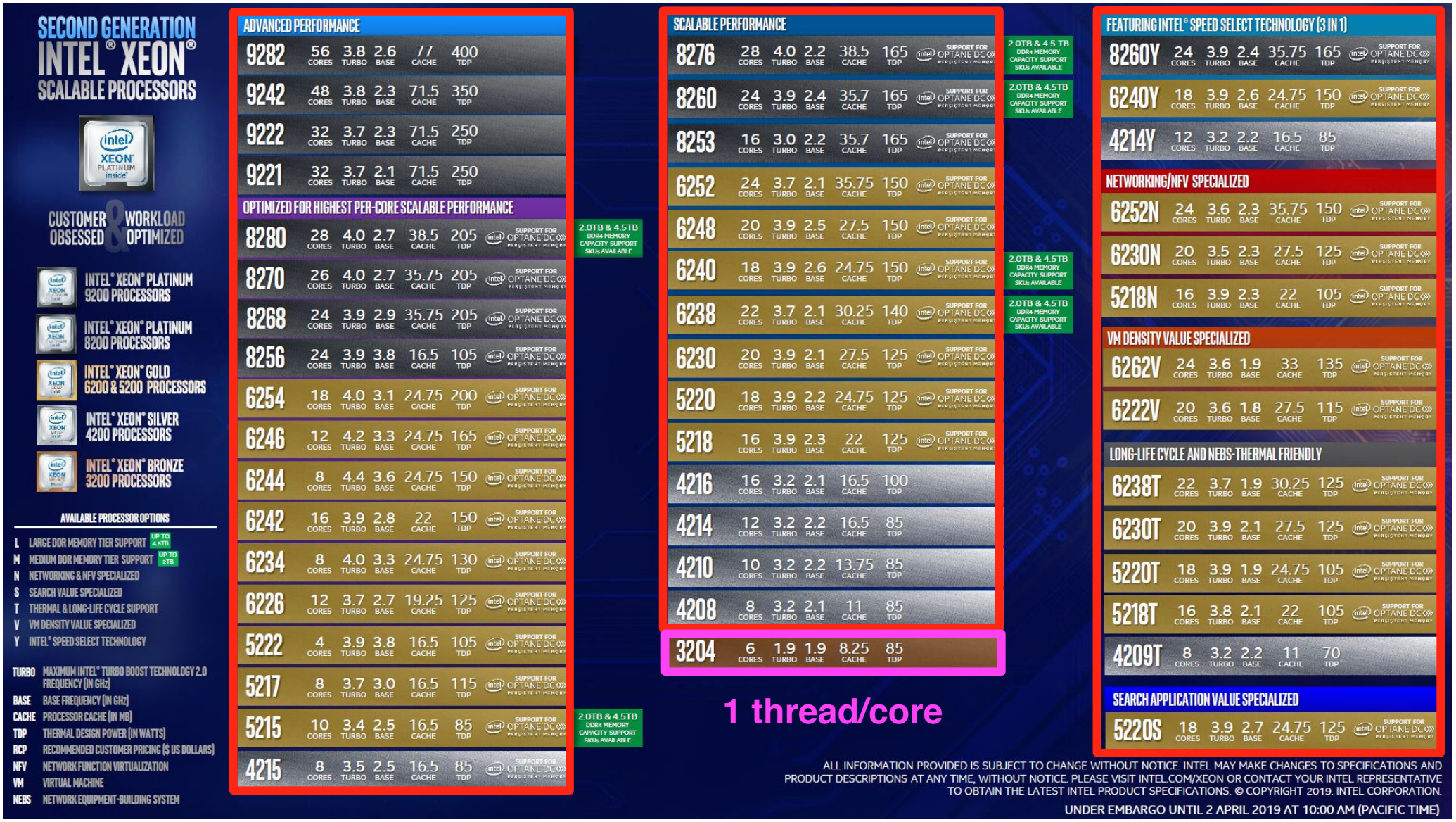
- 2019: AMD EPYC 7742
  - —64 cores; 2-way SMT; 256MB cache
- 2017: IBM Power9
  - —24 cores; 4-way SMT; 120MB cache
- 2016: Intel Knight's Landing
  - —72 cores; 4-way SMT; 36MB cache
- 2015: Oracle SPARC M7
  - —32 cores; 8-way SMT; 64MB cache
- Fall 14: Intel Haswell
  - —18 cores; 2-way SMT; 45MB cache
- June 14: IBM Power8
  - —12 cores; 8-way SMT; 96MB cache
- Sept 13: SPARC M6
  - —12 cores; 8-way SMT; 48MB cache
- May 12: AMD Trinity
  - —4 CPU cores; 384 graphics cores



IBM Power9

https://en.wikichip.org/wiki/ibm/microarchitectures/power9

Intel Scalable Processors (April 2, 2019)



2 SMT threads/core for all but one processor

Latency-optimized cores

Application Pull

- **Complex problems require computation on large-scale data**
- **Sufficient performance is 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"

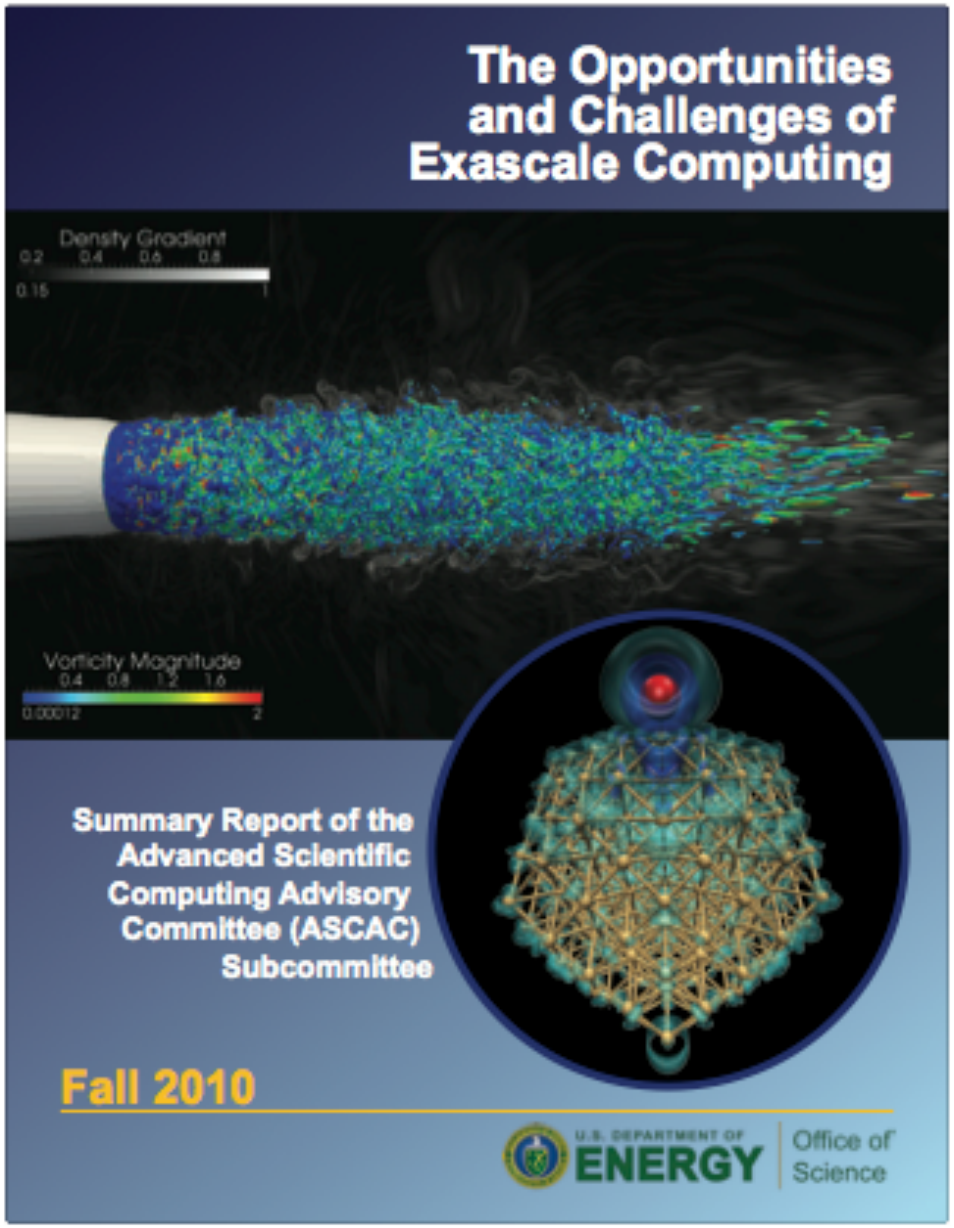
— "Scientific Discovery through Advanced Computing"
DOE Office of Science, 2000

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
  - —multiscale simulations of metal additive manufacturing processes
  - —understanding quantum properties of materials
  - —understanding reaction dynamics of heterogeneous catalysts
  - —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

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

March 11, 2011 Fukushima Daiichi Nuclear Power Plant suffered major damage from a 9.0 earthquake and subsequent tsunami that hit Japan.

The earthquake and tsunami disabled the reactor cooling systems, leading to radiation leaks and triggering a 30 km evacuation zone around the plant.

Confirmed deaths: 19,575 as of September 2017

Earthquake Research Institute, University of Tokyo

Tonankai-Tokai Earthquake Scenario

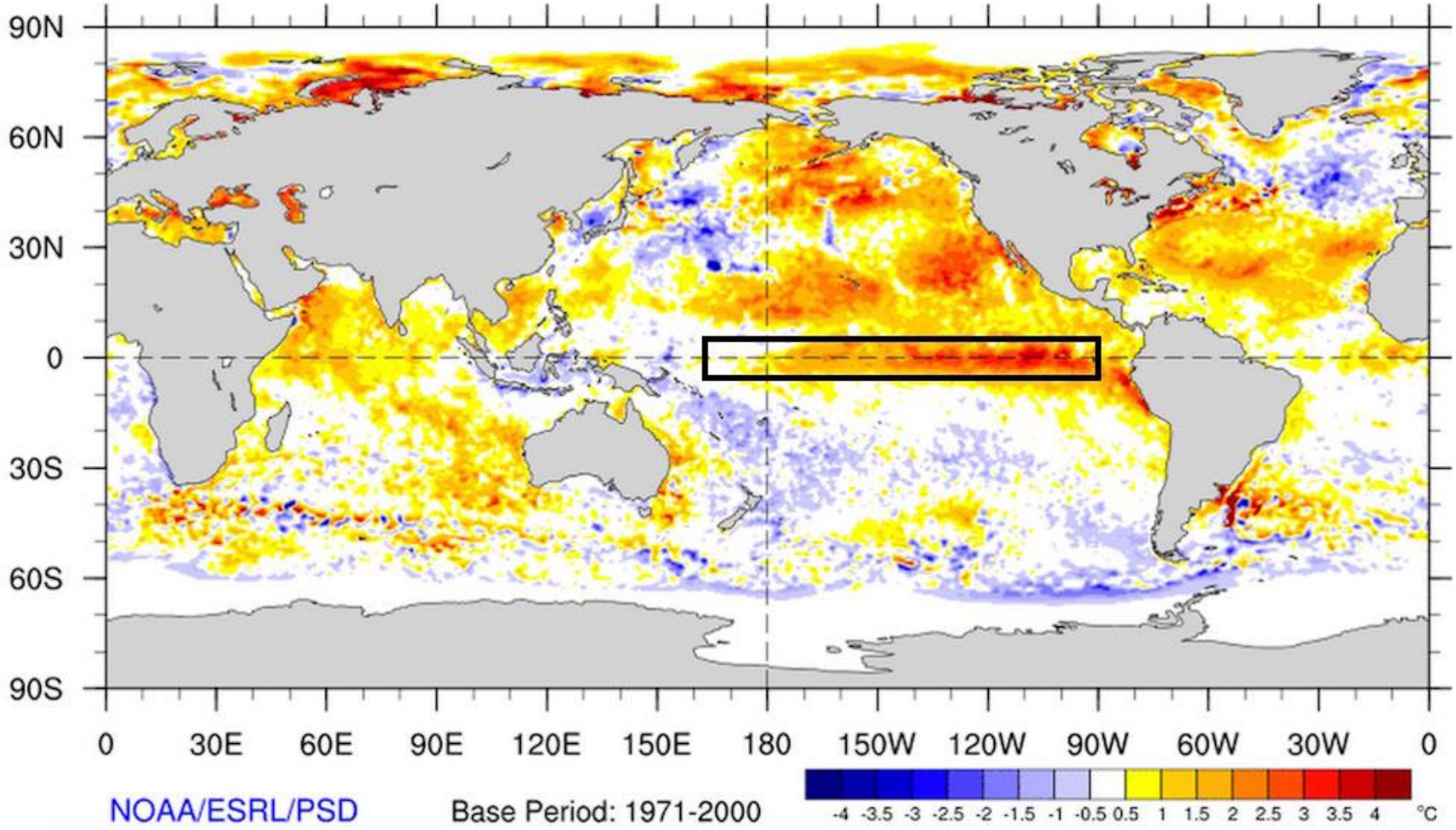
Video Credit: The Earth Simulator Art Gallery, CD-ROM, March 2004

Ocean Circulation Simulation

El Niño is an anomalous, yet periodic, warming of the central and eastern equatorial Pacific Ocean. For reasons still not well understood, every 2-7 years, this patch of ocean warms for six to 18 months

https://www.climate.gov/enso

El Niño was strong through the Northern Hemisphere winter 2015-16, with a transition to ENSO-neutral in May 2016.



Source: http://www.weather.com/storms/winter/news/january-march-outlook-2016-noaa-wsi

Community Earth System Model (CESM)

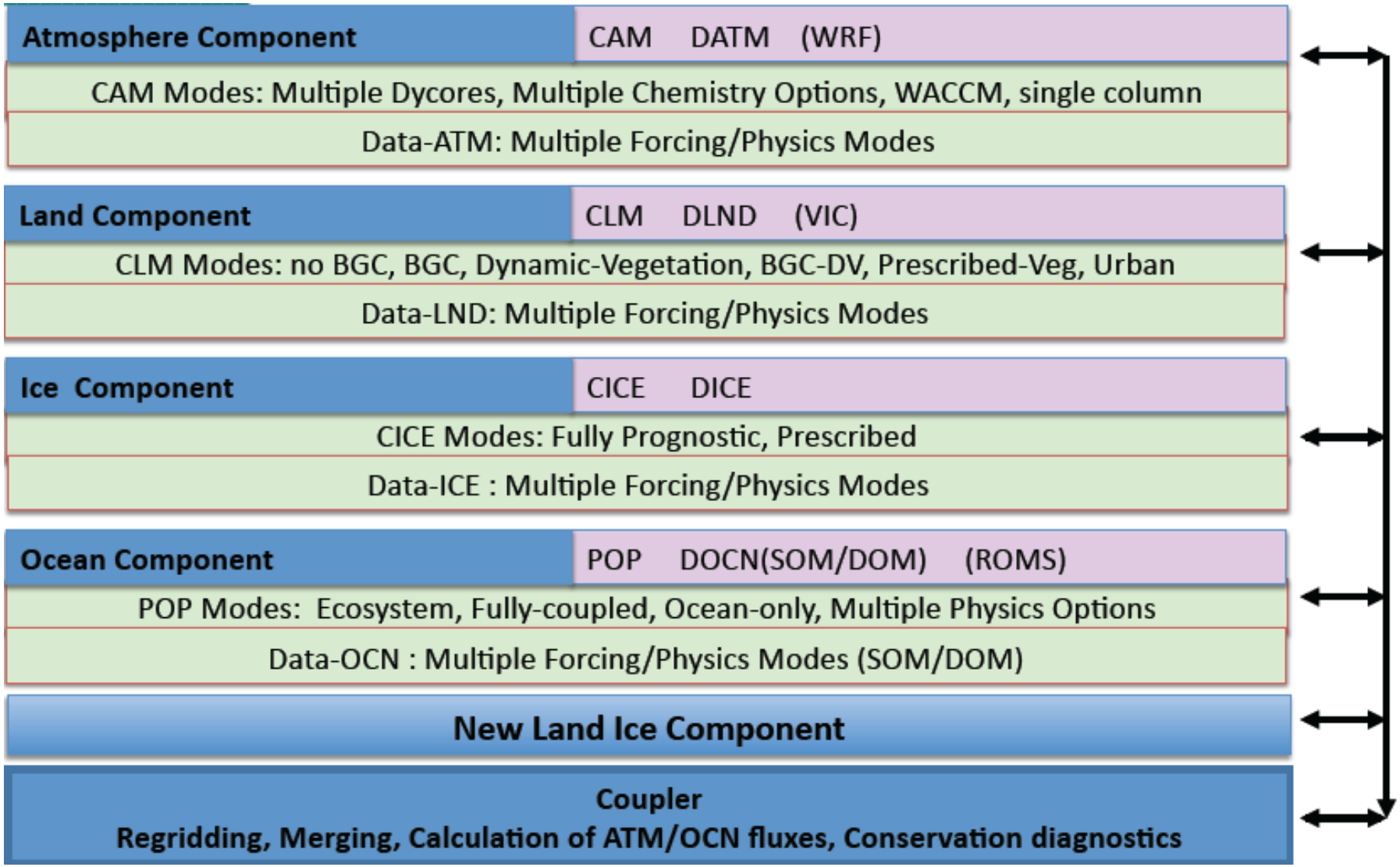


Figure courtesy of M. Vertenstein (NCAR)

CESM Execution Configurations

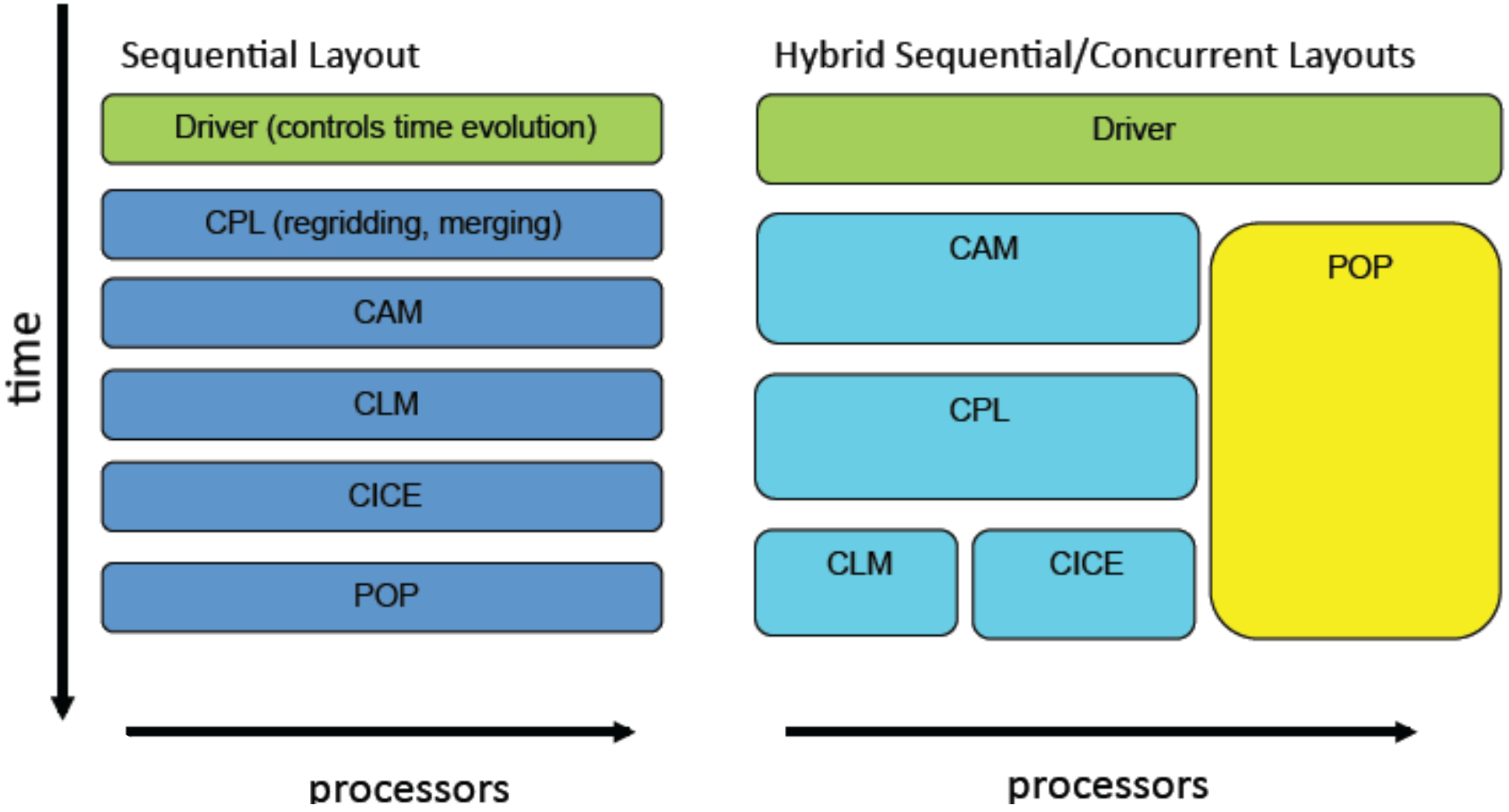
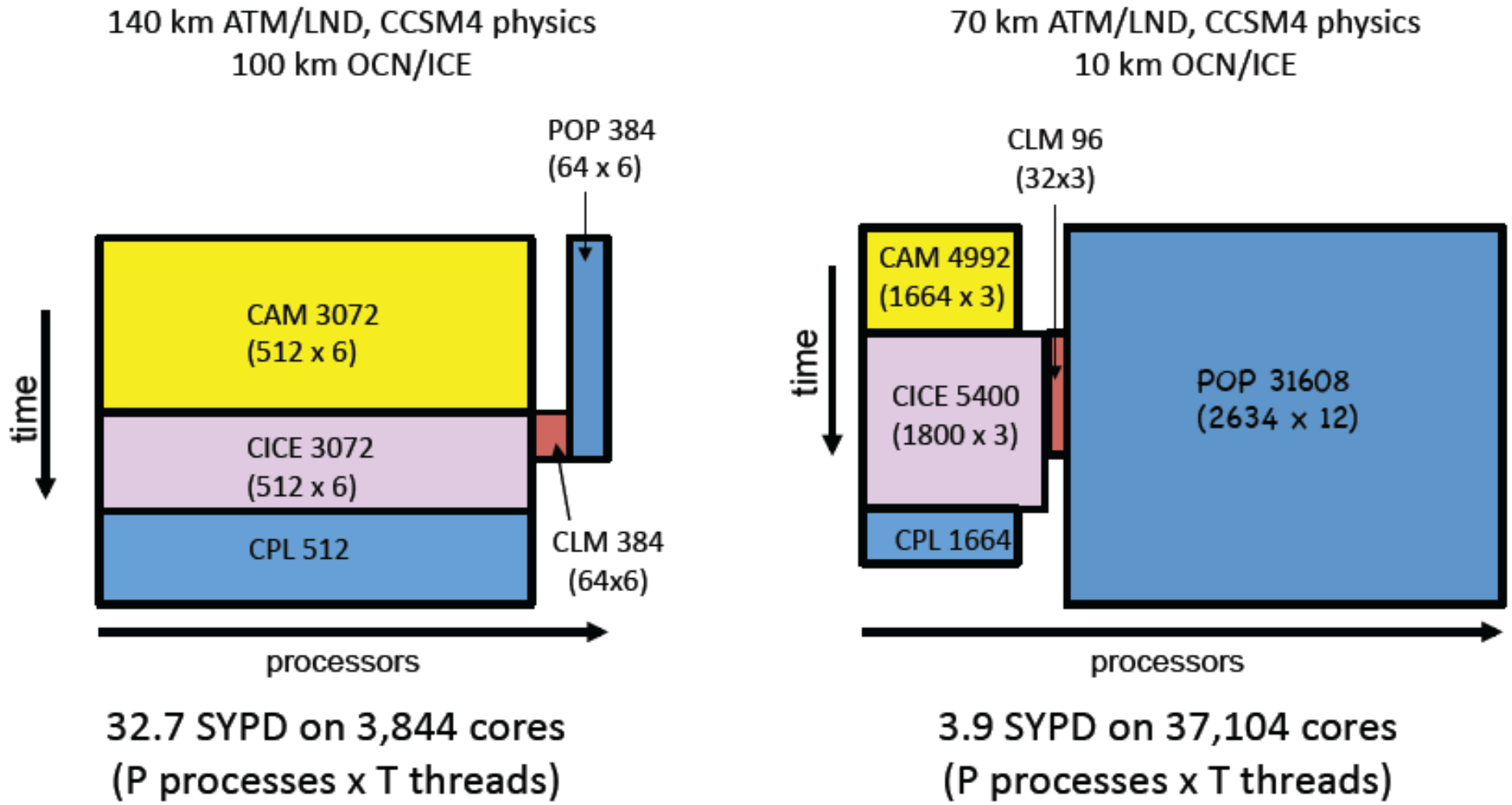


Figure courtesy of M. Vertenstein (NCAR)

CESM Simulations on a Cray Supercomputer

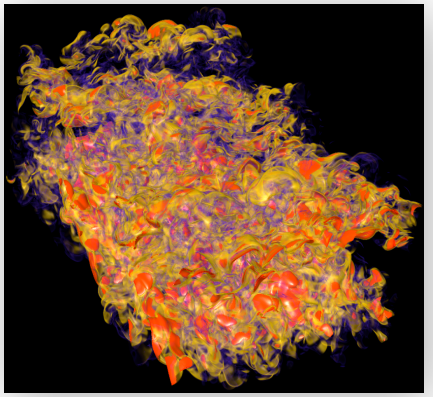


Performance Limiters: Left is CAM; Right is POP.

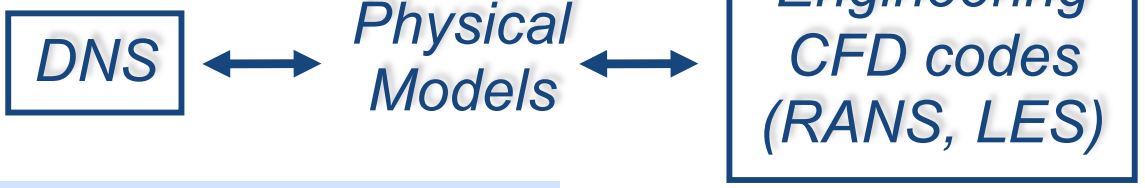
Figure courtesy of Pat Worley (ORNL)

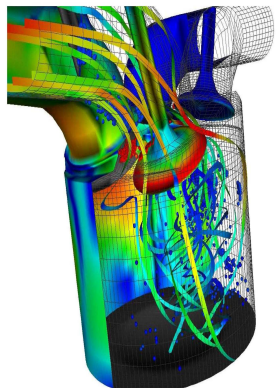
Simulating Turbulent Reacting Flows: S3D

- **Direct numerical simulation (DNS) of turbulent combustion**
  - **state-of-the-art code developed at CRF/Sandia**
    - **PI: Jaqueline H. Chen, SNL**
  - **2020: 600K hours, IBM AC922 2xP9+6xV100**
  - **"DNS of Turbulent Combustion Towards Efficient Engines with In Situ Analytics"**



- **Science**
  - **study micro-physics of turbulent reacting flows**
    - **physical insight into chemistry turbulence interactions**
  - **simulate chemistry and 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

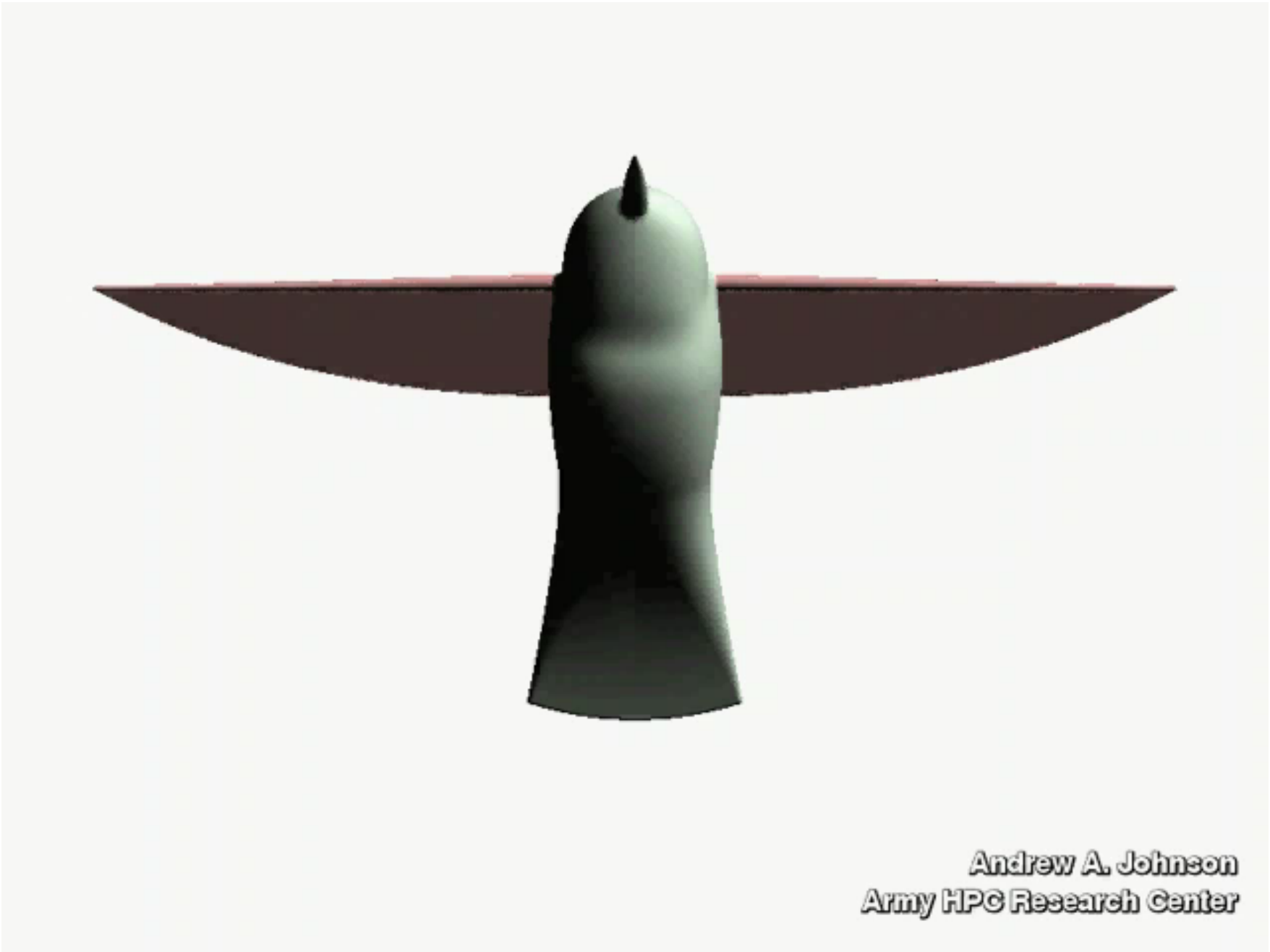
Fluid-Structure Interactions

- **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**

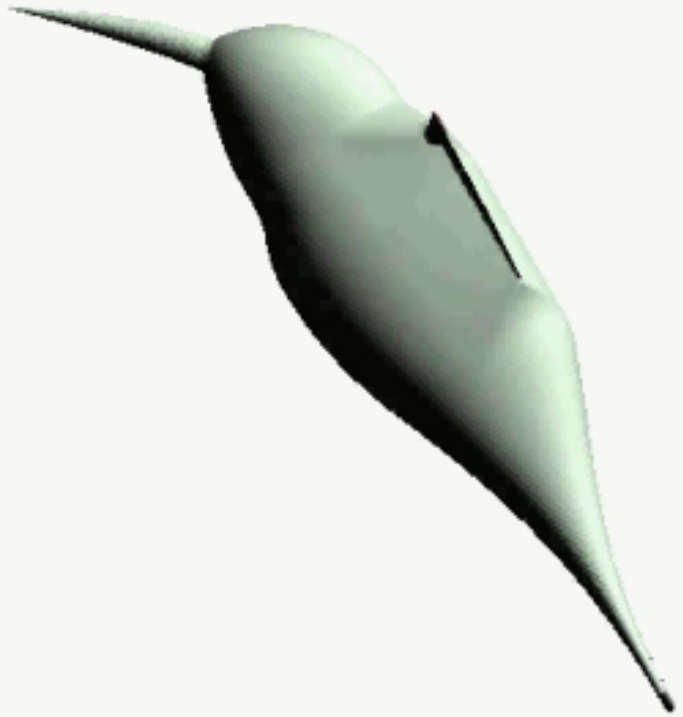
[Andrew Johnson, AHPCRC 2005]

- **Another example: 3D heart simulation [2014]**
  - —https://youtu.be/2LPboySOSvo

Air Velocity (Front)

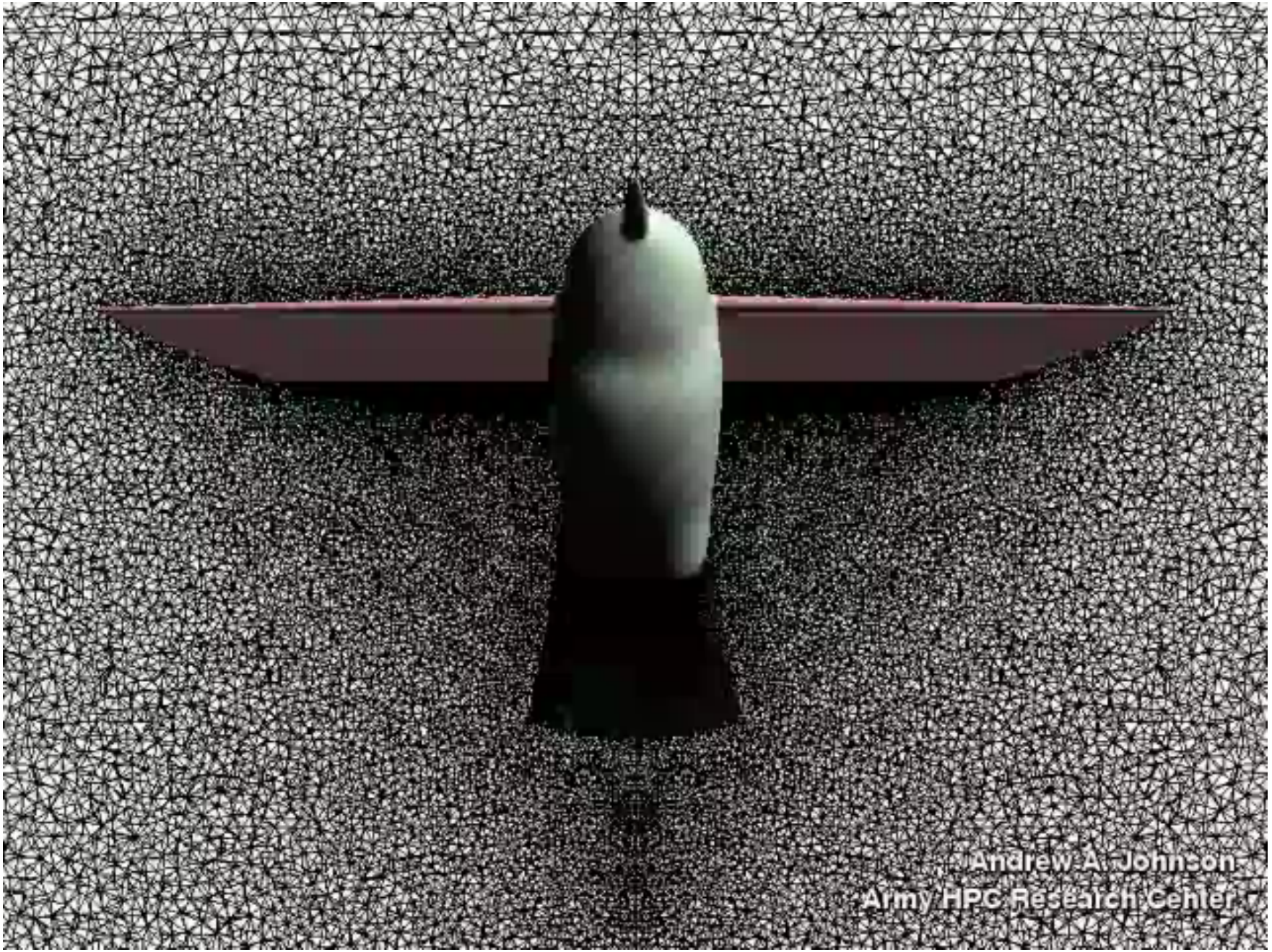


Air Velocity (Side)

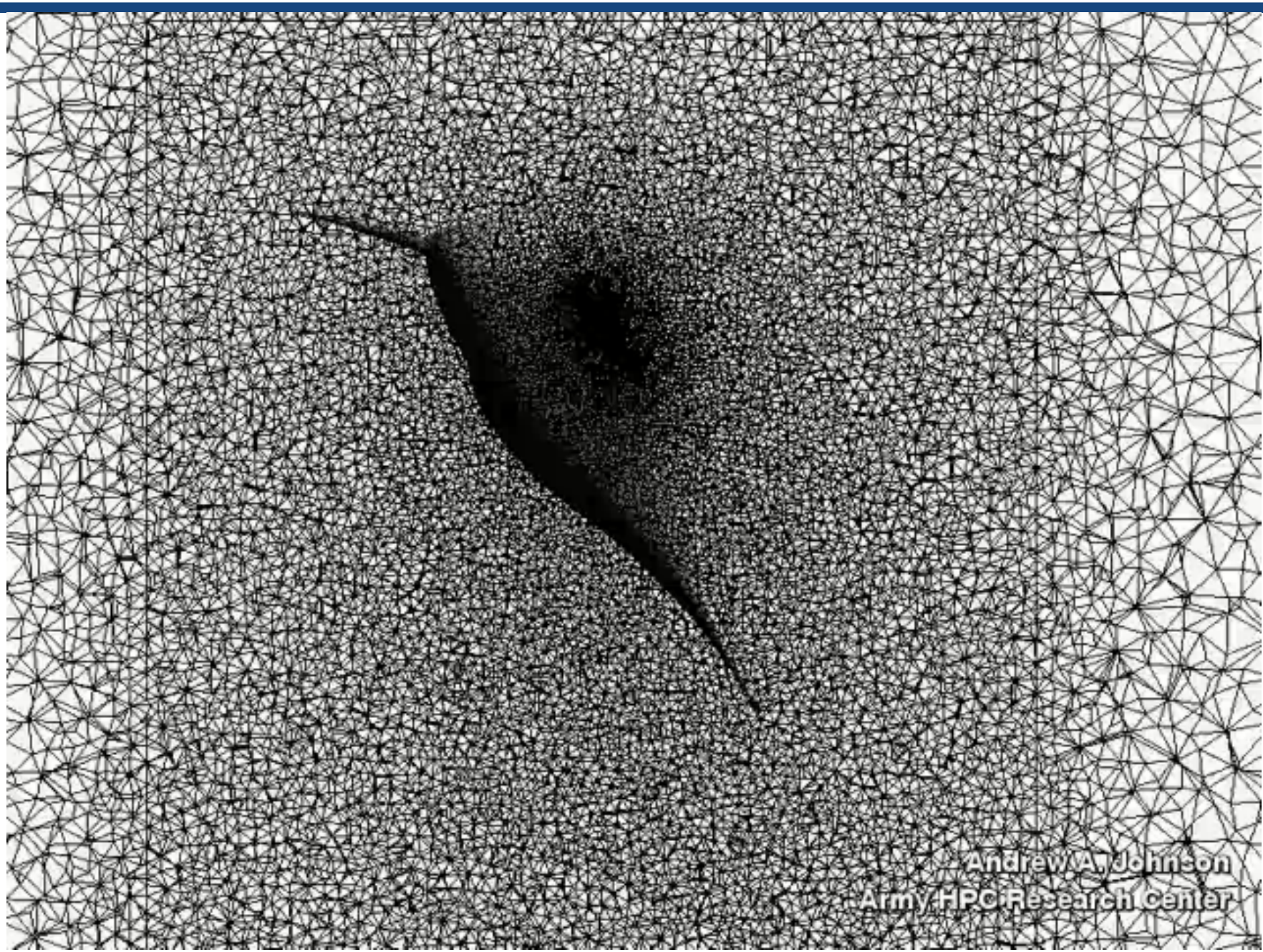


Andrew A. Johnson
Army HPC Research Center

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**
  - —supports a shared memory programming model on a cluster
- **Application overview**
  - —distribute mesh among the processors
  - —partition the mesh among the processors
  - —each processor 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

ORNL Summit Supercomputer

Summit Overview

200 x 10^{15} operations/second

Compute System

10.2 PB Total Memory
256 compute racks
4,608 compute nodes
Mellanox EDR IB fabric
200 PFLOPS
~13 MW

Compute Rack

18 Compute Servers
Warm water (70°F direct-cooled components)
RDHX for air-cooled components

39.7 TB Memory/rack
55 KW max power/rack

OpenPOWER™

Compute Node

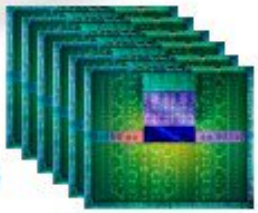
2 x POWER9
6 x NVIDIA GV100
NVMe-compatible PCIe 1600 GB SSD

25 GB/s EDR IB- (2 ports)
512 GB DRAM- (DDR4)
96 GB HBM- (3D Stacked)
Coherent Shared Memory

Components

IBM POWER9
- 22 Cores
- 4 Threads/core
- NVLink

NVIDIA GV100
- 7 TF
- 16 GB @ 0.9 TB/s
- NVLink



GPFS File System

250 PB storage
2.5 TB/s read, 2.5 TB/s write

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Scale of the Largest HPC Systems (Nov 2019)

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
2	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR , Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray/HPE Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384
7	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray/HPE DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578
8	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
9	**SuperMUC-NG** - ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , Lenovo Leibniz Rechenzentrum Germany	305,856	19,476.6	26,873.9	
10	**Lassen** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Tesla V100 , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	288,288	18,200.0	23,047.2	

IBM Power9 (22C/88T) 6x Nvidia Volta (80SM)

IBM Power9 (22C/88T) 4x Nvidia Volta (80SM)

Sunway 4C + 260C

Intel Xeon (12C/24T) + Matrix-2000 (128C)

Intel Xeon (28C/56T)

Intel Xeon (12C/24T) + NVIDIA P100 (56SM)

Intel Xeon (16C/22T)

Intel KNL (68C/272T)

Intel Xeon (20C/80T) 2x Nvidia Volta (80SM)

Intel Xeon (24C/48T)

IBM Power9 (22C/88T) 4x Nvidia Volta (80SM)

all 10 > 250K cores

> 1.5M cores

heterogeneous manycore

hybrid CPU+accelerator

homogeneous manycore

Source https://www.top500.org

Top500 Chart Generator

Achieving High Performance on Parallel Systems

Computation is only part of the picture

- **Memory latency and bandwidth**
  - —CPU rates are > 200x faster than memory
  - —bridge speed gap using memory hierarchy
  - —more cores exacerbates demand
- **Interprocessor communication**
- **Input/output**
  - — I/O bandwidth to disk typically needs to grow linearly with the # processors

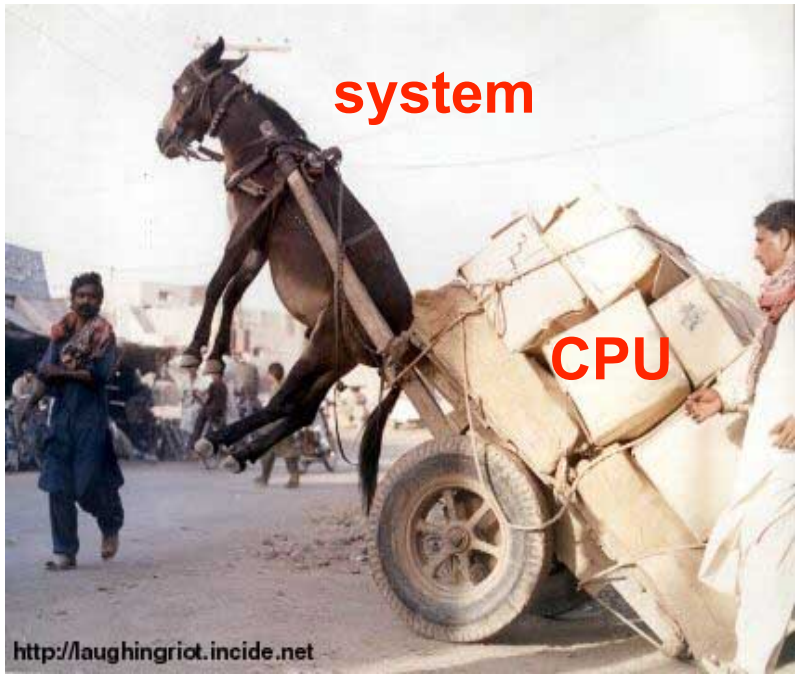


Image Credit: Bob Colwell, ISCA 1995

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 System for the Course

- **NOTS**
  - —226 nodes, each with two 8C/16T Intel Xeon processors
  - —2 nodes with 2 x NVIDIA K80 GPGPUs
  - —no global shared memory