Programming for the Next Decade

(Perspectives from a Systems Architect)

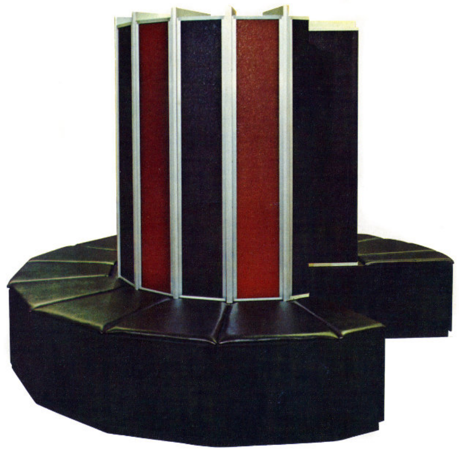
Steve Scott
Cray CTO

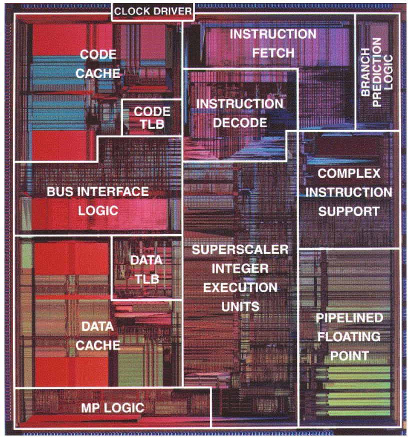
Blue Waters Symposium
May 12, 2015

Jim Goodman
University of Wisconsin

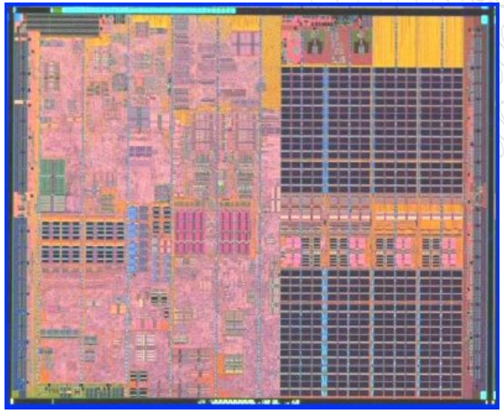
“Each year the questions remain the same, only the answers change.”

Copyright 2015 Cray Inc.

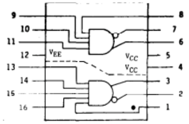

Cray 1, 1976



Intel Pentium, 1993


Intel Pentium 4 Cedar Mill, 2006

- ECL 5/4 NAND gate ICs (95%)



- 75K gates. (3400 PCBs!)
- RISC design
- Vector ISA
- Memory latency 11 clocks

- CMOS VLSI IC
- 3M transistors
- CISC design
- Deep pipelines, complex predictions

- 184M transistors!
- *Very* CISC design
- 31-stage pipeline
- 3.6 GHz in 65nm
- Last of its breed….

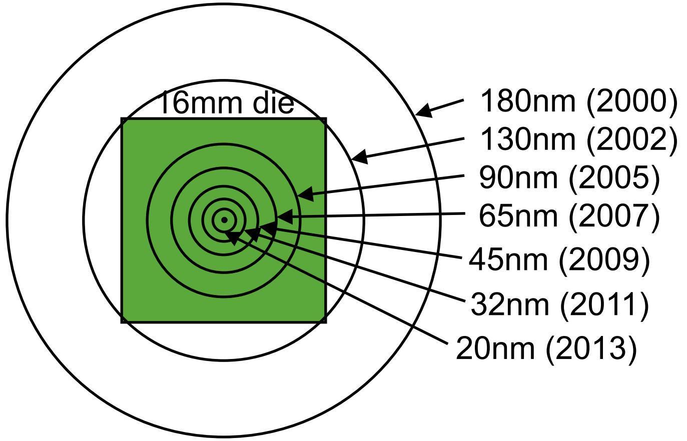
Copyright 2015 Cray Inc.

Post Dennard Scaling and the Power Wall
(2005 onward)

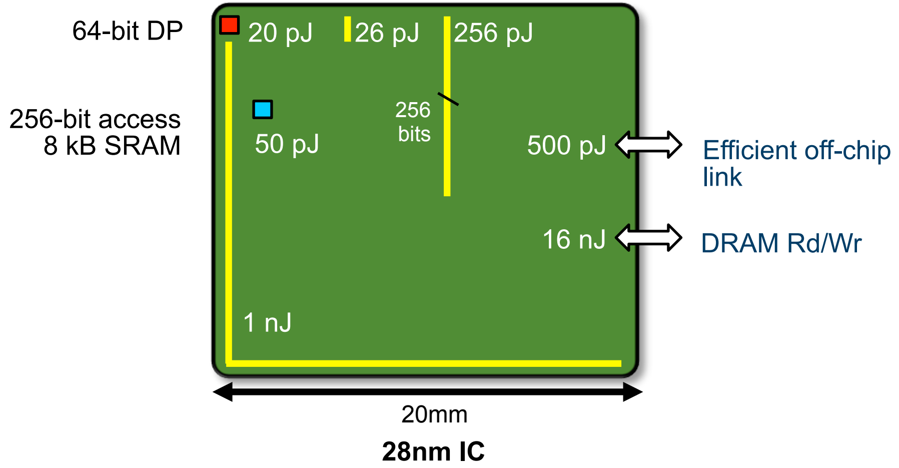
Voltage no longer drops with feature size

- ⇒ **perf/W/year** has slowed **dramatically (70% ➔ 20% CAGR)**
    - ⇒ Have become *power* constrained

Signal reach dropping:



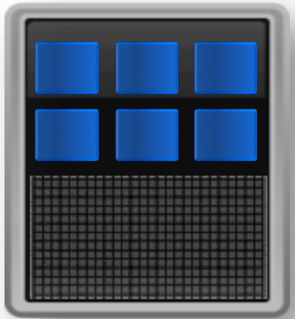
Communication much more expensive than computation



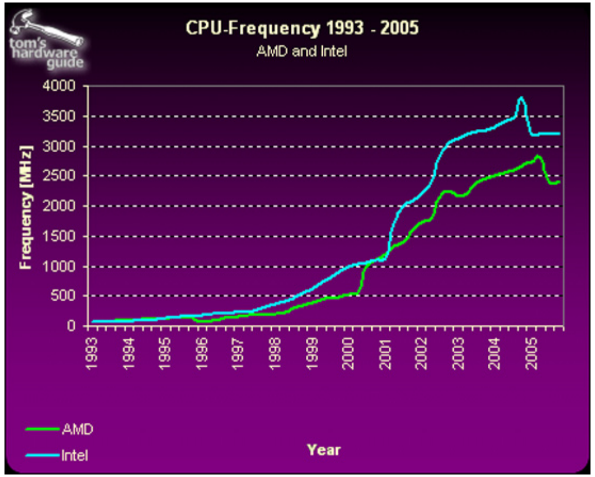
Copyright 2015 Cray Inc.

Architectural Response

1) Stop making it worse…



Multicore CPUs



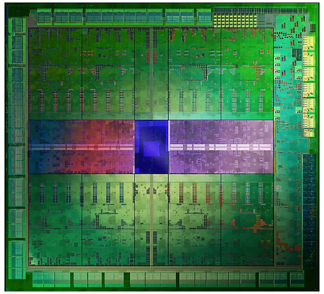
But still only a tiny fraction of CPU power spent on flops

2) Continue to innovate in circuits (e.g.: low voltage SRAMs)

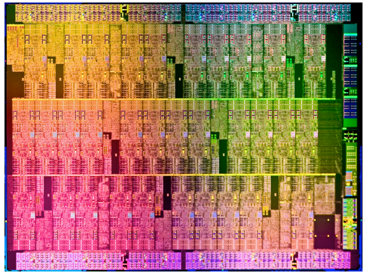
3) Unwind all that complexity we threw at single thread performance (reclaim the lost performance/W potential)

Copyright 2015 Cray Inc.

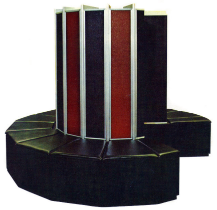
New Processor Landscape

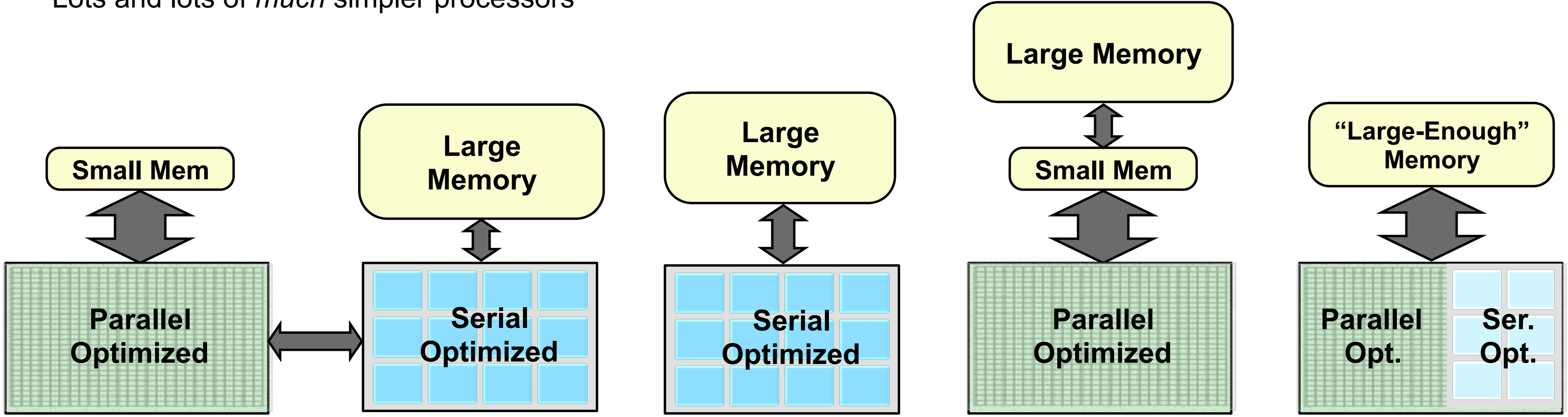


GPU computing (Nvidia Kepler)
Lots and lots of *much* simpler processors



Vectors are back!
(Intel Xeon Phi)
Parallelism with low complexity and control overhead

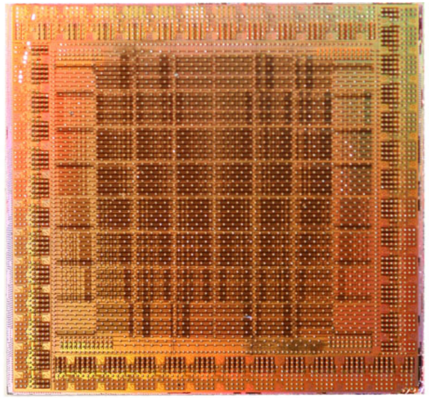




Copyright 2015 Cray Inc.

Power-Efficient Networks

- Cray pioneered the use of high radix routers in HPC
  - Became optimal due to *technology shift*
    - Pin bandwidth growing relative to packet length
    - Reduces serialization latency of narrow links
  - Reduced network diameter (number of hops)
    - Lowers network latency and cost
  - But higher radix network require longer cable lengths
    - Limits electrical signaling speed


64 port YARC router in Cray X2

- Advent of cost-effective optics allows longer cable lengths
  - Optics are now cost effective above a few meters (and dropping)
  - Cost, bandwidth and power are relatively insensitive to cable length

- Current and future Cray systems based on hybrid, electrical-optical networks
  - Cost-effective, scalable global bandwidth
  - Very low network diameter (small number of hops) $\Rightarrow$ very energy efficient

Copyright 2015 Cray Inc.

Future of HPC Programming

Summary of Future Machines

- **Computers are not getting faster… just wider**
  - O(EF) with O(GHz) clocks → O(B) way parallelism!
- ***Vertical* locality much more important than *horizontal* locality**

Dimension	Latency Hit	Bandwidth Hit	Energy Hit
Within node	~200x	~200x	> 500x
Across nodes	~25x	~8x	~5x

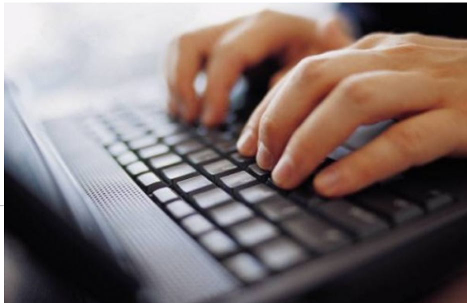
** If include local NVM, within node grows, across nodes shrinks*

- **Parallelism is multi-dimensional (and heterogeneous?)**
  - Vectorization + threading + multi-node
  - Processors optimized for serial performance ***or*** power efficiency (not both)
- **Interconnects won't look that different than today**

Copyright 2015 Cray Inc.

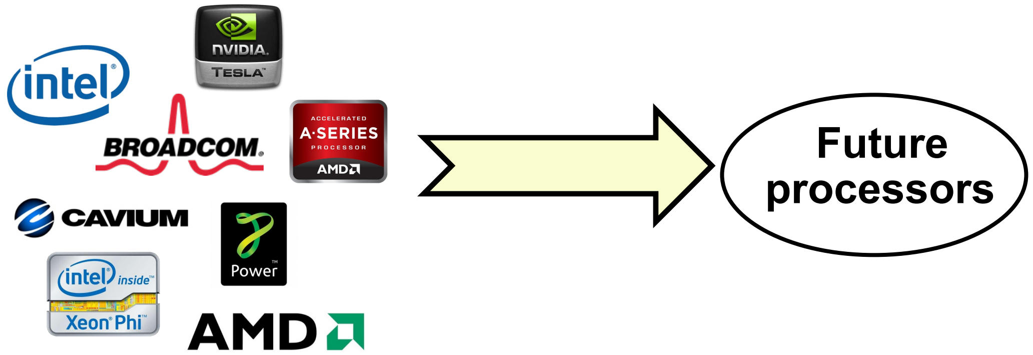
Implications for Programmers

- **Must move to more threading on the node**
  - All-MPI won't deliver maximum performance
- **Must vectorize low-level loops**
  - 8-30x performance improvement on array operations
- **Must avoid scalar code**
  - On "accelerated" nodes, creates traffic between accelerator and host, or runs 3-4x slower than on a serial-optimized core
  - Inherently slower and less power-efficient
- **Must pay a *lot* more attention to locality within node**
  - Think about data placement and movement
  - Consider "sub-optimal" algorithms that limit data motion



Would like to code for future machines in a portable way

- **Spatial and Temporal Portability**



- **Separation of labor**
  - Programmer *exposes* parallelism and locality
  - Compiler, tools, and runtime map onto specific hardware
  - Optimized libraries for various platforms

Copyright 2015 Cray Inc.

Bold Prediction:

- **Future HPC Programming Model: MPI + OpenMP**
- **Can we make this easier?**
  - Threading, vectorization, data placement
- **Recent poll at NERSC found 80% of apps use single level of parallelism**
- **Why & when to convert to hybrid programming model?**
  - When code becomes network bound
  - Load balancing and synchronization overheads become large
  - Excessive memory used by straight MPI
  - To take advantage of hybrid compute nodes

Copyright 2015 Cray Inc.

Approach to Adding Parallelism

1. **Identify key high-level loops**
   - Determine where to add additional levels of parallelism
2. **Perform parallel analysis and scoping**
3. **Add OpenMP layer of parallelism**
   - Insert OpenMP directives
4. **Analyze performance for further optimizations**
   - Specifically vectorization of inner loops

Copyright 2015 Cray Inc.

Which of these profiles display what is important?

```
 Time% |      Time |    Imb. |  Imb. |    Calls |Group
       |           |    Time | Time% |          | Function
       |           |         |       |          |  PE=HIDE

 100.0% | 1.545303 |       -- |     -- | 721995.0 |Total
|-----------------------------------------------------------------
|  84.7% | 1.308232 |       -- |     -- | 721990.0 |USER
||----------------------------------------------------------------
||  20.6% | 0.317597 | 0.067854 |  17.9% | 276480.0 |parabola_
||   9.8% | 0.151749 | 0.024599 |  14.2% |  30720.0 |riemann_
||   8.7% | 0.134020 | 0.214959 |  62.6% |     20.0 |sweepy_
||   8.6% | 0.133213 | 0.057049 |  30.5% |     10.0 |sweepz_
||   8.1% | 0.125045 | 0.028330 |  18.8% |  30720.0 |remap_
||   4.9% | 0.075581 | 0.085175 |  53.8% |     10.0 |dtcon_
||   3.4% | 0.052096 | 0.012666 |  19.9% |  61440.0 |paraset_
||   3.2% | 0.049156 | 0.010650 |  18.1% |  30720.0 |evolve_
||   3.1% | 0.048344 | 0.013154 |  21.7% |  92160.0 |forces_
||   3.1% | 0.047601 | 0.013329 |  22.2% |  92160.0 |volume_
||   2.6% | 0.040902 | 0.008664 |  17.8% |  15360.0 |ppmlr_
||   2.1% | 0.032670 | 0.007914 |  19.8% |  30720.0 |states_
||   1.8% | 0.027902 | 0.006195 |  18.5% |  30720.0 |flatten_
||   1.6% | 0.024737 | 0.005826 |  19.4% |  30720.0 |boundary_
||   1.1% | 0.017032 | 0.016742 |  50.4% |      2.0 |vhone_
||   0.8% | 0.013135 | 0.004581 |  26.3% |     10.0 |sweepx1_
||   0.6% | 0.009817 | 0.004406 |  31.5% |     10.0 |sweepx2_
||   0.5% | 0.007235 | 0.009223 |  56.9% |      2.0 |init_
||   0.0% | 0.000399 | 0.000207 |  34.7% |      6.0 |grid_
||=================================================================
```

Typical profile showing exclusive wall-time

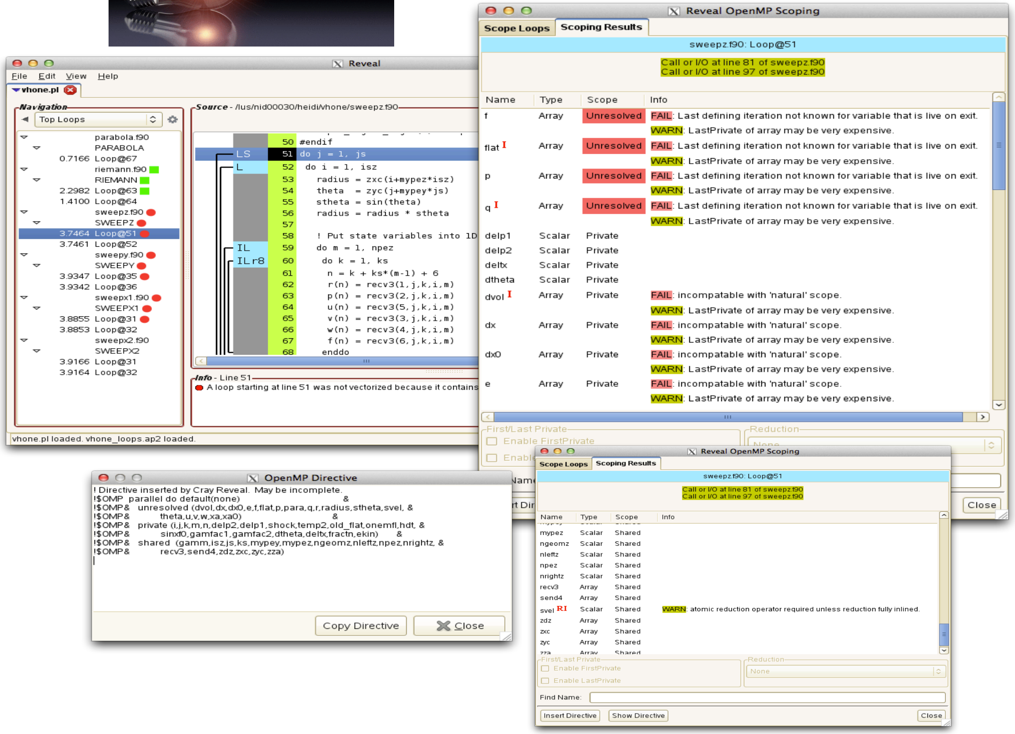
```
 Time% |      Time |    Calls |Calltree

 100.0% | 1.545303 |       -- |Total
|--------------------------------------------------
| 100.0% | 1.545303 |    2.0 |vhone_
||-------------------------------------------------
||  83.1% | 1.283566      -- |vhone_.LOOP.2.li.205
|||------------------------------------------------
3||  22.8% | 0.352952 | 20.0 |sweepy_     ← SWEEPY  22.8% of inclusive time
||||-----------------------------------------------
4|||  14.2% | 0.218932 | --   | sweepy_.LOOP.1.li.32   ← High level grid loop
5|||        |          |      | sweepy_.LOOP.2.li.33   ← High level grid loop
6|||  14.2% | 0.218932 |  2560.0 |  ppmlr_
|||||||--------------------------------------------
7||||||    5.8% | 0.089807 |  5120.0 |remap_
||||||||------------------------------------------
8|||||||    3.4% | 0.052046 | 30720.0 |parabola_
8|||||||    1.8% | 0.028345 |  5120.0 |remap_(exclusive)
8|||||||    0.4% | 0.006467 |  5120.0 |paraset_
8|||||||    0.2% | 0.002949 |  5120.0 |volume_
||||||||==========================================
7||||||    3.0% | 0.047088 |  5120.0 |riemann_
7||||||    1.7% | 0.026442 | 15360.0 |parabola_
7||||||    1.4% | 0.021188 |  5120.0 |evolve_
||||||||------------------------------------------
8|||||||    0.7% | 0.010535 |  5120.0 |evolve_(exclusive)
8|||||||    0.4% | 0.005505 | 10240.0 |forces_
8|||||||    0.3% | 0.005147 | 10240.0 |volume_
```

^ Nesting level everything below 3 is called by 3

Copyright 2015 Cray Inc.

Simplifying the Task with Reveal



```
   parabola.f90
   PARABOLA
0.7166 Loop@67
   riemann.f90 🟩
   RIEMANN 🟩
2.2982 Loop@63 🟩
1.4100 Loop@64
   sweepz.f90 ●
   SWEEPZ ●
3.7464 Loop@51 ●
3.7461 Loop@52
   sweepy.f90 ●
   SWEEPY ●
3.9347 Loop@35 ●
3.9342 Loop@36
   sweepx1.f90 ●
   SWEEPX1 ●
3.8855 Loop@31 ●
3.8853 Loop@32
   sweepx2.f90
   SWEEPX2
3.9166 Loop@31
3.9164 Loop@32
```

Source - /lus/nid00030/heidi/vhone/sweepz.f90

```
      50 #endif
LS    51 do j = 1, js
L     52  do i = 1, isz
      53    radius = zxc(i+mypez*isz)
      54    theta  = zyc(j+mypey*js)
      55    stheta = sin(theta)
      56    radius = radius * stheta
      57
      58    ! Put state variables into 1D
IL    59    do m = 1, npez
IL r8 60     do k = 1, ks
      61      n = k + ks*(m-1) + 6
      62      r(n) = recv3(1,j,k,i,m)
      63      p(n) = recv3(2,j,k,i,m)
      64      u(n) = recv3(5,j,k,i,m)
      65      v(n) = recv3(3,j,k,i,m)
      66      w(n) = recv3(4,j,k,i,m)
      67      f(n) = recv3(6,j,k,i,m)
      68     enddo
```

Info - Line 51
● A loop starting at line 51 was not vectorized because it contains

vhone.pl loaded. vhone_loops.ap2 loaded.

sweepz.f90: Loop@51

Call or I/O at line 81 of sweepz.f90
Call or I/O at line 97 of sweepz.f90

Name	Type	Scope	Info
f	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
			WARN: LastPrivate of array may be very expensive.
flat I	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
			WARN: LastPrivate of array may be very expensive.
p	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
			WARN: LastPrivate of array may be very expensive.
q I	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
			WARN: LastPrivate of array may be very expensive.
delp1	Scalar	Private	
delp2	Scalar	Private	
deltx	Scalar	Private	
dtheta	Scalar	Private	
dvol I	Array	Private	FAIL: incompatable with 'natural' scope.
			WARN: LastPrivate of array may be very expensive.
dx	Array	Private	FAIL: incompatable with 'natural' scope.
			WARN: LastPrivate of array may be very expensive.
dx0	Array	Private	FAIL: incompatable with 'natural' scope.
			WARN: LastPrivate of array may be very expensive.
e	Array	Private	FAIL: incompatable with 'natural' scope.
			WARN: LastPrivate of array may be very expensive.

```
! Directive inserted by Cray Reveal.  May be incomplete.
!$OMP  parallel do default(none)                                  &
!$OMP&   unresolved (dvol,dx,dx0,e,f,flat,p,para,q,r,radius,stheta,svel, &
!$OMP&          theta,u,v,w,xa,xa0)                               &
!$OMP&   private (i,j,k,m,n,delp2,delp1,shock,temp2,old_flat,onemfl,hdt, &
!$OMP&          sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn,ekin)      &
!$OMP&   shared  (gamm,isz,js,ks,mypey,mypez,ngeomz,nleftz,npez,nrightz, &
!$OMP&          recv3,send4,zdz,zxc,zyc,zza)
```

sweepz.f90: Loop@51

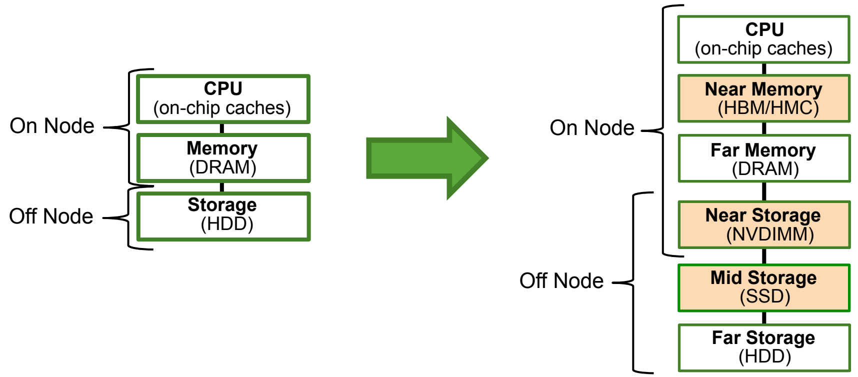
Call or I/O at line 81 of sweepz.f90
Call or I/O at line 97 of sweepz.f90

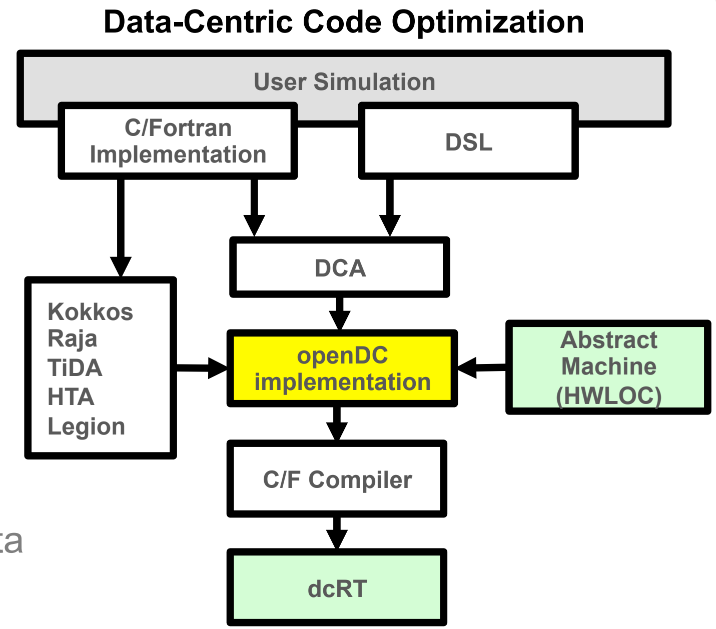
Name	Type	Scope	Info
mypez	Scalar	Shared	
ngeomz	Scalar	Shared	
nleftz	Scalar	Shared	
npez	Scalar	Shared	
nrightz	Scalar	Shared	
recv3	Array	Shared	
send4	Array	Shared	
svel RI	Scalar	Shared	WARN: atomic reduction operator required unless reduction fully inlined.
zdz	Array	Shared	
zxc	Array	Shared	
zyc	Array	Shared	
zza	Array	Shared	

- Navigate to relevant loops to parallelize
- Identify parallelization and scoping issues
- Get feedback on issues down the call chain (e.g.: shared reductions)
- Shows vectorization and other compiler optimizations
- Optionally insert parallel directives into source
- Validate scoping correctness on existing directives

Copyright 2015 Cray Inc.

Data Management in the Memory Hierarchy





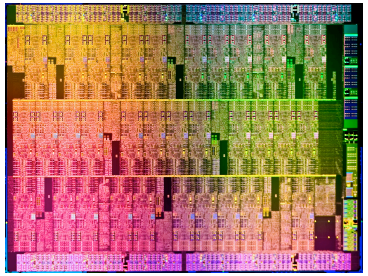
- **Two levels of interest:**
  - Memory hierarchy accessed as memory
    (caches, HBM, DDR4, NVM, remote SSD?)
  - Network attached NVM that is accessed as storage
- **At each level, want a dual approach**
  - APIs, directives, and tools for users to manage/access data
  - System software to automatically manage the memory

Copyright 2015 Cray Inc.

BIG
DATA
USE
ONE
SINCE
SOFTWARE
USING
TYPES
WORKING
UBIQUITOUS
REQUIRING
RADIO-FREQUENCY
COMPLEXITY
GARTNER
MASSIVELY
SAN
SOLID
WIRELESS
BUSINESS
MOVING
ANALYTICS
TOLERABLE
RELATED
BIOLOGICAL
PERFORMANCE
ALSO
HUNDREDS
CREATED
PARALLEL
SIZE
NEEDED
QUALITIES
BURIED
DISK
RELATIONAL
DESKTOP
ORGANIZATIONS
PETABYTES
PROCESSING
ZETTABYTES
GENOMICS
COMPUTING
TOOLS
SET
SHARED
TIME
CURRENTLY
TENS
PRACTITIONERS
LOGS
INTERNET
WITHIN
COMBAT
SIGNIFICANT
PROCESS
INCLUDE
SYSTEMS
CAPACITY
TECHNOLOGIES
FC
DEFINITION
BIOGEOCHEMICAL
NETWORKS
FORMS
USED
DISTRIBUTED
CAPTURE
SEARCH
RECONSIDER
OPPORTUNITIES
INFORMATION
PRESENTATIONS
NOW
MAY
MANAGEMENT
MANAGE
GROW
CONNECTOMICS
RECORDS
WORLD'S
SOCIAL
LARGE
DEFINING
STORE
COST
CONTINUES
TERABYTES
MPP
RESEARCH
STORAGE
EVERY
CASE
CITATION
COMPLEX
DATABASES
LARGER
INDEXING
DIFFICULTY
TARGET
ABILITY
SETS
EXAMPLES
ELAPSED
CURRENT
THOUGHT
APPLIED
SENSOR
ARCHIVES
AMOUNT
DESCRIBING

Big Data vs. HPC

&&



Common Needs:

- Compute power
- Interconnect bandwidth
- Memory capacity & bandwidth
- Storage system capacity & bandwidth
- Workload management
- Scaling
- Resiliency
- Visualization
- System management
- etc.

Copyright 2015 Cray Inc.

A Matter of Balance

Network Bandwidth

File System Capacity & Bandwidth

Compute

Memory Capacity & BW

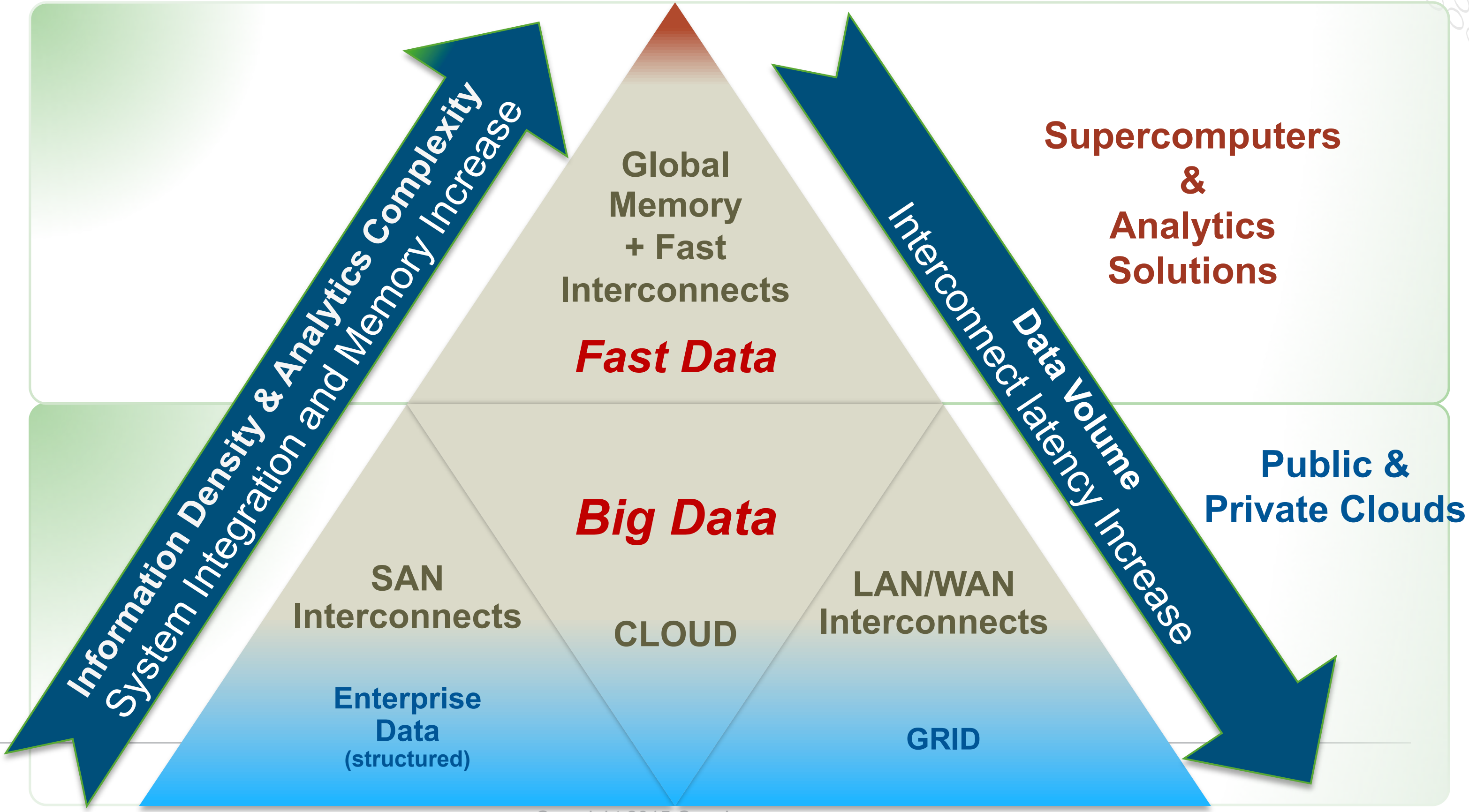
We have these same trade-offs *within* HPC

May lean towards larger memories, and more network & storage bandwidth

Copyright 2015 Cray Inc.

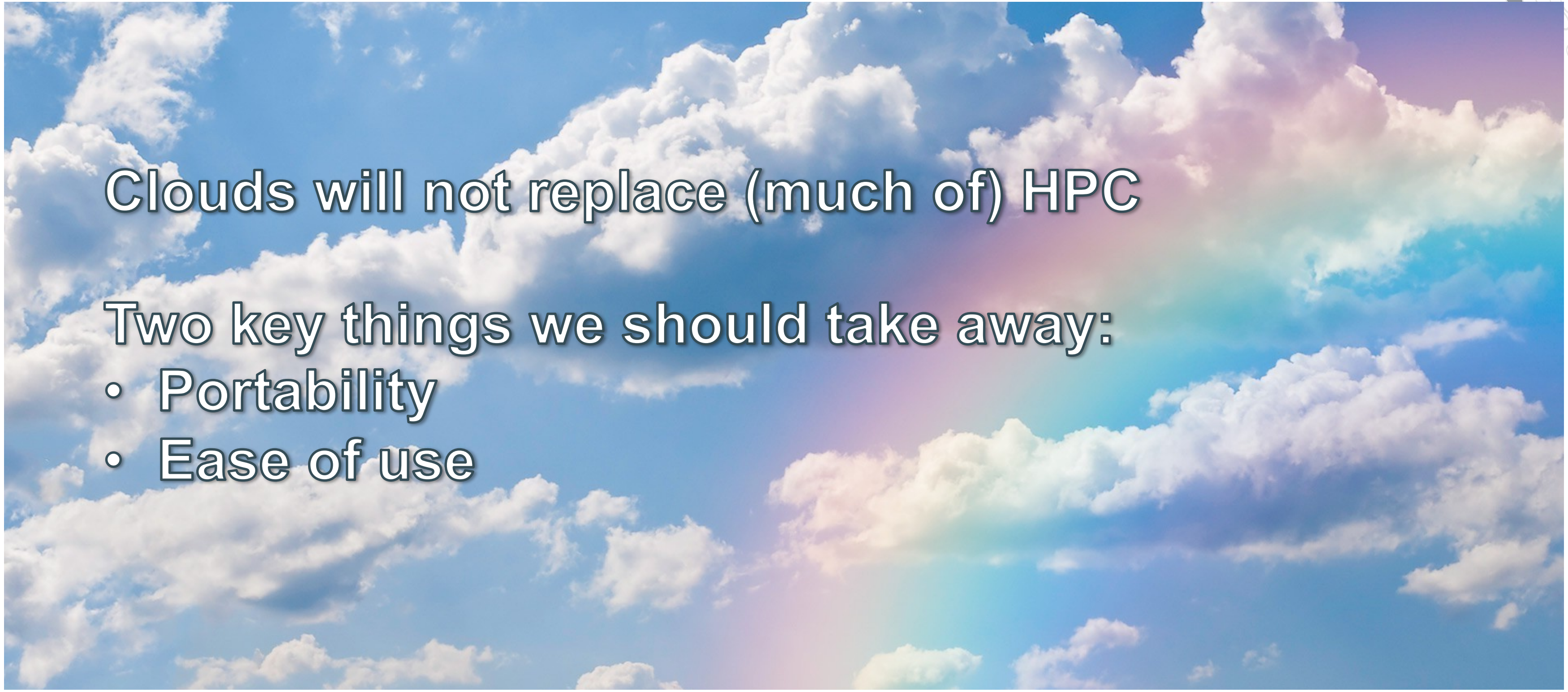
Enabling More Complexity & Capability…

Big Data → Fast Data


Information Density & Analytics Complexity
System Integration and Memory Increase
Global Memory + Fast Interconnects
Fast Data
Big Data
SAN Interconnects
CLOUD
LAN/WAN Interconnects
Enterprise Data (structured)
GRID
Data Volume
Interconnect latency Increase
Supercomputers & Analytics Solutions
Public & Private Clouds

Copyright 2015 Cray Inc.

I've Looked at Clouds from Both Sides Now



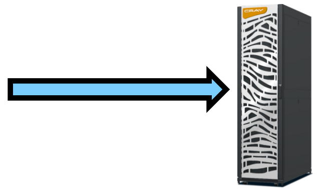
Clouds will not replace (much of) HPC

Two key things we should take away:

- Portability
- Ease of use

Copyright 2015 Cray Inc.

System Monitoring and Operational Analytics



Analytics Appliance

- Predictive failure analysis
- Job failure/performance diagnosis
- Cyber-threat detection
- System optimization
  - Power management
  - Job scheduling and placement
  - IO and network configuration
- Proactive detection of performance issues
- System dashboards

Currently collect logs in multiple places

- SMW, SDB & login nodes, Lustre service nodes

Types of data:

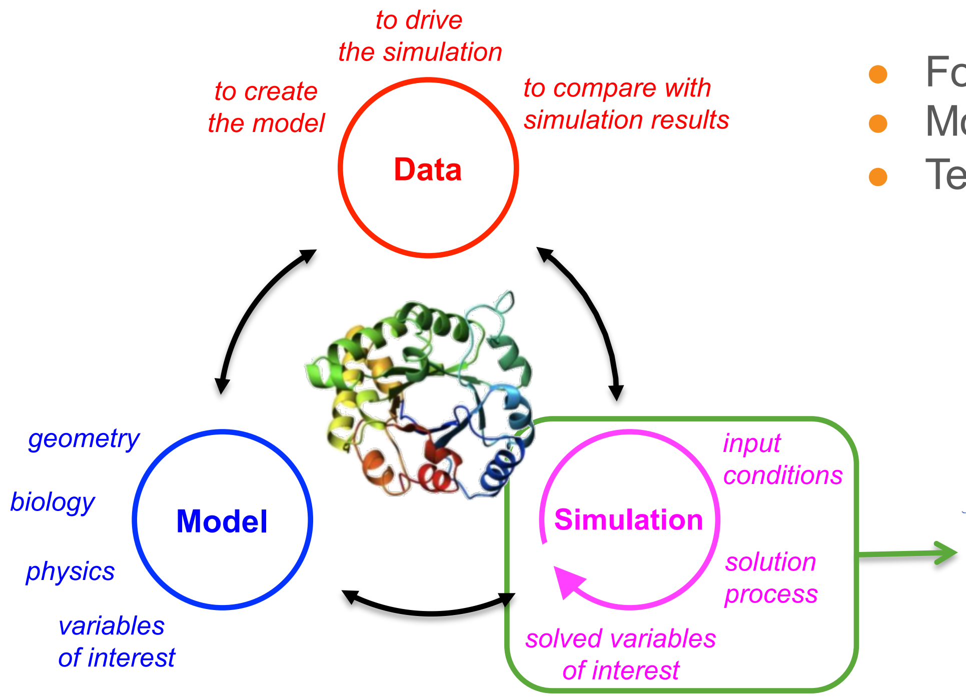
- Network health
- Console traffic (node-level OS errors)
- Temp, power, perf & status of all components
- Job scheduling and placement information
- Job performance data
- File system and network logs
- Etc.

Hard to diagnose performance problems or failures

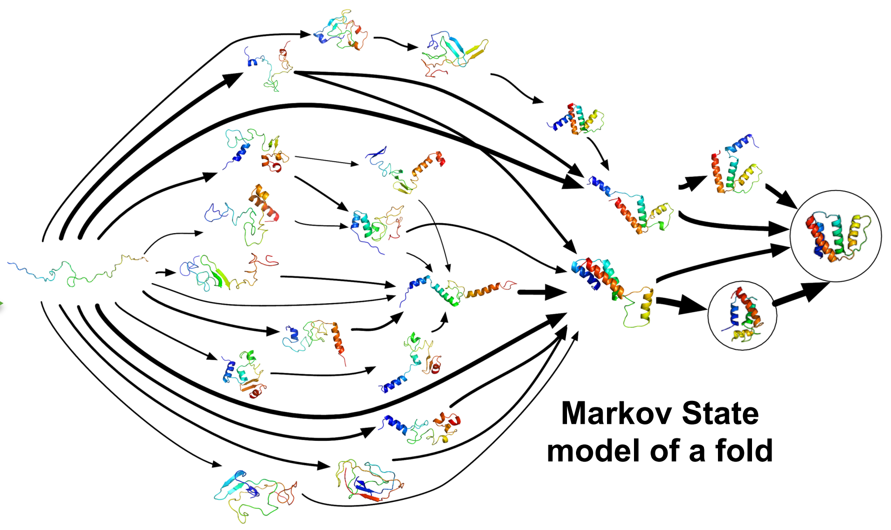
- SSA is a first step…

Copyright 2015 Cray Inc.

Protein Folding – Mixed Simulation and Analytics

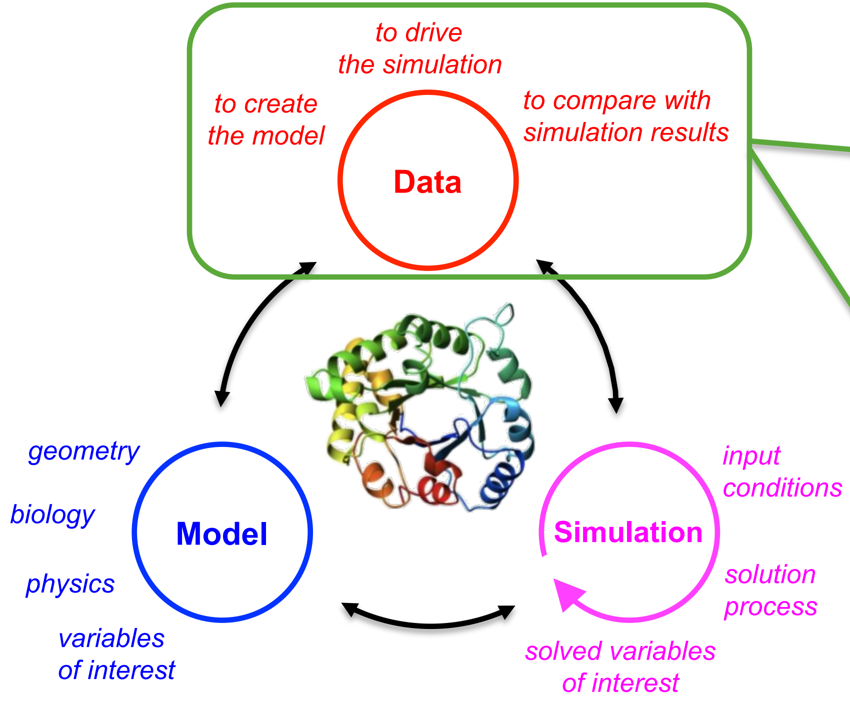


- Focuses on *how* protein folding happens
- Model possible paths to folded end-state
- Temporal resolution matters, but **drives data size**

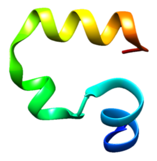


Copyright 2015 Cray Inc.

Protein Folding – Mixed Simulation and Analytics



Enabling in-situ data analysis for large protein-folding trajectory datasets



(a)

	1	2	3	4	...	N
1	0.0	2.0	1.8	3.2	...	10.4
2	2.0	0.0	2.3	2.1	...	9.3
3	1.8	2.3	0.0	0.7	...	9.4
4	3.2	2.1	0.7	0.0	...	8.9
...	...	...	...	...	0.0	...
N	10.4	9.3	9.4	8.9	...	0.0

(b)

	x	y	z
	1	2	3
1	1.0	2.3	1.4
2	2.0	0.4	1.3
3	1.8	2.3	2.0
4	1.2	3.1	1.7
...	...	...	...
N	4.4	2.3	3.4

(c)

Figure 1: One conformation of the villin HP-35 protein (a); part of its distance matrix using only its backbone atoms in the conformation (b); and three eigenvectors and the associated eigenvalues capturing and synthesizing the conformation geometry (c).

Abstract—This paper presents a one-pass, distributed method that enables in-situ data analysis for large protein-folding trajectory datasets by executing sufficiently fast, avoiding moving trajectory data, and limiting the memory usage.

In-situ analysis of folds – Dimensionality reduction using PCA & MDS

CATH

STRUCTURAL CLASSIFICATION OF PROTEINS

Automated classification in protein databases

Copyright 2015 Cray Inc.

One Interesting Difference Between Data Analytics and HPC Markets

- **The Data Analytics crowd seems to *really* like productivity**
  - Map/Reduce is easy, scalable, resilient, and…. *low performance!*
  - Spark is much more flexible, and higher performance, but still pretty high overhead by HPC standards
- **We've had little luck explaining that they really ought to be using C + MPI instead**
  - Much more interest in Hadoop/Spark/R, etc. than MPI
- **Provocative idea of the night:**
  - Chapel as HPDA language?
  - Also has growing appeal for HPC on new architectures
  - Separates *structural* aspects of code (hierarchical parallelism, locality) from *algorithmic* code
  - Recent work on performance closing gap with C + MPI

Copyright 2015 Cray Inc.

What does “Productivity” mean to you?

Recent Graduate:

“something similar to what I used in school: Python, Matlab, Java, …”

Seasoned HPC Programmer:

“that sugary stuff that I can’t use because I require full control to ensure good performance”

Computational Scientist:

“something that lets me express my parallel computations without having to wrestle with architecture-specific details”

Chapel Team:

“something that lets the computational scientists express what they want,
without taking away the control the HPC programmers want,
implemented in a language as attractive as recent graduates want.”

Copyright 2015 Cray Inc.

Chapel in a Nutshell

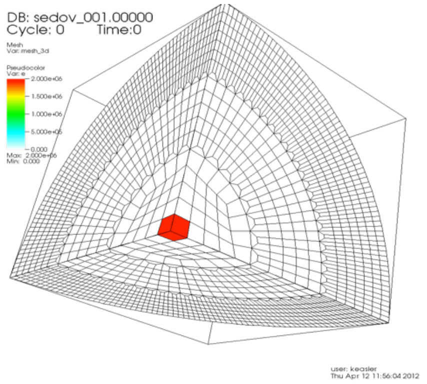
Chapel: a parallel language that has emerged from DARPA HPCS

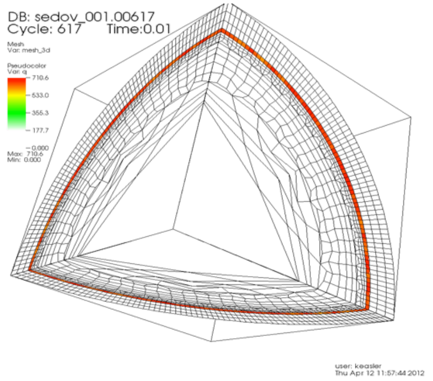
- **general parallelism:**
  - data-, task-, and nested parallelism
  - highly dynamic multithreading or static SPMD-style
- **multiresolution philosophy:** high-level features built on low-level
  - to provide "manual overrides"
  - to support a separation of concerns (application vs. parallel experts)
- **locality control:**
  - explicit or data-driven placement of data and tasks
  - locality expressed distinctly from parallelism
- **features for productivity:** type inference, iterators, rich array types
- **portable:** designed and implemented to support diverse systems
- **open source:** developed and distributed under Apache v2.0 CLAs

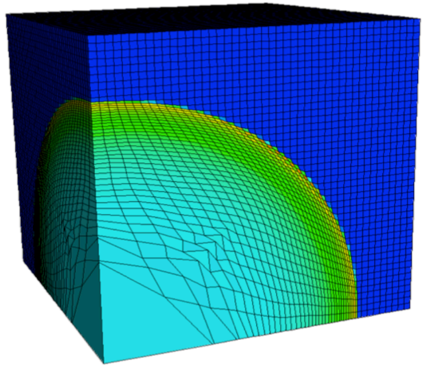
Copyright 2015 Cray Inc.

LULESH: a DOE Proxy Application

Goal: Solve one octant of the spherical Sedov problem (blast wave using Lagrangian hydrodynamics for a single material



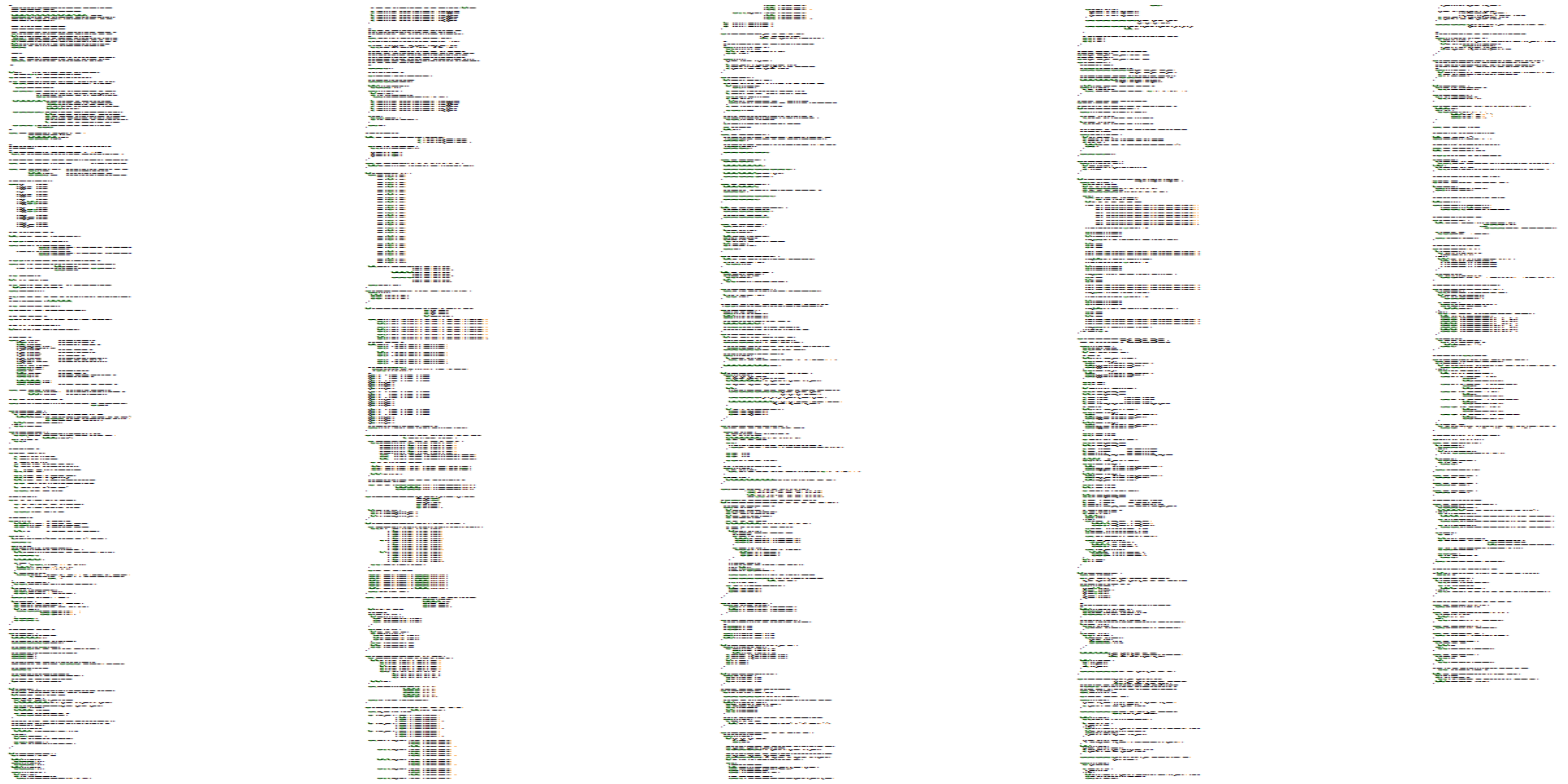




pictures courtesy of Rob Neely, Bert Still, Jeff Keasler, LLNL

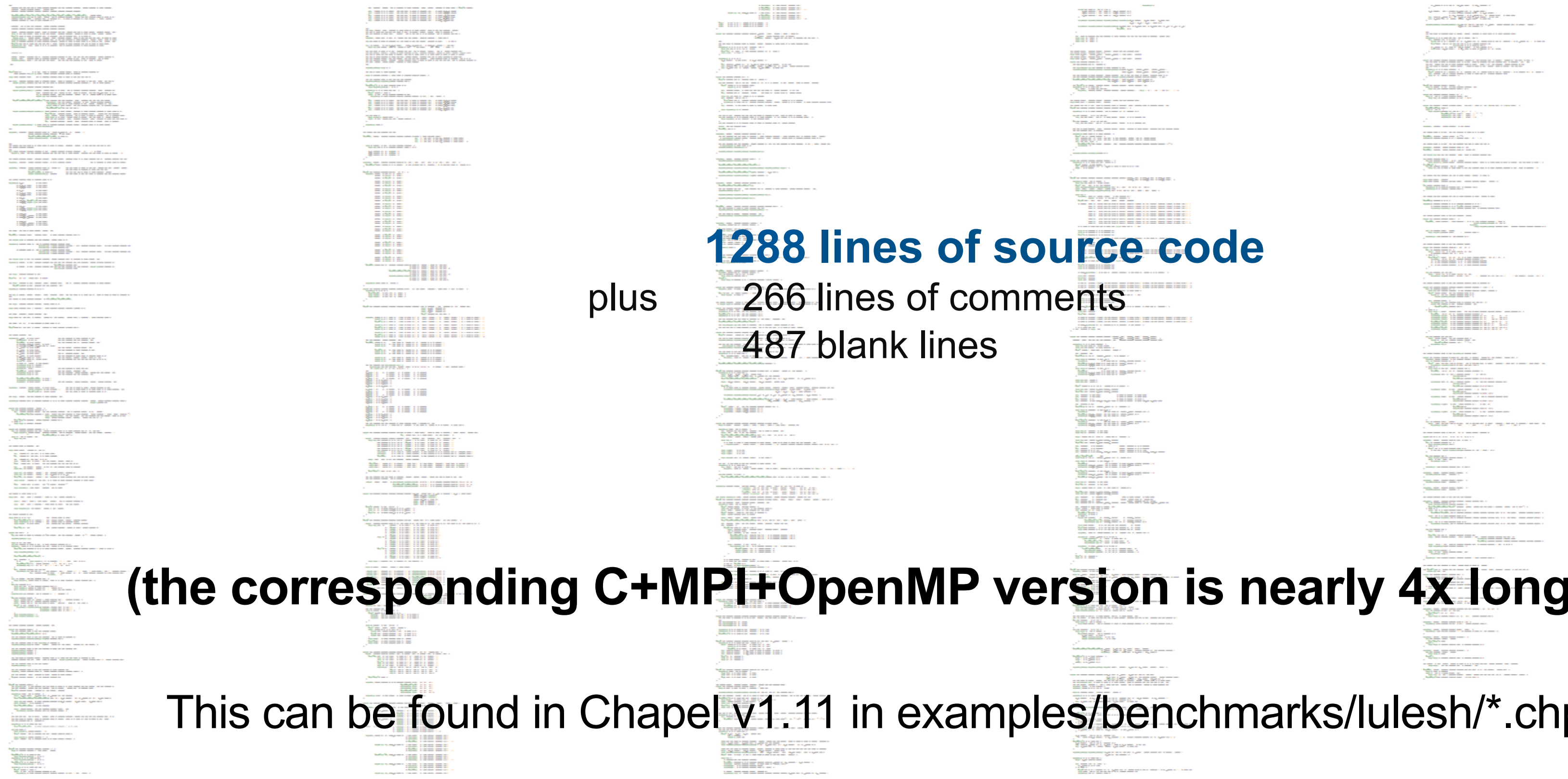
Copyright 2015 Cray Inc.

LULESH in Chapel



Copyright 2015 Cray Inc.

LULESH in Chapel



1288 lines of source code

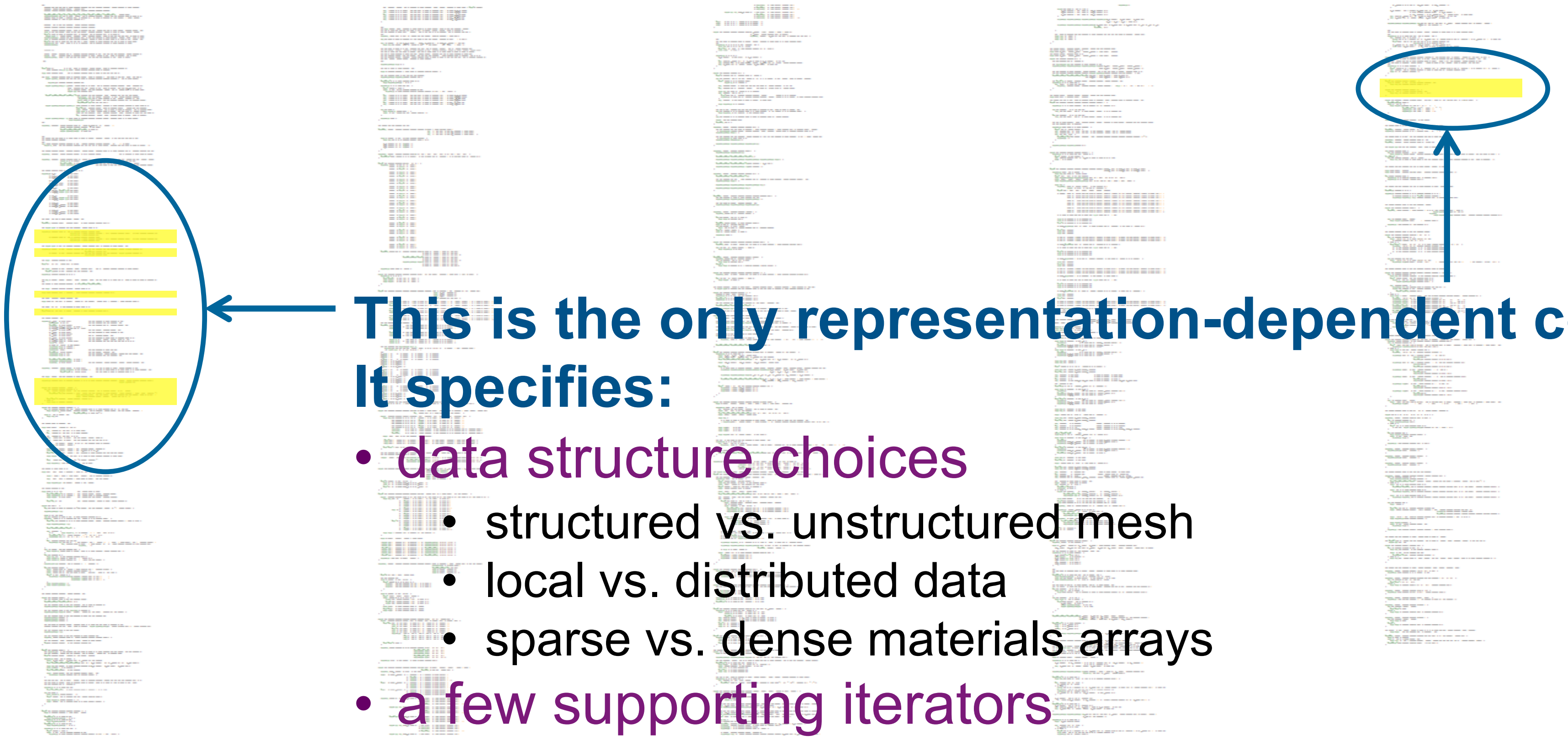
plus 266 lines of comments
487 blank lines

(the corresponding C+MPI+OpenMP version is nearly 4x longer)

This can be found in Chapel v1.11 in examples/benchmarks/lulesh/*.chpl

Copyright 2015 Cray Inc.

LULESH in Chapel



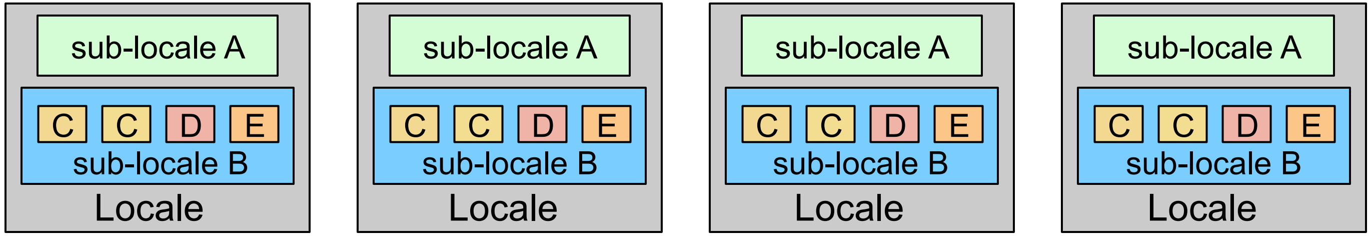
This is the only representation-dependent code.
It specifies:

- data structure choices
    - structured vs. unstructured mesh
    - local vs. distributed data
    - sparse vs. dense materials arrays
- a few supporting iterators

Copyright 2015 Cray Inc.

Hierarchical Locales for Emerging Architectures

- Support locales within locales to describe architectural sub-structures within a node (e.g., memories, processors)



- *On-clauses* and *domain maps* map tasks and variables to sub-locales
- Supports intra-node NUMA regions and hybrid processors

Copyright 2015 Cray Inc.

Summary

- **Technology changes are driving significant changes to node architecture and memory hierarchies**
  - Billion-way parallelism, hybrid processors, deeply hierarchical memories
- **As a first step, need to transition codes to hierarchical parallelism**
  - Distributed memory + threading + vectorization
- **…and focus more on data placement in memory hierarchy**
  - Data motion is *much* more expensive than computation
- **Cloud won't take over, but can we play nicely together?**
- **Lots of opportunities to combine HPC and analytics**
- **Let's see if we can bridge the gap HPC and analytics communities**
  - Goal: Performance + Productivity

Copyright 2015 Cray Inc.

Thank You.

Questions?