



SiCortex

Broadening the HPC Franchise

Matt Reilly
Chief Engineer
SiCortex, Inc

The Problem: Too few people are competent at High Processor Count Computing



The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.

The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.

The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.



The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.



PETRONAS Twin Towers - Front View
87 kb
Copyright 2005 PETRONAS

The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.
- And I haven't even mentioned data mining...

The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.
- And I haven't even mentioned data mining...
- Our compute problems are rapidly outstripping and outclassing the talent pool we have to address them.

The Problem: Too few people are competent at High Processor Count Computing

- Technology permits rapid increase in system complexity and enterprise ambition.
- The chief means of coping with complexity and mitigating risk are modeling, simulation, and synthetic experimentation.
- And I haven't even mentioned data mining...
- Our compute problems are rapidly outstripping and outclassing the talent pool we have to address them.
- And the primary growth in computing capacity is coming from parallelism.

The Problem: Too few people are competent at High Processor Count Computing



**We need to make high processor count
computing available to more of our
colleagues.**

What can we do?

What can we do?

- Education

What can we do?

- Education
 - Improve curriculum and exposure for “geeks in training”

What can we do?

- Education
 - Improve curriculum and exposure for “geeks in training”
- Current Practice

What can we do?

- Education
 - Improve curriculum and exposure for “geeks in training”
- Current Practice
 - Reduce barriers to adoption of HPC parallel systems

What can we do?

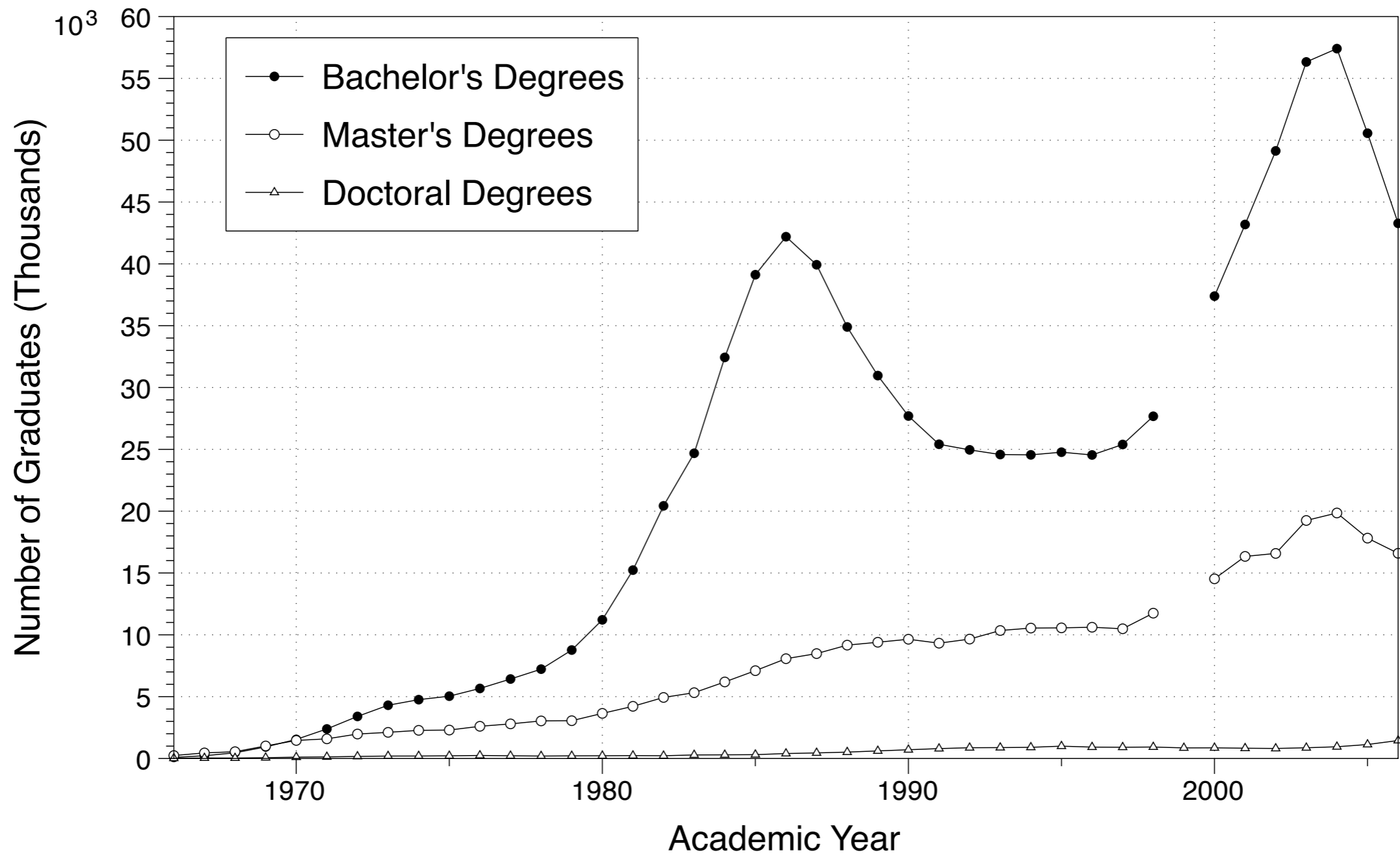
- Education
 - Improve curriculum and exposure for “geeks in training”
- Current Practice
 - Reduce barriers to adoption of HPC parallel systems
 - Develop domain specific infrastructures (libraries)

What can we do?

- Education
 - Improve curriculum and exposure for “geeks in training”
- Current Practice
 - Reduce barriers to adoption of HPC parallel systems
 - Develop domain specific infrastructures (libraries)
 - New language development (again?)

Computer Science Graduates

Computer Science Degrees Conferred by Year 1966 to 2006



Parallel Thinking and the ACM CS Curricula (A cursory survey)

Keyword	Number of “relevant” courses	
	CS 2001	SWVE 2004
Parallel	3 (none in “core”)	0
Distributed	4 (1 in core)	3
Concurrent	1 (none in core)	1
Multicore	0	0
Multiprocessor	0	0
Thread	0	0

“Relevant” requires term to appear in context of course title or syllabus related to program or algorithm development or software engineering practice. (excludes OS and architecture courses)

(Sources: The Joint Task Force on Computing Curricula IEEE CS and ACM, “Software Engineering 2004” and “Computing Curricula 2001, Computer Science”)

Curriculum



Curriculum

- Not a “specialty,” new major, or even a new emphasis.

Curriculum

- Not a “specialty,” new major, or even a new emphasis.
- Parallelism is a fundamental method, make it part of the fundamental toolbox.

Curriculum

- Not a “specialty,” new major, or even a new emphasis.
 - Parallelism is a fundamental method, make it part of the fundamental toolbox.
- Make large processor count systems available in all “project” related courses.

Curriculum

- Not a “specialty,” new major, or even a new emphasis.
 - Parallelism is a fundamental method, make it part of the fundamental toolbox.
- Make large processor count systems available in all “project” related courses.
- Build playgrounds.

A Playground?

A Playground?

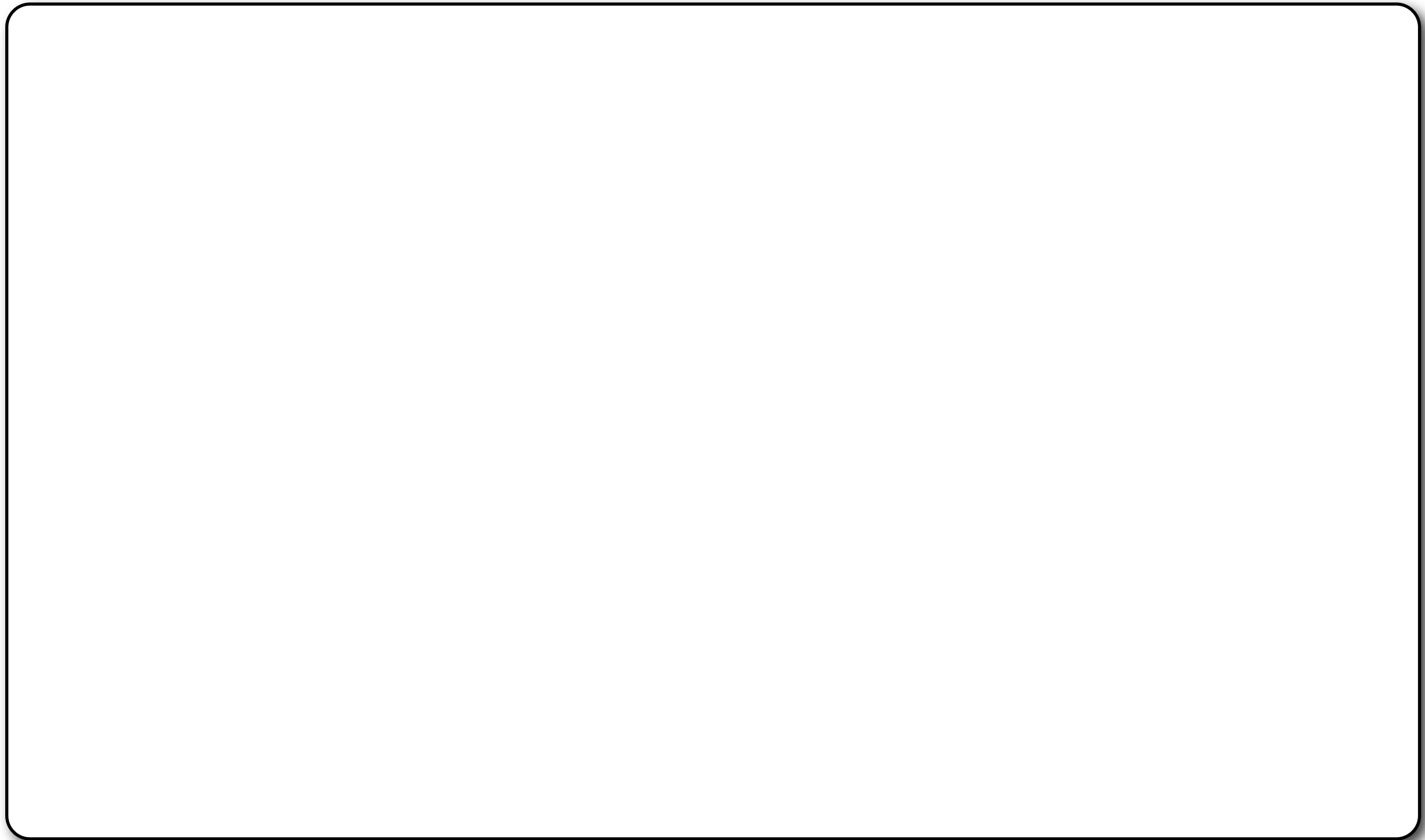
- Simple
- Inexpensive (per student)
- Extra curricular
- Scalable N from dozens to thousands

Is this practical?

A Playground



A Playground



A Playground

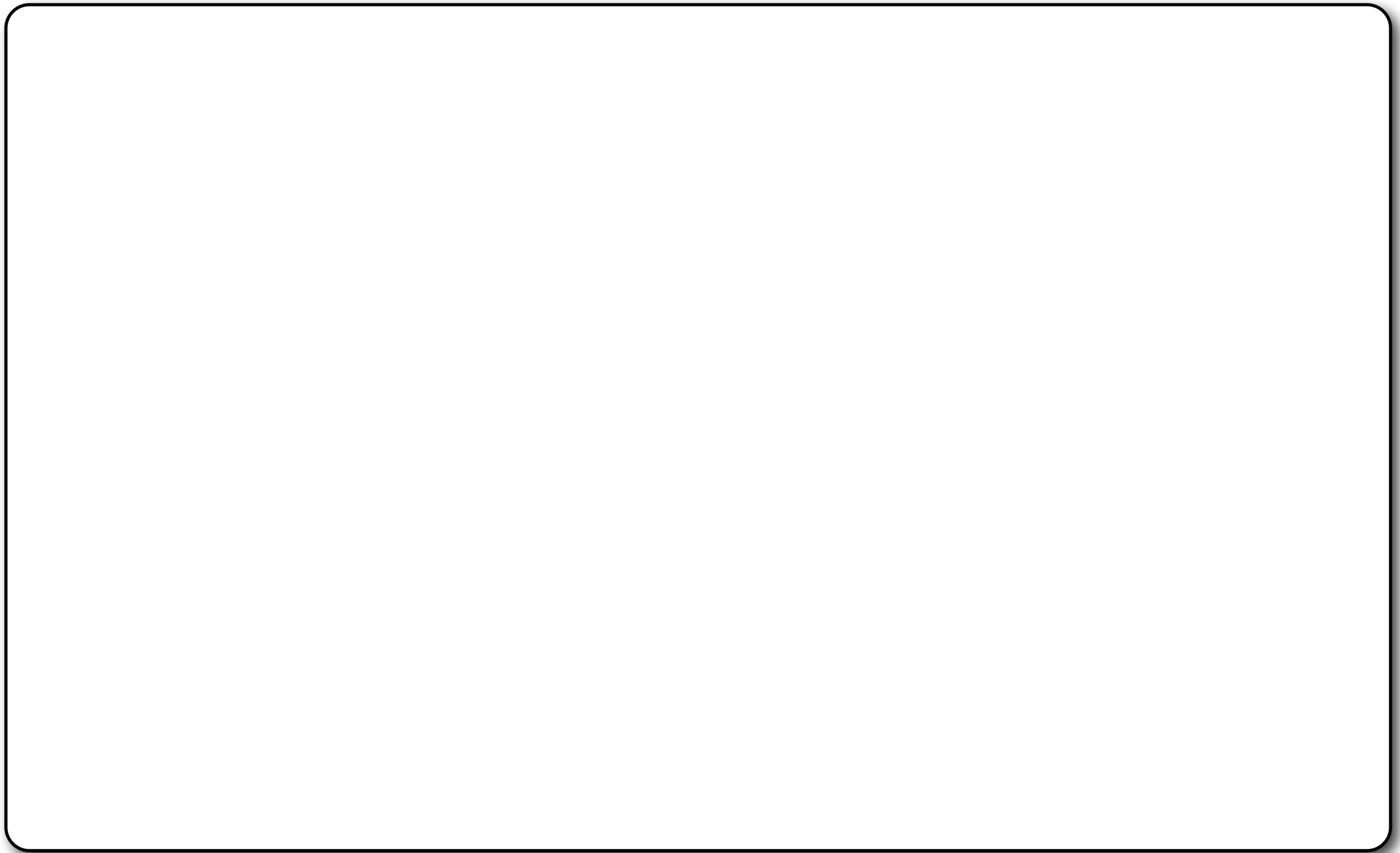


A Playground



Another Playground

Another Playground



Another Playground



Another Playground



Changing Current Practice

Drop the Barrier to Entry



Drop the Barrier to Entry

- Handbooks, examples of good practice, “app notes” from vendors and developers

Drop the Barrier to Entry

- Handbooks, examples of good practice, “app notes” from vendors and developers
 - Make use of the open source bazaar

Drop the Barrier to Entry

- Handbooks, examples of good practice, “app notes” from vendors and developers
 - Make use of the open source bazaar
 - e.n.s.g. <http://www-unix.mcs.anl.gov/dbpp/text/book.html> Ian Foster’s 1995 handbook for “Designing and Building Parallel Programs”

Drop the Barrier to Entry

- Handbooks, examples of good practice, “app notes” from vendors and developers
 - Make use of the open source bazaar
 - e.n.s.g. <http://www-unix.mcs.anl.gov/dbpp/text/book.html> Ian Foster’s 1995 handbook for “Designing and Building Parallel Programs”
- Vendors commit to downward scalability

Drop the Barrier to Entry

- Handbooks, examples of good practice, “app notes” from vendors and developers
 - Make use of the open source bazaar
 - e.n.s.g. <http://www-unix.mcs.anl.gov/dbpp/text/book.html> Ian Foster’s 1995 handbook for “Designing and Building Parallel Programs”
- Vendors commit to downward scalability
 - Explore the world between a dozen processors and a bazillion processors.

Use What We've Got



Use What We've Got

- Today's parallel geeks should be building infrastructure

Use What We've Got

- Today's parallel geeks should be building infrastructure
 - FFTW, PETSc are good examples

Use What We've Got

- Today's parallel geeks should be building infrastructure
 - FFTW, PETSc are good examples
- Build “back ends” and candy coatings for parallel services

Use What We've Got

- Today's parallel geeks should be building infrastructure
 - FFTW, PETSc are good examples
- Build “back ends” and candy coatings for parallel services
 - *P is an interesting path.

Use What We've Got

- Today's parallel geeks should be building infrastructure
 - FFTW, PETSc are good examples
- Build “back ends” and candy coatings for parallel services
 - *P is an interesting path.
- Build domain specific umbrella libraries and frameworks.

An Umbrella Framework

```
...
C Global MPI Initialization, required in every MPI program
  call MPI_Init(ierr)
C Create an SCDD Context record
  INTEGER :: SCDD_CONTEXT
  CALL SCDD_Init( MPI_COMM_WORLD , SCDD_CONTEXT , ierr )
  CALL SCDD_Info( SCDD_CONTEXT , my_pid, nproc, ierr )
C Declare the problem geometry
C and get the boundaries of our local subvolume
  CALL SCDD_Local_Size( SCDD_CONTEXT, npad, nx_glob, ny_glob, nz_glob
    & nx_tile_sz , ny_tile_sz, nx_tile_coord, ny_tile_coord, ierr )
C Create the local storage
  ...
C Initialize all arrays
  ...
C Iterate
  DO ...
    call F(U,...)
    call SCDD_Halo_Exchange(SCDD_CONTEXT, u2, ierr)
  END DO
C Output and results
  ...
```

Boilerplate

Application Specific Code

Do We Need New Languages?

Do We Need New Languages?

- UPC and CAF look promising

Do We Need New Languages?

- UPC and CAF look promising
- Sponsored “language design tournaments”

Do We Need New Languages?

- UPC and CAF look promising
- Sponsored “language design tournaments”
- New languages must recognize hardware/firmware reality

Do We Need New Languages?

- UPC and CAF look promising
- Sponsored “language design tournaments”
- New languages must recognize hardware/firmware reality
 - Latency and inter-processor bandwidth must be respected

Do We Need New Languages?

- UPC and CAF look promising
- Sponsored “language design tournaments”
- New languages must recognize hardware/firmware reality
 - Latency and inter-processor bandwidth must be respected
 - Probably has to look like C or FORTRAN.

Takeaway



Takeaway

- Academic:
 - Add high processor count computing toolbox to the current curriculum where it belongs.
 - Build playgrounds.

Takeaway

- Academic:
 - Add high processor count computing toolbox to the current curriculum where it belongs.
 - Build playgrounds.
- Vendors:
 - Build downward scalable high processor count systems.

Takeaway

- Academic:
 - Add high processor count computing toolbox to the current curriculum where it belongs.
 - Build playgrounds.
- Vendors:
 - Build downward scalable high processor count systems.
- Developers/Leaders:
 - Build the knowledge pool with frameworks, libraries, app notes, blogs, billboards, skywriting...

Achieve more. Consume Less.



SiCortex