Lecture 24:

The Future of High-Performance Computing

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2016

Executive Order
July 29, 2015

EXECUTIVE ORDER

- - - - - - -

CREATING A NATIONAL STRATEGIC COMPUTING INITIATIVE

By the authority vested in me as President by the Constitution and the laws of the United States of America, and to maximize benefits of high-performance computing (HPC) research, development, and deployment, it is hereby ordered as follows:

The NSCI is a whole-of-government effort designed to create a cohesive, multi-agency strategic vision and Federal investment strategy, executed in collaboration with industry and academia, to maximize the benefits of HPC for the United States.

Strategic Objectives

(1) Accelerating delivery of a capable exascale computing system that integrates hardware and software capability to deliver approximately 100 times the performance of current 10 petaflop systems across a range of applications representing government needs.

(2) Increasing coherence between the technology base used for modeling and simulation and that used for data analytic computing.

(3) Establishing, over the next 15 years, a viable path forward for future HPC systems even after the limits of current semiconductor technology are reached (the "post-Moore's Law era").

(4) Increasing the capacity and capability of an enduring national HPC ecosystem by employing a holistic approach that addresses relevant factors such as networking technology, workflow, downward scaling, foundational algorithms and software, accessibility, and workforce development.

(5) Developing an enduring public-private collaboration to ensure that the benefits of the research and development advances are, to the greatest extent, shared between the United States Government and industrial and academic sectors.

Comparing Two Large-Scale Systems

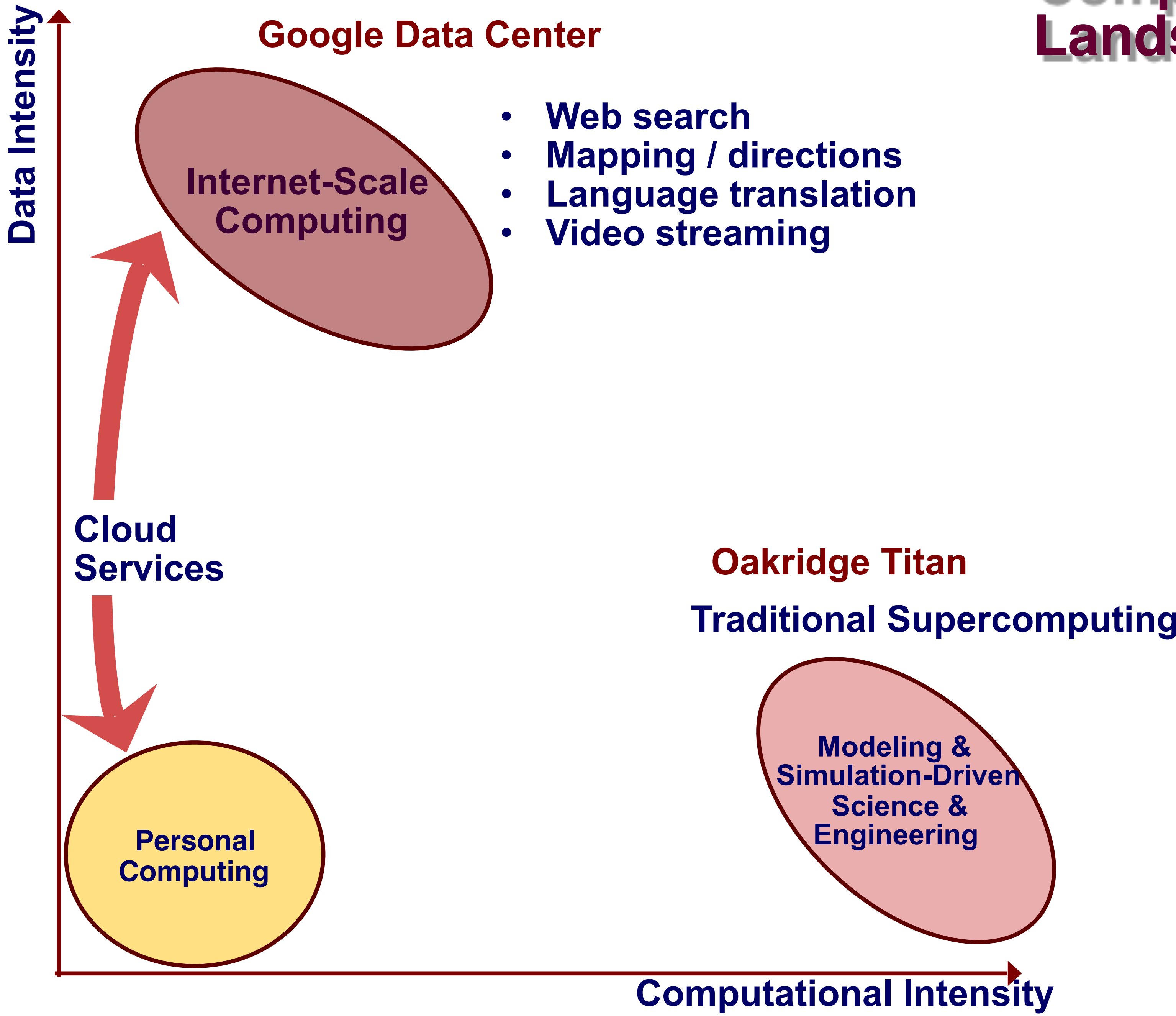
Oakridge Titan

- Monolithic supercomputer (2nd fastest in world)
- Designed for compute-intensive applications

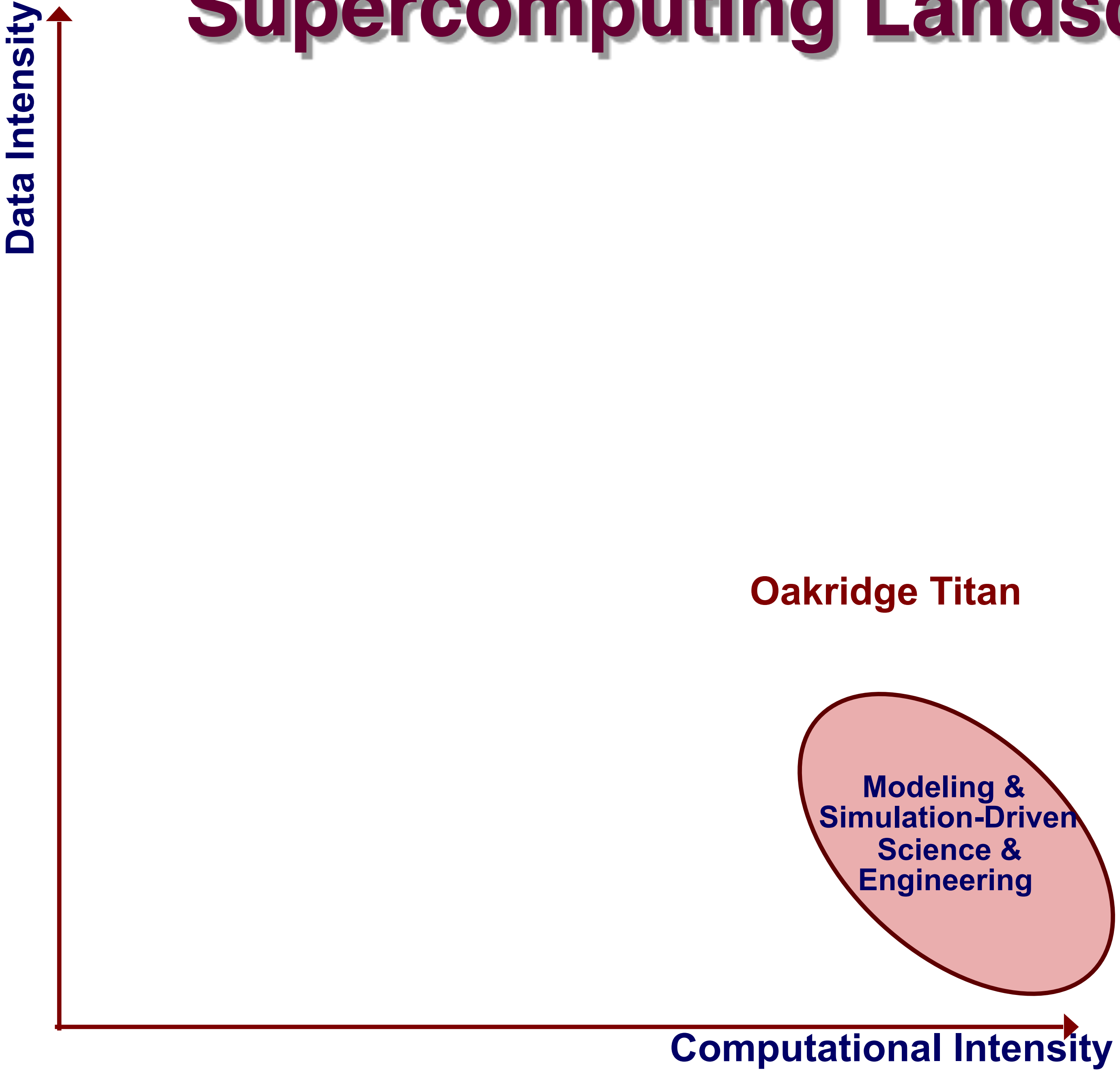
Google Data Center

- Servers to support millions of customers
- Designed for data collection, storage, and analysis

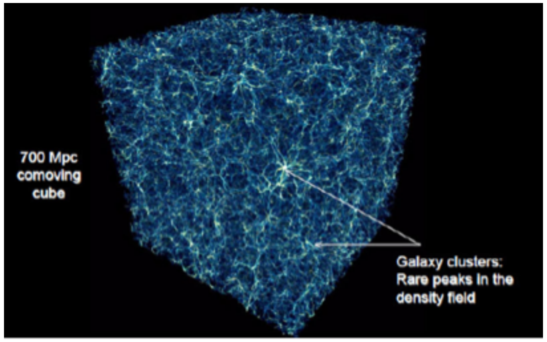
Computing Landscape



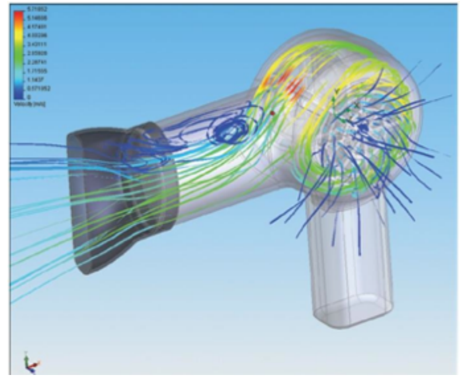
Supercomputing Landscape



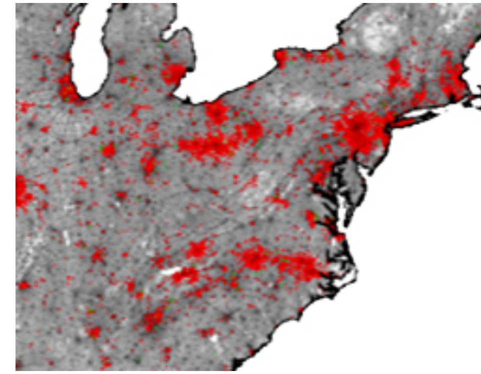
Supercomputer Applications



Science



Industrial Products



Public Health

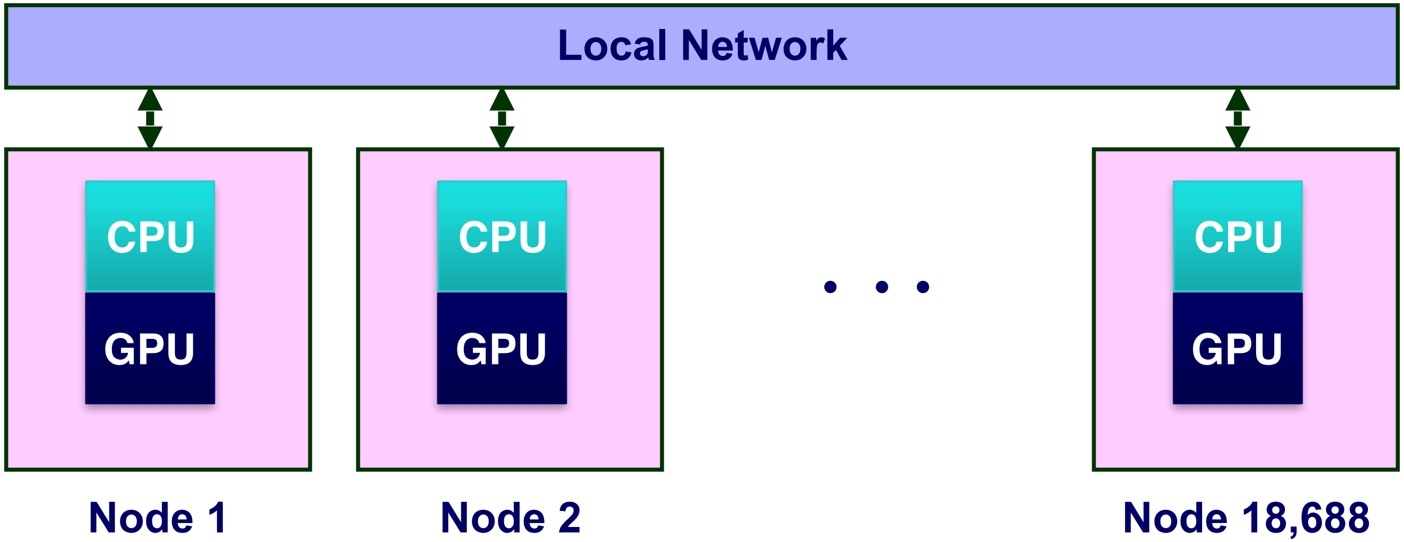
Simulation-Based Modeling

- System structure + initial conditions + transition behavior
- Discretize time and space
- Run simulation to see what happens

Requirements

- Model accurately reflects actual system
- Simulation faithfully captures model

Titan Hardware



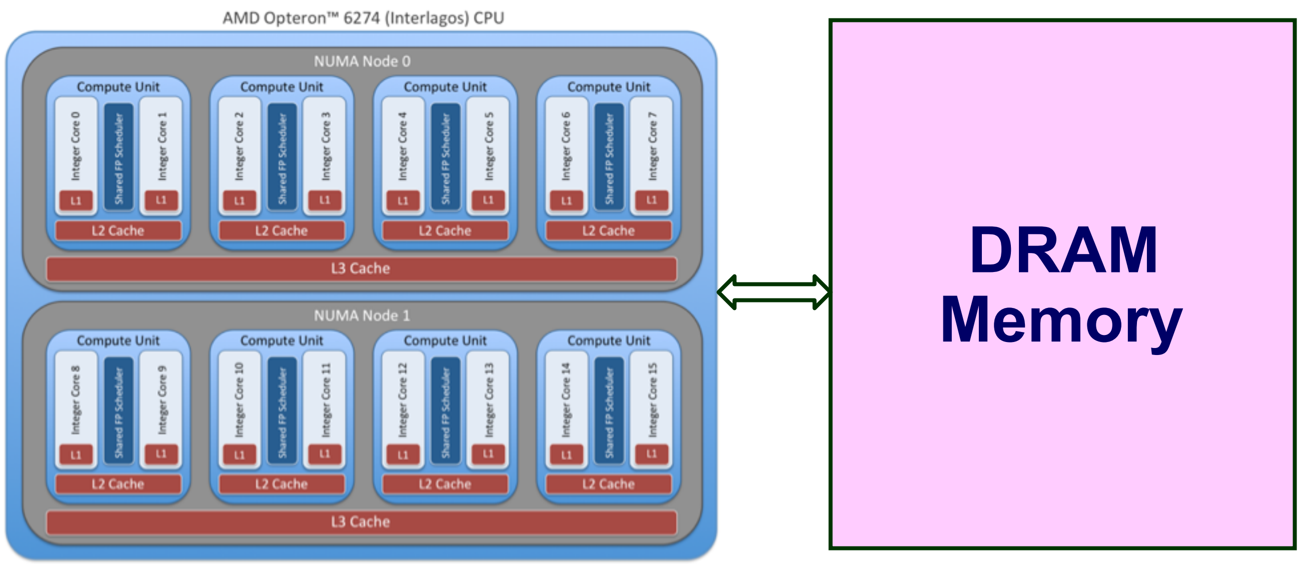
Each Node

- AMD 16-core processor
- nVidia Graphics Processing Unit
- 38 GB DRAM
- *No disk drive*

Overall

- 7MW, $200M

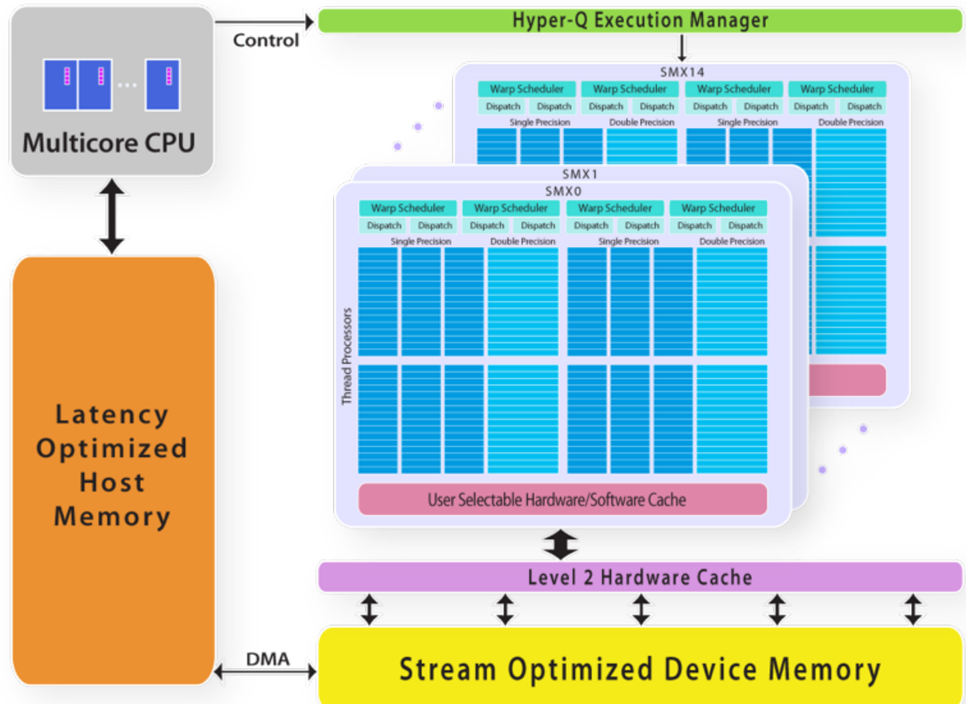
Titan Node Structure: CPU



CPU

- 16 cores sharing common memory
- Supports multithreaded programming
- ~0.16 x 10^{12} floating-point operations per second (FLOPS) peak performance

Titan Node Structure: GPU



Kepler GPU

- **14 multiprocessors**
- **Each with 12 groups of 16 stream processors**
  - 14 X 12 X 16 = 2688
- **Single-Instruction, Multiple-Data parallelism**
  - Single instruction controls all processors in group
- **4.0 x 10^{12} FLOPS peak performance**

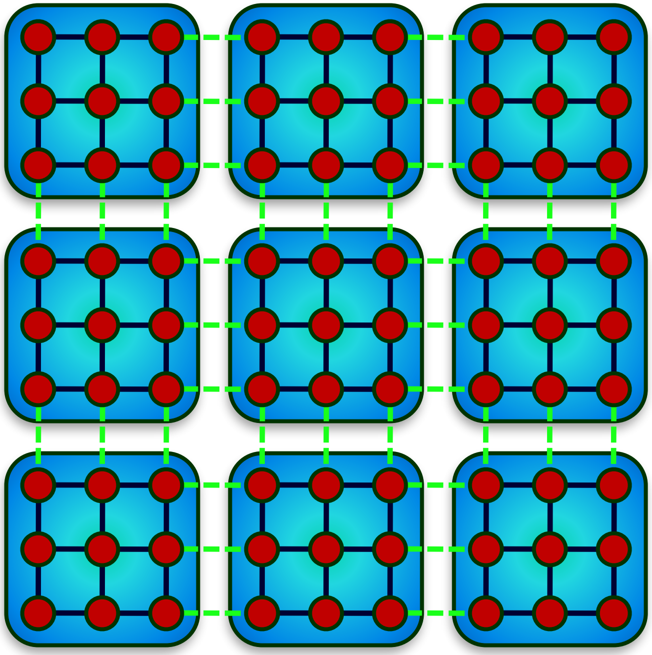
Titan Programming: Principle

Solving Problem Over Grid

- E.g., finite-element system
- Simulate operation over time

Bulk Synchronous Model

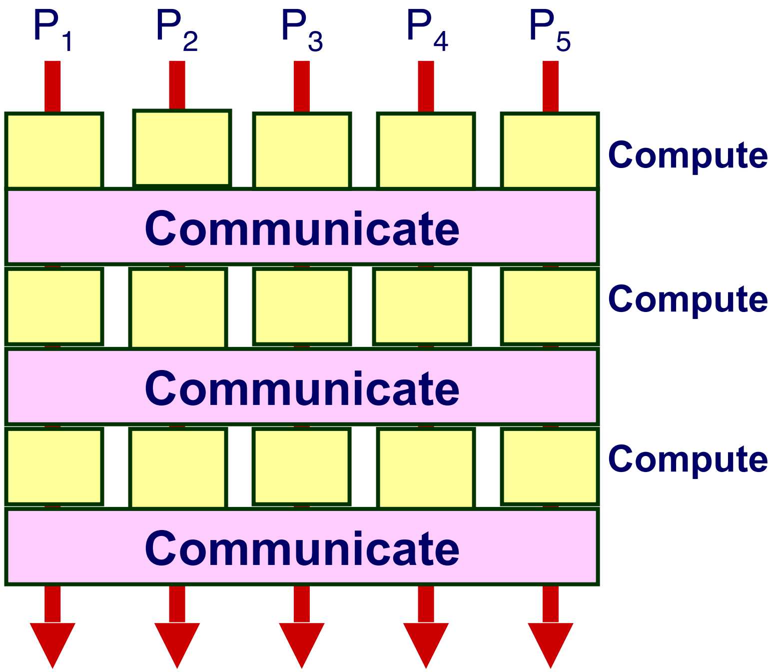
- Partition into Regions
  - p regions for p-node machine
- Map Region per Processor



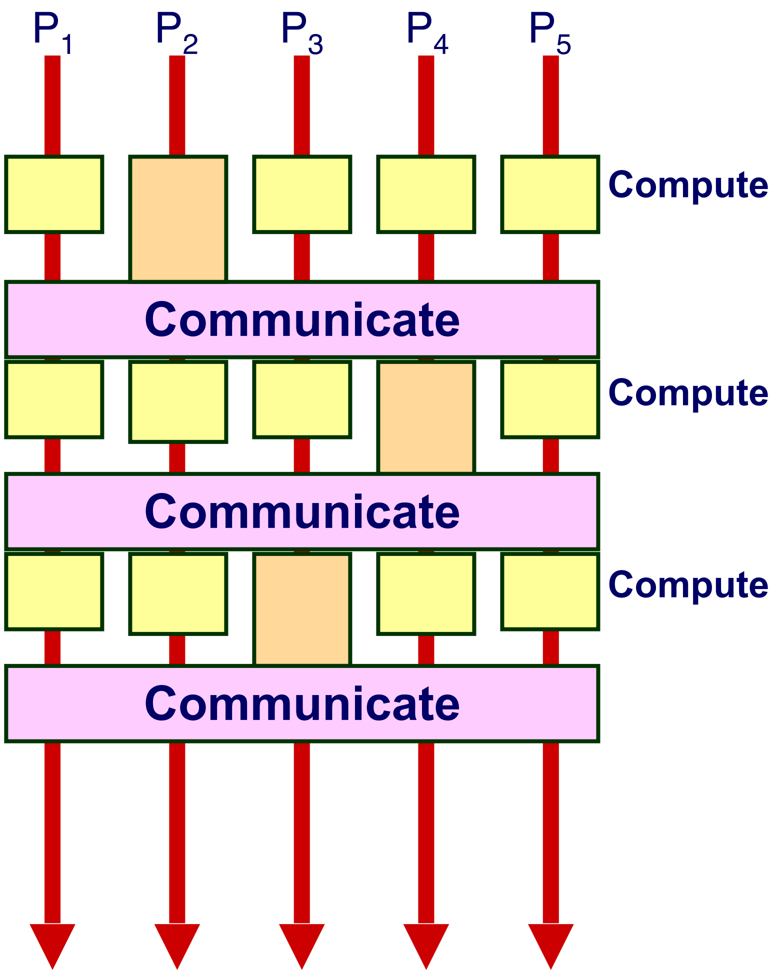
Titan Programming: Principle (cont)

Bulk Synchronous Model

- **Map Region per Processor**
- **Alternate**
  - All nodes compute behavior of region
    - » Perform on GPUs
  - All nodes communicate values at boundaries



Bulk Synchronous Performance



- Limited by performance of slowest processor

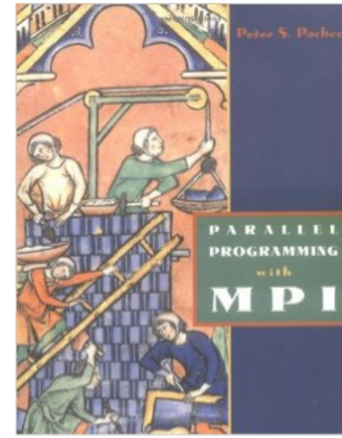
Strive to keep perfectly balanced

- Engineer hardware to be highly reliable
- Tune software to make as regular as possible
- Eliminate “noise”
  - Operating system events
  - Extraneous network activity

Titan Programming: Reality

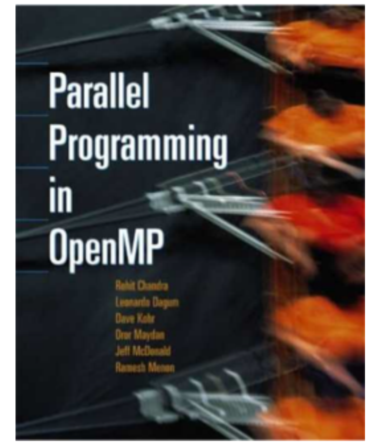
System Level

- **Message-Passing Interface (MPI) supports node computation, synchronization and communication**



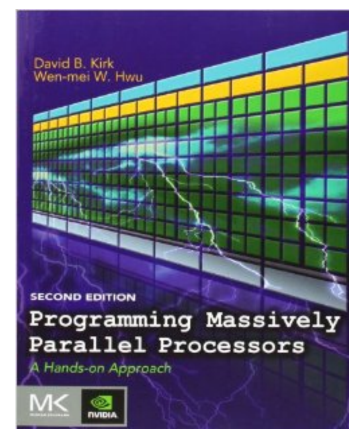
Node Level

- **OpenMP supports thread-level operation of node CPU**
- **CUDA programming environment for GPUs**
  - Performance degrades quickly if don't have perfect balance among memories and processors

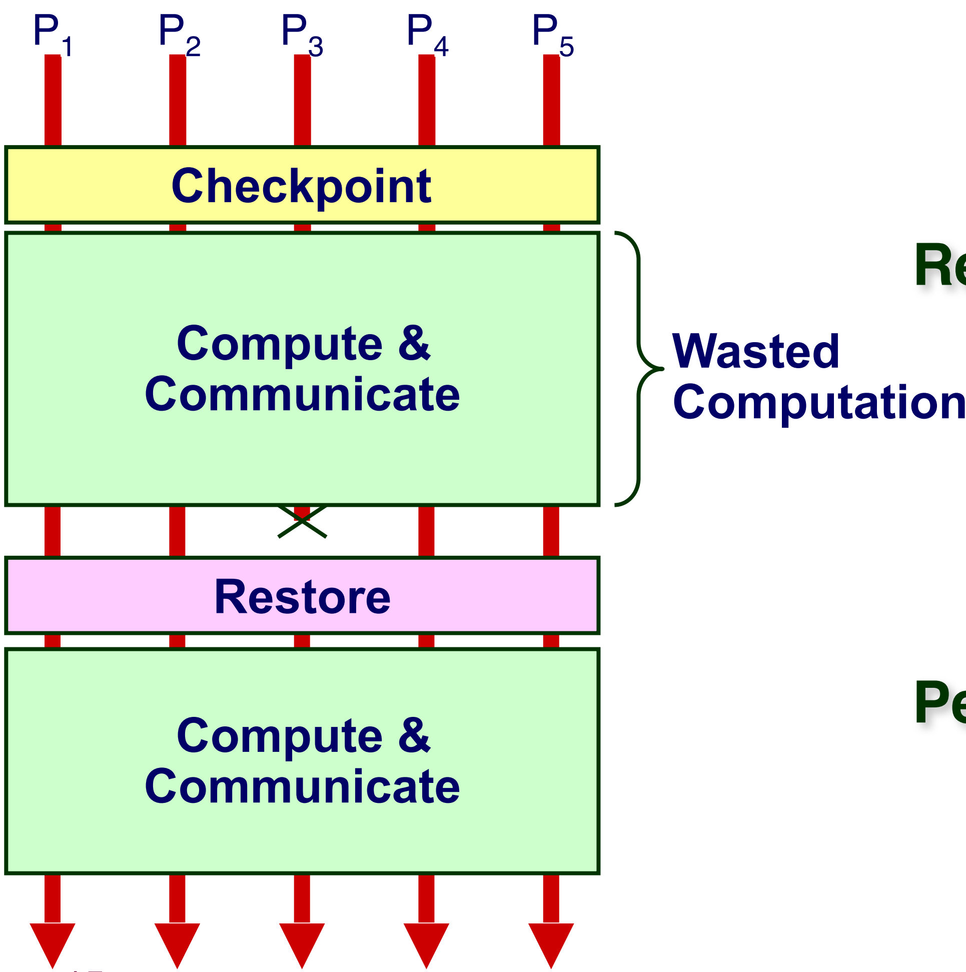


Result

- **Single program is complex combination of multiple programming paradigms**
- **Tend to optimize for specific hardware configuration**



MPI Fault Tolerance



Checkpoint

- Periodically store state of all processes
- Significant I/O traffic

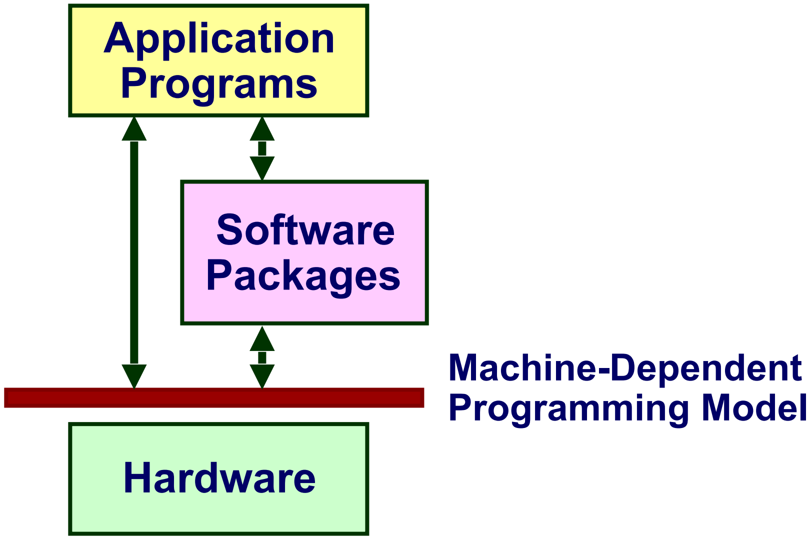
Restore

- When failure occurs
- Reset state to that of last checkpoint
- All intervening computation wasted

Performance Scaling

- Very sensitive to number of failing components

Supercomputer Programming Model



- Program on top of bare hardware

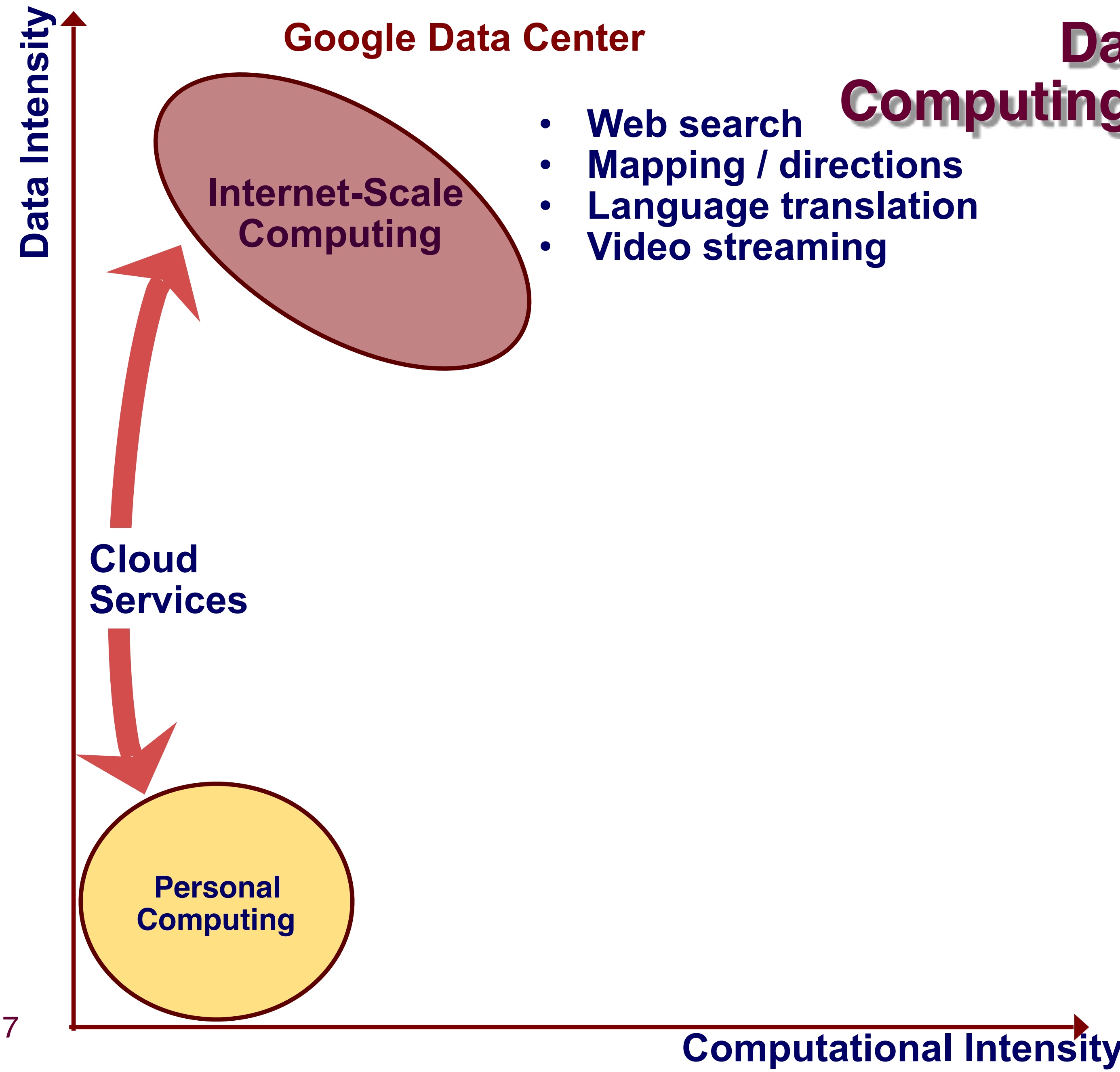
Performance

- Low-level programming to maximize node performance
- Keep everything globally synchronized and balanced

Reliability

- Single failure causes major delay
- Engineer hardware to minimize failures

Data-Intensive Computing Landscape



Internet Computing

Web Search

- Aggregate text data from across WWW
- No definition of correct operation
- Do not need real-time updating

Mapping Services

- Huge amount of (relatively) static data
- Each customer requires individualized computation

Online Documents

- Must be stored reliably
- Must support real-time updating
- (Relatively) small data volumes

Other Data-Intensive Computing Applications

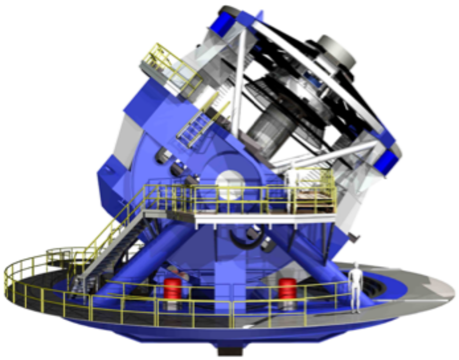
Wal-Mart

- 267 million items/day, sold at 6,000 stores
- HP built them 4 PB data warehouse
- Mine data to manage supply chain, understand market trends, formulate pricing strategies

LSST
Large Synoptic Survey Telescope

LSST

- Chilean telescope will scan entire sky every 3 days
- A 3.2 gigapixel digital camera
- Generate 30 TB/day of image data



Data-Intensive Application Characteristics

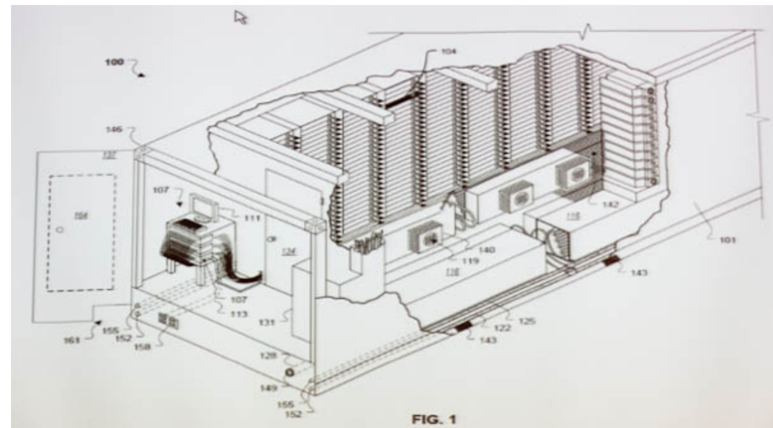
Diverse Classes of Data

- Structured & unstructured
- High & low integrity requirements

Diverse Computing Needs

- Localized & global processing
- Numerical & non-numerical
- Real-time & batch processing

Google Data Centers

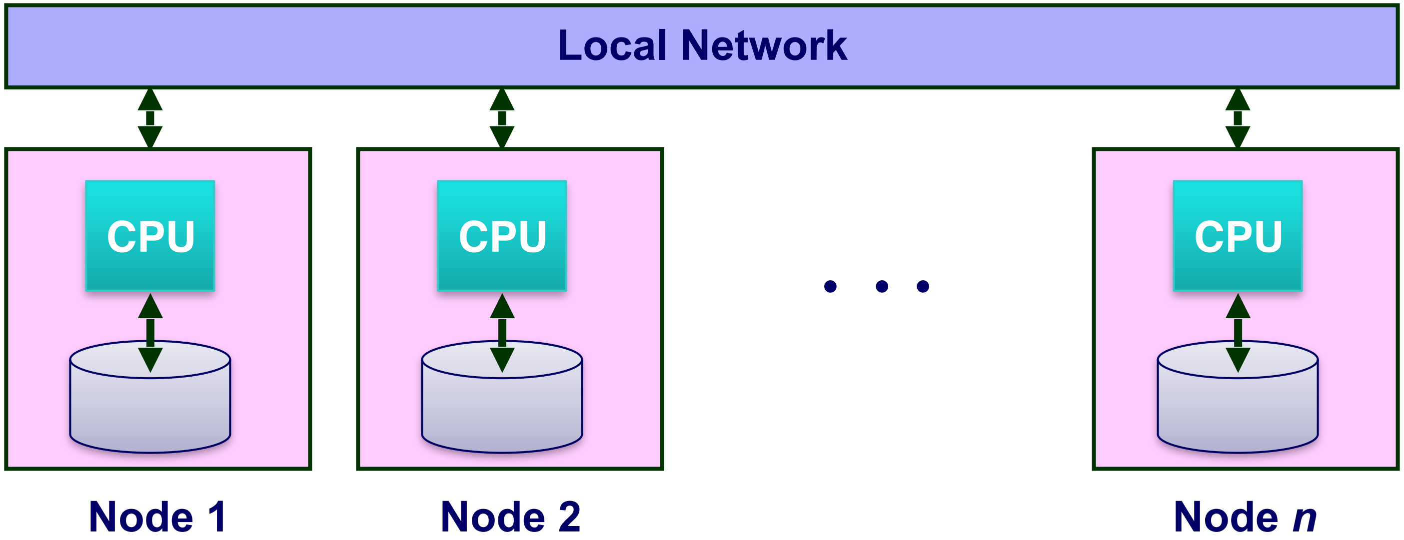


Dalles, Oregon

- **Hydroelectric power @ 2¢ / KW Hr**
- **50 Megawatts**
  - Enough to power 60,000 homes

- **Engineered for low cost, modularity & power efficiency**
- **Container: 1160 server nodes, 250KW**

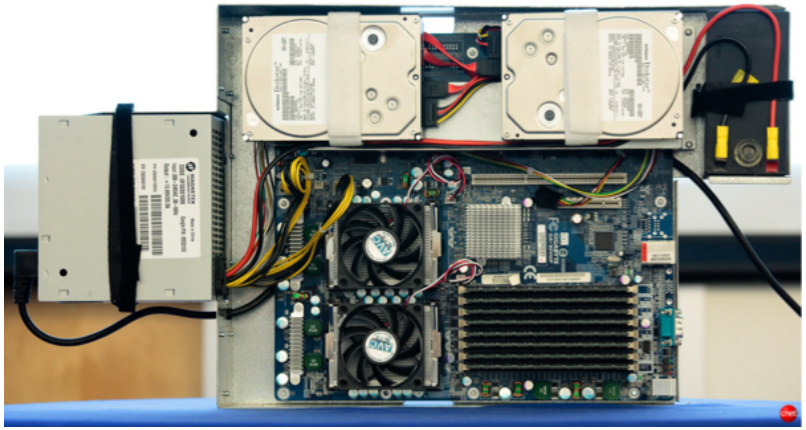
Google Cluster



- Typically 1,000–2,000 nodes

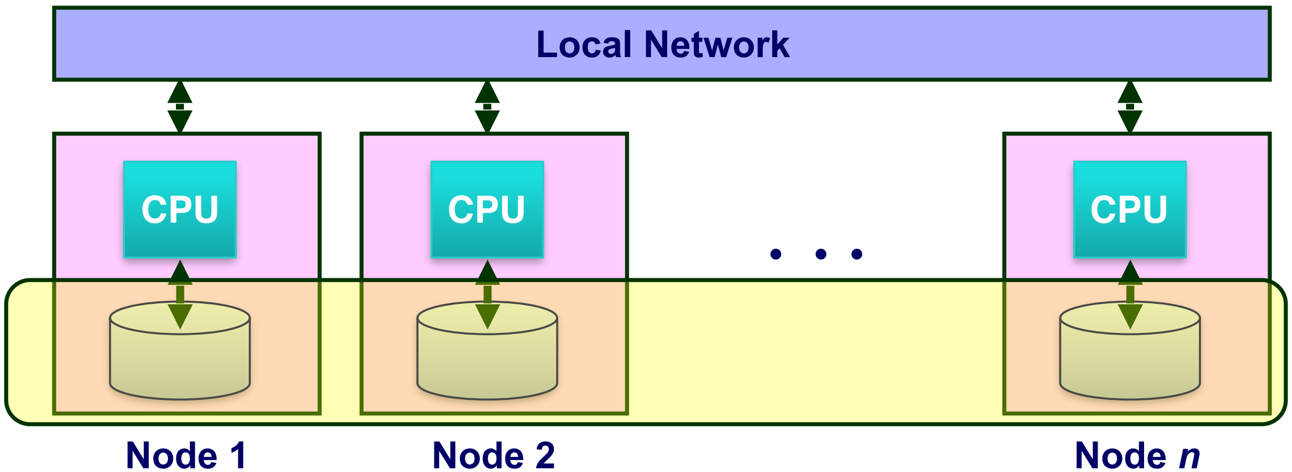
Node Contains

- 2 multicore CPUs
- 2 disk drives
- DRAM



Hadoop Project

File system with files distributed across nodes

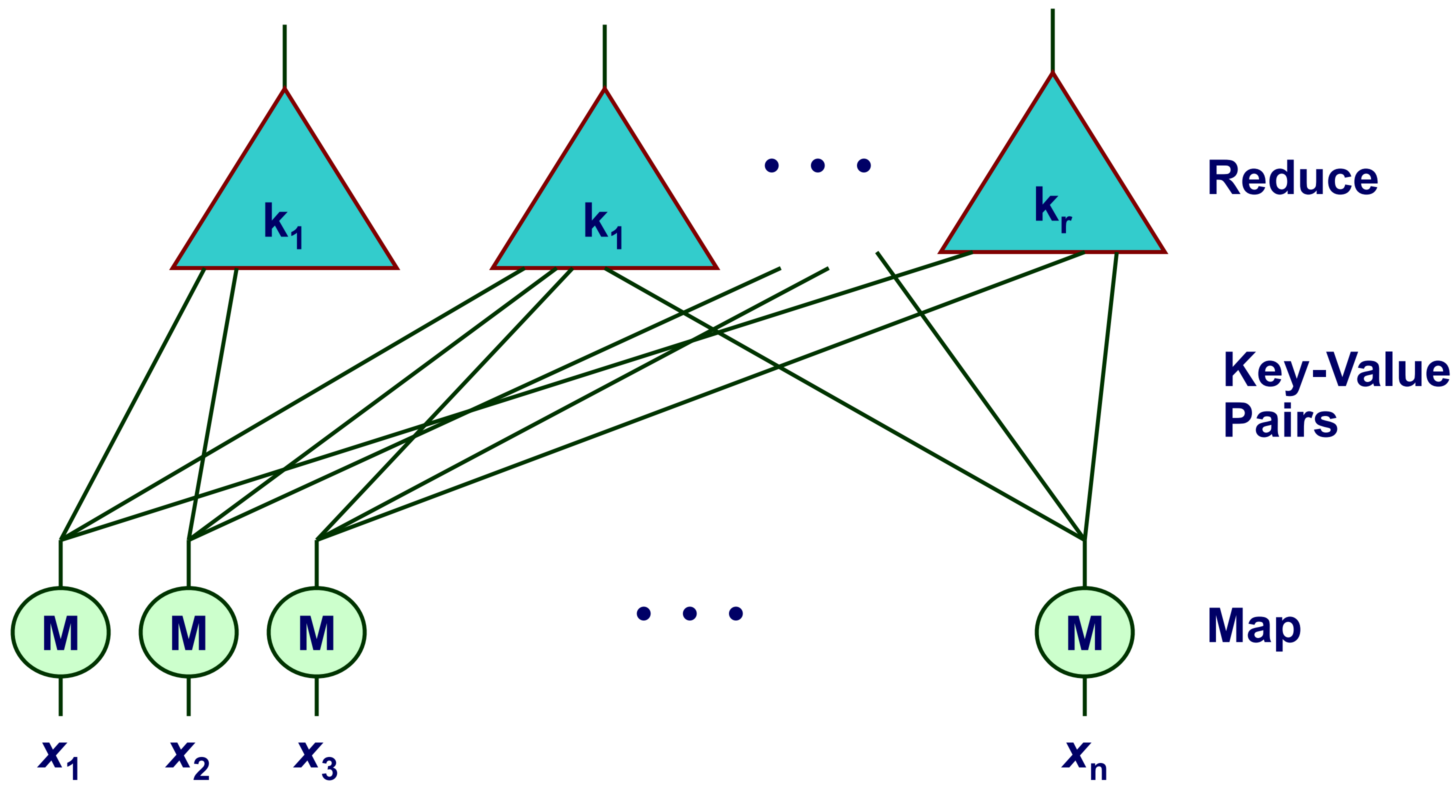


- **Store multiple (typically 3 copies of each file)**
  - If one node fails, data still available
- **Logically, any node has access to any file**
  - May need to fetch across network

Map / Reduce programming environment

- **Software manages execution of tasks on nodes**

Map/Reduce Programming Model

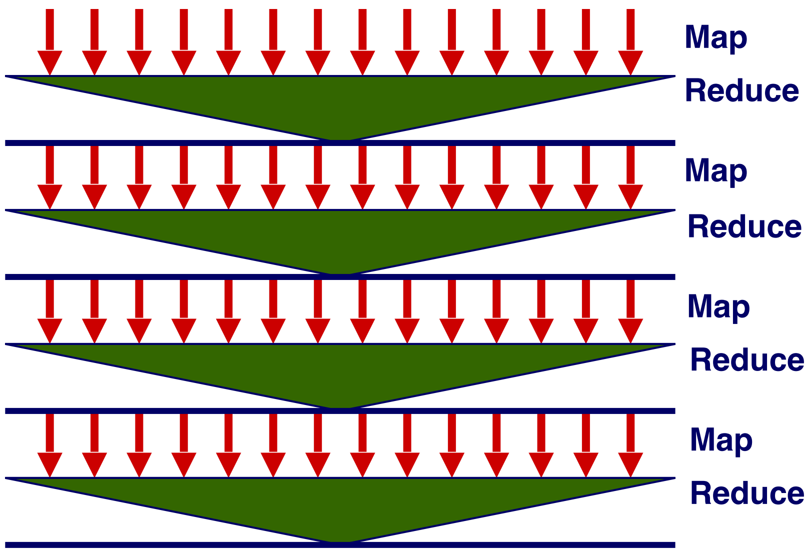


- **Map computation across many objects**
  - E.g., 10^{10} Internet web pages
- **Aggregate results in many different ways**

Dean & Ghemawat: "MapReduce: Simplified Data Processing on Large Clusters", OSDI 2004

Map/Reduce Operation

Map/Reduce



Characteristics

- **Computation broken into many, short-lived tasks**
  - Mapping, reducing
- **Tasks mapped onto processors dynamically**
- **Use disk storage to hold intermediate results**

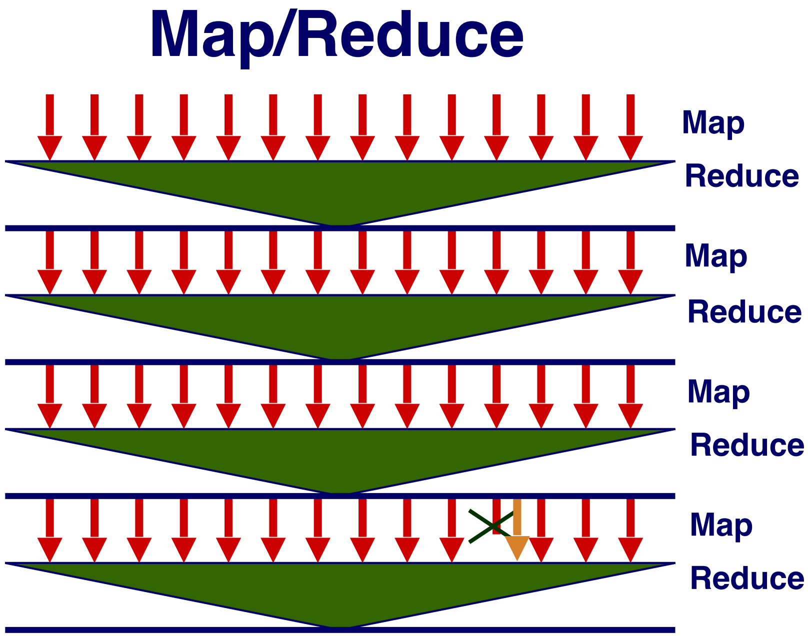
Strengths

- **Flexibility in placement, scheduling, and load balancing**
- **Can access large data sets**

Weaknesses

- **Higher overhead**
- **Lower raw performance**

Map/Reduce Fault Tolerance



Data Integrity

- Store multiple copies of each file
- Including intermediate results of each Map / Reduce
  - Continuous checkpointing

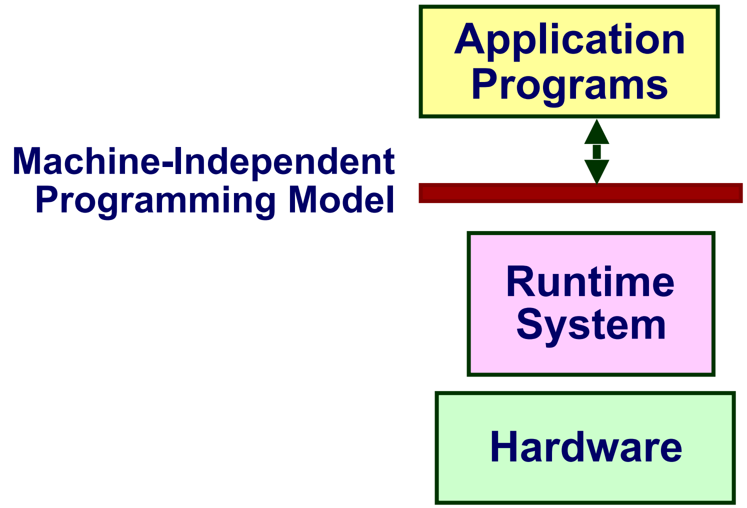
Recovering from Failure

- Simply recompute lost result
  - Localized effect
- Dynamic scheduler keeps all processors busy

Use software to build reliable system on top of unreliable hardware

Cluster Programming Model

- Application programs written in terms of high-level operations on data
- Runtime system controls scheduling, load balancing, …

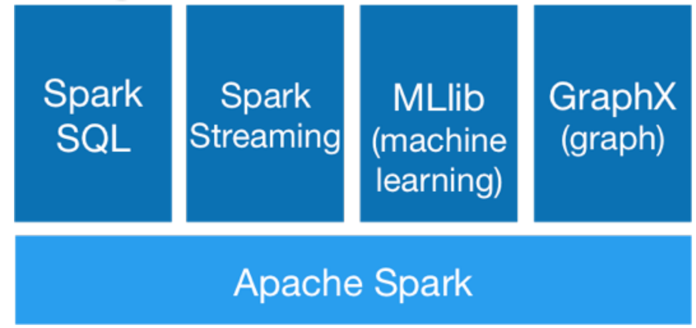


Scaling Challenges

- Centralized scheduler forms bottleneck
- Copying to/from disk very costly
- Hard to limit data movement
  - Significant performance factor

Recent Programming Systems

Spark Project



- at U.C., Berkeley
- Grown to have large open source community

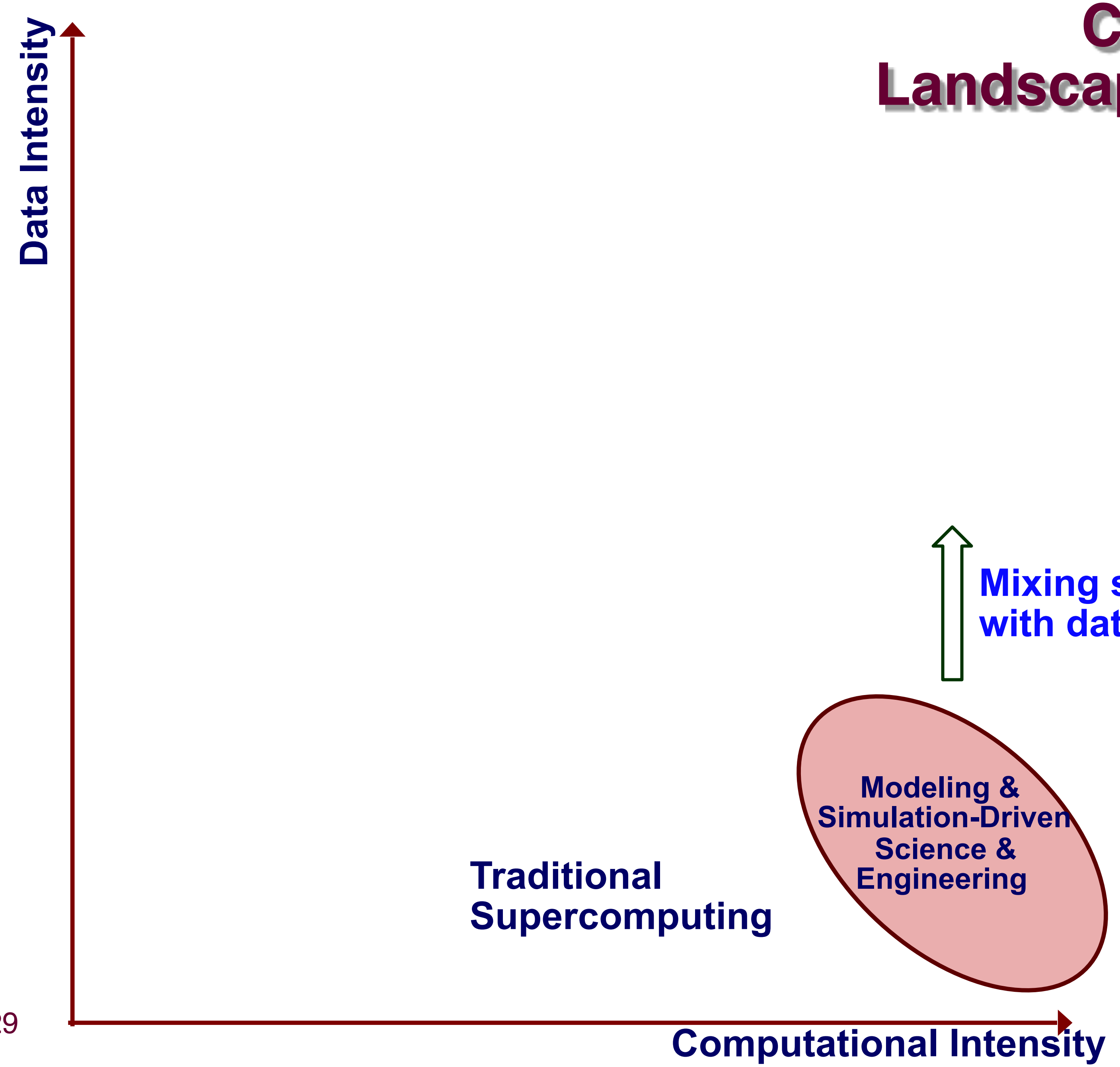
GraphLab

Machine Learning Startup GraphLab Gets A New Name And An $18.5M Check

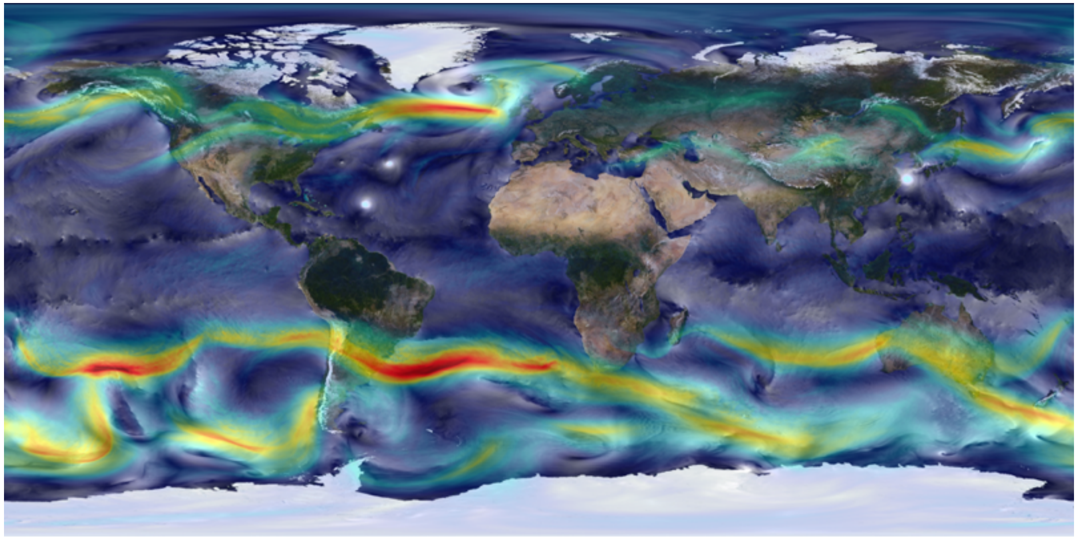
Posted Jan 8, 2015 by Jonathan Shieber (@jshieber)

- Started as project at CMU by Carlos Guestrin
- Environment for describing machine-learning algorithms
  - Sparse matrix structure described by graph
  - Computation based on updating of node values

Computing Landscape Trends



Combining Simulation with Real Data



Limitations

- Simulation alone: Hard to know if model is correct
- Data alone: Hard to understand causality & "what if"

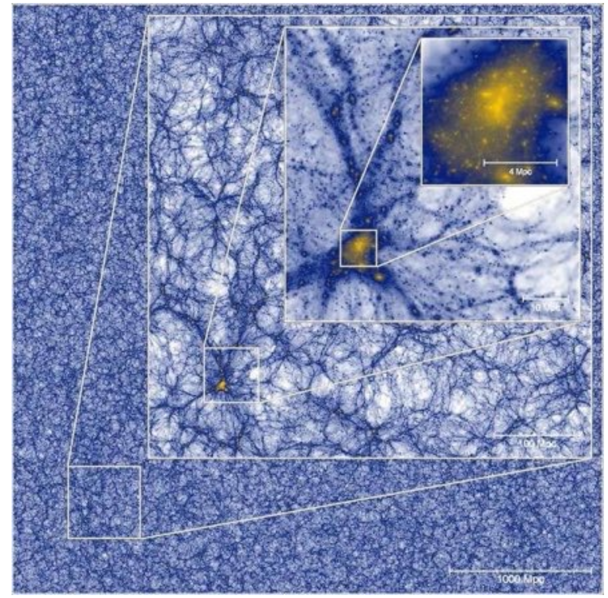
Combination

- Check and adjust model during simulation

Real-Time Analytics

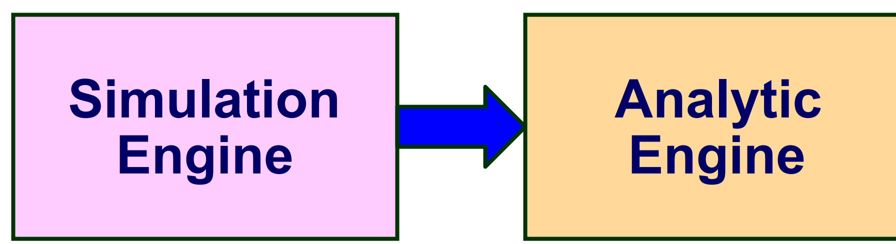
Millenium XXL Simulation (2010)

- **3×10^9 particles**
- **Simulation run of 9.3 days on 12,228 cores**
- **700TB total data generated**
  - Save at only 4 time points
  - 70 TB
- **Large-scale simulations generate large data sets**

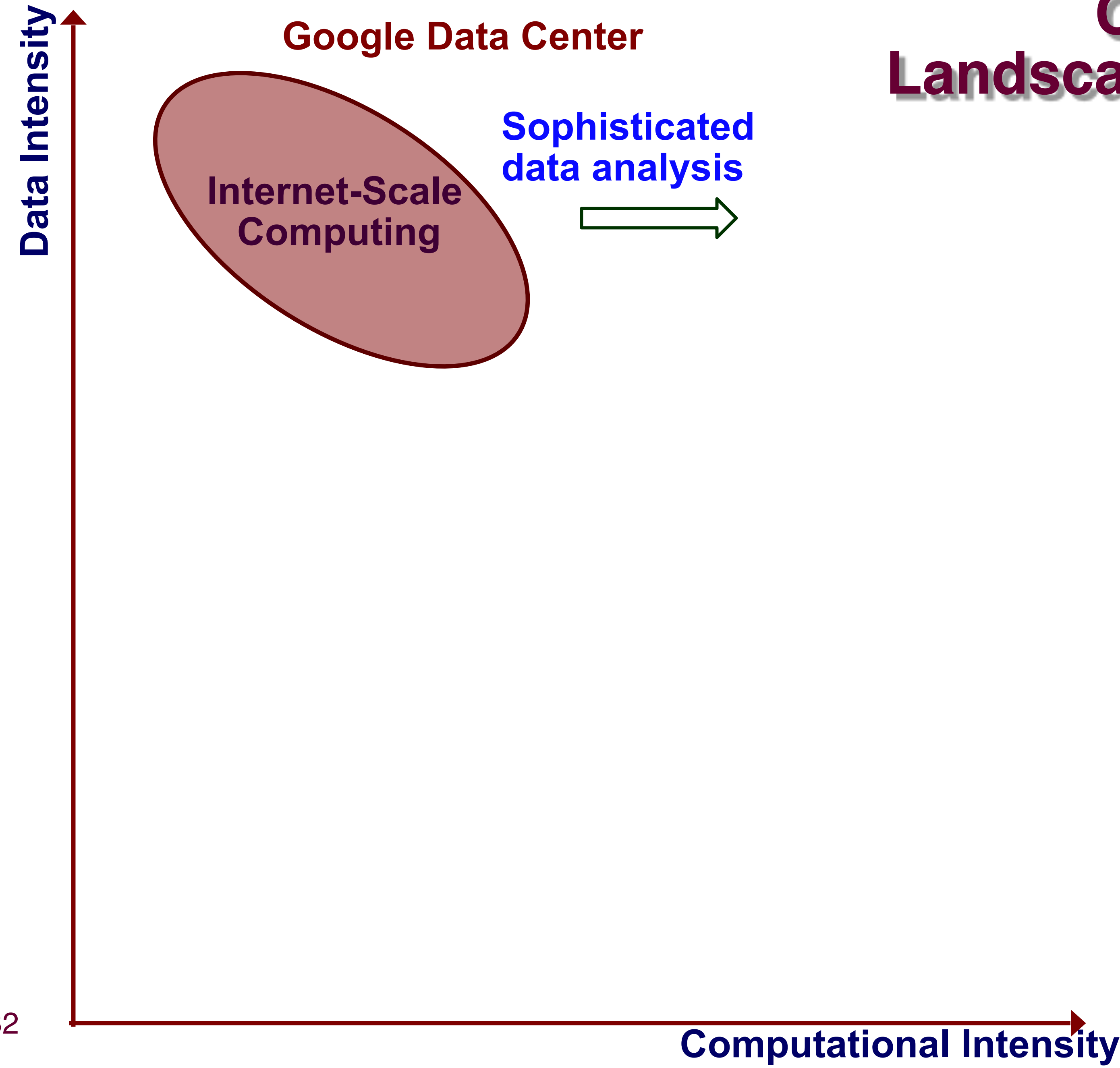


What If?

- **Could perform data analysis while simulation is running**

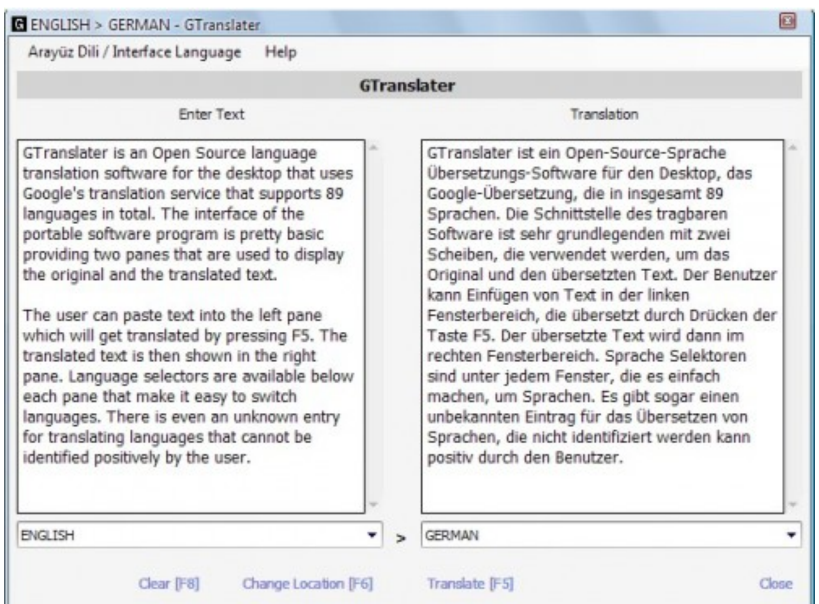


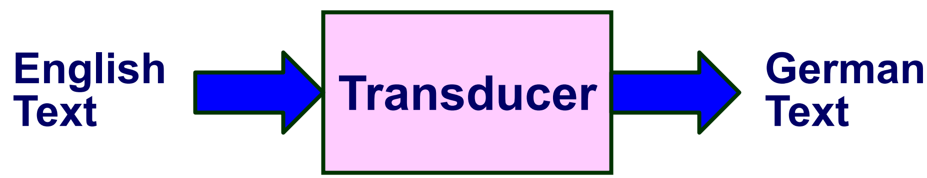
Computing Landscape Trends



Example Analytic Applications

Microsoft Project Adam

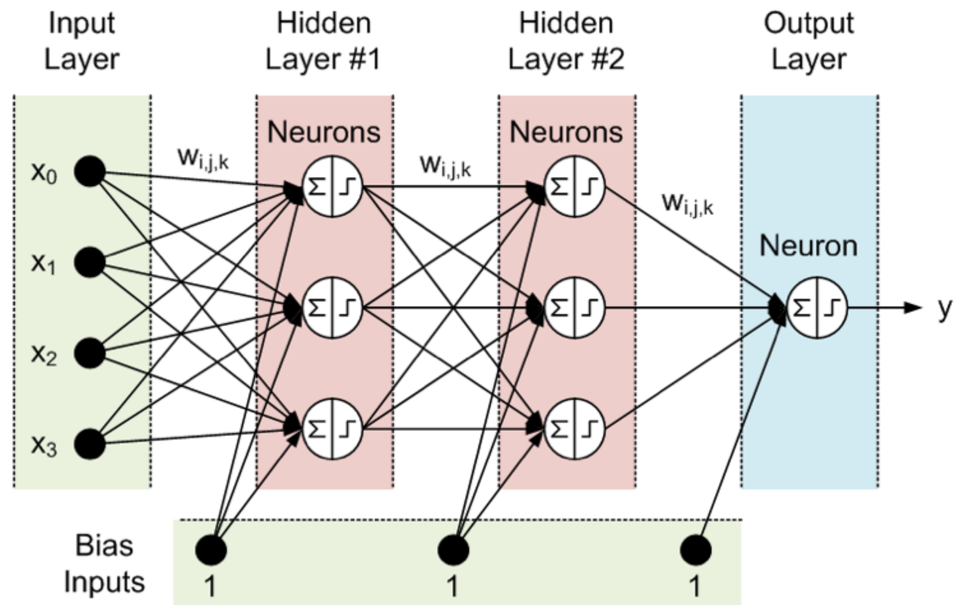




Data Analysis with Deep Neural Networks

Task:

- Compute classification of set of input signals



Training

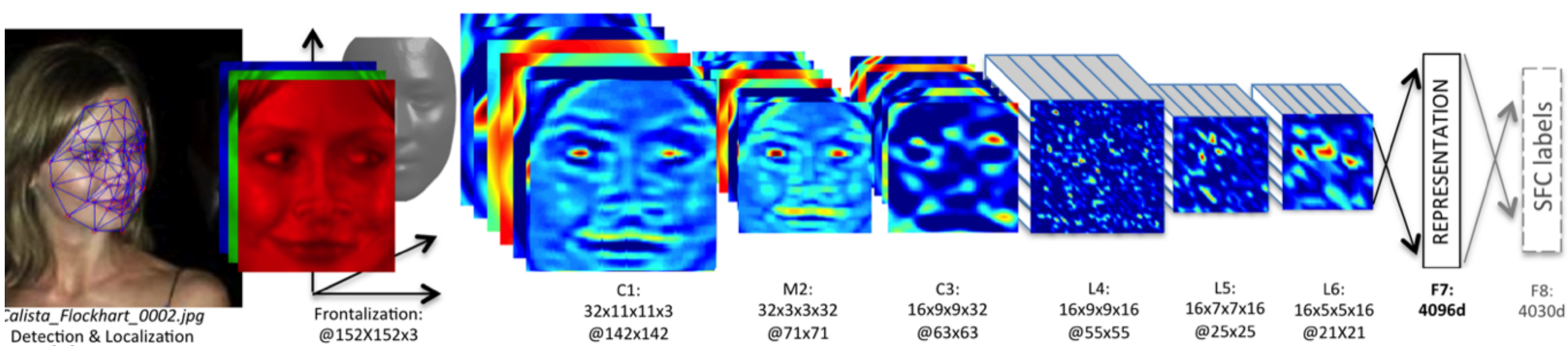
- Use many training samples of form input / desired output
- Compute weights that minimize classification error

Operation

- Propagate signals from input to output

DNN Application Example

Facebook DeepFace Architecture



Training DNNs

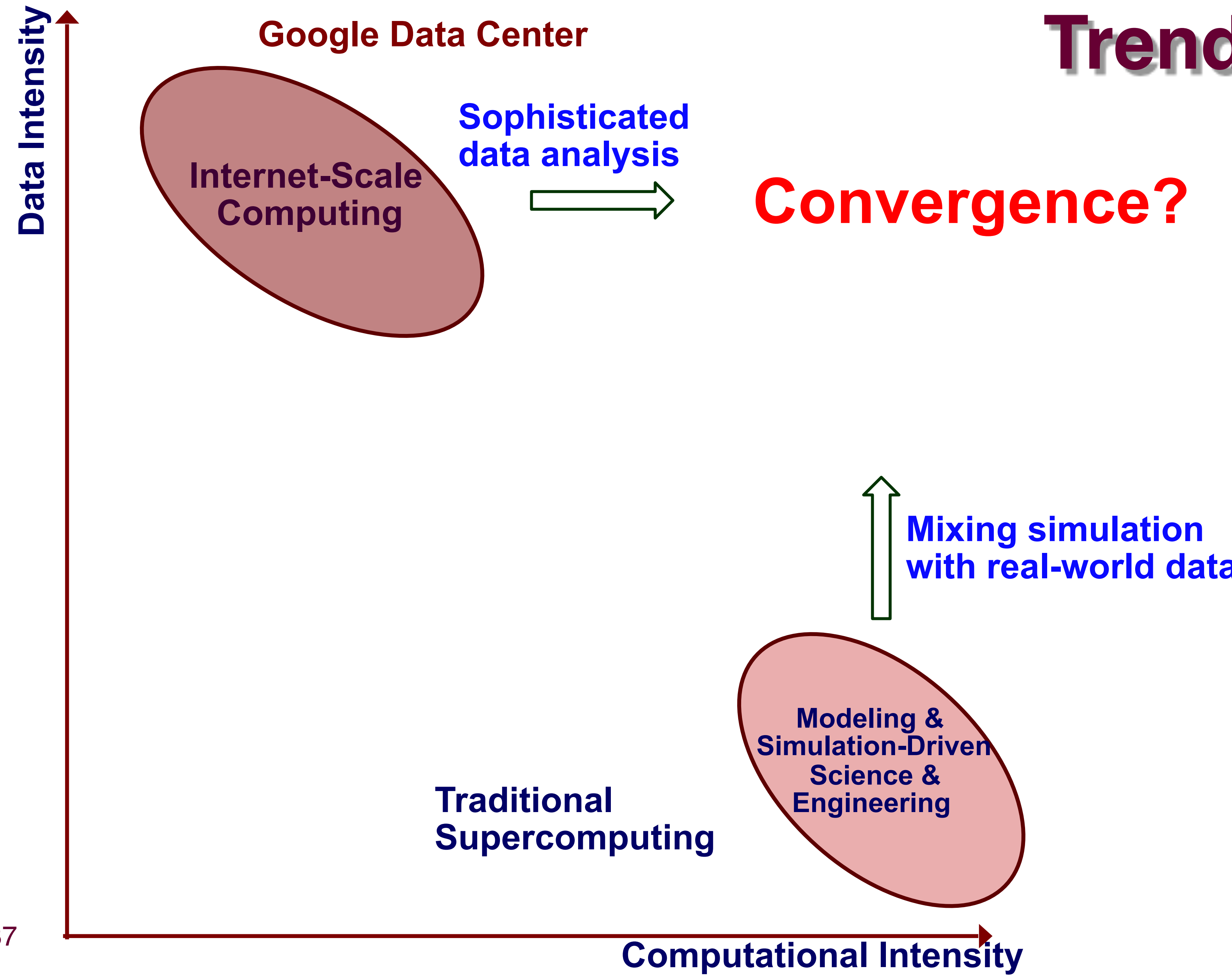
Characteristics

- Iterative numerical algorithm
- Regular data organization

Project Adam Training

- 2B connections
- 15M images
- 62 machines
- 10 days

Trends



Google Data Center
Internet-Scale Computing
Sophisticated data analysis
Convergence?
Mixing simulation with real-world data
Modeling & Simulation-Driven Science & Engineering
Traditional Supercomputing
Data Intensity
Computational Intensity

Challenges for Convergence

Supercomputers	Data Center Clusters
Hardware	
Customized	Consumer grade
Optimized for reliability	Optimized for low cost
Run-Time System	
Source of "noise"	Provides reliability
Static scheduling	Dynamic allocation
Application Programming	
Low-level, processor-centric model	High level, data-centric model

Summary: Computation/Data Convergence

Two Important Classes of Large-Scale Computing

- **Computationally intensive supercomputing**
- **Data intensive processing**
  - Internet companies + many other applications

Followed Different Evolutionary Paths

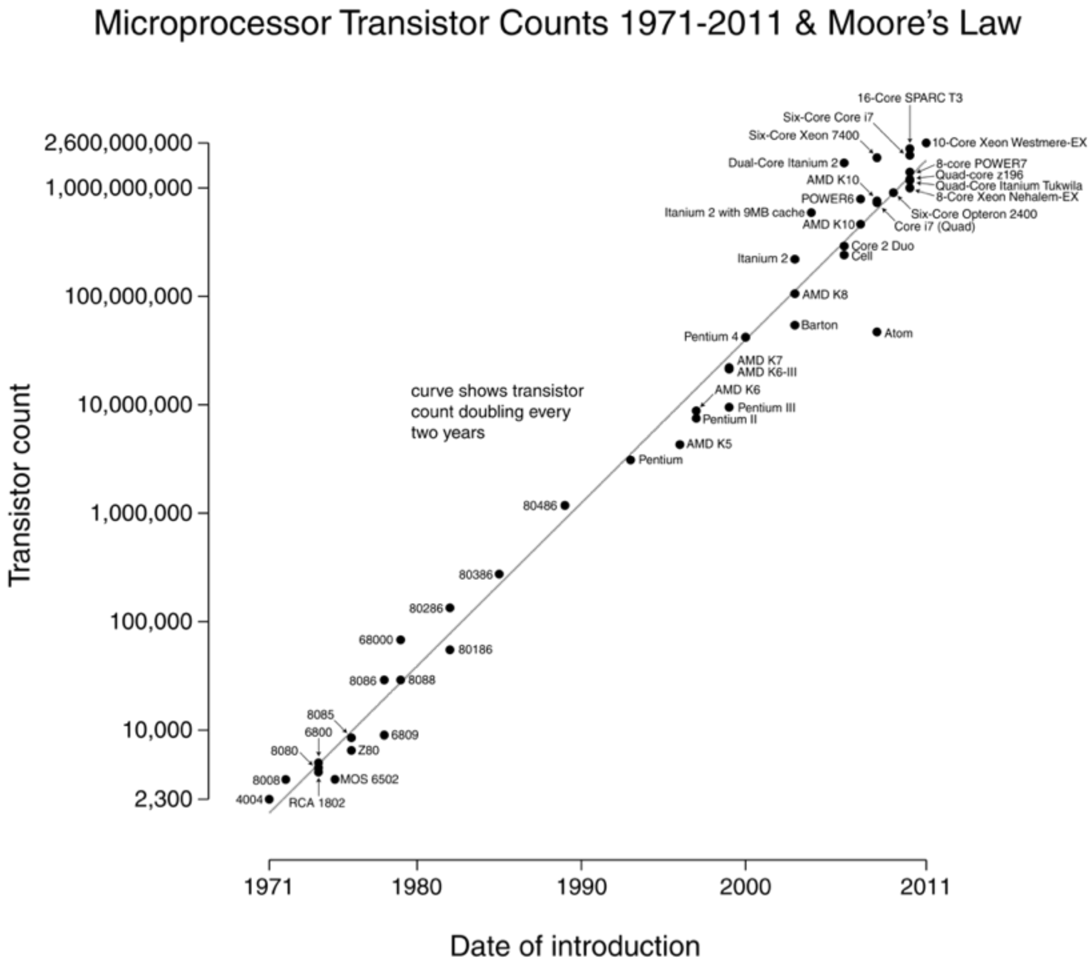
- **Supercomputers: Get maximum performance from available hardware**
- **Data center clusters: Maximize cost/performance over variety of data-centric tasks**
- **Yielded different approaches to hardware, runtime systems, and application programming**

A Convergence Would Have Important Benefits

- **Computational *and* data-intensive applications**
- **But, not clear how to do it**

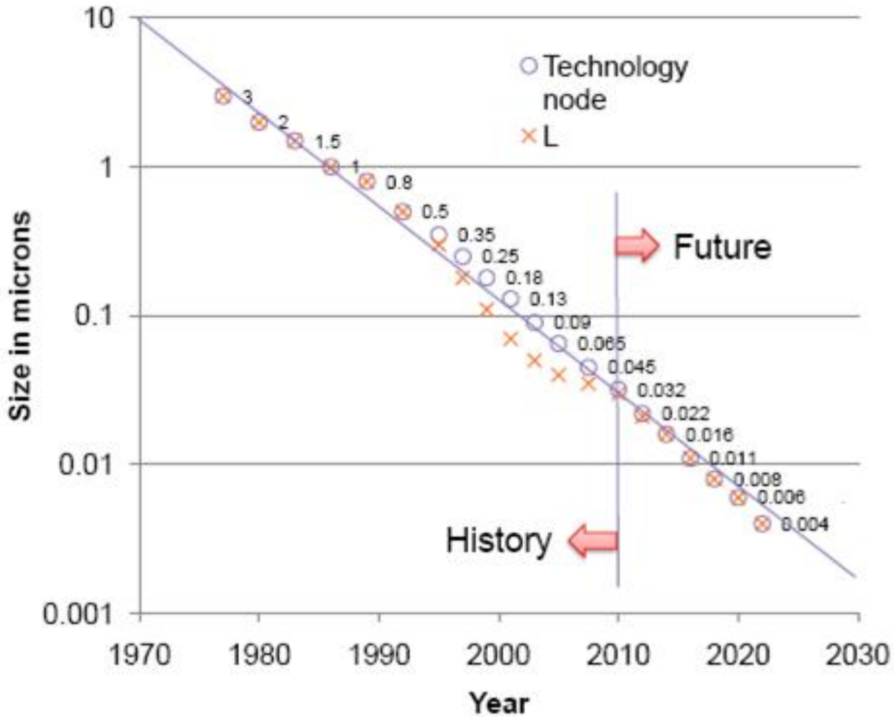
TECHNOLOGY CHALLENGES

Moore's Law



- **Basis for ever-increasing computer power**
- **We've come to expect it will continue**

Challenges to Moore's Law: Technical



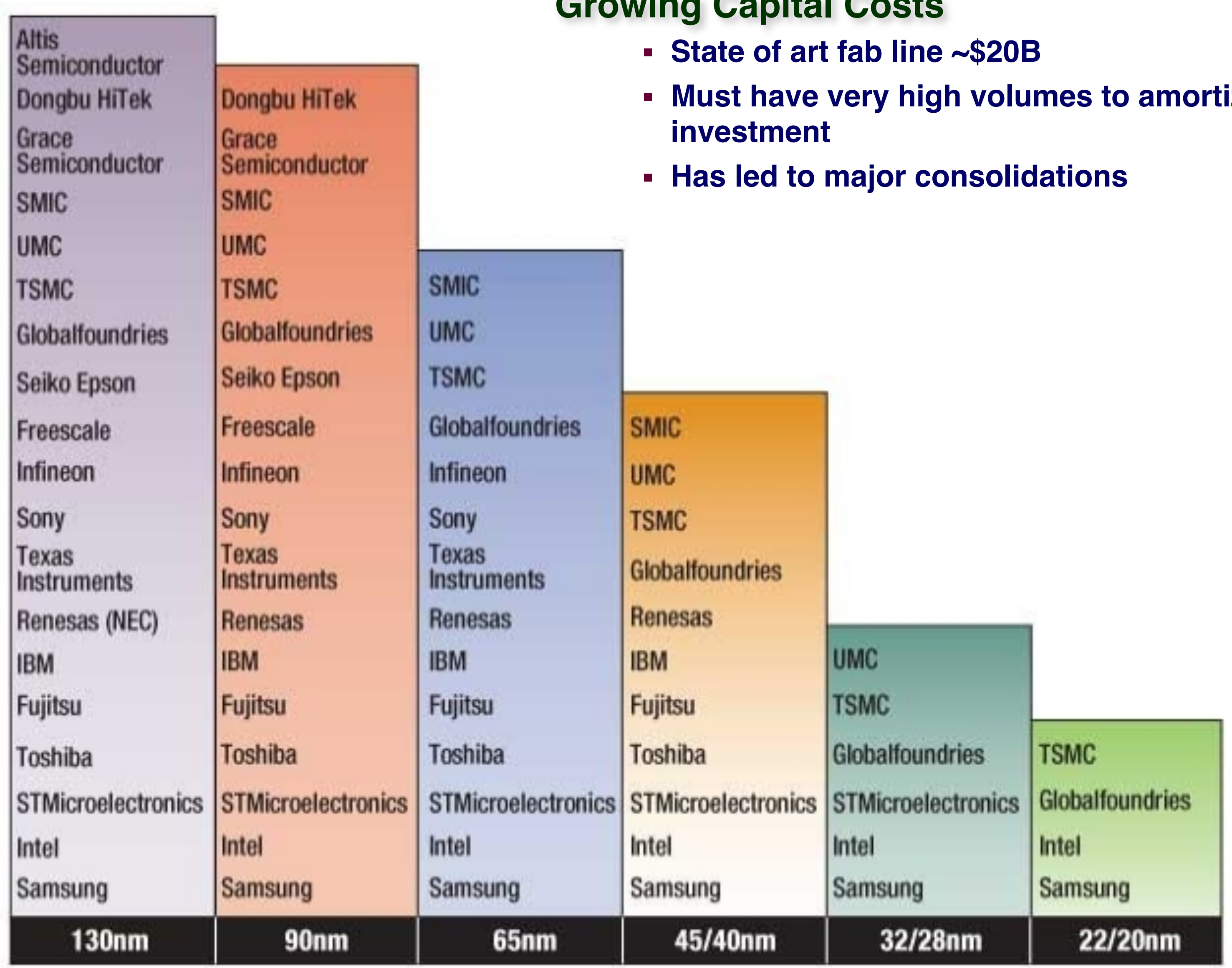
- 2022: transistors with 4nm feature size
- Si lattice spacing 0.54nm

- Must continue to shrink features sizes
- Approaching atomic scale

Difficulties

- Lithography at such small dimensions
- Statistical variations among devices

Challenges to Moore's Law: Economic



Growing Capital Costs

- **State of art fab line ~$20B**
- **Must have very high volumes to amortize investment**
- **Has led to major consolidations**

Dennard Scaling

- Due to Robert Dennard, IBM, 1974
- Quantifies benefits of Moore's Law

How to shrink an IC Process

- Reduce horizontal and vertical dimensions by *k*
- Reduce voltage by *k*

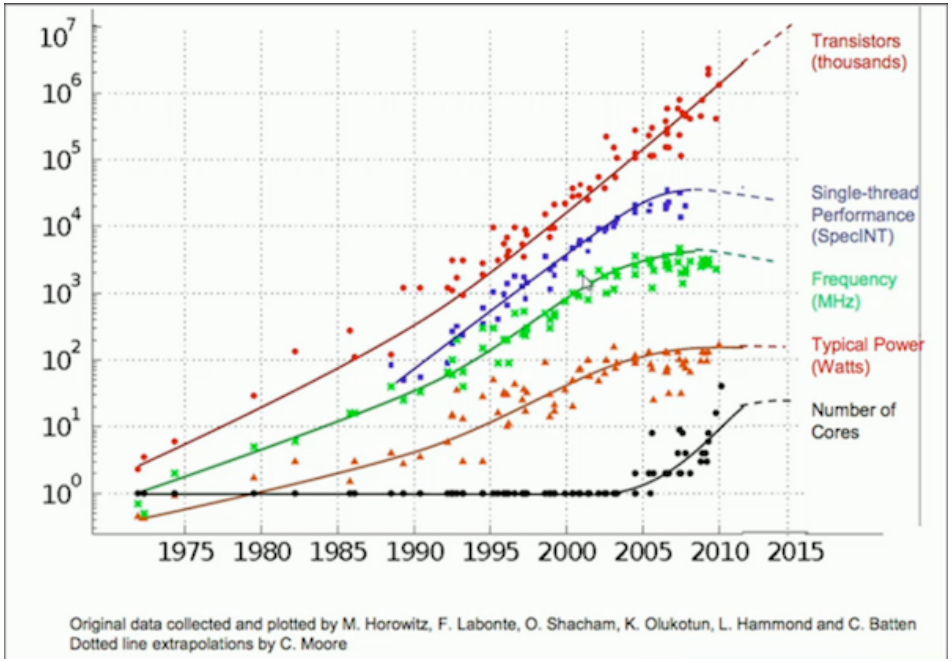
Outcomes

- Devices / chip increase by k^2
- Clock frequency increases by *k*
- Power / chip constant

Significance

- Increased capacity and performance
- No increase in power

End of Dennard Scaling



What Happened?

- **Can't drop voltage below ~1V**
- **Reached limit of power / chip in 2004**
- **More logic on chip (Moore's Law), but can't make them run faster**
  - Response has been to increase cores / chip

Research Challenges

Supercomputers

- **Can they be made more dynamic and adaptive?**
  - Requirement for future scalability
- **Can they be made easier to program?**
  - Abstract, machine-independent programming models

Data-Intensive Computing

- **Can they be adapted to provide better computational performance?**
- **Can they make better use of data locality?**
  - Performance & power-limiting factor

Technology / Economic

- **What will we do when Moore's Law comes to an end for CMOS?**
- **How can we ensure a stable manufacturing environment?**