Software for Exascale

Pete Beckman

Argonne National Laboratory

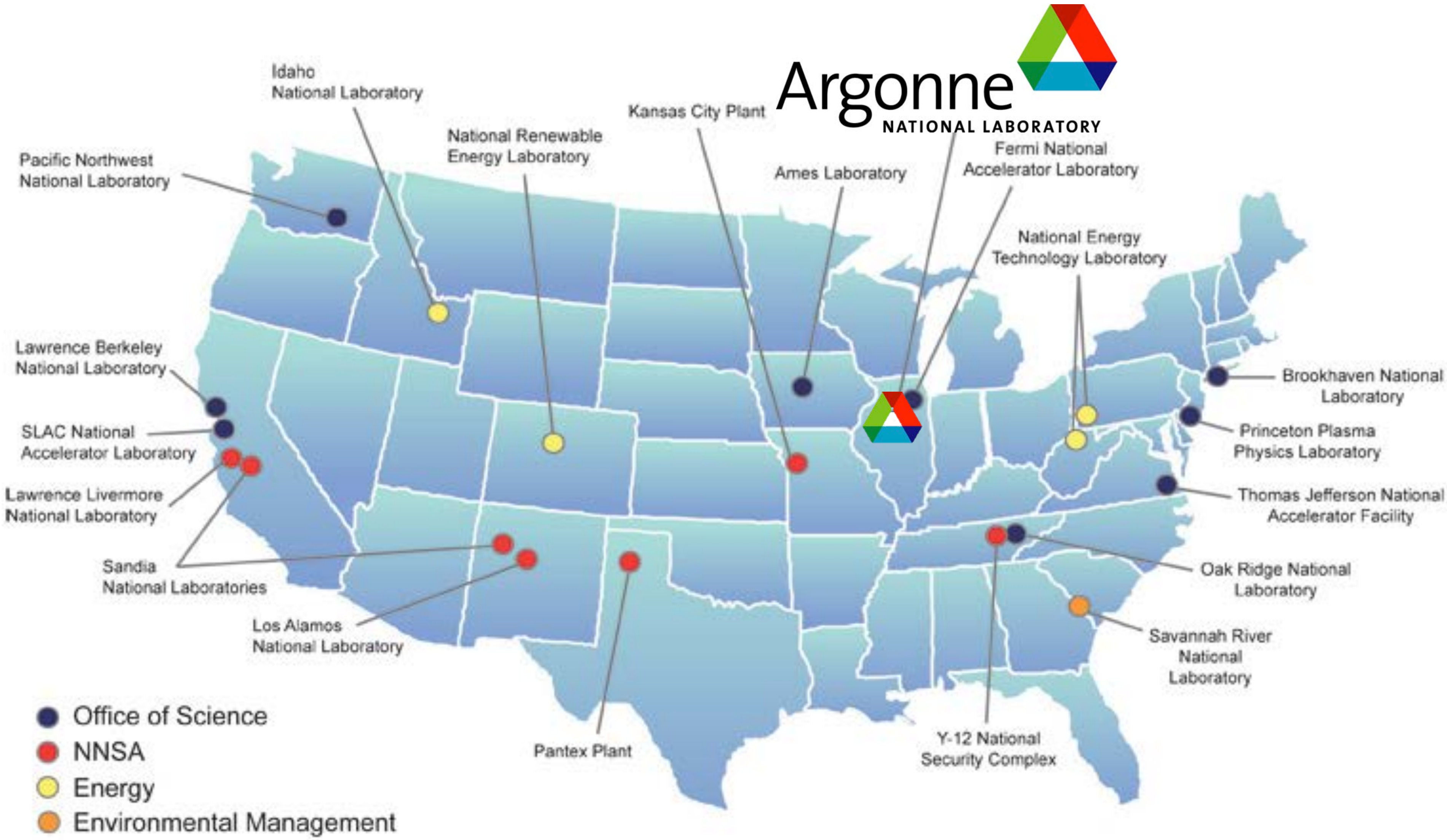
Northwestern University

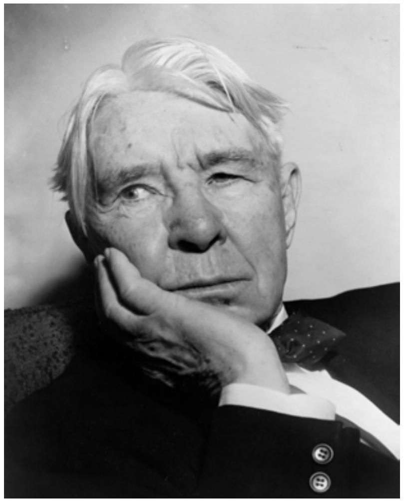
Exascale Applications and Software Conference

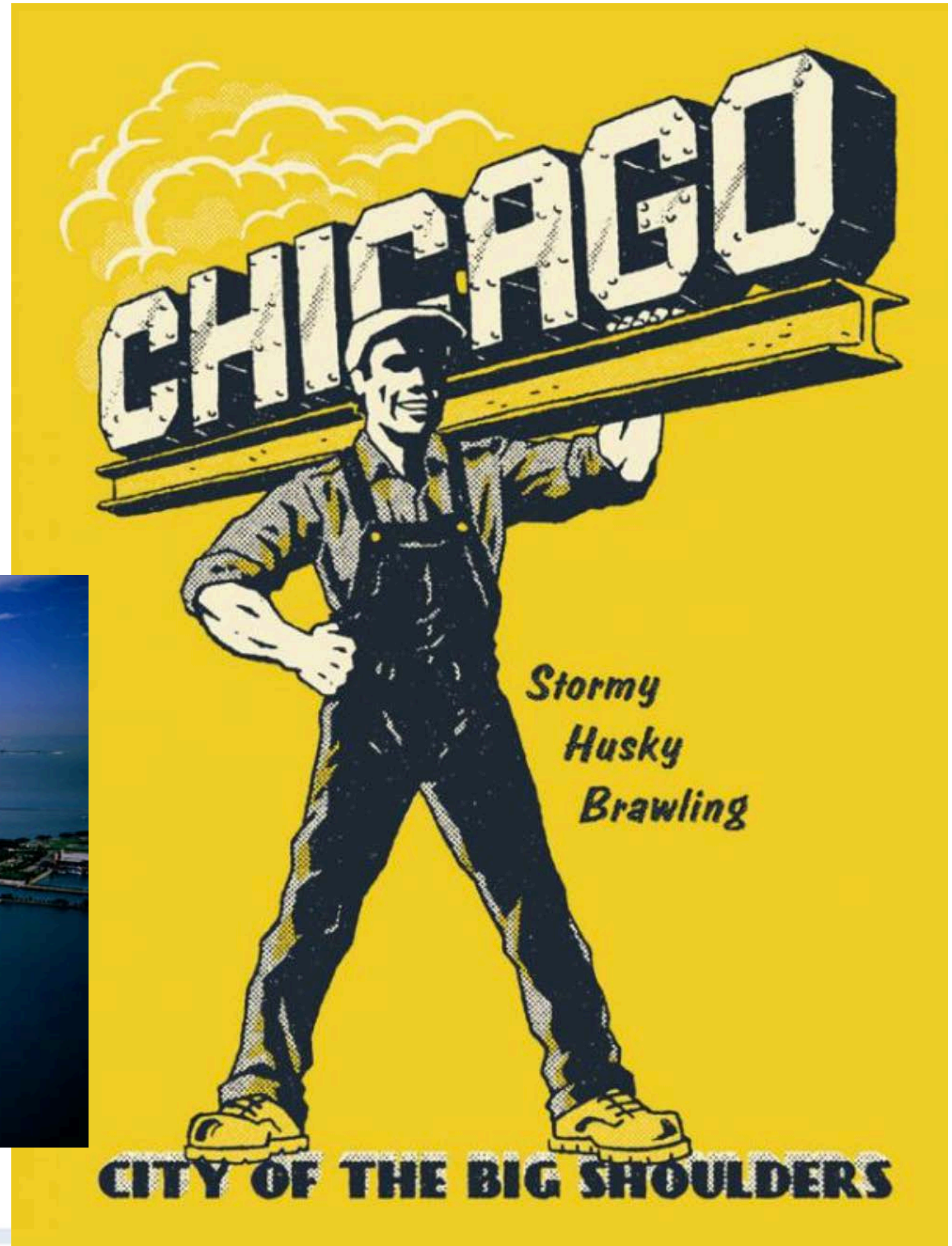
EASC 2015, Edinburgh

April 21, 2015

Argonne: Part of DOE National Laboratory System






CHICAGO
Stormy
Husky
Brawling
CITY OF THE BIG SHOULDERS

Argonne National Laboratory
$675M /yr budget
3,200 employees
1,450 scientists/eng
750 Ph.D.s

HPC has been pretty successful…

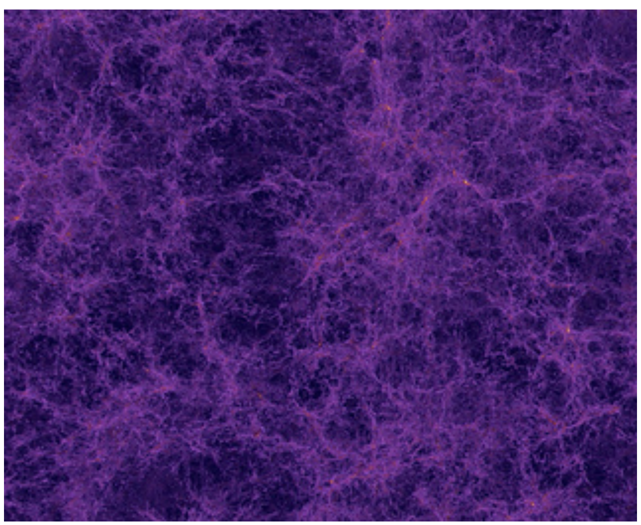
Tianhe-2

Sequoia

K Computer

Example: HACC Cosmology Code

- HACC cosmology code from Argonne (Salman Habib) achieved **14 PFlops/s** on Sequoia (Blue Gene/Q at LLNL)
  - Ran on full Sequoia system using MPI + OpenMP hybrid
  - Used 16 MPI ranks * 4 OpenMP threads per rank on each node, which matches the architecture: 16 cores per node with 4 hardware threads each
  - **~ 6.3 million way concurrency: 1,572,864 MPI ranks * 4 threads/rank**
  - http://www.hpcwire.com/hpcwire/2012-11-29/sequoia_supercomputer_runs_cosmology_code_at_14_petaflops.html
  - SC12 Gordon Bell prize finalist



The HACC code has been used to run one of the largest cosmological simulations ever, with 1.1 trillion particles

Old Wisdom: Moore's Law = free exponential speedups!

THE WALL STREET JOURNAL.

WSJ

WEEKEND

SATURDAY/SUNDAY, APRIL 18 - 19, 2015

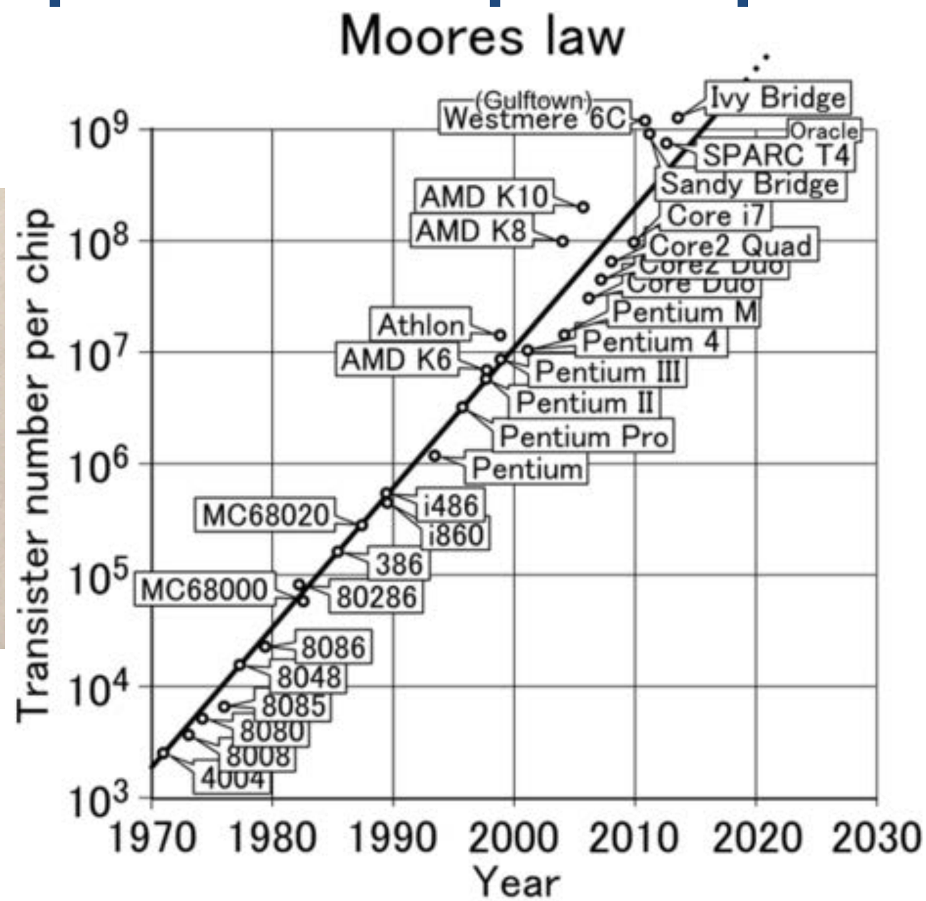
Turning 50, Tech Axiom Moore's Law Shows Age

By Don Clark

But the industry's costs keep rising, with new chip-fabrication plants costing as much as $10 billion. Cost pressures led International Business Machines Corp. last year to pay $1.5 billion to another company to take over its semiconductor operations.

Companies that can afford to keep pushing Moore's Law are finding it increasingly hard to keep up the pace. Intel's introduction of 14-nanometer technology was two quarters late be-

While companies say they likely can keep shrinking the size of silicon chips for another decade or so, that work is bringing diminishing financial returns. Some chip designers already are limiting their use of the newest technology to high-end products where performance is more important than cost.

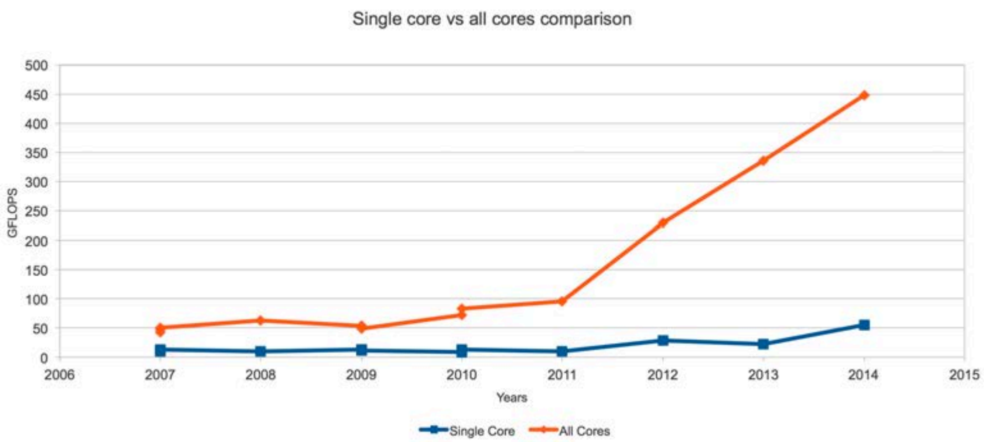


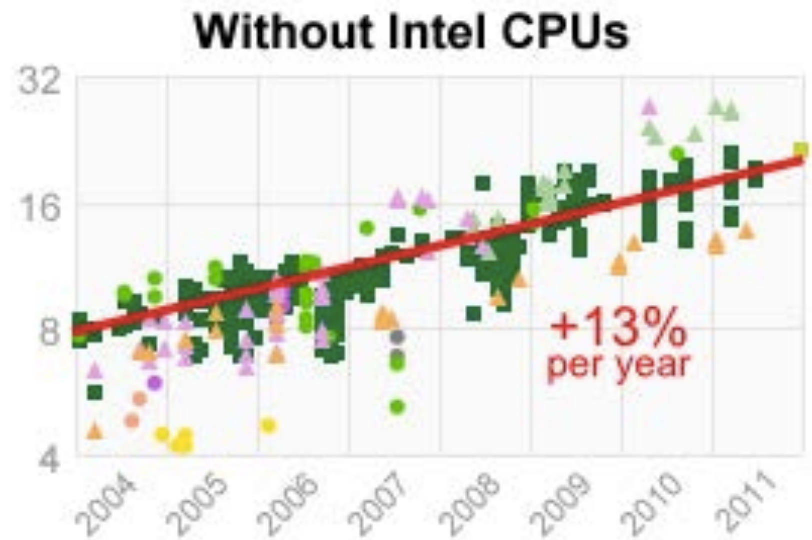
Intel: Moore's Law will continue through 7nm chips

Mark Hachman | @markhachman
Senior Editor, PCWorld
Feb 22, 2015 12:00 PM

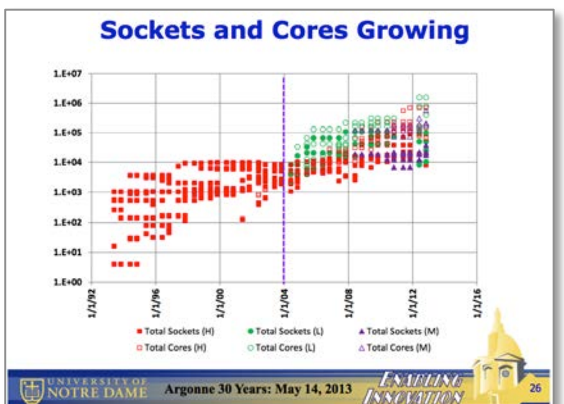
Reality: Computing improvements have slowed dramatically over the past Decade

Transistors you can buy for a fixed # of dollars in leading technology is no longer increasing!





Single thread performance improvement is slow. (Specint)



"Herbert Stein's Law: "If something cannot go on forever, it will stop,"

*"Intel has done a little better over this period, Increasing at 21% per year.

*"No Moore?", Economist, Nov 2013.
Src: Linley Group

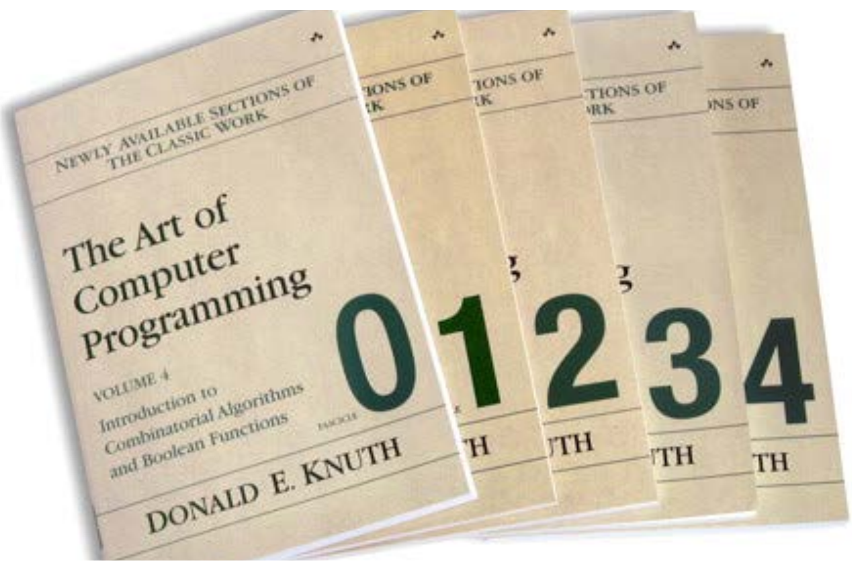
Courtesy: Andrew Chien

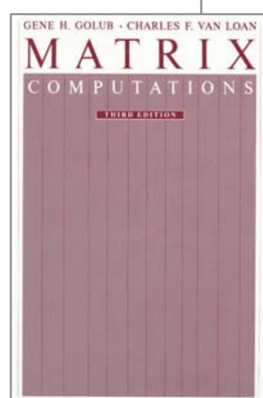
Old Wisdom:
Efficient Algorithms minimize operations

Classic Analysis of Algorithms: Ops = Time

Make algorithm quicker: minimize flops, compares

Ops: Best, Worst, Average, *Space*





1996

1.4.5 Thinking About Data Motion

Another important attribute of a matrix algorithm concerns the actual volume of data that has to be moved around during execution. Matrices sit in memory but the computations that involve their entries take place in functional units. The control of memory traffic is crucial to performance in many computers. To continue with the factory metaphor used at the beginning of this section: *Can we keep the superfast arithmetic units busy with enough deliveries of matrix data and can we ship the results back to memory fast enough to avoid backlog?* FIG. 1.4.3 depicts the typical situation in an advanced uniprocessor environment. Details vary from machine

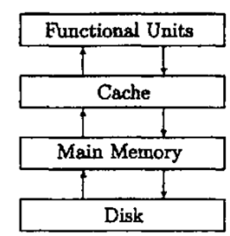


FIG. 1.4.3 *Memory Hierarchy*

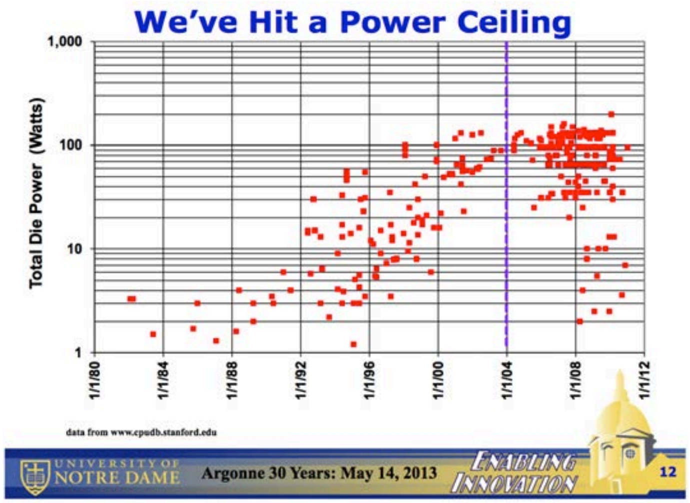
chine, but two "axioms;; prevail:

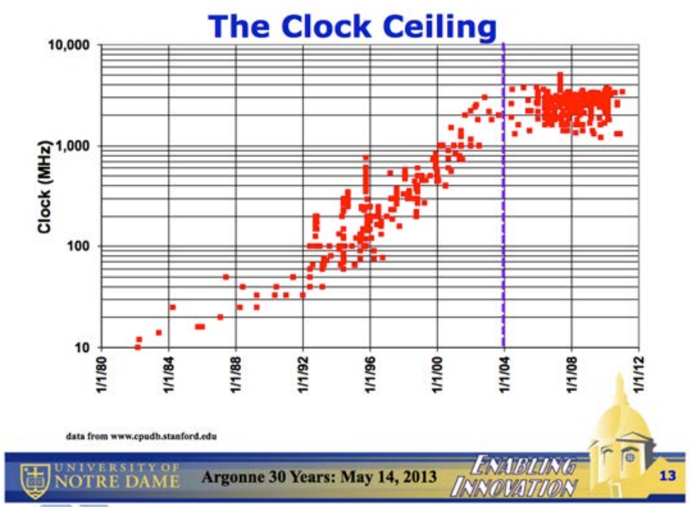
Each level in the hierarchy has a limited capacity and for economic reasons this capacity is usually smaller as we ascend the hierarchy.

There is a cost, sometimes relatively great, associated with the moving of data between two levels in the hierarchy.

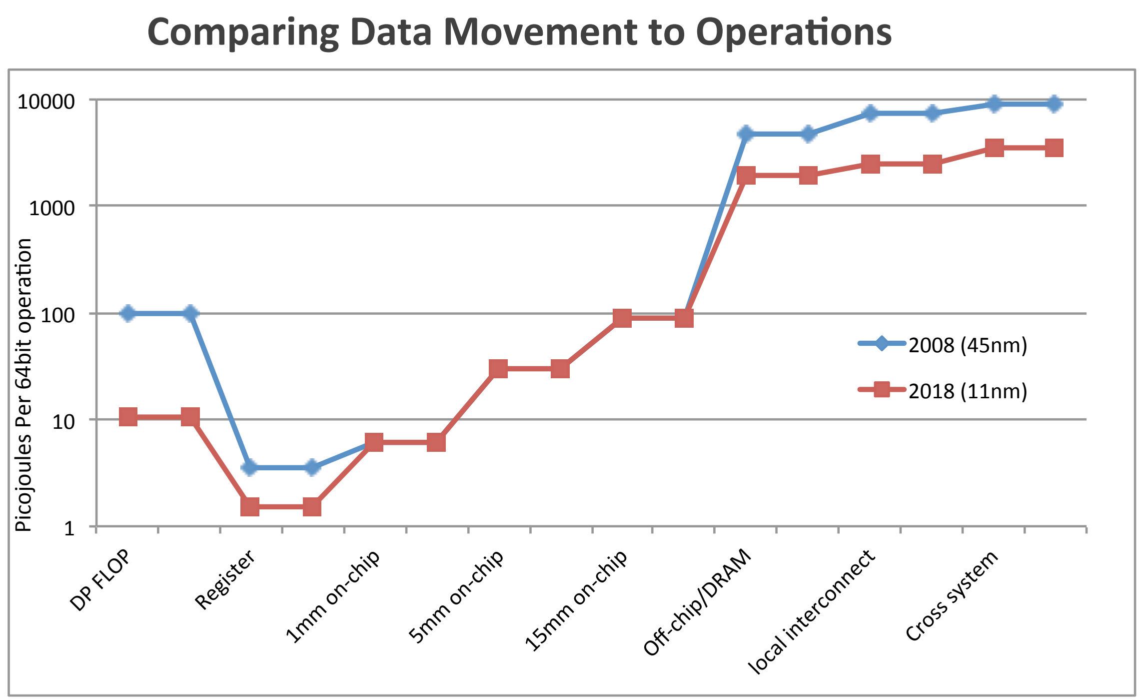
The design of an efficient matrix algorithm requires careful thinking about the flow of data in between the various levels of storage. The vector touch and data re-use issues are important in this regard.

Reality:
Efficient = optimize data movement (and power)





Courtesy: Peter Kogge



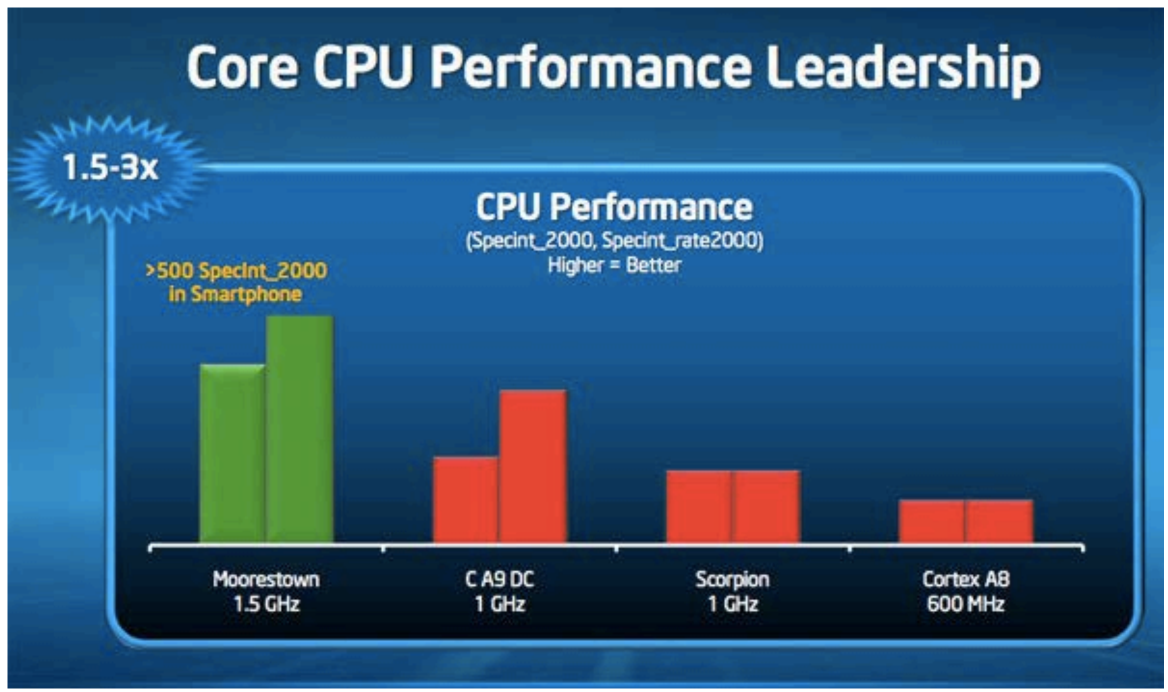
Courtesy: John Shalf

Pipelining, load/store, GPGU…

Old Wisdom:
Parallel Algorithms: Equal Work = Equal Time
(computers run at predictable speeds)

SPMD Code: Divide data into equal sized chucks across *p* processors

```
For all timesteps {
    exchange data with neighbors
    compute on local data
    barrier
}
```

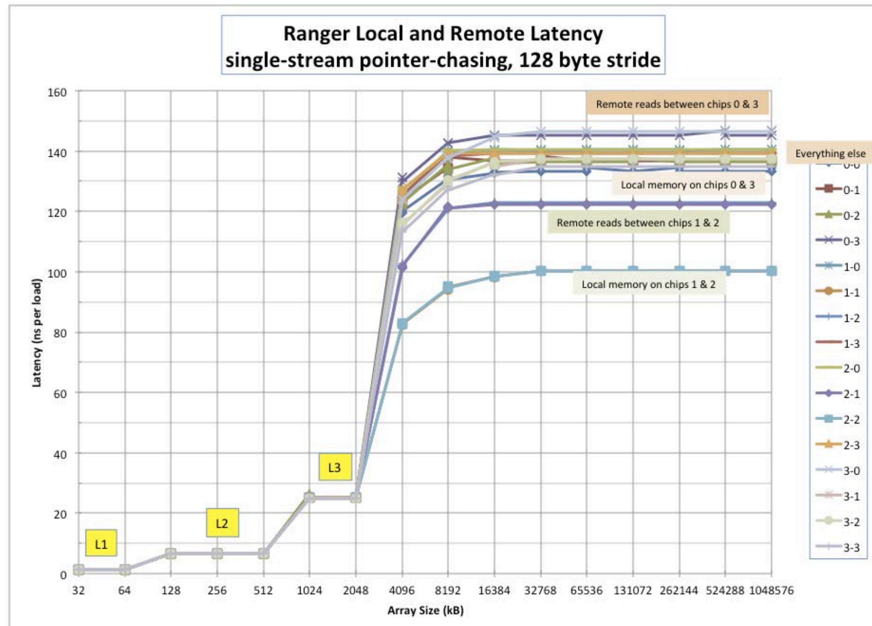


Courtesy: Andrew Chien

Reality: Performance is Highly Variable

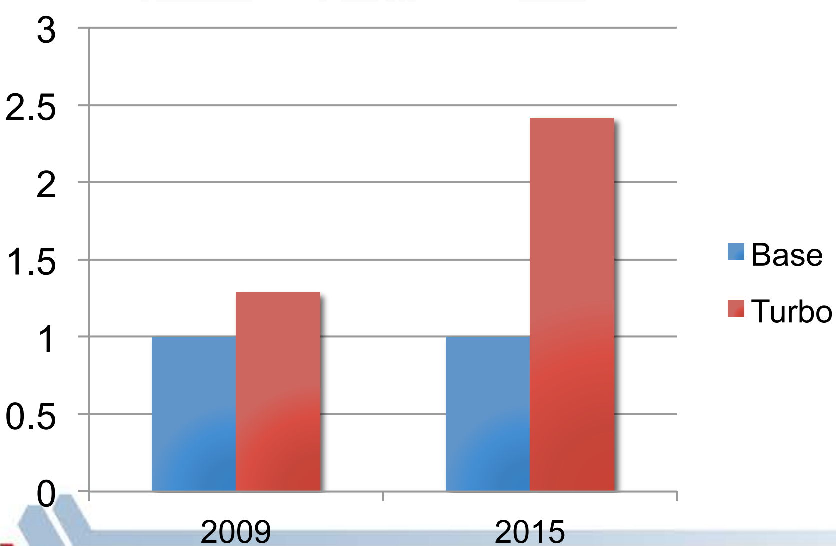
≠

Memory Hierarchy Depth (1-150-?)



Courtesy: McCalpin

Turbo Boost (1.2-2.5-?)



+ new Non-volatile memory (3,000 cycles)

+ old Non-volatile memory Flash (150,000 cycles)

Courtesy: Andrew Chien

The New Exascale Reality

- ~~Computing rapidly gets faster and cheaper for free~~
  - Rapid exponential improvement is over, slow improvement will continue for awhile... Parallelism explodes, SQUEEEEZE!
- ~~Efficient programs minimize operations~~
  - More operations can better, optimize for locality, data movement, power
- ~~Computers run at fixed, predictable speed~~
  - Increasing dynamic and flexible, complication and advantage

What Prevents Scalability?

(in the large and in the small)

- **Insufficient parallelism**
  - As the problem scales, more parallelism must be found
- **Insufficient latency hiding**
  - As the problem scales, more latency must be hidden
- **Insufficient resources** (Memory, BW, Flops)
  - As the problem scales, so must the resources needed

What Prevents Scalability?

(in the large and in the small)

- **Insufficient parallelism**
  - As the problem scales, more parallelism must be found
- **Insufficient latency hiding**
  - As the problem scales, more latency must be hidden
- **Insufficient resources** (Memory, BW, Flops)
  - As the problem scales, so must the resources needed

As we scale machine, system becomes more dynamic
As we squeeze power, system becomes more dynamic
As we address resilience, system becomes more dynamic
As we share networks, system becomes more dynamic

… Google (re-discovers) OS Noise & Contention

Component-Level Variability Amplified By Scale

A common technique for reducing latency in large-scale online services is to parallelize sub-operations across many different machines, where each sub-operation is co-located with its portion of a large dataset. Parallelization happens by fanning out a request from a root to a large number of leaf servers and merging responses via a request-distribution tree. These sub-operations must all complete within a strict deadline for the

Living with Latency Variability

The careful engineering techniques in the preceding section are essential for building high-performance interactive services, but the scale and complexity of modern Web services make it infeasible to eliminate all latency variability. Even if such perfect behavior could

Reducing Component Variability

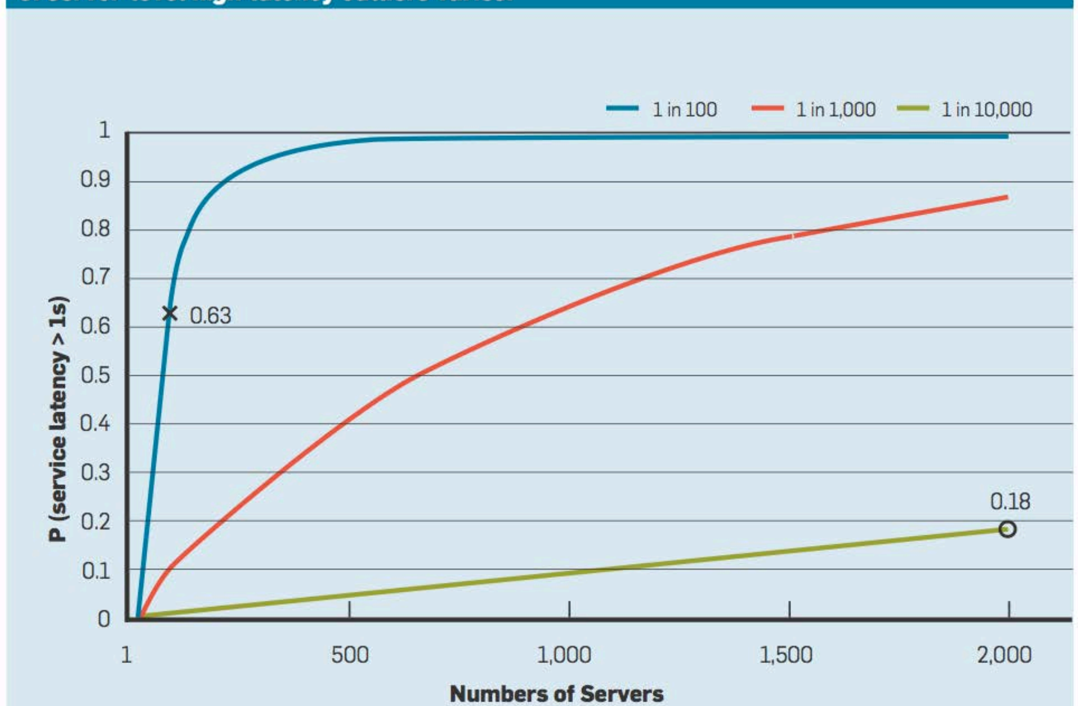
Interactive response-time variability can be reduced by ensuring interactive requests are serviced in a timely manner

Software techniques that tolerate latency variability are vital to building responsive large-scale Web services.

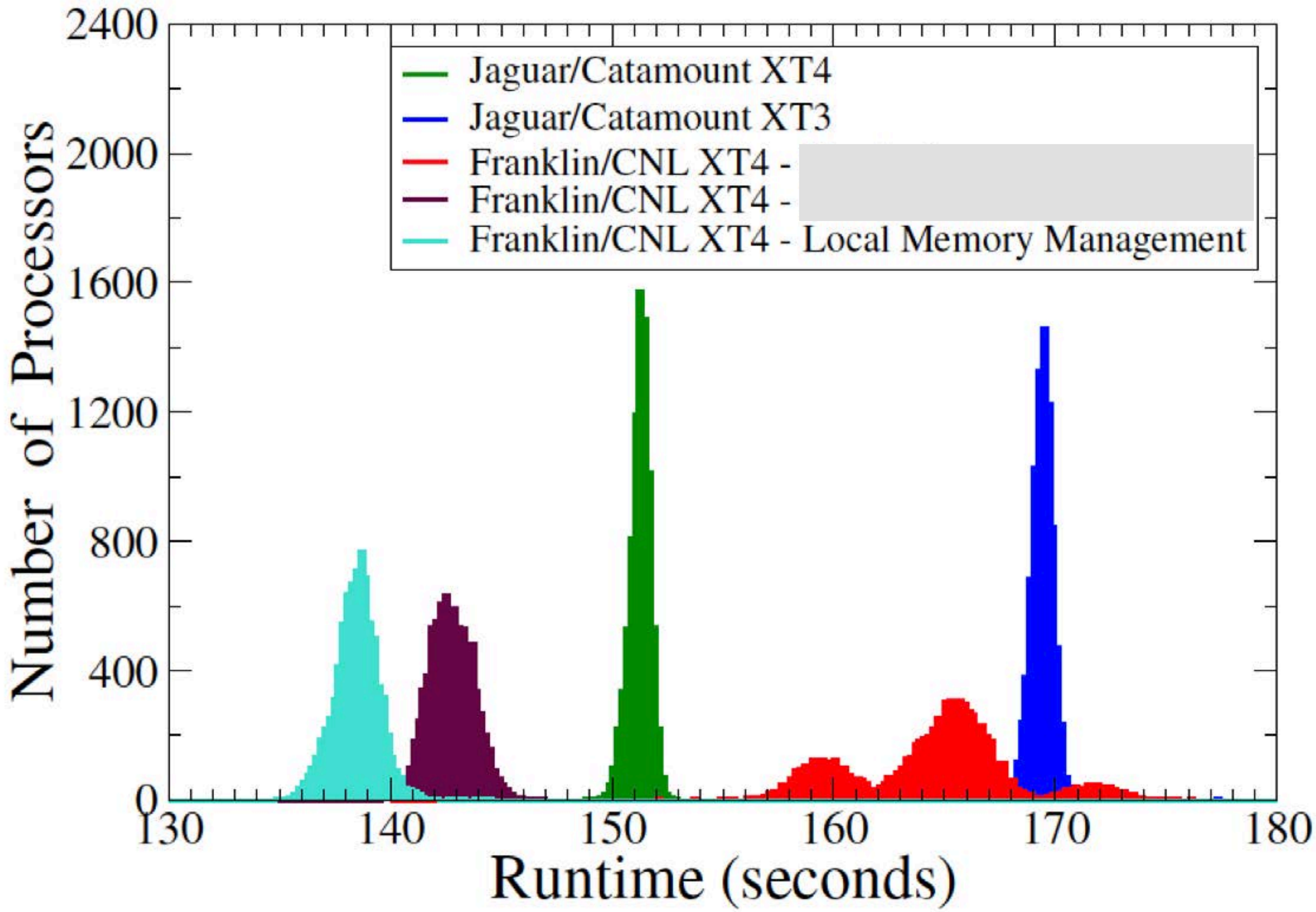
BY JEFFREY DEAN AND LUIZ ANDRÉ BARROSO

The Tail at Scale

Probability of one-second service-level response time as the system scales and frequency of server-level high-latency outliers varies.



Exploring Dynamic

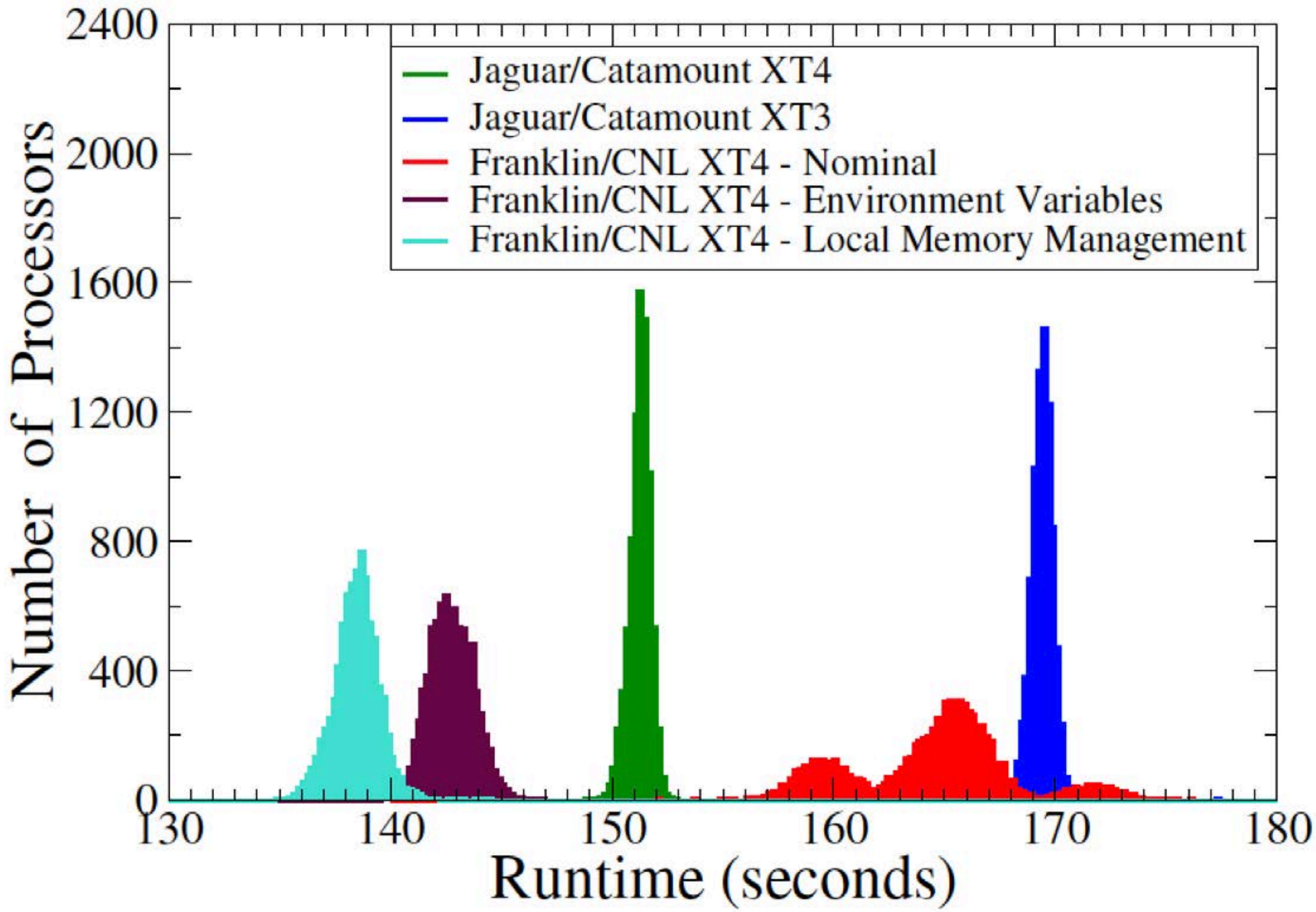


Dynamic Choices: Fast and Variable….. Slow and Steady…

"Scalability challenges for massively parallel AMR applications", IPDPS 2009

Credit: Brian Van Straalen, John Shalf

Exploring Dynamic

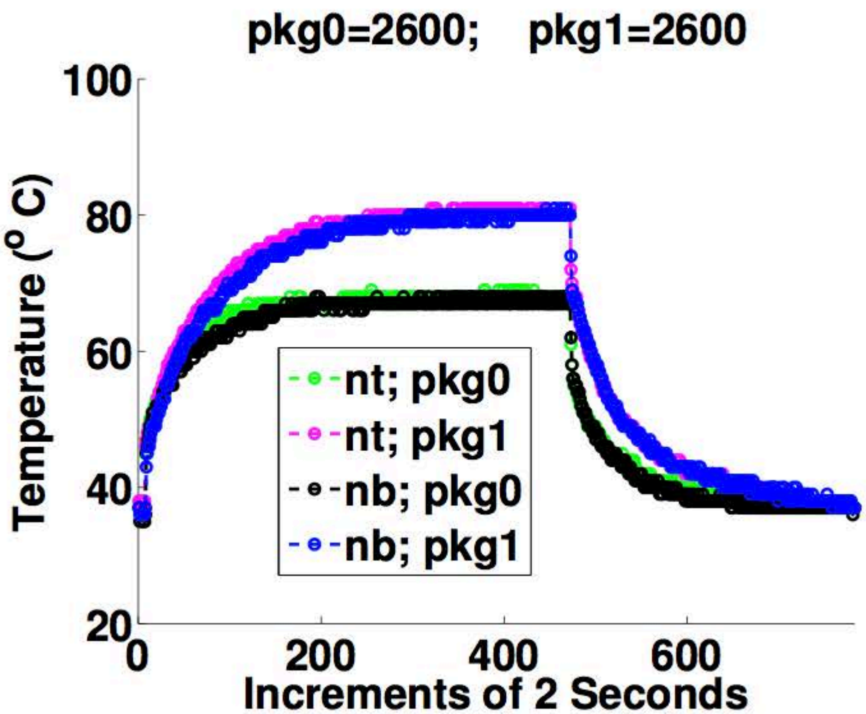


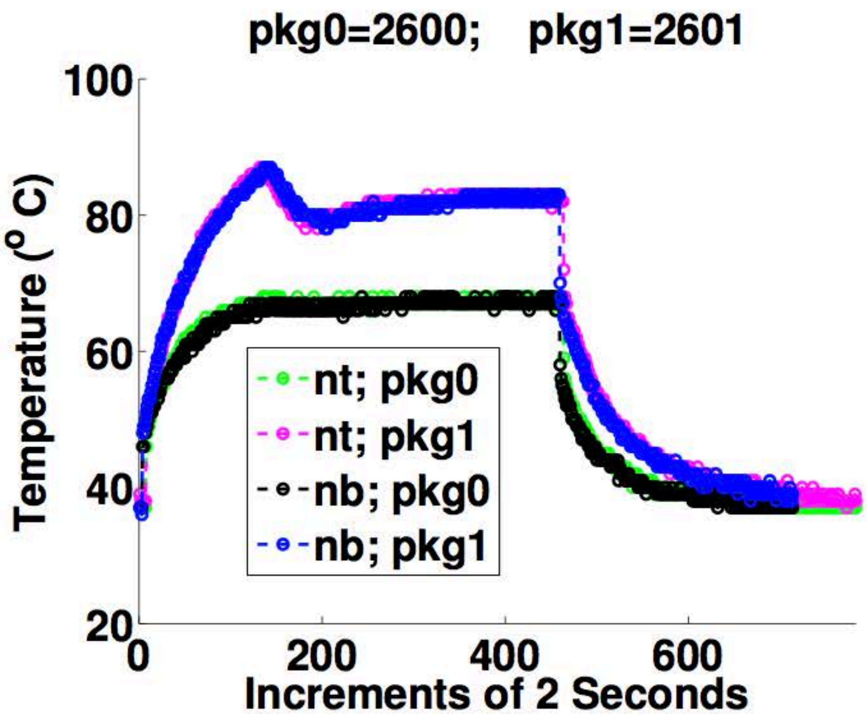
Dynamic Choices: Fast and Variable….. Slow and Steady…

"Scalability challenges for massively parallel AMR applications", IPDPS 2009

Credit: Brian Van Straalen, John Shalf

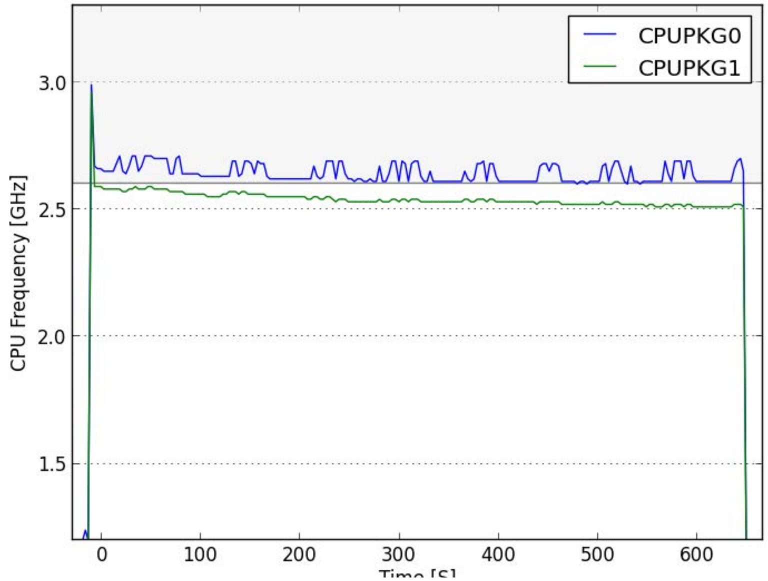
Dynamic Power and Temp from Turboboost

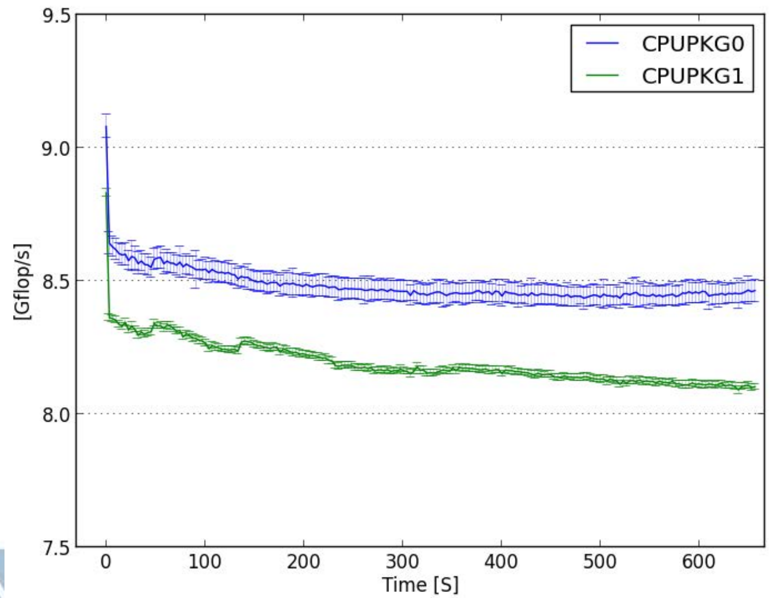




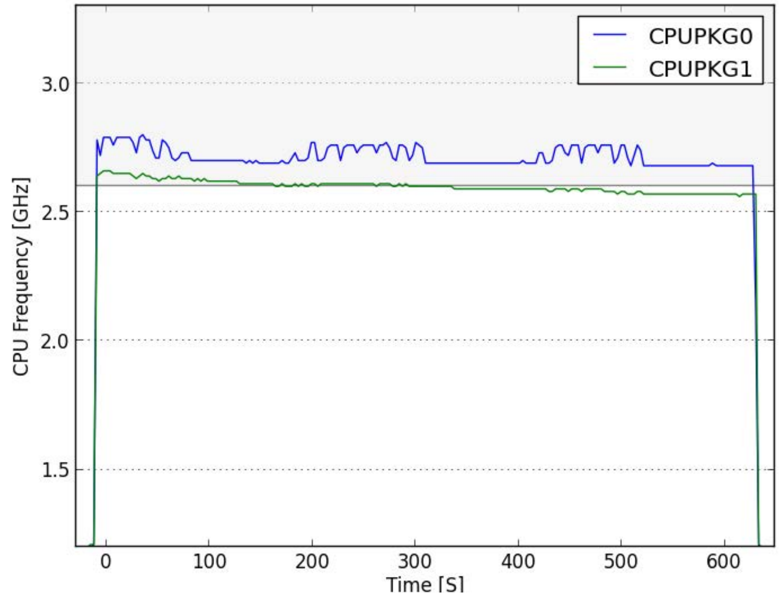
We live with dynamic now… More examples

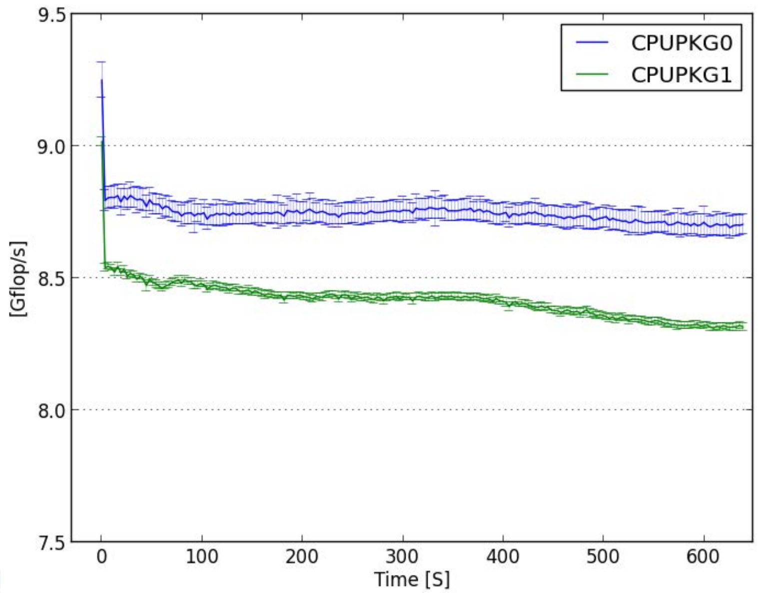
110W Limit





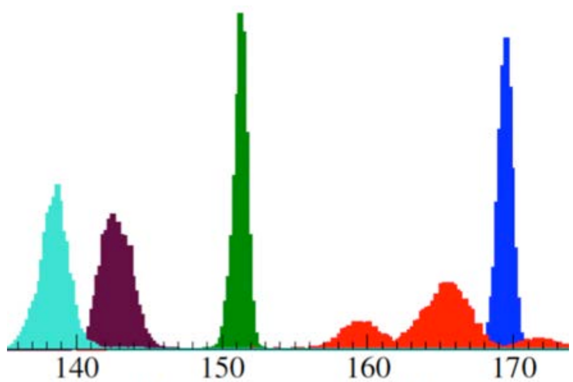
115W Limit





Our Hardware is Dynamic, Adaptive Today!
(the future is even more dynamic)

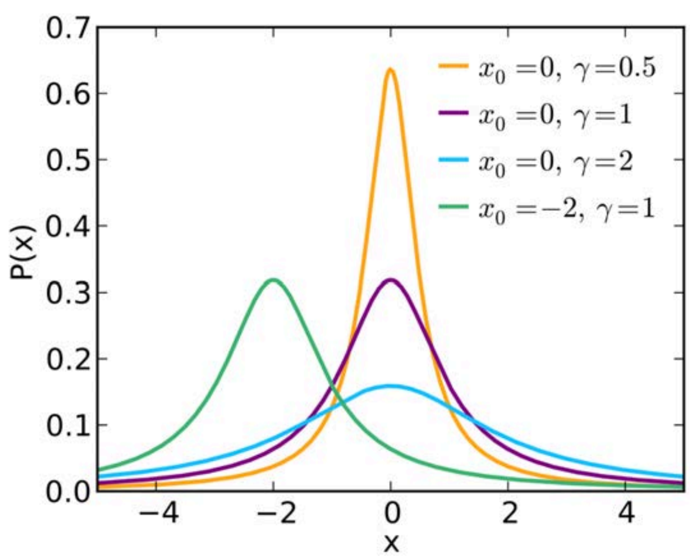
- **Bulk Synchronous is our scaling problem**
  - **≠MPI (library that moves data with put/get or send/recv)**
  - **We must focus on dynamic behavior**
- **"OS Noise" and "jitter" is a legacy distraction**
  - **OS & Runtime must be VERY active…**
  - **Forget that old-school "get out of the way" stuff**
- **Load balancing is necessary, but not sufficient…**

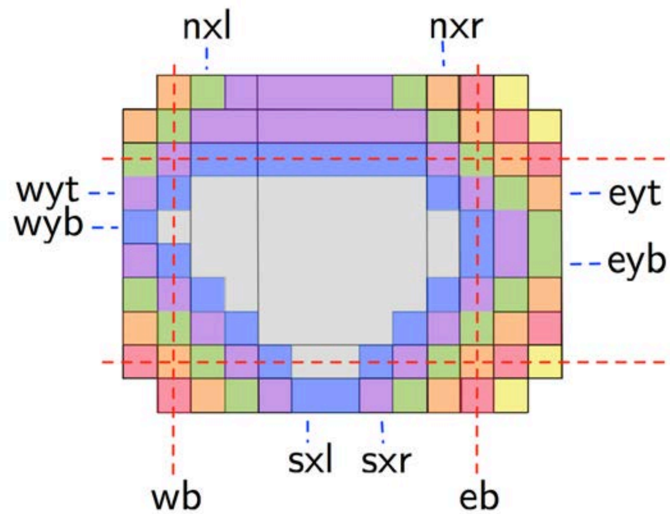


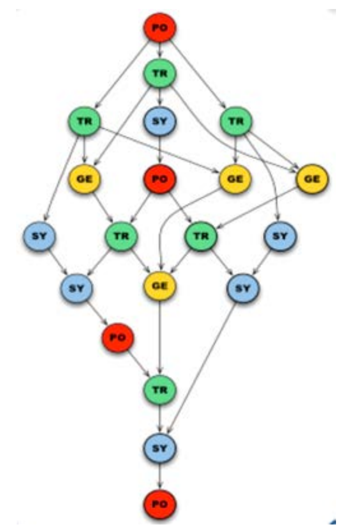
- How do we design software in this new era?
- How do we build latency tolerant algs?
- Can we create tools that measure, learn, predict, and then improve performance?

How Pliable does node code need to be?
How do we measure pliability?

- What is the shape of performance distribution?
- How much latency do we need to hide?
- What is the cost of dynamic execution?
- Can we build in predictive models?



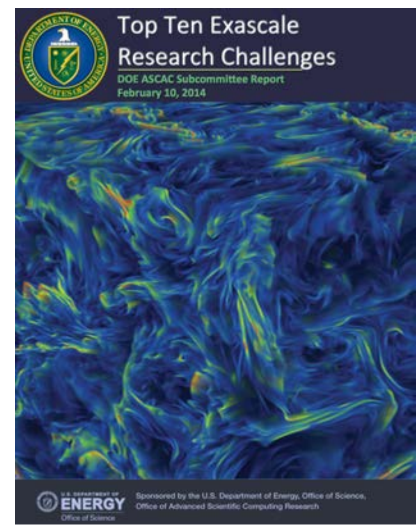




But yet, We Pretend our World is Not Dynamic

- **Trinity/NERSC-8:** ?

"The system shall provide correct and consistent runtimes. An application's runtime (i.e. wall clock time) shall not change by more than 3% from run-to-run in dedicated mode and 5% in production mode."

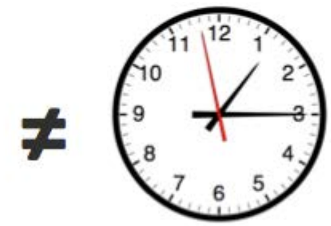


ASCAC Top 10 Research Challenges for Exascale

- "[…] power management [..] through dynamic adjustment of system balance to fit within a fixed power budget"
- […] Enabling […] dynamic optimizations […] (power, performance, and reliability) will be crucial to scientific productivity. "
- " […] Next-generation runtime systems are under development that support different mixes of several classes of dynamic adaptive functionality. "

"dynamic" mentioned 43 times in 86 pg report

Exascale Lesson:



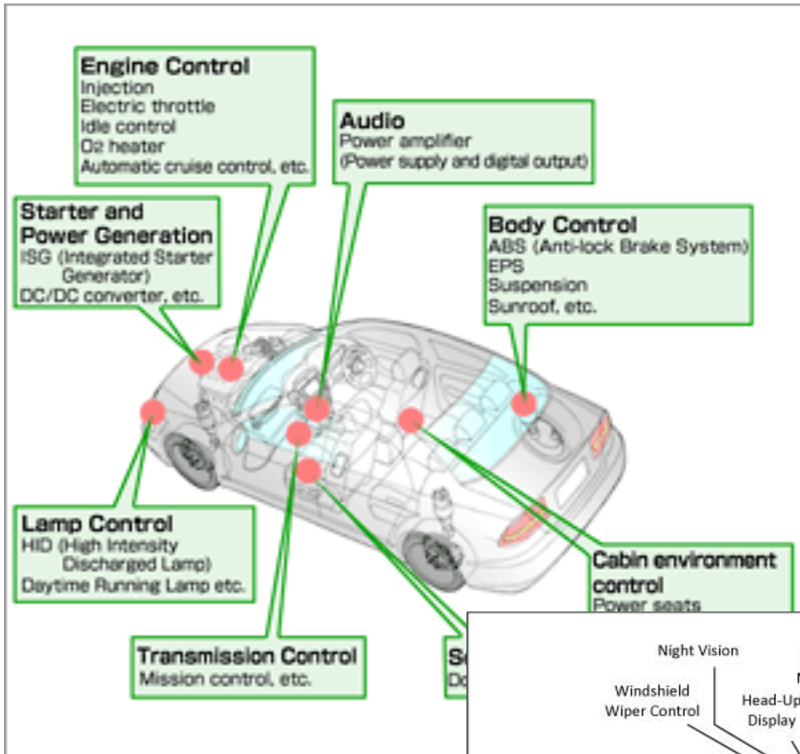
- **Code should be as static as possible, but no more so**
- 1) Prepare: Create flexibility via over-decomposition, clear expression of dependencies
- 2) Take small steps to becoming more pliable…. statically
  - (static) mapping of resource (slow/fast; heat)
  - (static) load balancing (periodic repartitioning)
  - (static) dependency graph tiling of stencils to match communication
- 3) Find ***goal-oriented optimization***
  - Dynamic lightweight work-sharing
  - Dynamic power management
  - Dynamic data movement across hierarchy

Code should not consider dynamic a performance error (e.g. NERSC)

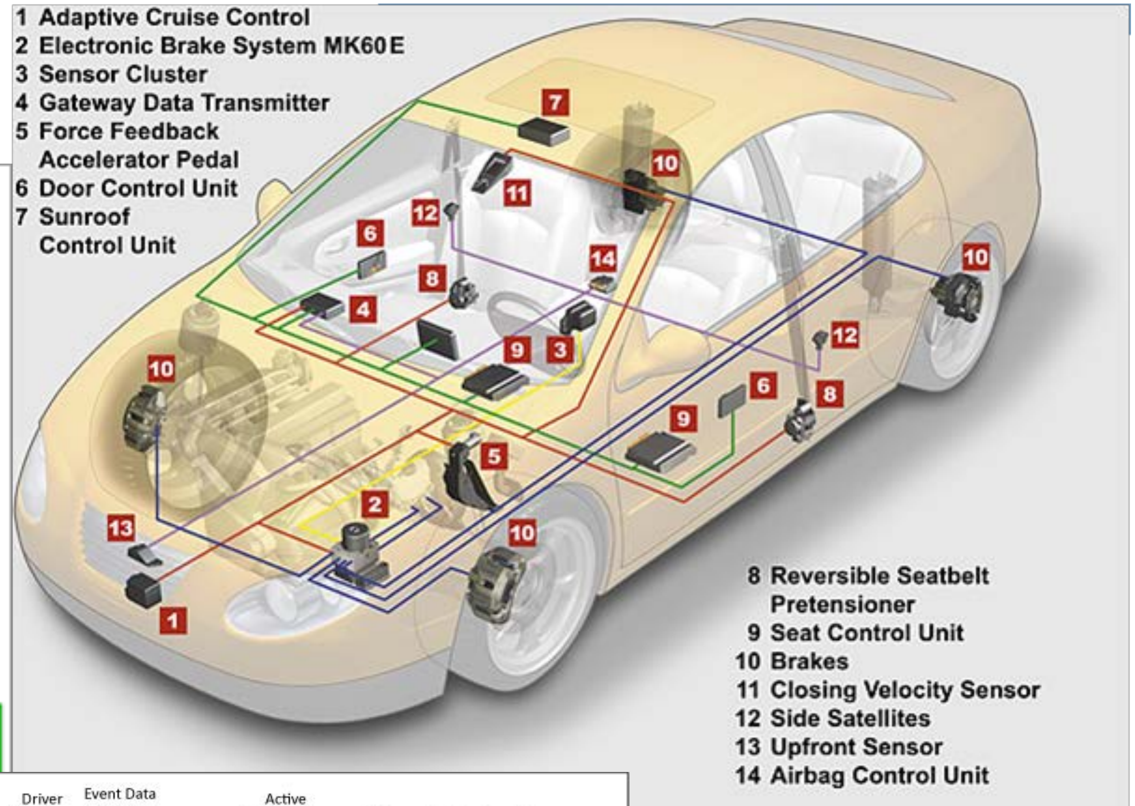

Harold's
TEXACO
BOTHELL, WN.
Ford


HOLLEY
POWERED BY FORD
Stant
CHANGING COOLANT
CUSTOM CLASSIC TRUCKS

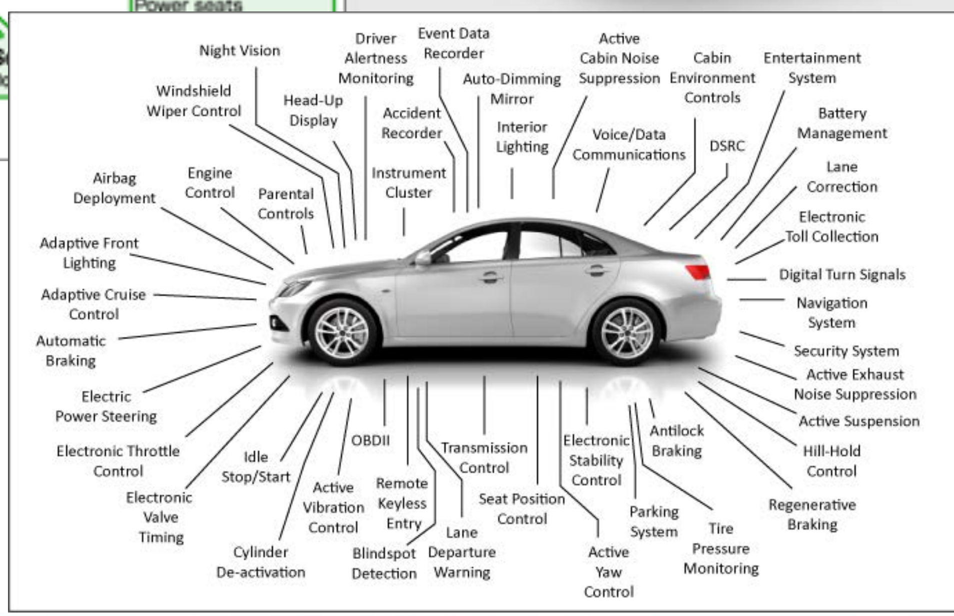

AMG



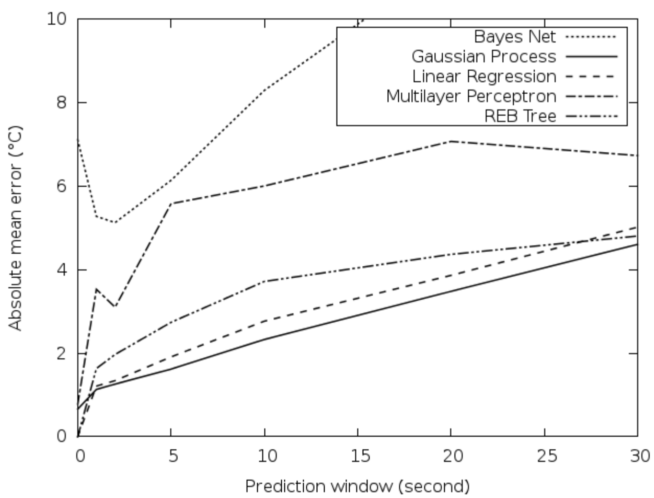
Engine Control
Injection
Electric throttle
Idle control
O2 heater
Automatic cruise control, etc.
Audio
Power amplifier
(Power supply and digital output)
Starter and Power Generation
ISG (Integrated Starter Generator)
DC/DC converter, etc.
Body Control
ABS (Anti-lock Brake System)
EPS
Suspension
Sunroof, etc.
Lamp Control
HID (High Intensity Discharged Lamp)
Daytime Running Lamp etc.
Cabin environment control
Power seats
Transmission Control
Mission control, etc.

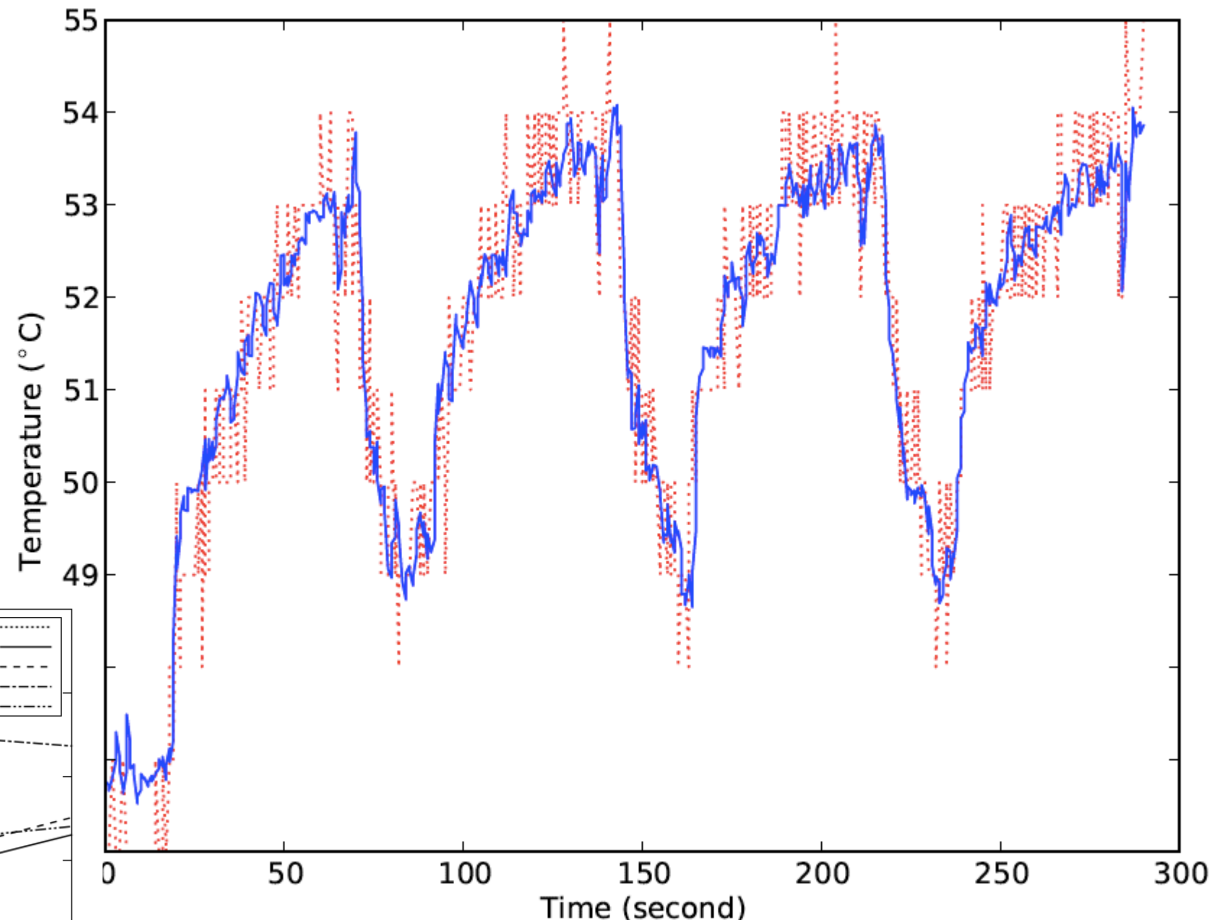


1 Adaptive Cruise Control
2 Electronic Brake System MK60E
3 Sensor Cluster
4 Gateway Data Transmitter
5 Force Feedback Accelerator Pedal
6 Door Control Unit
7 Sunroof Control Unit
8 Reversible Seatbelt Pretensioner
9 Seat Control Unit
10 Brakes
11 Closing Velocity Sensor
12 Side Satellites
13 Upfront Sensor
14 Airbag Control Unit



Night Vision
Driver Alertness Monitoring
Event Data Recorder
Active Cabin Noise Suppression
Cabin Environment Controls
Entertainment System
Windshield Wiper Control
Head-Up Display
Accident Recorder
Auto-Dimming Mirror
Interior Lighting
Voice/Data Communications
DSRC
Battery Management
Lane Correction
Airbag Deployment
Engine Control
Parental Controls
Instrument Cluster
Electronic Toll Collection
Adaptive Front Lighting
Digital Turn Signals
Adaptive Cruise Control
Navigation System
Automatic Braking
Security System
Active Exhaust Noise Suppression
Electric Power Steering
Active Suspension
Electronic Throttle Control
Idle Stop/Start
OBDII
Transmission Control
Electronic Stability Control
Antilock Braking
Hill-Hold Control
Electronic Valve Timing
Active Vibration Control
Remote Keyless Entry
Seat Position Control
Parking System
Regenerative Braking
Cylinder De-activation
Blindspot Detection
Lane Departure Warning
Active Yaw Control
Tire Pressure Monitoring



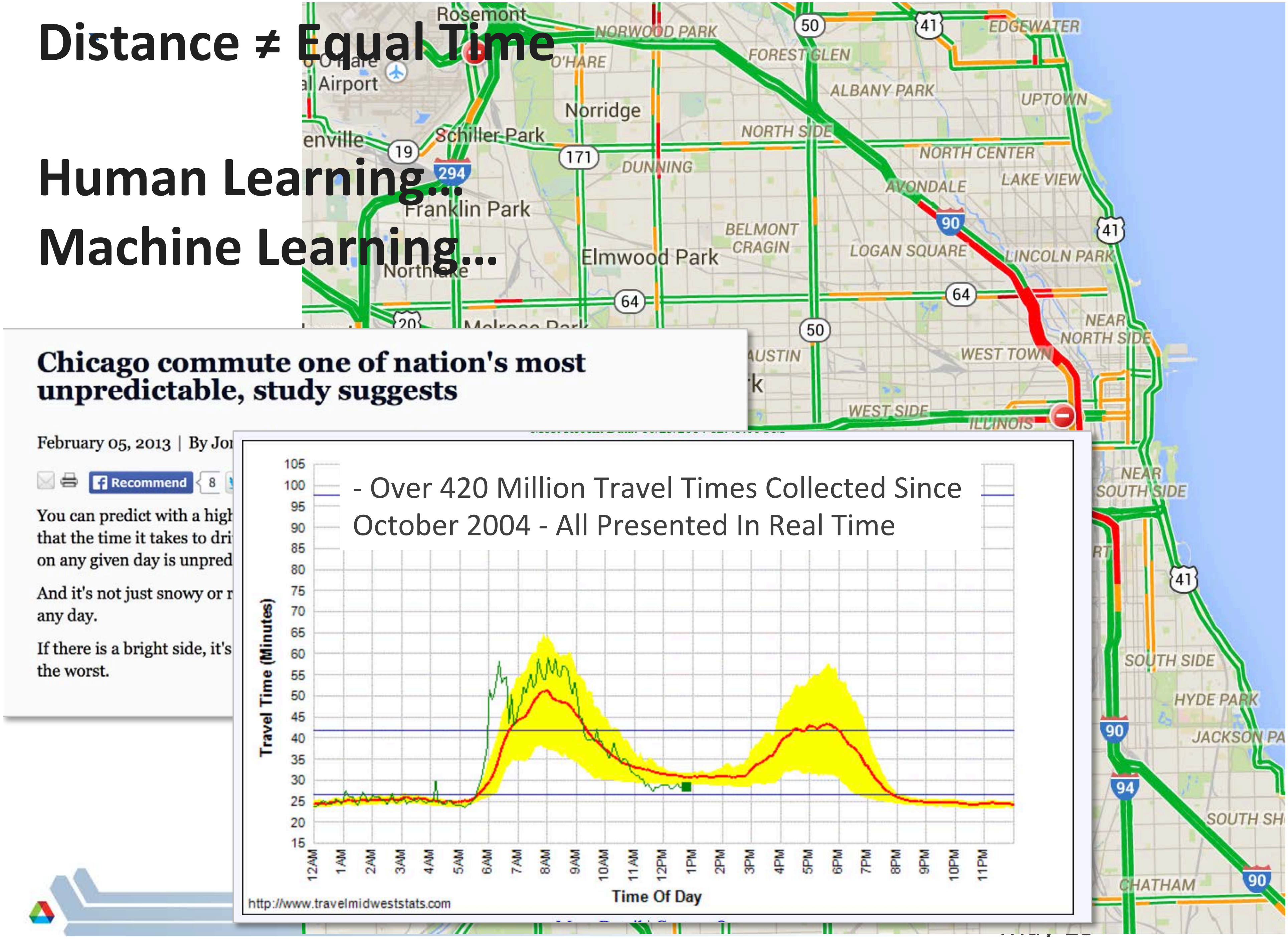


Online temperature predictions (blue solid line) versus actual sensor readings (red dotted line)

Distance ≠ Equal Time

Human Learning…

Machine Learning…



Chicago commute one of nation's most unpredictable, study suggests

February 05, 2013 | By Jo

Recommend

You can predict with a high
that the time it takes to dri
on any given day is unpred

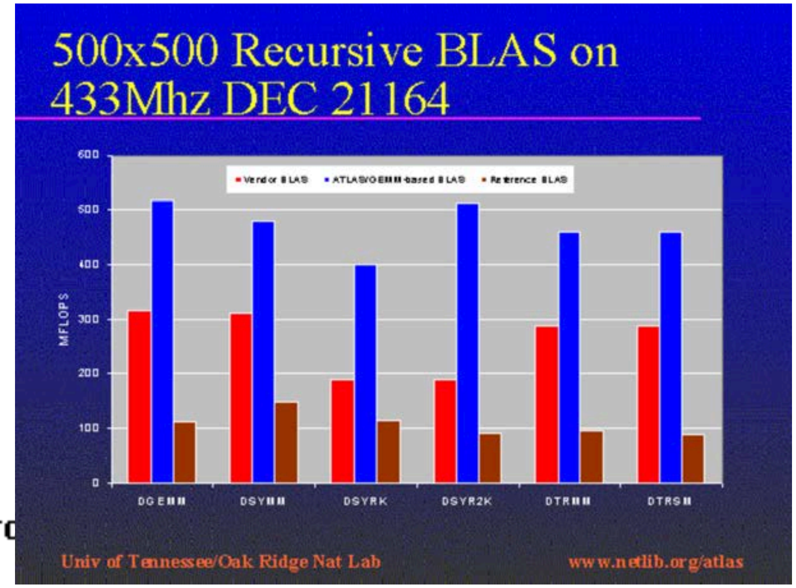
And it's not just snowy or r
any day.

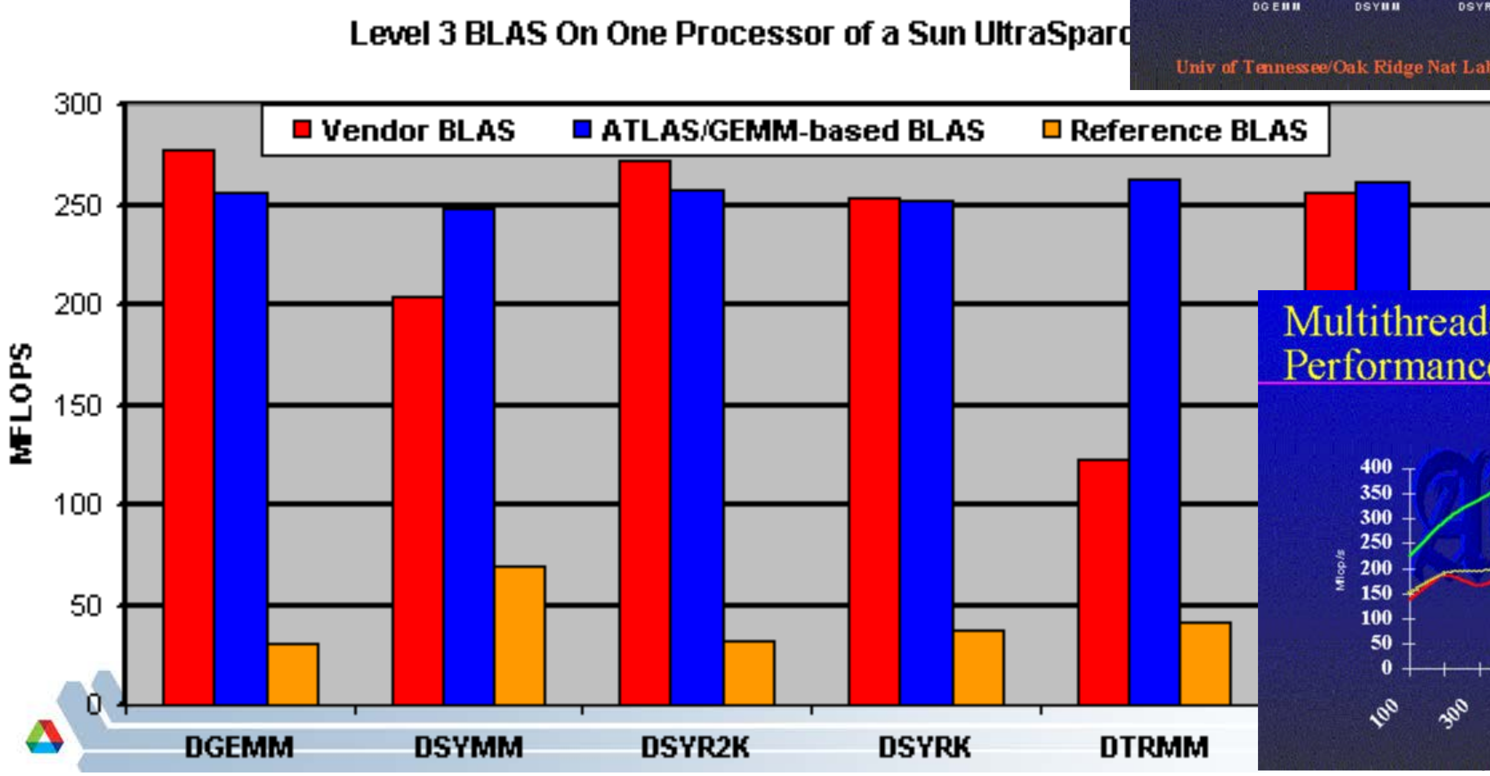
If there is a bright side, it's
the worst.

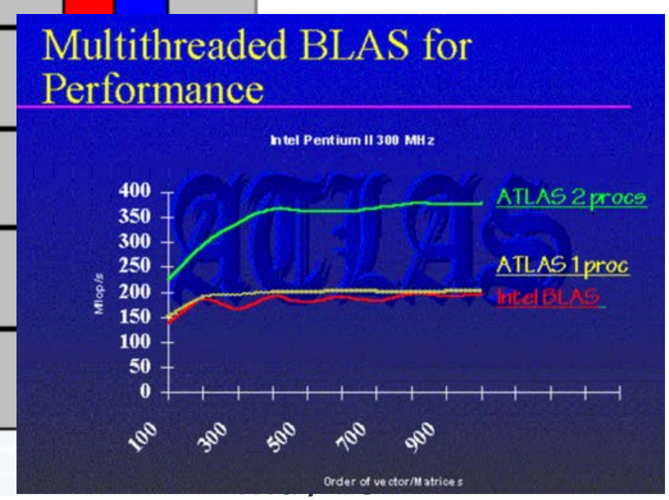
Automatically Tuned Linear Algebra Software (ATLAS) … 15 yrs ago…

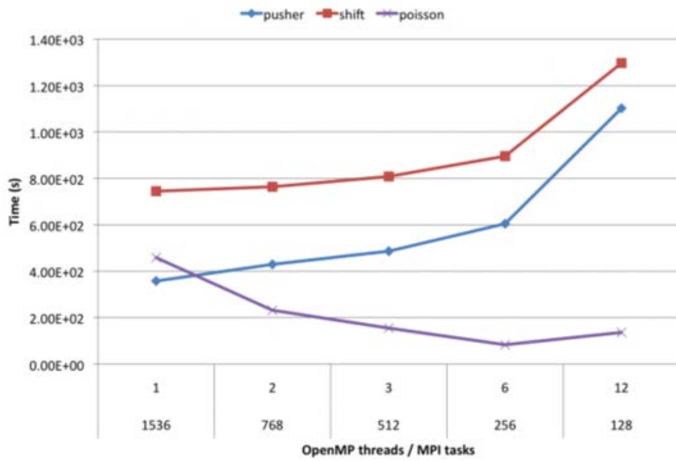
Primitive Machine Learning:
"Search and Select" (no humans)

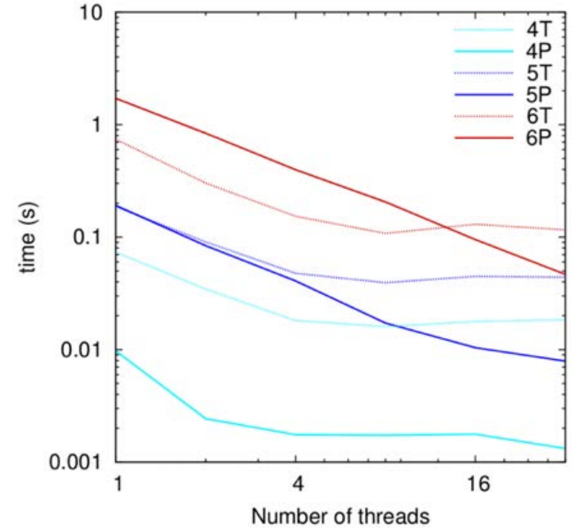
But embarrassingly static….

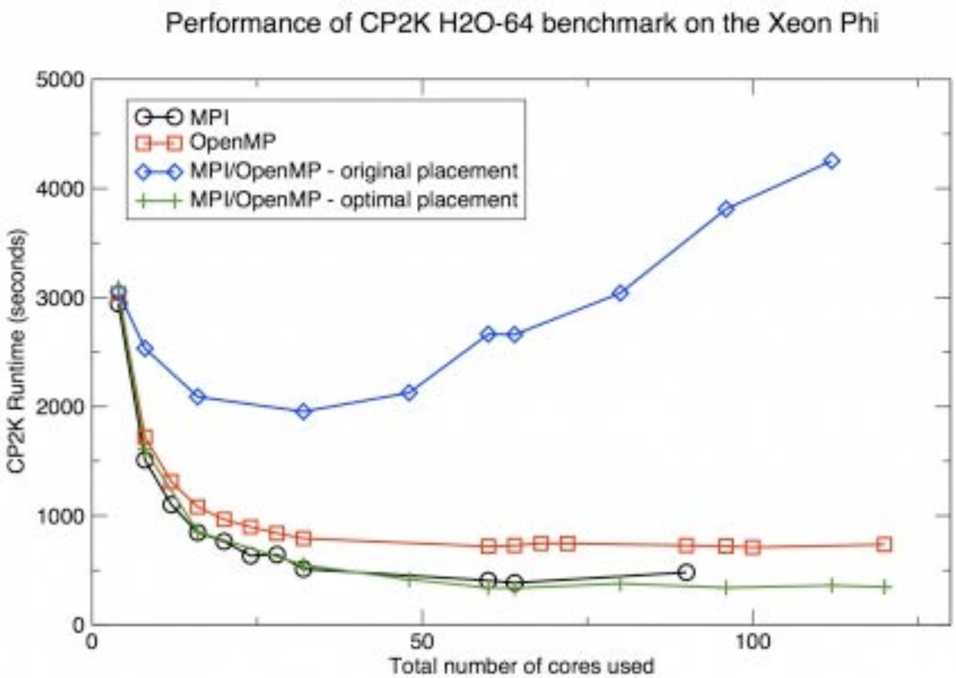








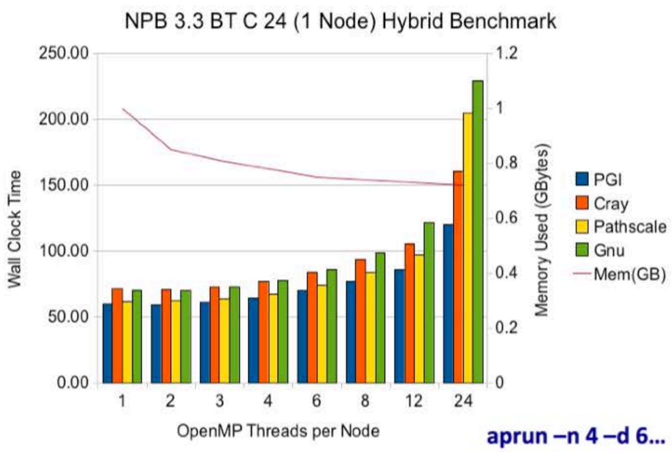




"The figure shows the performance of MPI, OpenMP and MPI+OpenMP versions of CP2K. The blue diamonds show the original performance with poor task placement. The green line shows the final result with optimal placement. This obtained better performance than both the MPI and OpenMP versions and enabled more virtual threads to be used. The best placement was found to be a balanced approach where each of the 60 physical cores have as few threads as possible whilst also keeping the threads belonging to a particular MPI process physically close to one another."

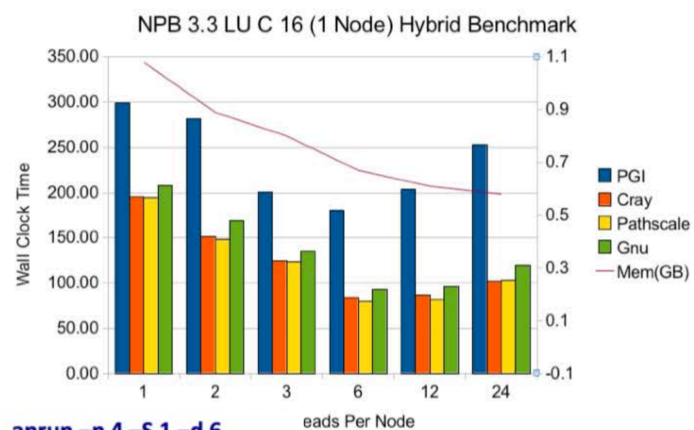
Why Mixed OpenMP/MPI Code is Sometimes Slower?

- OpenMP has less scalability due to implicit parallelism while MPI allows multi-dimensional blocking.
- All threads are idle except one while MPI communication.
  - Need overlap comp and comm for better performance.
  - Critical Section for shared variables.
- Thread creation overhead
- Cache coherence, false sharing.
- Data placement, NUMA effects.
- Natural one level parallelism problems.
- Pure OpenMP code performs worse than pure MPI within node.
- Lack of optimized OpenMP compilers/libraries.

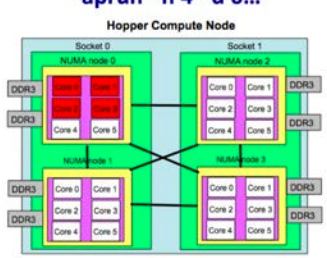


Debug and Tune Hybrid Codes

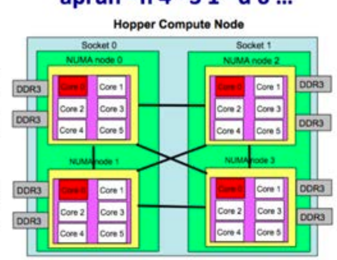
- Debugger tools: DDT, Totalview, gdb, Valgrind.
- Profiling: IPM, CrayPat, TAU.
- Decide which loop to parallelize. Better to parallelize outer loop. Decide whether Loop permutation, fusion or exchange is needed. Use NOWAIT clause if possible.
- Choose between loop-based or SPMD.
- Use different OpenMP task scheduling options.
- Experiment with different combinations of MPI tasks and number of threads per MPI task. Less MPI tasks may not saturate inter-node bandwidth.
- Adjust environment variables.
- Aggressively investigate different thread initialization options and the possibility of overlapping communication with computation.
- Try OpenMP TASK.
- Leave some cores idle on purpose: memory capacity or bandwidth capacity.
- Try different compilers.



aprun –n 4 –d 6...



aprun –n 4 –S 1 –d 6 ...



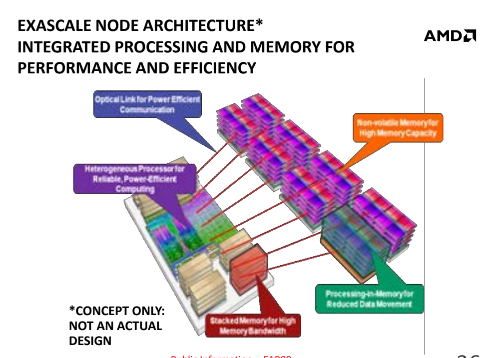
Argonne's Next Big Machine: Aurora

Argonne's Aurora Details

System Feature	Aurora
Peak System performance (FLOPs)	180 - 450 PetaFLOPS
Processor	3rd Generation Intel® Xeon Phi™ processor (code name Knights Hill)
Number of Nodes	>50,000
Compute Platform	Cray Shasta next generation supercomputing platform
High Bandwidth On-Package Memory, Local Memory, and Persistent Memory	>7 PetaBytes
System Interconnect	2nd Generation Intel® Omni-Path Architecture with silicon photonics
Interconnect interface	Integrated
Burst Storage Buffer	Intel® SSDs, 2nd Generation Intel® Omni-Path Architecture
File System	Intel Lustre* File System
File System Capacity	>150 PetaBytes
File System Throughput	>1 TeraByte/s
Intel Architecture (x86-64) Compatibility	Yes
Peak Power Consumption	13 Megawatts
FLOPS/watt	>13 GFLOPS/watt
Delivery Timeline	2018
Facility Area	~3,000 sq. ft.

Conclusions: The Times They are A-Changin’

- Embrace DYNAMIC!
  - Work ≠ Time
- Optimize algorithms for data movement
- Learn to love runtime systems
- Explore adaptive, learning, predictive software stacks that takes humans out of the loop…
  - Sorry humans, you are too slow.
  - Reject human tuning papers…
  - System software stack must stop being forgetful…..
    - mpiexec -n 1048576 a.out
    - mpiexec -n 1048576 a.out



Questions?