# Reflections on Programming Environments and Productivity
## (based on experiences with HPCS and Chapel)

Brad Chamberlain, Chapel Team, Cray Inc.

ASCR Exascale Computing Systems Productivity Workshop

June 3rd, 2014

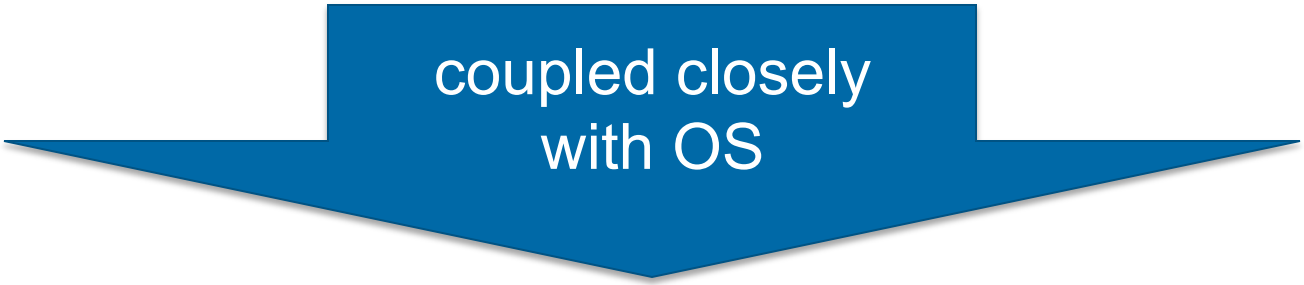
# Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

# Programming Environments


- **Programming Notations / Programming Models**
  - languages, language extensions, DSLs, pragmas, libraries, …
- **Tools**
  - debuggers, profilers, autotuners, IDEs, …
- **Runtime Support**
  - communication, tasking, memory, I/O

my bias



# Chapel (part of the reason for my bias)

- **An emerging parallel programming language**
  - Design and development led by Cray Inc.
    - in collaboration with academia, labs, industry
- **Goal: Improve productivity of parallel programming**
- **A work-in-progress**

# Productivity: Traditional, pre-Exascale Concerns

# What does "Productivity" mean to you?

**Recent Graduates:**

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

**Seasoned HPC Programmers:**

"that sugary stuff that I don't need because I ~~was born to suffer~~ want full control to ensure performance"

**Computational Scientists:**

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

**Chapel Team:**

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

# Productivity: My nightmare scenario

## Scenario:

- A mainstream computing buddy wants to do some scalable parallelism
  - Accustomed to using Python, Matlab, or Java, say
  - Also IDEs with auto-completion, refactoring, integrated debugging, …
- Knowing you're an expert in the field, wants recommendations

## The source of my fears:

"As the HPC community, do we have anything we can recommend as a productive solution to such a person with a straight face?"

"Do any of us even recognize what productivity means to most programmers anymore? Would we know it if it bit us on the leg?"

# How to attract/retain HPC programmers?

# One Answer: "Decent" Parallel Languages

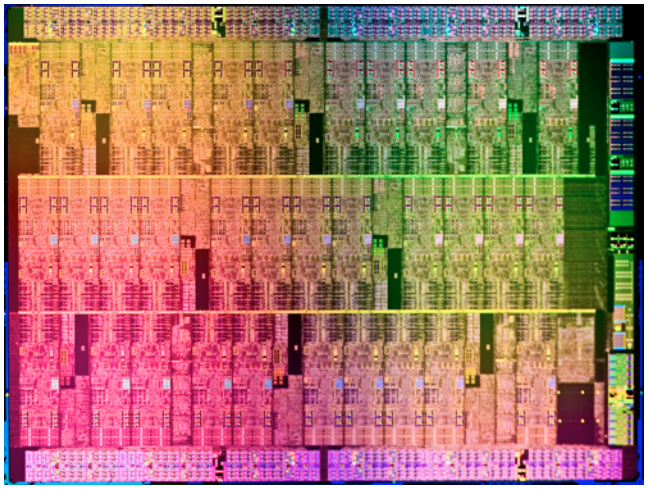
**What was the last parallel notation you used that felt:**

- productive?
- high-level?
- powerful?
- flexible?
- effective?
- modern?
- fun?
- (…all the things we judge good software by…)?

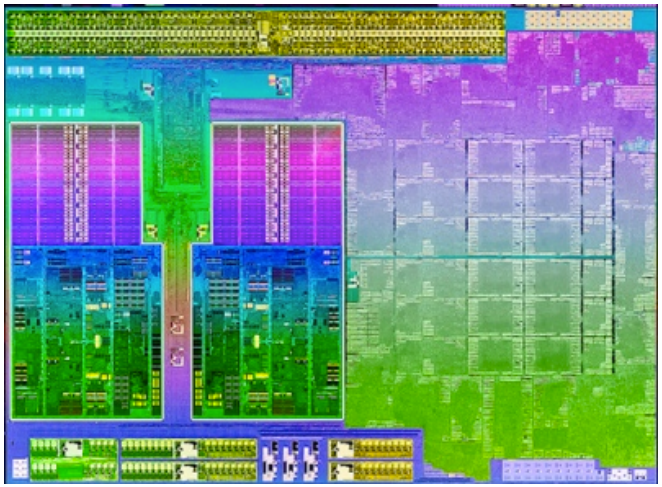

***(Because that's what we're competing against when it comes to attracting new talent)***

# Productivity in the Exascale Era

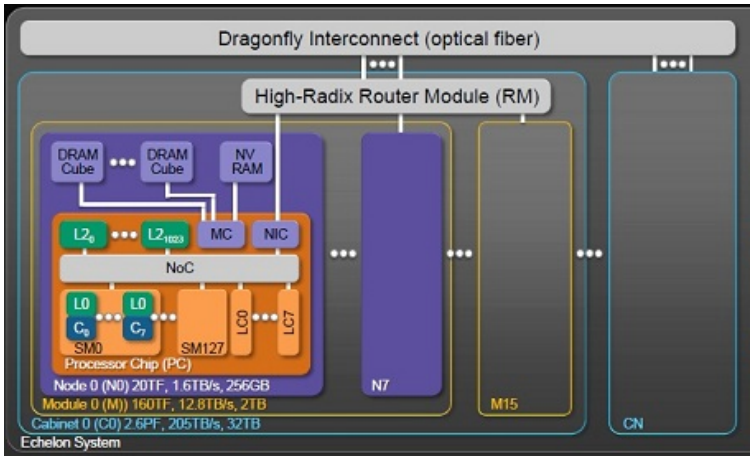
# Prototypical Exascale Processor Technologies



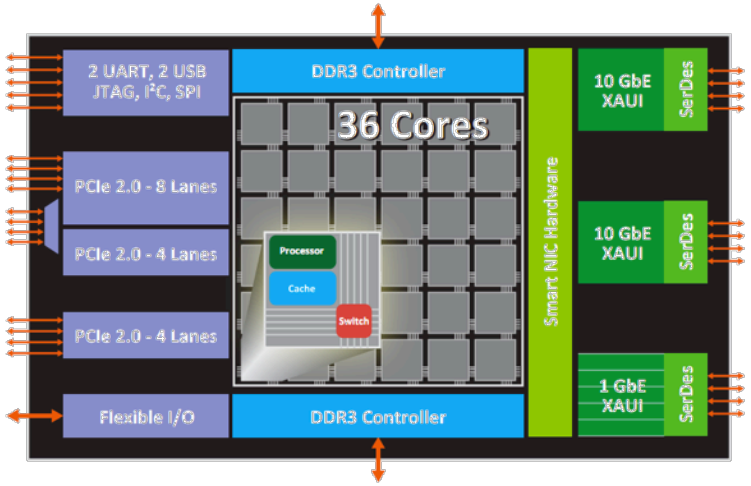
Intel MIC



AMD APU



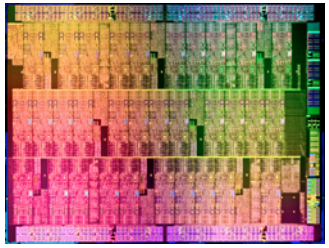

Nvidia Echelon

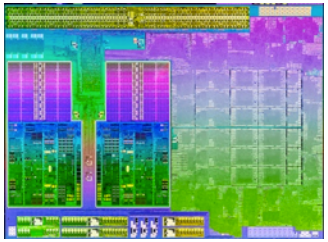


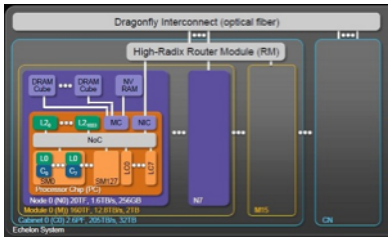

Tilera Tile-Gx

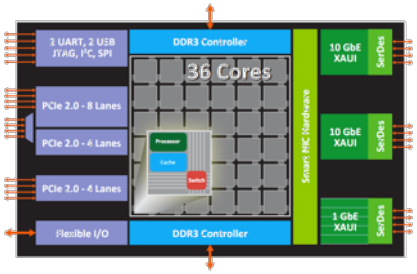
http://download.intel.com/pressroom/images/Aubrey_Isle_die.jpg http://www.zdnet.com/amds-trinity-processors-take-on-intels-ivy-bridge-3040155225/
http://insidehpc.com/2010/11/26/nvidia-reveals-details-of-echelon-gpu-designs-for-exascale/ http://tilera.com/sites/default/files/productbriefs/Tile-Gx%203036%20SB012-01.pdf

# Exascale: Programmer Productivity Challenges








**Emerging processor designs…**

**…are increasingly locality-sensitive**

**…potentially have multiple processor/memory types**

⇒ Exascale programmers will have a lot more to think about at the node level than in the past

⇒ What will it take to keep the productivity bar level (to say nothing of improving it?)

# Summarizing: Three Productivity Challenges

1. How to improve productivity relative to current practice?
2. How to improve productivity to entice new users?
3. How to maintain productivity in the face of exascale?
   - let alone improve it?

**Happily, #3 gives us a renewed excuse to work on #'s 1 & 2:**

If we're going to have to switch to something new anyway, it's a great opportunity to change to something truly productive

# Productivity, HPCS, and Cray:

# A Brief History/Review

# Productivity, as defined by HPCS

**Productivity (10x improvement goal) =**

performance
+ programmability
(readability, writeability, maintainability, modifiability, tunability, …)
+ portability
+ robustness

A reasonable starting point…

…yet, how to combine four areas down to a single metric?

- particularly given that most of them are hard to measure individually?

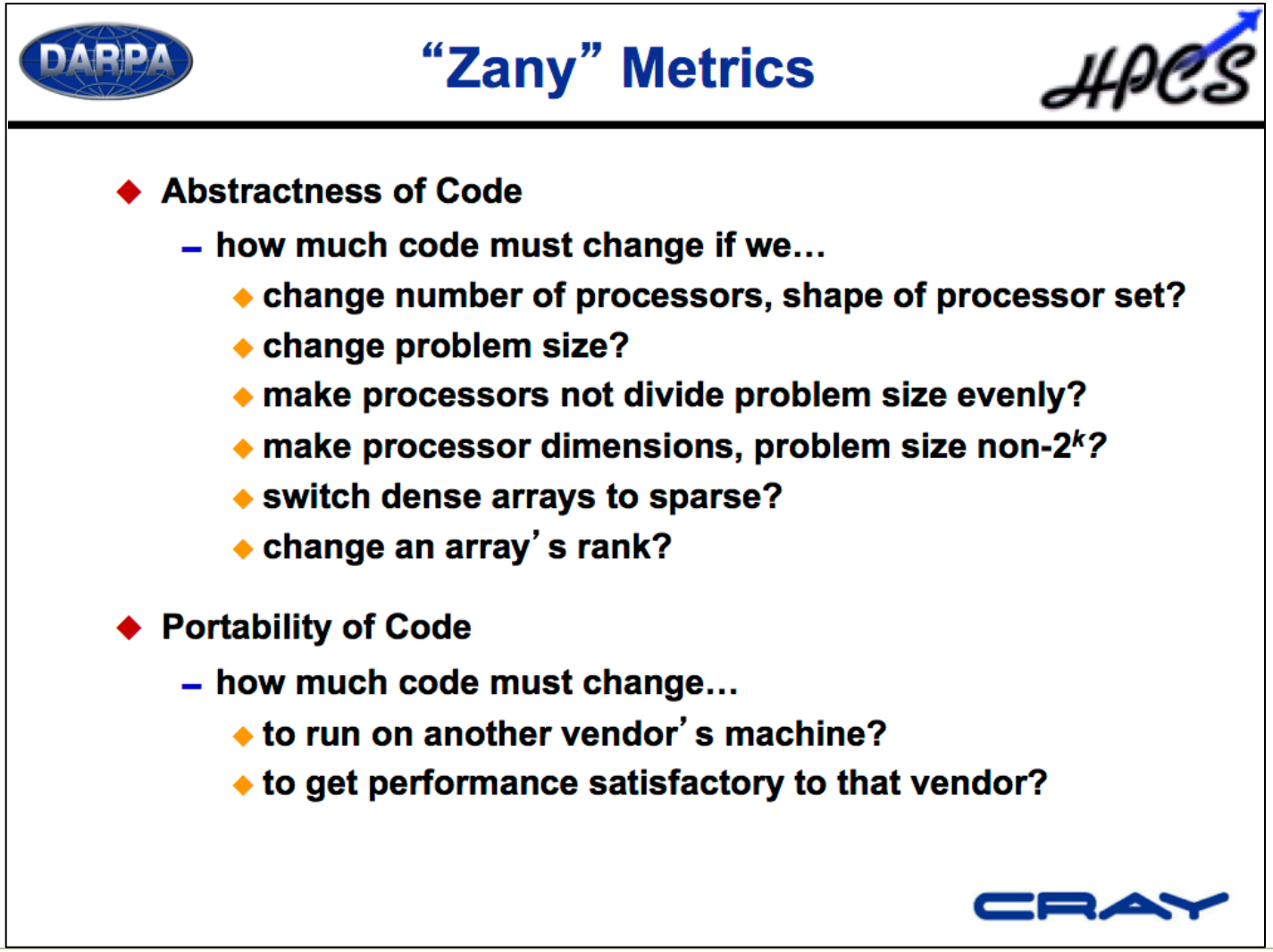
Also some unreasonable (IMO) goals/expectations:

- Initially, a stated desire to see the establishment of Moore's Law-style productivity improvements year after year

# Productivity: Played Out?

- **There's some sense that productivity isn't "hot" anymore**
  - "Didn't we [solve | fail to solve] that in HPCS?"
- **Arguably analogous to "peace"**
  - not particularly "new" or "hip" as an concept
  - reasonable causes for skepticism about our ability to achieve it
  - but clearly something to desire over the alternatives
- **Personally, I prefer not to throw in the towel** (in either case)

# My "Zany Metrics" (a brainstorming exercise)








(HPCS Phase II Metrics Kickoff: 2003-8-5)

# “Language Bingo”

## Language Comparison

| | MPI | SHMEM | Java | UPC | CAF | HPF | OpenMP | Fortran/ C |
|---|---|---|---|---|---|---|---|---|
| Performs Well | O | O | ? | ? | O | ? | ? | ~ |
| Portable | O | ? | O | ? | ? | ~ | X | ~ |
| Performance Model | O | O | O | O | O | X | X | X |
| Global View | X | X | X | X | X | O | O | O |
| Post-scalar | ~/X | ~/X | O | X | ~ | ~ | ~/X | ~/X |
| Abstractions | X | X | O | ~ | ~ | X | X | X |
| Succinct | X | X | X | X | X | ~ | ~ | ~ |
| General | O | O | O | ~ | X | X | X | O |

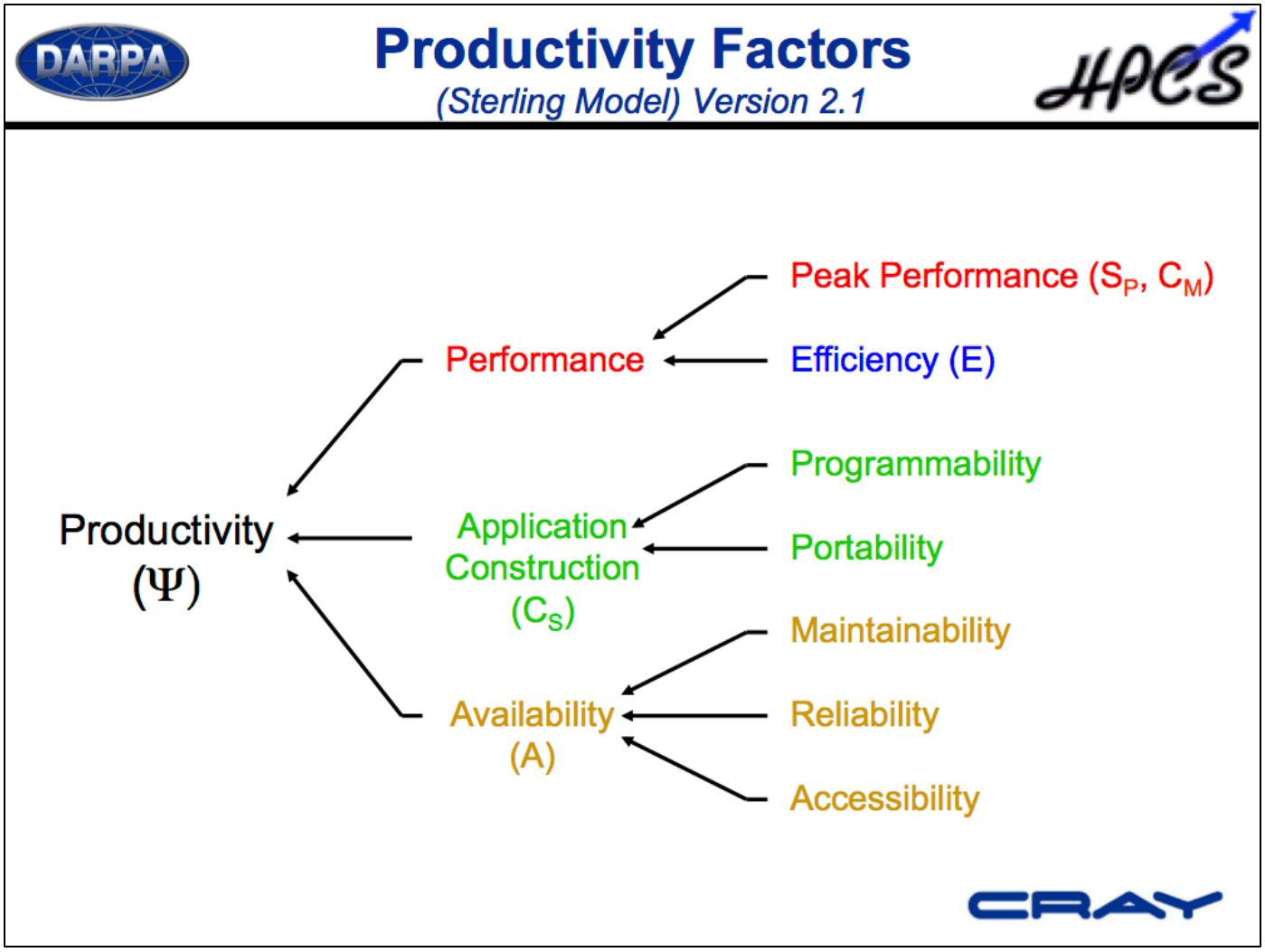
-- = no comment  O = good  ~ = so-so  X = poor  ? = unproven





(slide from some HPCS productivity meeting)

# Sterling's Model of Productivity








(HPCS Phase II Metrics Kickoff: 2003-8-5)

# Sterling's Model of Productivity

## General Model of Productivity

$R_i \equiv i^{th}$ result product

$T_i \equiv$ time to compute result $R_i$

$T_L \equiv$ total lifetime of machine

$T_V \equiv$ total overhead time of machine

$T_Q \equiv$ quiescent time of machine

$T_R \equiv$ working time of machine

$N_R \equiv$ total number of result products during $T_L$

$C_L \equiv$ all costs associated with machine during $T_L$

$C_{LS} \equiv$ application software costs during $T_L$

$C_{LO} \equiv$ costs of ownership during $T_L$

$C_M \equiv$ cost of procurement and initial installation

$C_{Si} \equiv$ cost of application software for result $R_i$

$\Psi \equiv$ productivity

$$T_R = \sum_i^{N_R} T_i$$

$$T_L = T_R + T_V + T_Q$$

$$R_L = \sum_i^{N_R} R_i$$

$$C_L = C_{LS} + C_M + C_{LO}$$

$$C_{LS} = \sum_i^{N_R} C_{Si}$$

$$\Psi \equiv \frac{R_L}{C_L \times T_L}$$

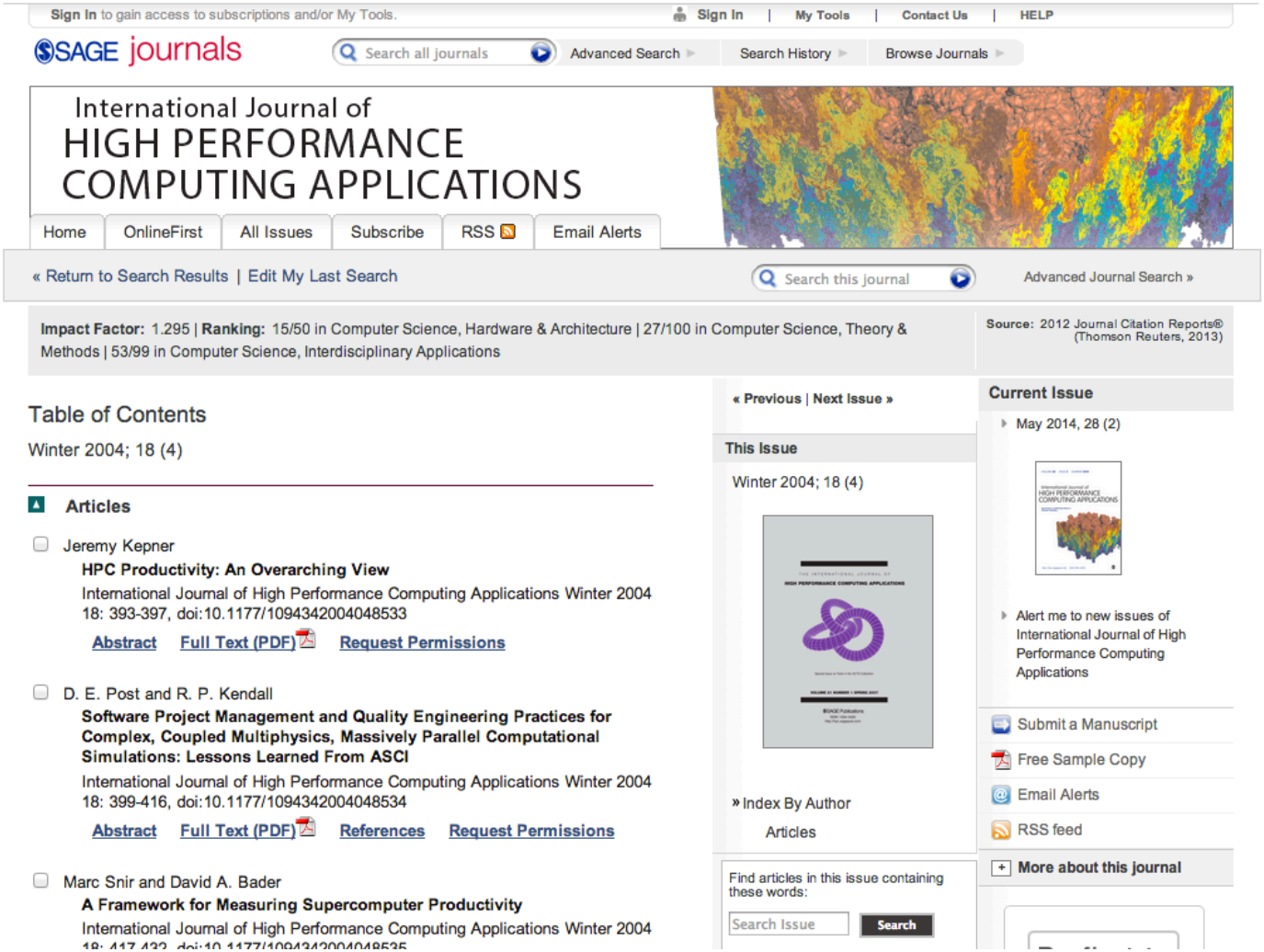
…coincidence?!? I think not…





(HPCS Phase II Metrics Kickoff: 2003-8-5)

# Various Others' Models of Productivity



Sign In to gain access to subscriptions and/or My Tools. Sign In | My Tools | Contact Us | HELP

SAGE journals — Search all journals — Advanced Search — Search History — Browse Journals

International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

Home | OnlineFirst | All Issues | Subscribe | RSS | Email Alerts

« Return to Search Results | Edit My Last Search — Search this journal — Advanced Journal Search »

**Impact Factor:** 1.295 | **Ranking:** 15/50 in Computer Science, Hardware & Architecture | 27/100 in Computer Science, Theory & Methods | 53/99 in Computer Science, Interdisciplinary Applications

**Source:** 2012 Journal Citation Reports® (Thomson Reuters, 2013)

Table of Contents

Winter 2004; 18 (4)

**Articles**

Jeremy Kepner
**HPC Productivity: An Overarching View**
International Journal of High Performance Computing Applications Winter 2004 18: 393-397, doi:10.1177/1094342004048533
Abstract | Full Text (PDF) | Request Permissions

D. E. Post and R. P. Kendall
**Software Project Management and Quality Engineering Practices for Complex, Coupled Multiphysics, Massively Parallel Computational Simulations: Lessons Learned From ASCI**
International Journal of High Performance Computing Applications Winter 2004 18: 399-416, doi:10.1177/1094342004048534
Abstract | Full Text (PDF) | References | Request Permissions

Marc Snir and David A. Bader
**A Framework for Measuring Supercomputer Productivity**
International Journal of High Performance Computing Applications Winter 2004

« Previous | Next Issue »

This Issue

Winter 2004; 18 (4)

» Index By Author

Articles

Find articles in this issue containing these words:

Search Issue — Search

Current Issue

May 2014, 28 (2)

Alert me to new issues of International Journal of High Performance Computing Applications

Submit a Manuscript

Free Sample Copy

Email Alerts

RSS feed

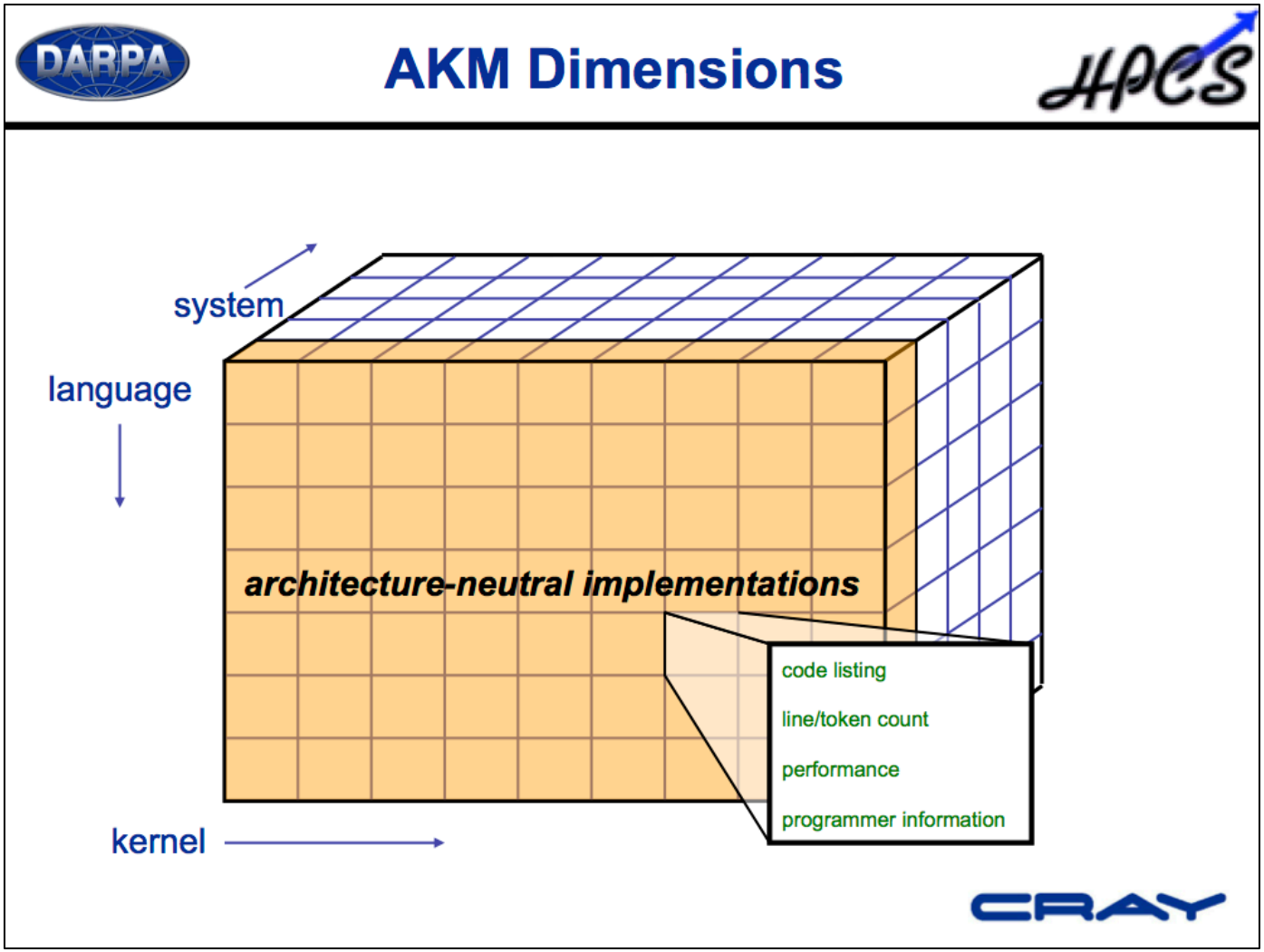
More about this journal





http://hpc.sagepub.com/content/18/4.toc

# The Application Kernel Matrix

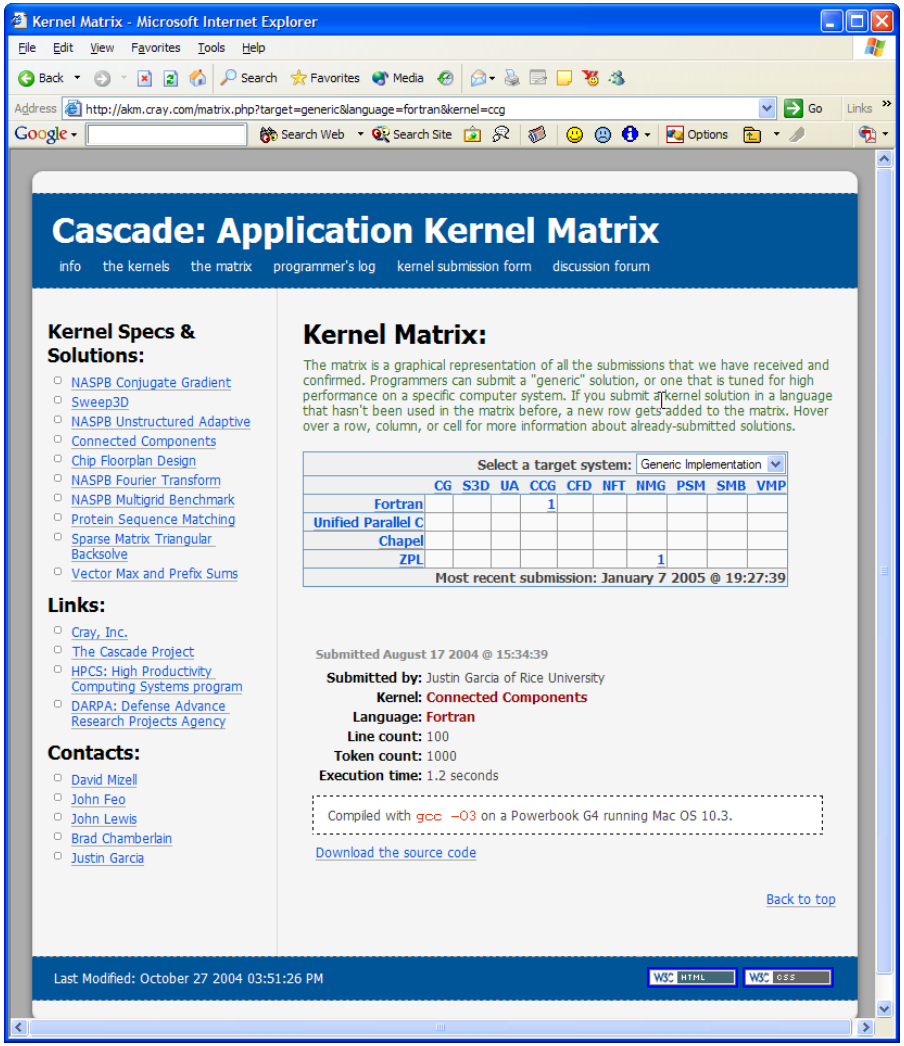







(Marina del Mar SW Productivity Workshop, 2005?)

# The Application Kernel Matrix Website (R.I.P.)




(While the AKM did not catch on, other similar comparison sites have, at least in the mainstream

One of my favorites (content and form):

- The Computer Language Benchmarks Game http://benchmarksgame.alioth.debian.org/

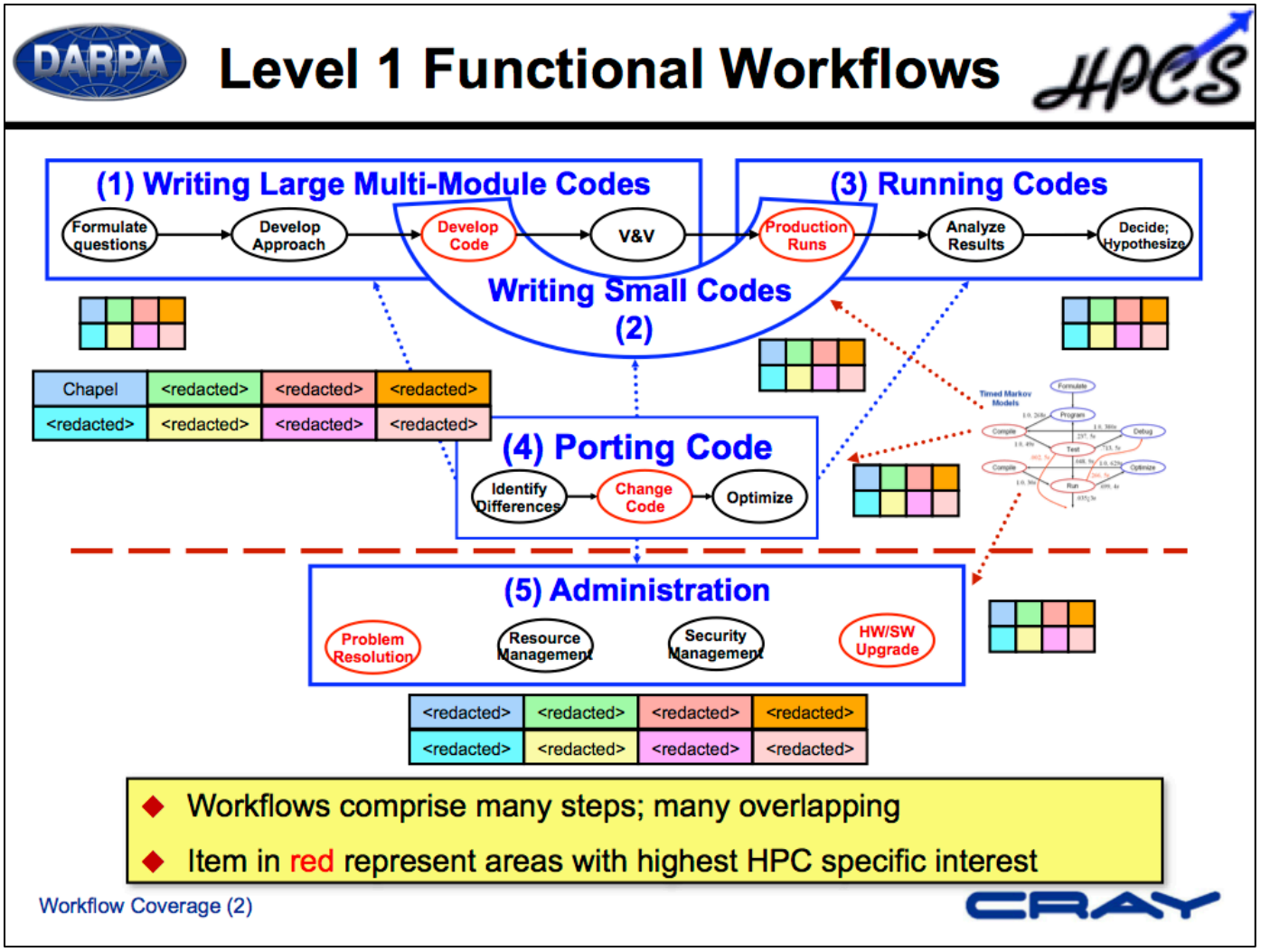
The HPC Challenge, Berkeley Dwarfs, and Graph-500 arguably play similar roles).





(defunct website, formerly http://akm.cray.com)

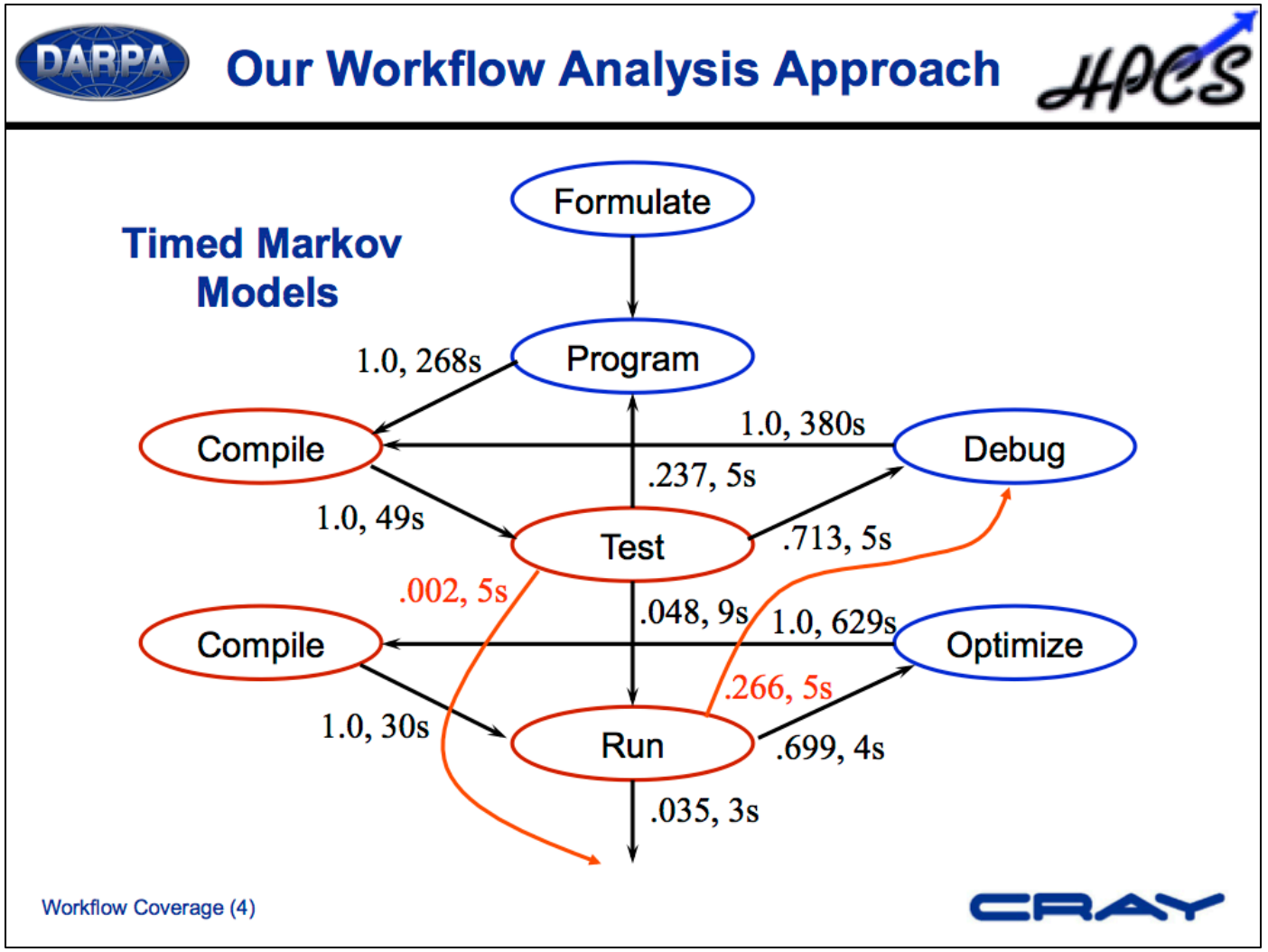
# HPCS Workflows






(slide from internal Cray government review)

# Timed Markov Models



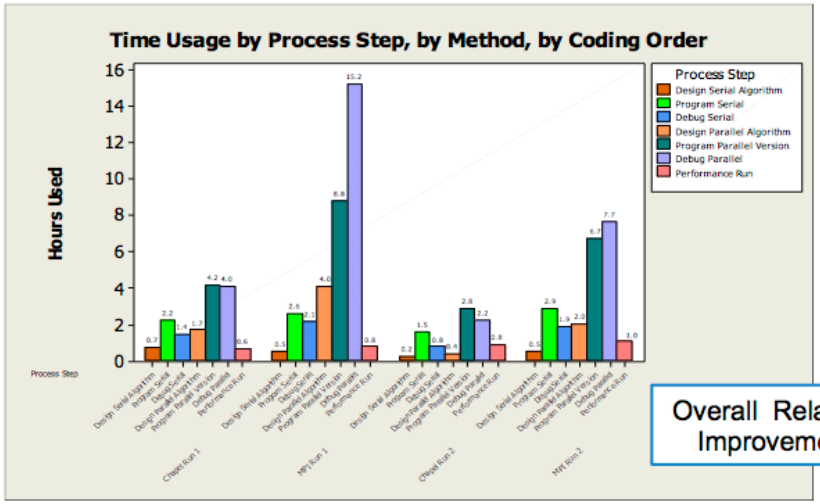





(slide from internal Cray government review)

# User Studies: Quantitative Evaluation

## Workflow 2 – Productivity Improvement Summary




Overall Relative Productivity Improvement 2.3 to 2.6 X

- Debug Parallel Code
  - Programmers created less bugs when programming in Chapel
    - Global view of data simplifies programming
- Program Parallel
  - Fewer changes needed when going from serial to parallel
    - Multi-resolution of parallelism makes it easier to go from serial to parallel
- Designing Parallel Algorithm
  - MPI programs frequently require restructuring of the serial C code, finding right MPI lib routine





(slide from internal Cray government review)

# User Studies: Qualitative Evaluation

"The biggest feature from a broad perspective for me was domains. Especially for scientific codes, it is invaluable to be able to define the couple problem domains you're working with. It makes it trivial to change the size or layout or distribution if you decide you need to, it helps guarantee that all of your different arrays match up. **A 3D rectangular grid is infinitely more clear in Chapel with domains than in C**, where you have to figure out how they laid it out (is it one giant array? what is the major dimension? x? z? y?)."

> "**I loved not having to think as hard** about offsets and counts for the parallel version of the code **in Chapel, as opposed to the MPI version**, where I almost always had to chase down two or three indexing errors."

"Lastly, **I'm a huge huge fan of the type inference used in Chapel.** I like that I don't have to specify types everywhere--they can just be inferred from how I'm using them, but if I mess something up, the compiler catches it."

# Summary: Many Useful Concepts/Techniques…

# BUT…

# The Catch with Productivity Metrics

**My suspicion:** If I were to tell you that any of these metrics…

…demonstrated that Chapel was 10x better than Fortran+MPI
…showed that X10 was 2x better than Chapel
…indicated that Perl was 5x better than Python
…proved that emacs was 8.5x better than vim
…dared suggest that alpine is 7x better than Outlook

…would you believe me? Enough to change your practices?

**Maxim #1:** "I can't define productivity, but I know it when I see it"

- "And seeing is believing"

**Maxim #2:** "Productivity is in the eye of the beholder"

- "To each his/her own productivity solutions"

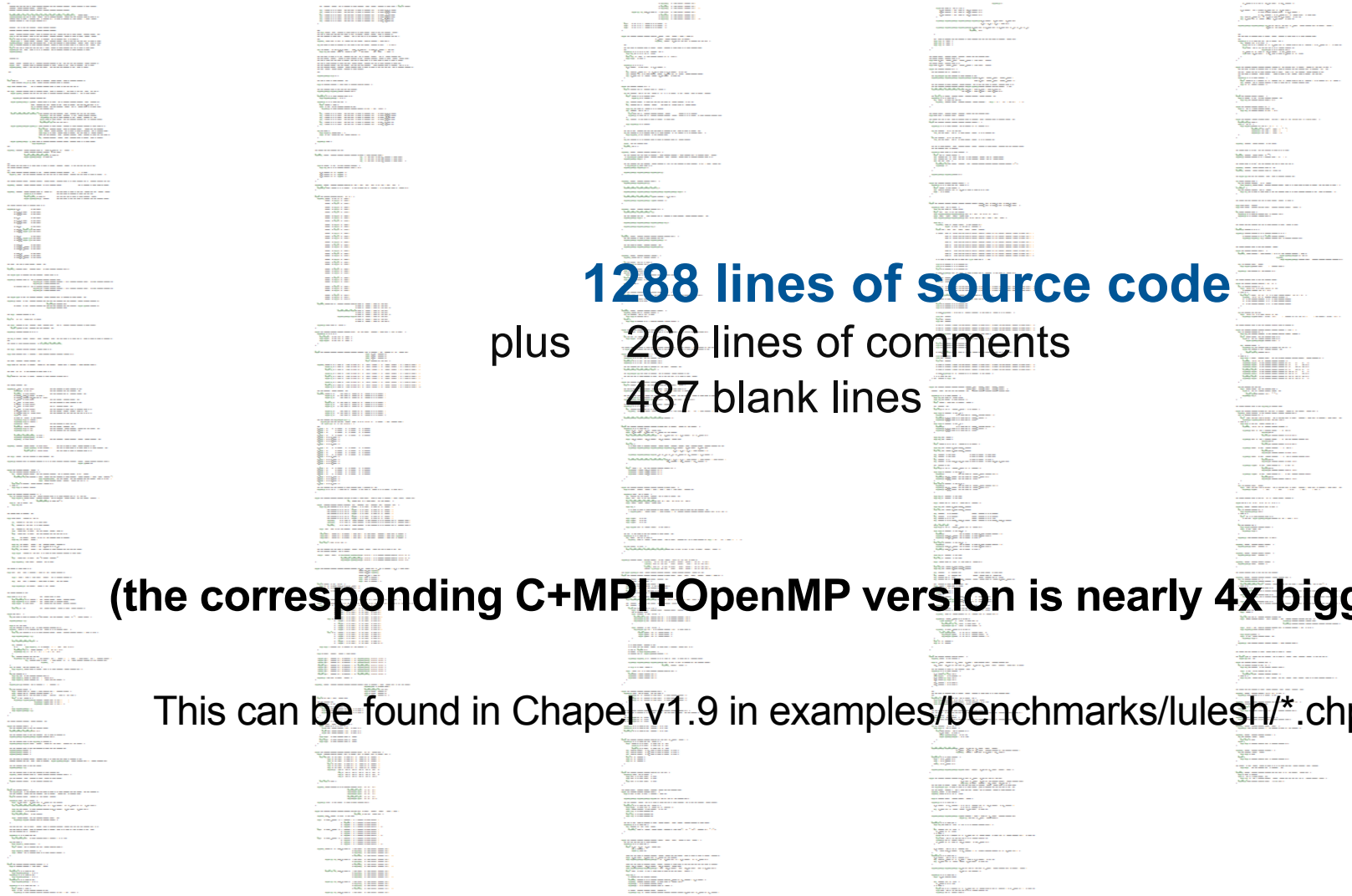
# If Not Metrics, Then What?

**For many of us, case studies can be compelling...**

**Here are three examples from Chapel**

# Case 1: LULESH in Chapel

# LULESH in Chapel



**1288 lines of source code**

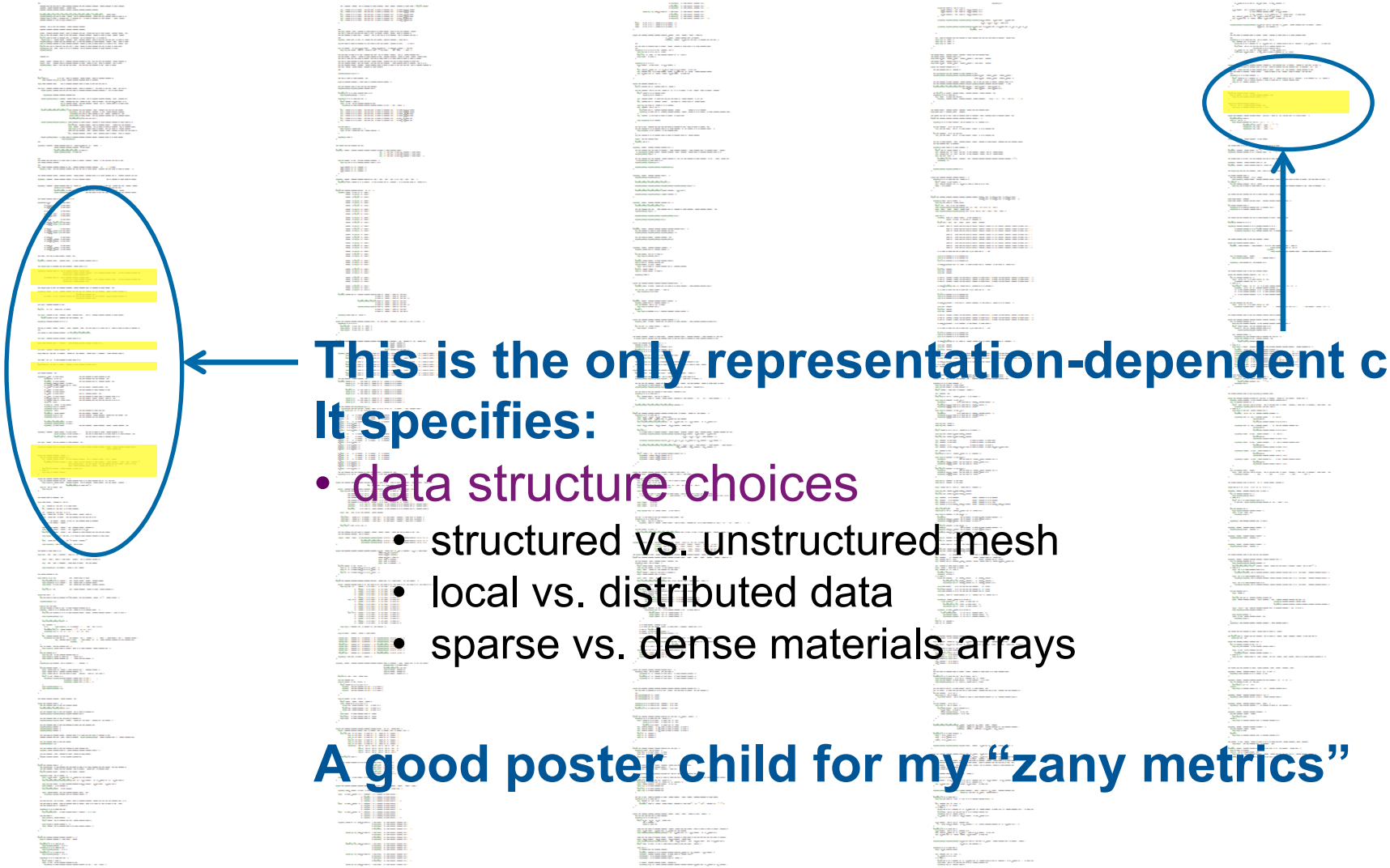
plus 266 lines of comments

487 blank lines

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

This can be found in Chapel v1.9 in examples/benchmarks/lulesh/*.chpl

# LULESH in Chapel



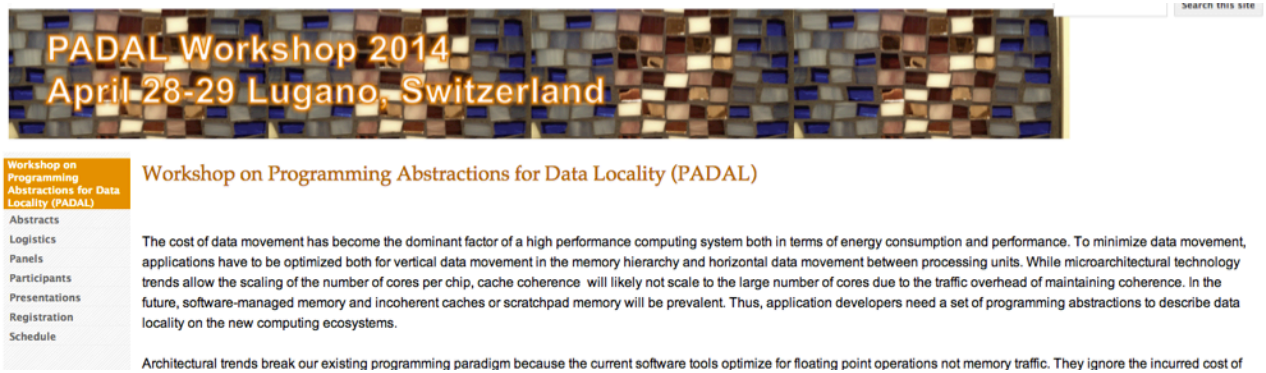
**This is the only representation-dependent code. It specifies:**

- data structure choices
  - structured vs. unstructured mesh
  - local vs. distributed data
  - sparse vs. dense materials arrays

**A good poster child for my "zany metrics"**

# LULESH's Technical Productivity Lesson

- **Put hardware-mapping-specific choices in declarations**
  - Makes computation independent of key decisions like:
    - memory layouts
    - distributions
    - sparse vs. dense
    - # of dimensions
  - Supports switching between options easily
- **This is a design trend that (happily) seems to be growing**
  - Several similar designs reported on at PADAL 2014 workshop






http://padalworkshop.org

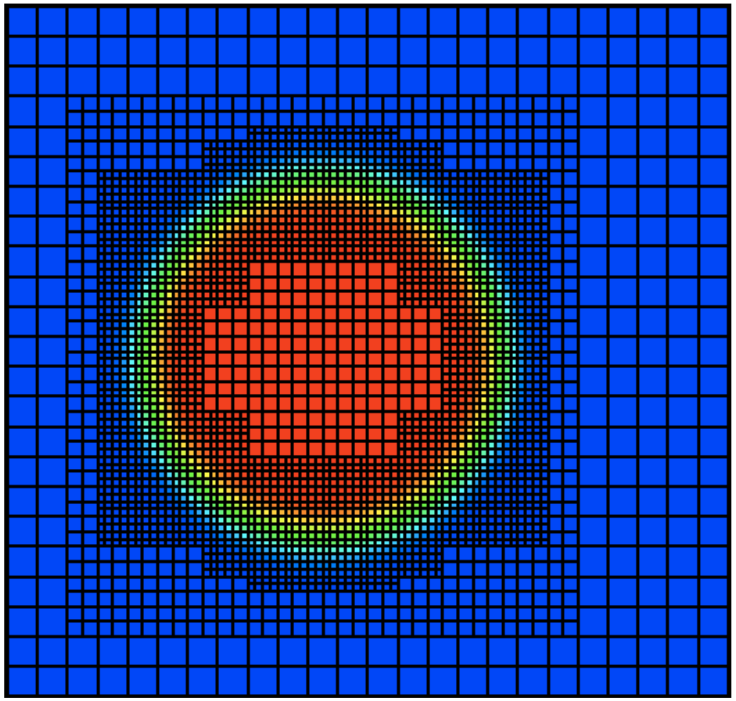
# LULESH's Social Productivity Lesson

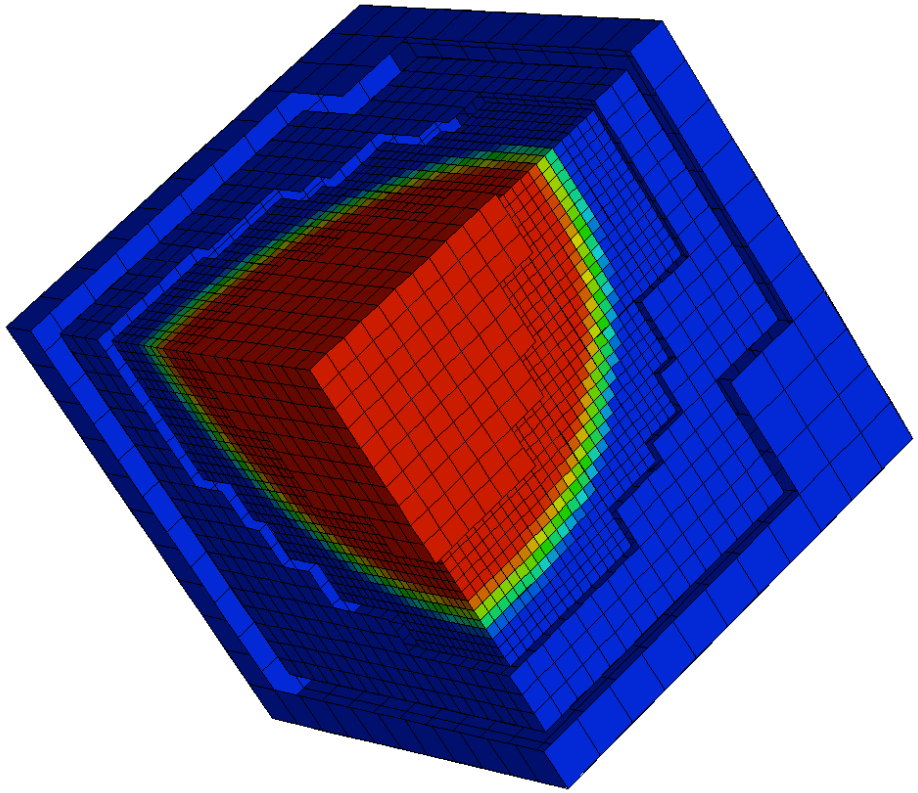
- **Written by intern in final 2 weeks as a "bonus" project**
  - productive!
- **Illustrated Chapel use in a familiar setting to scientists**
  - ones who'd heard many Chapel talks previously, and yet…
- **Served as a medium for discussion, collaboration**
- **Demonstrated productivity of features thought not to be**
  - global indexing, sparse domains
- **Provided bidirectional feedback/knowledge transfer**
  - co-design!

**(a nice win for DOE's proxy apps effort)**

# Case 2: Chapel Rank-Independent AMR Framework

**UW applied mathematician wrote one code that could be used to produce results in 2D, 3D, 6D, 17D…**



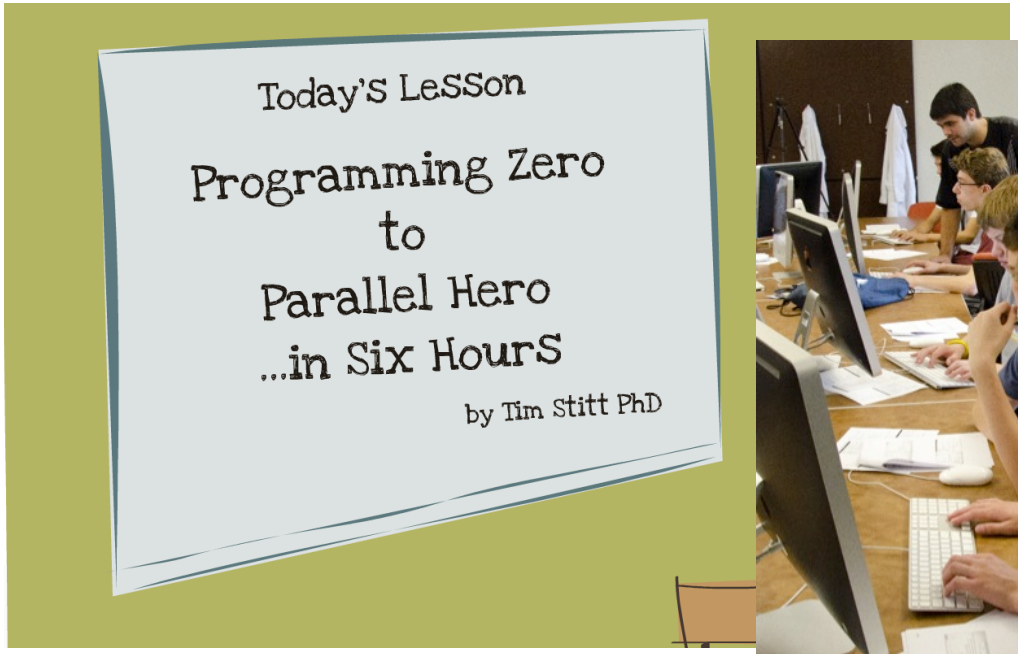






(pictures courtesy of Jonathan Claridge, UW Amath)

# Case 3: Chapel's Appeal to Educators/Students






http://prezi.com/wp13iqmsl1di/summer-scholars/?utm_campaign=share&utm_medium=copy

SIGCSE ATLANTA 2014

March 5-8, 2014
Atlanta, Georgia

OpenConf Peer Review & Conference Management System

OpenConf Home | Email Chair

Full Program »

**Chapel: A versatile tool for teaching undergraduates parallel programming**

Chapel is a programming language being developed for high-performance applications. It is well-suited for teaching parallelism in a wide variety of undergrad courses. Chapel is easy to learn since it supports a low-overhead style like a scripting language as well as

http://faculty.knox.edu/dbunde/teaching/chapel/SIGCSE14/

## CSEP524: Parallel Computation

### Software

**Pthreads:** *(included with the above)*

**OpenMP:** *(included with the above)*

**Chapel:** chapel-fedora17-1.6.1.1.tar.gz
*(This is a pre-release of the Chapel 1.6.1 sources, pre-compiled for the Fedora 17 VM; and skipping step 2 if you're using the VM)*

**MPI:** Setup MPI
*This describes how to install MPI and how to run it locally and on our course VM cluster.*

http://courses.cs.washington.edu/courses/csep524/13wi/

# Chapel Wrap-up

- (Many other productive demonstrations in addition…)
- Chapel's Productivity Scorecard
  - Performance
  - Programmability
  - Portability
  - Robustness
- Work is ongoing to improve Chapel's weak areas
  - Productivity often requires long-term investment and patience

# Summary / Takeaways

- **Productivity still matters**
  - even if the term is well-worn
- **Exascale brings new productivity challenges**
  - but also an opportunity to improve upon past approaches
- **Not convinced we can measure productivity**
  - nor that it matters whether or not we can
- **Proxy apps are a useful medium for productivity studies**
  - serve as a place to demonstrate productivity features
  - serve as meeting place for distinct communities
- **Educators and Students are a good resource**
  - can't stand in for today's experts, but may be tomorrow's
- **Productivity gains may not happen overnight**

# Funding Productivity: Personal Opinions

- **Invest in software to compensate for hardware challenges**
- **Pursue a combination of evolution and revolution**
  - can't afford to just do one
- **Evaluate based on user (or prospective user) viewpoints**
  - computer science viewpoints are not necessarily as relevant
- **Must budget for evaluation/study of new technologies**
  - can't expect such studies to be spare time activities
  - requires time from experts, not simply novices
  - again, well-designed proxy apps can be useful here

# Legal Disclaimer

*Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.*

*Cray Inc. may make changes to specifications and product descriptions at any time, without notice.*

*All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.*

*Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.*

*Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.*

*Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.*

*The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.*


CRAY
THE SUPERCOMPUTER COMPANY

http://chapel.cray.com chapel_info@cray.com http://sourceforge.net/projects/chapel/