High-Performance Scientific Computing
Lecture 1: Intro

MATH-GA 2011 / CSCI-GA 2945 · September 5, 2012

Today

About this class

HPC: A look around

A taste of what's to come

Extra stuff

Outline

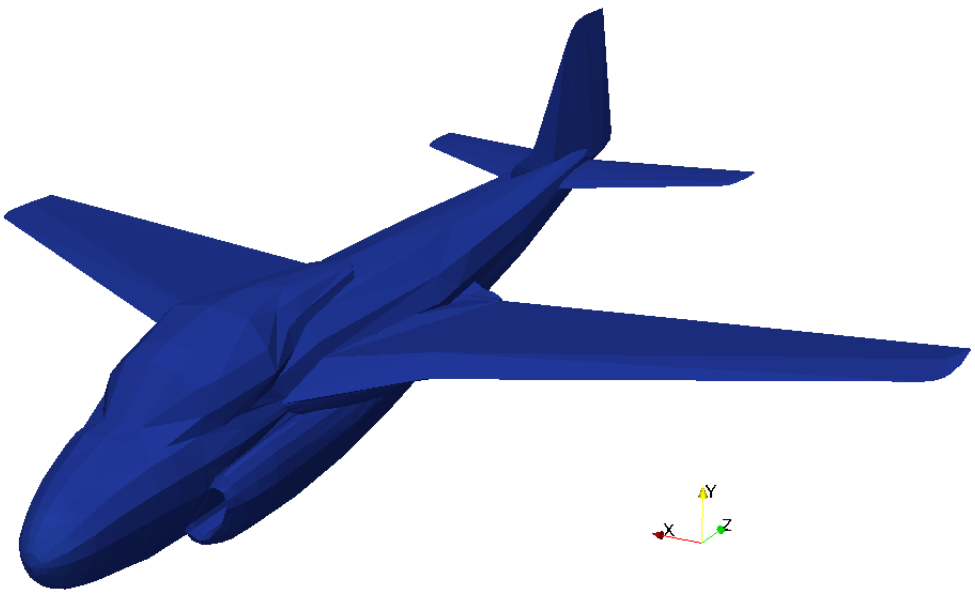
Course Goal

Slow code goes in.

Speedy code goes out.

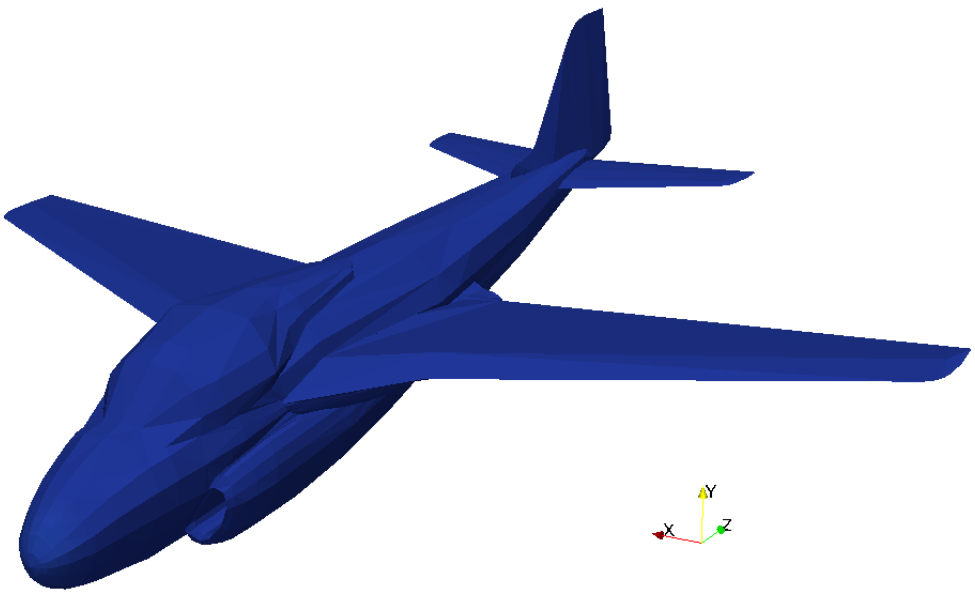
Define slow?

Define slow?



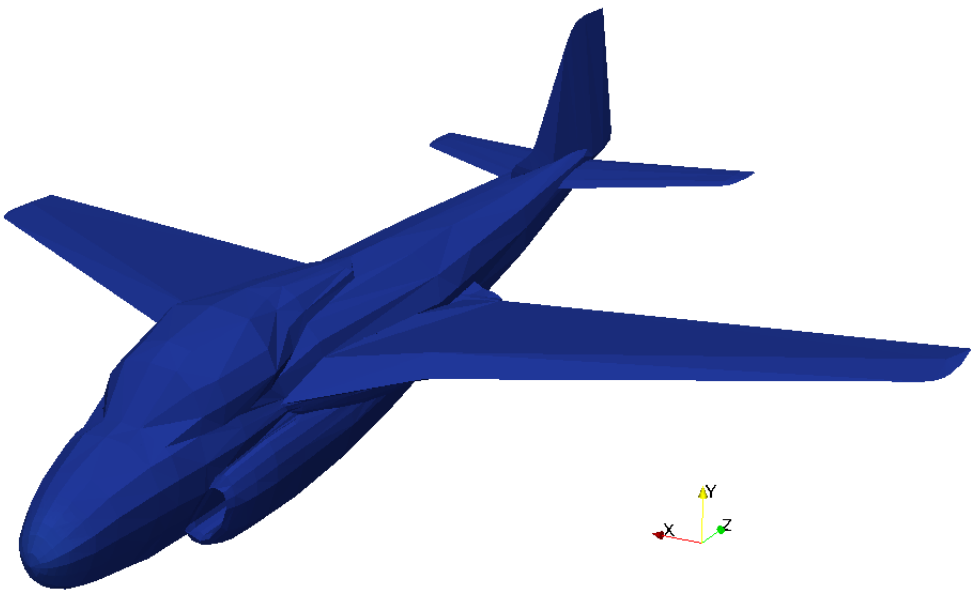
- Took about 30 days on a single PC.

Define slow?



- Took about 30 days on a single PC.
- Took about a day on a GPU.

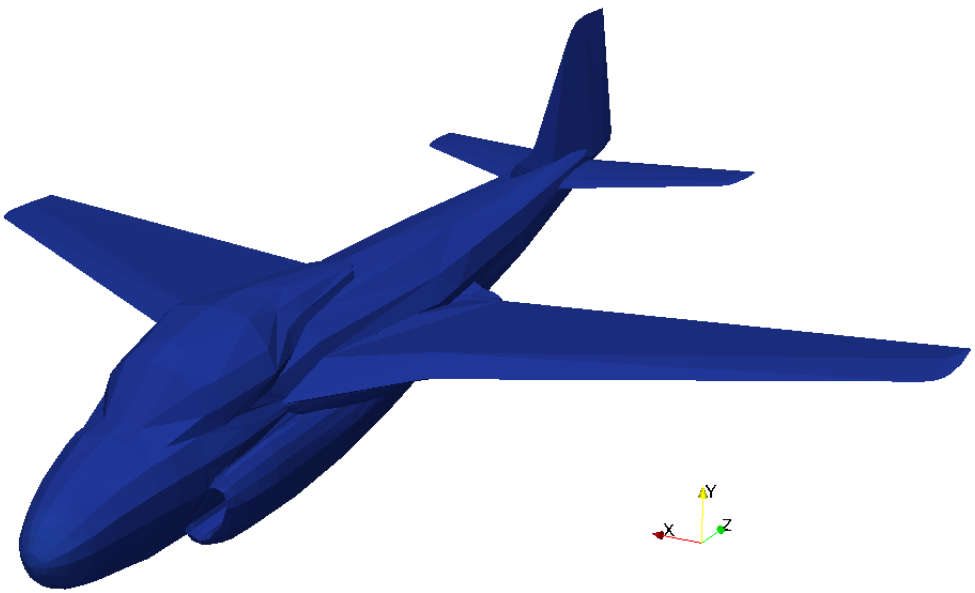
Define slow?



- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Define slow?

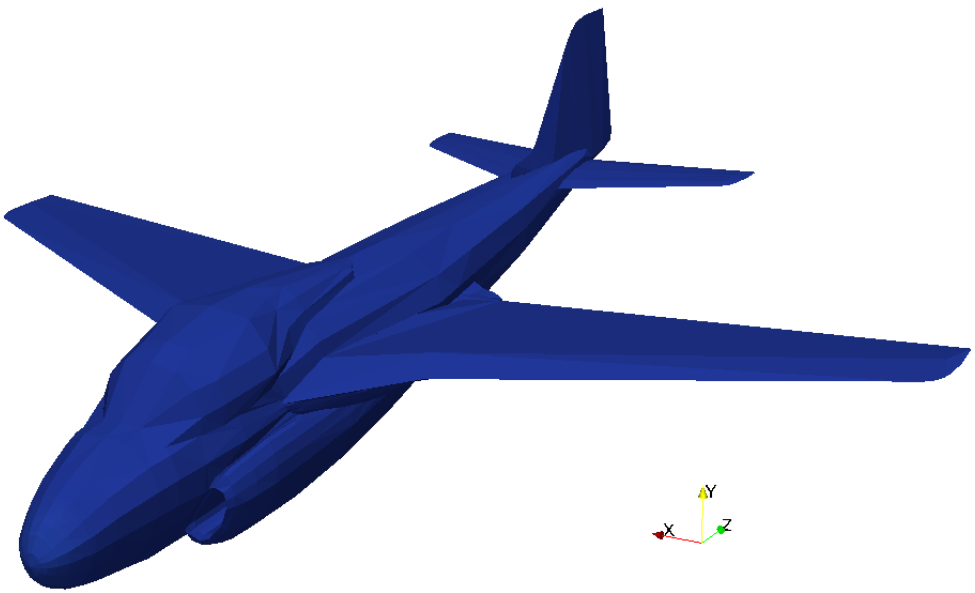


- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

Define slow?



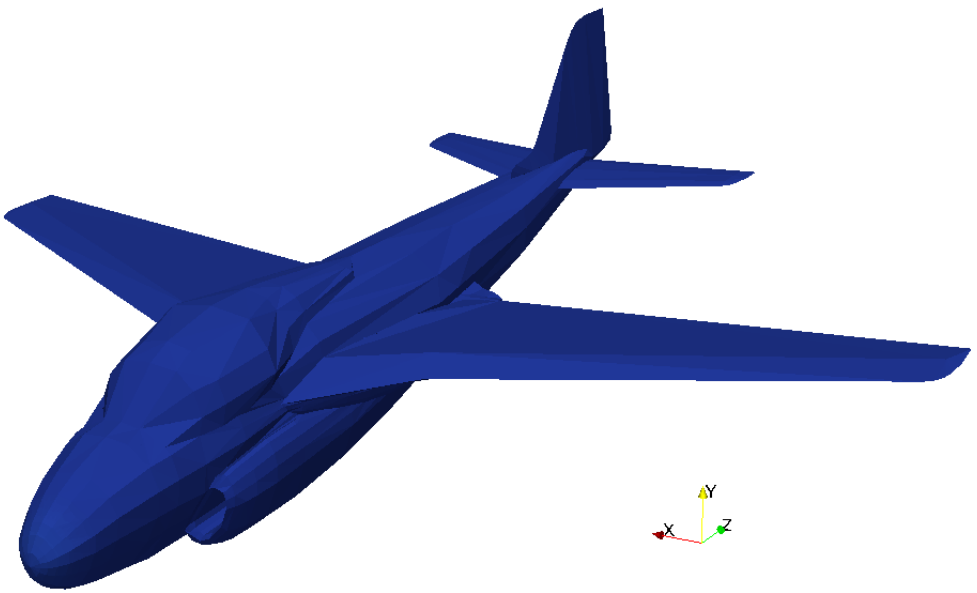
- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?

Define slow?



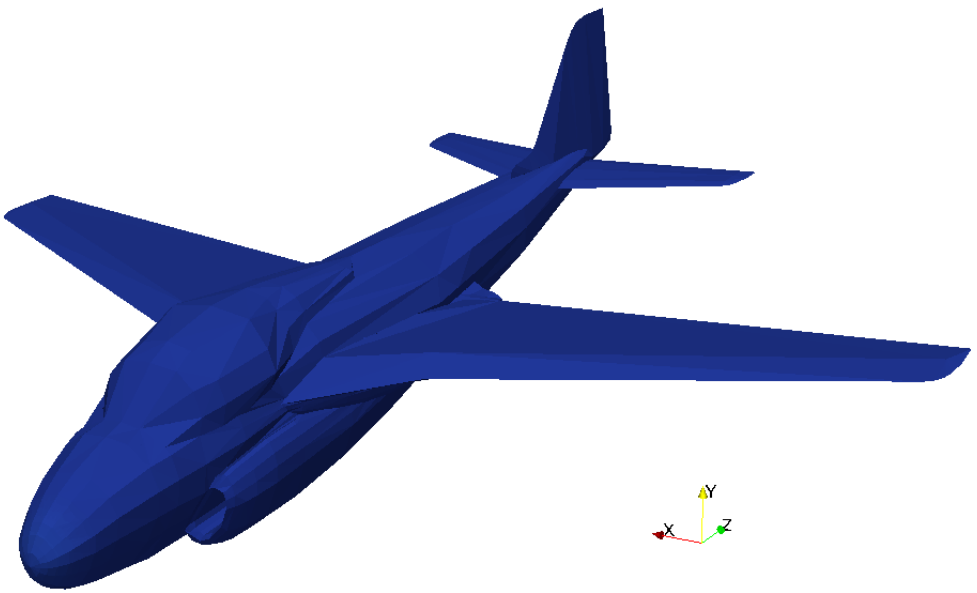
- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?
- Anything else?

Define slow?



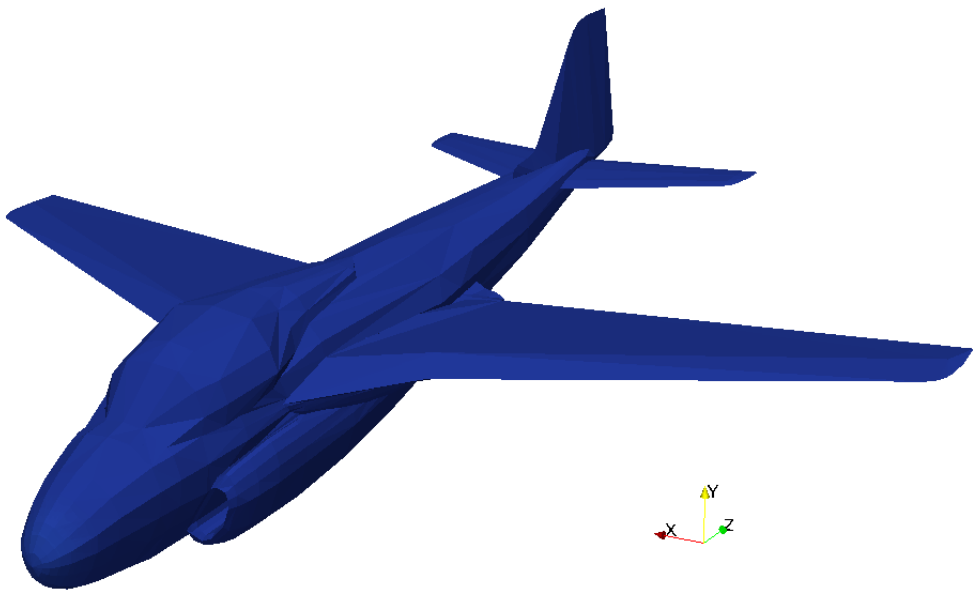
- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?
- Anything else?
- $16\times$ the cost!

Define slow?



- Took about 30 days on a single PC.
- Took about a day on a GPU.

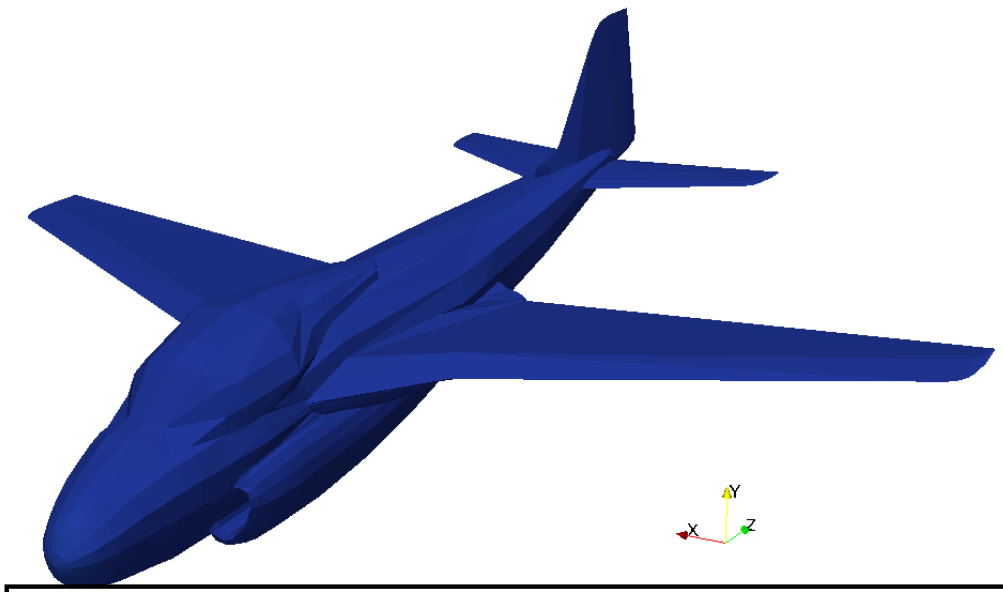
That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?
- Anything else?
- $16\times$ the cost!

Realistic (high-fidelity) problems are big.

Define slow?



- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

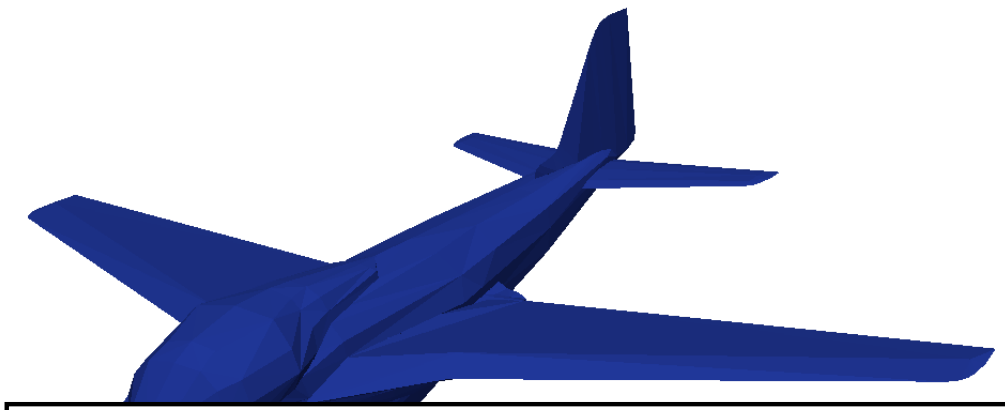
Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?
- Anything else?
- $16\times$ the cost!

Realistic (high-fidelity) problems are big.

$\rightarrow$ You'll need a bigger hammer.

Define slow?



Realistic (high-fidelity) problems are big.

$\rightarrow$ You'll need a bigger hammer.

You'll need to know how to use the bigger hammer.

- Took about 30 days on a single PC.
- Took about a day on a GPU.

That's still pretty crude-looking.

Suppose I'd like to double the resolution. (i.e. cut the mesh width h in half.)

- Had K elements. Now?
- Anything else?
- $16\times$ the cost!

Course Outline

Part 1: Do (~ 4)

- Write, run programs (C)
- Use tools (make, git, gdb)
- OpenMP, MPI, OpenCL
- Correctness in each

Part 2: Understand ($\sim$3)

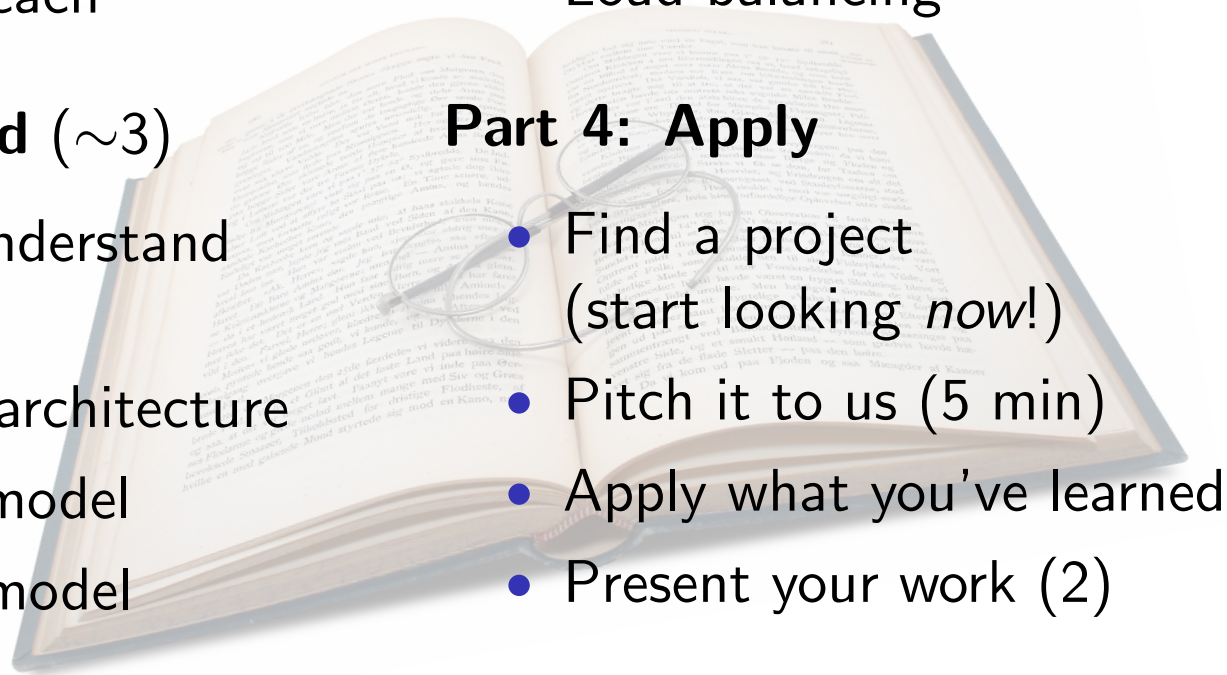
- Measure and understand performance
- Basic machine architecture
- CPU machine model
- GPU machine model

Part 3: Refine ($\sim$3)

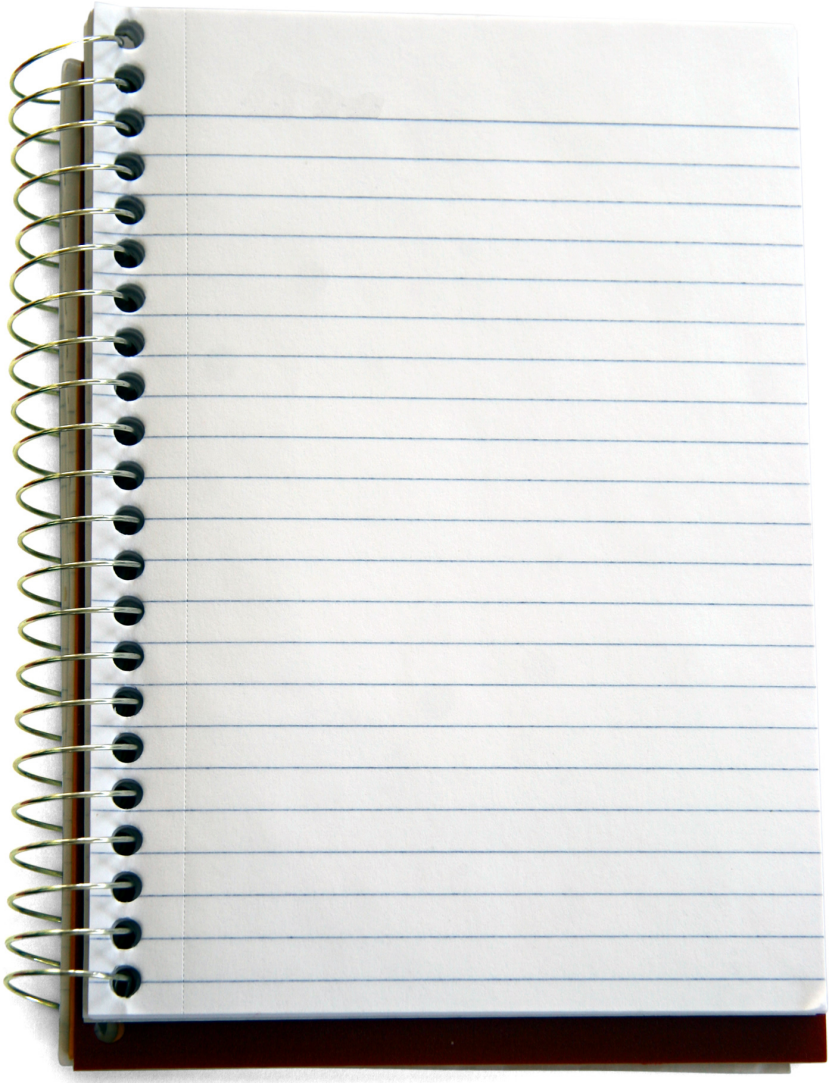
- Advanced tools & languages
- Work partitioning
- Common patterns
- Load balancing

Part 4: Apply

- Find a project (start looking *now*!)
- Pitch it to us (5 min)
- Apply what you've learned
- Present your work (2)



Sign-up sheet



Survey

- Home department

Survey

- Home department
- Degree

Survey

- Home department
- Degree
- Longest program ever written?

Survey

- Home department
- Degree
- Longest program ever written?
  - in C?

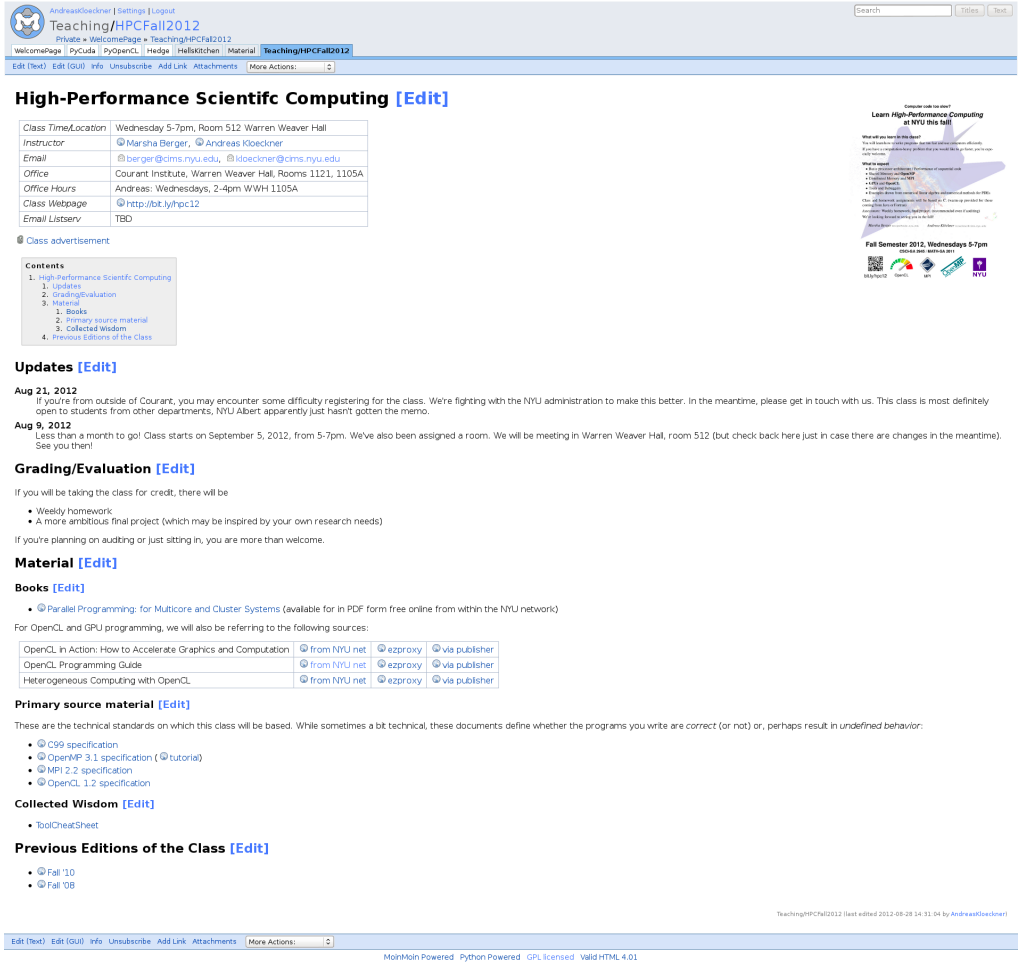
Survey

- Home department
- Degree
- Longest program ever written?
  - in C?
- Parallel?

Survey

- Home department
- Degree
- Longest program ever written?
  - in C?
- Parallel?
- Already have a project?

Class web page



AndreasKloeckner | Settings | Logout

Search | Titles | Text

Teaching/HPCFall2012

Private » WelcomePage » Teaching/HPCFall2012

WelcomePage | PyCuda | PyOpenCL | Hedge | HellsKitchen | Material | Teaching/HPCFall2012

Edit (Text) Edit (GUI) Info Unsubscribe Add Link Attachments More Actions:

High-Performance Scientifc Computing [Edit]

Class Time/Location	Wednesday 5-7pm, Room 512 Warren Weaver Hall
Instructor	Marsha Berger, Andreas Kloeckner
Email	berger@cims.nyu.edu, kloeckner@cims.nyu.edu
Office	Courant Institute, Warren Weaver Hall, Rooms 1121, 1105A
Office Hours	Andreas: Wednesdays, 2-4pm WWH 1105A
Class Webpage	http://bit.ly/hpc12
Email Listserv	TBD

Class advertisement

Contents

1. High-Performance Scientifc Computing
   1. Updates
   2. Grading/Evaluation
   3. Material
      1. Books
      2. Primary source material
      3. Collected Wisdom
   4. Previous Editions of the Class

Updates [Edit]

Aug 21, 2012
If you're from outside of Courant, you may encounter some difficulty registering for the class. We're fighting with the NYU administration to make this better. In the meantime, please get in touch with us. This class is most definitely open to students from other departments, NYU Albert apparently just hasn't gotten the memo.

Aug 9, 2012
Less than a month to go! Class starts on September 5, 2012, from 5-7pm. We've also been assigned a room. We will be meeting in Warren Weaver Hall, room 512 (but check back here just in case there are changes in the meantime). See you then!

Grading/Evaluation [Edit]

If you will be taking the class for credit, there will be

- Weekly homework
- A more ambitious final project (which may be inspired by your own research needs)

If you're planning on auditing or just sitting in, you are more than welcome.

Material [Edit]

Books [Edit]

- Parallel Programming: for Multicore and Cluster Systems (available for in PDF form free online from within the NYU network)

For OpenCL and GPU programming, we will also be referring to the following sources:

OpenCL in Action: How to Accelerate Graphics and Computation	from NYU net	ezproxy	via publisher
OpenCL Programming Guide	from NYU net	ezproxy	via publisher
Heterogeneous Computing with OpenCL	from NYU net	ezproxy	via publisher

Primary source material [Edit]

These are the technical standards on which this class will be based. While sometimes a bit technical, these documents define whether the programs you write are *correct* (or not) or, perhaps result in *undefined behavior*:

- C99 specification
- OpenMP 3.1 specification (tutorial)
- MPI 2.2 specification
- OpenCL 1.2 specification

Collected Wisdom [Edit]

- ToolCheatSheet

Previous Editions of the Class [Edit]

- Fall '10
- Fall '08

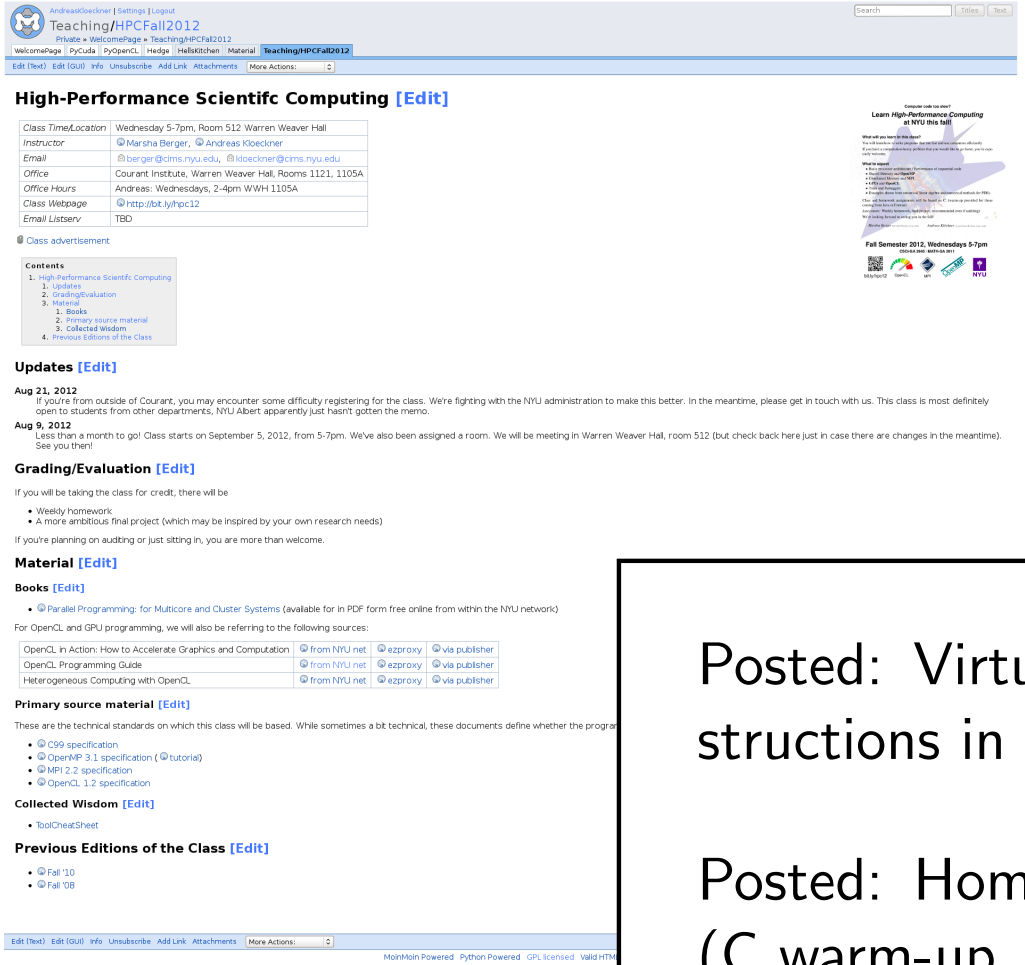
Teaching/HPCFall2012 (last edited 2012-08-28 14:31:04 by AndreasKloeckner)

Edit (Text) Edit (GUI) Info Unsubscribe Add Link Attachments More Actions:

MoinMoin Powered Python Powered GPL licensed Valid HTML 4.01

bit.ly/hpc12

Class web page



AndreasKloeckner | Settings | Logout

Teaching/HPCFall2012

Private » WelcomePage » Teaching/HPCFall2012

WelcomePage | PyCuda | PyOpenCL | Hedge | HellsKitchen | Material | Teaching/HPCFall2012

Edit (Text) Edit (GUI) Info Unsubscribe Add Link Attachments More Actions:

High-Performance Scientifc Computing [Edit]

Class Time/Location	Wednesday 5-7pm, Room 512 Warren Weaver Hall
Instructor	Marsha Berger, Andreas Kloeckner
Email	berger@cims.nyu.edu, kloeckner@cims.nyu.edu
Office	Courant Institute, Warren Weaver Hall, Rooms 1121, 1105A
Office Hours	Andreas: Wednesdays, 2-4pm WWH 1105A
Class Webpage	http://bit.ly/hpc12
Email Listserv	TBD

Class advertisement

Contents

1. High-Performance Scientifc Computing
   1. Updates
   2. Grading/Evaluation
   3. Material
      1. Books
      2. Primary source material
      3. Collected Wisdom
   4. Previous Editions of the Class

Updates [Edit]

Aug 21, 2012
If you're from outside of Courant, you may encounter some difficulty registering for the class. We're fighting with the NYU administration to make this better. In the meantime, please get in touch with us. This class is most definitely open to students from other departments, NYU Albert apparently just hasn't gotten the memo.

Aug 9, 2012
Less than a month to go! Class starts on September 5, 2012, from 5-7pm. We've also been assigned a room. We will be meeting in Warren Weaver Hall, room 512 (but check back here just in case there are changes in the meantime). See you then!

Grading/Evaluation [Edit]

If you will be taking the class for credit, there will be

- Weekly homework
- A more ambitious final project (which may be inspired by your own research needs)

If you're planning on auditing or just sitting in, you are more than welcome.

Material [Edit]

Books [Edit]

- Parallel Programming: for Multicore and Cluster Systems (available for in PDF form free online from within the NYU network)

For OpenCL and GPU programming, we will also be referring to the following sources:

OpenCL in Action: How to Accelerate Graphics and Computation	from NYU net	ezproxy	via publisher
OpenCL Programming Guide	from NYU net	ezproxy	via publisher
Heterogeneous Computing with OpenCL	from NYU net	ezproxy	via publisher

Primary source material [Edit]

These are the technical standards on which this class will be based. While sometimes a bit technical, these documents define whether the progra

- C99 specification
- OpenMP 3.1 specification (tutorial)
- MPI 2.2 specification
- OpenCL 1.2 specification

Collected Wisdom [Edit]

- ToolCheatSheet

Previous Editions of the Class [Edit]

- Fall '10
- Fall '08

Edit (Text) Edit (GUI) Info Unsubscribe Add Link Attachments More Actions:

MoinMoin Powered Python Powered GPL licensed Valid HTM

bit.ly/hpc12

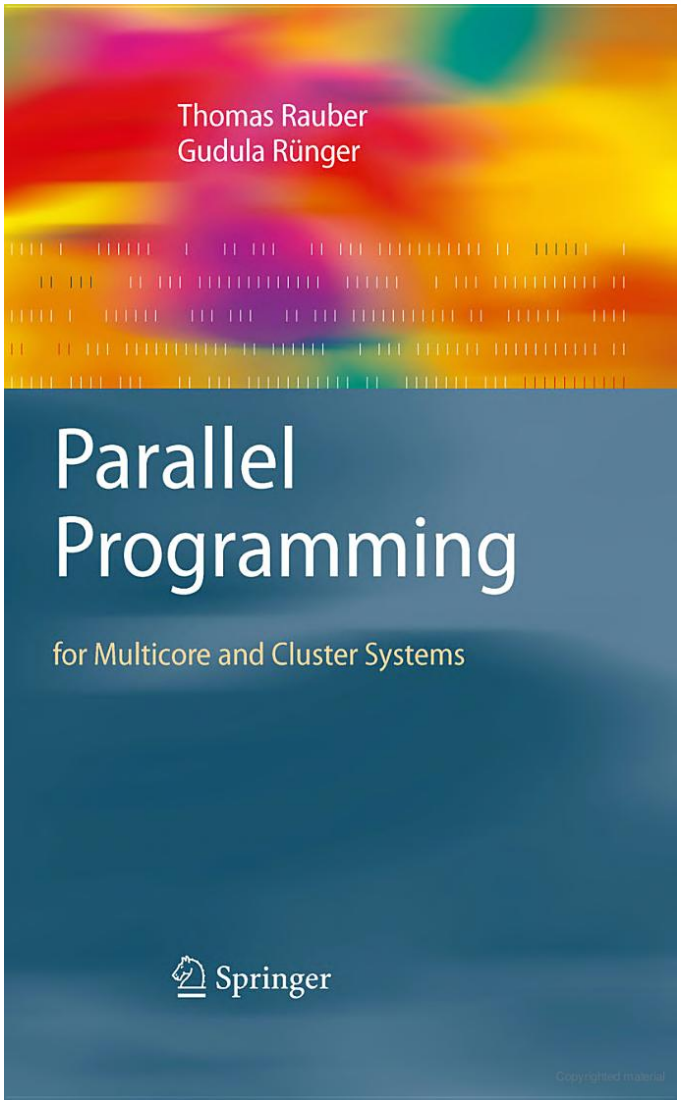
Posted: Virtual machine image (instructions in HW1)

Posted: Homework set 1
(C warm-up, git, mechanics)
Due next week.

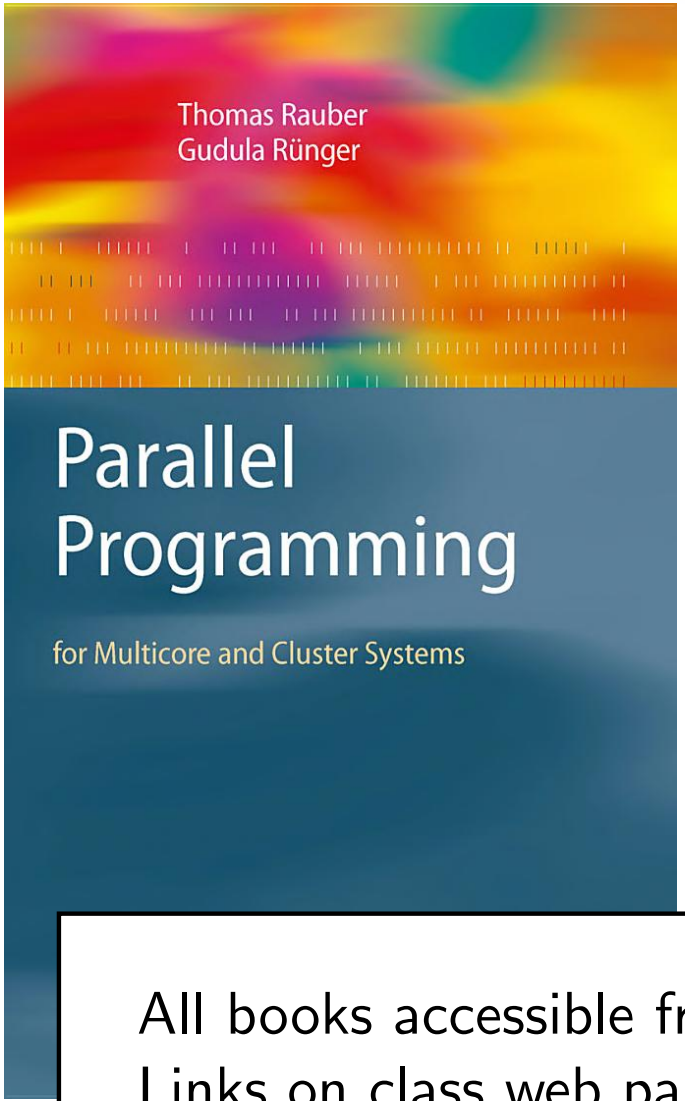
Listserv

hpc12@tiker.net

Book 1

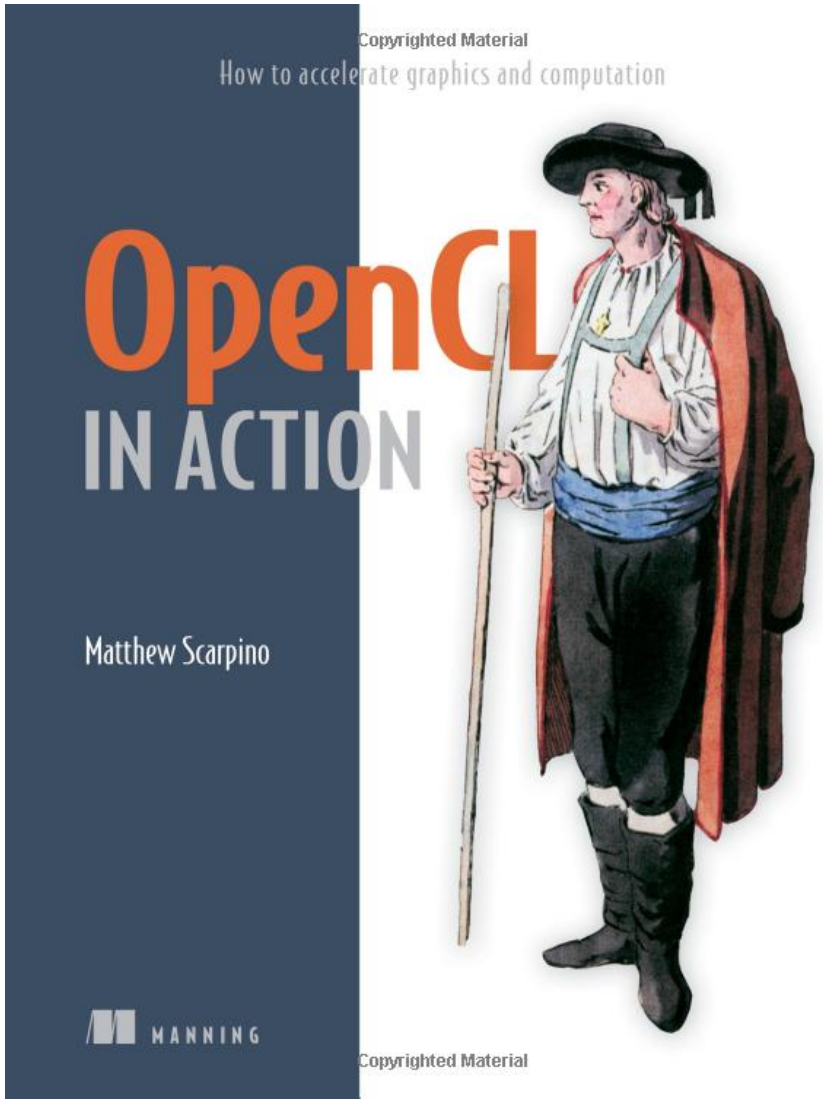

Thomas Rauber
Gudula Rünger
Parallel Programming
for Multicore and Cluster Systems
Springer
Copyrighted material

Book 1

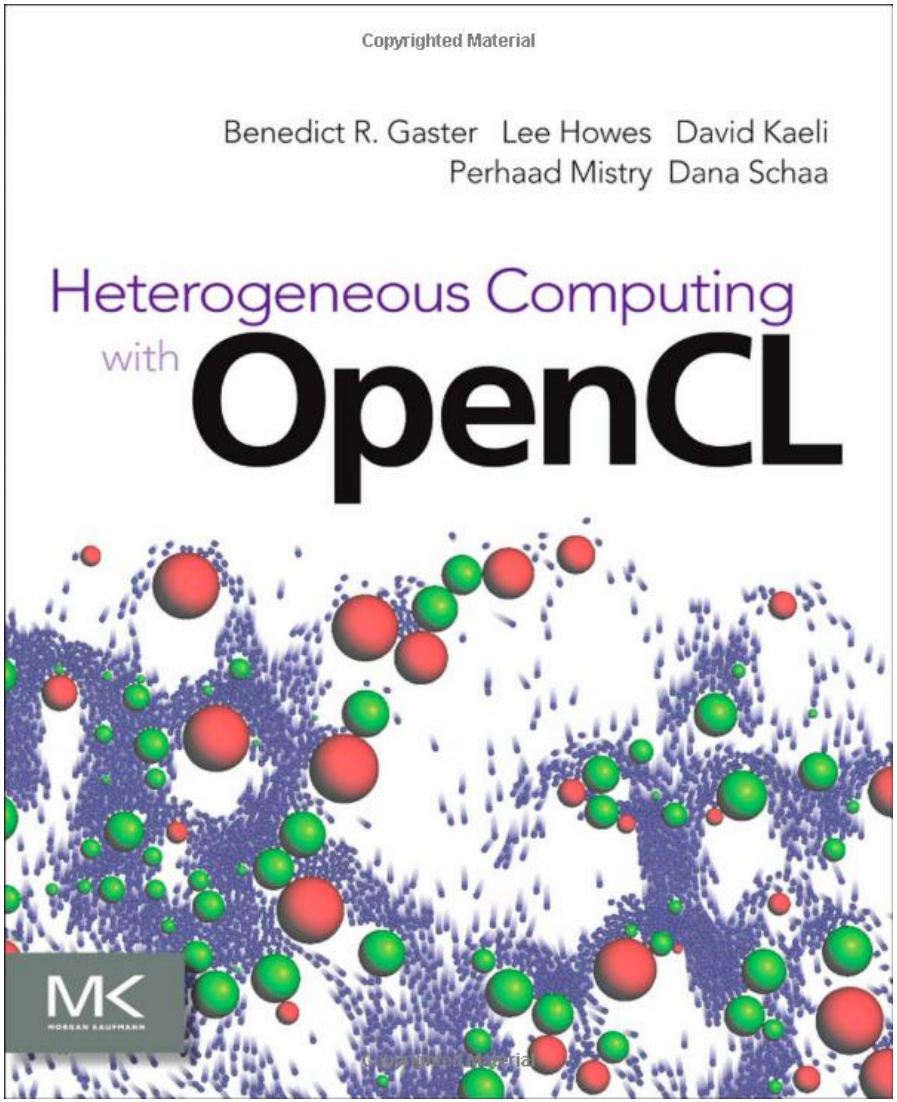


All books accessible from NYU network.
Links on class web page.

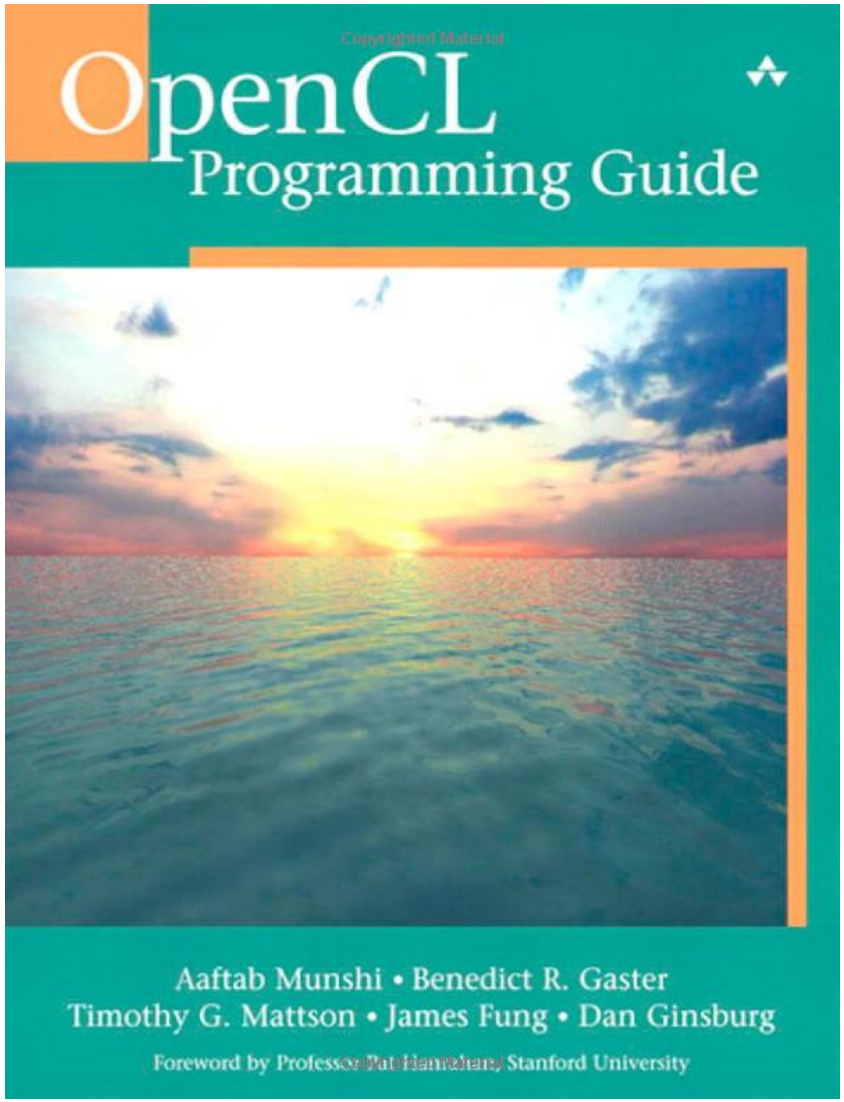
Books 2–4


Copyrighted Material
How to accelerate graphics and computation
OpenCL
IN ACTION
Matthew Scarpino
MANNING
Copyrighted Material

Books 2–4


Copyrighted Material
Benedict R. Gaster Lee Howes David Kaeli
Perhaad Mistry Dana Schaa
Heterogeneous Computing
with
OpenCL
MK
MORGAN KAUFMANN

Books 2–4


Copyrighted Material
OpenCL
Programming Guide
Aaftab Munshi • Benedict R. Gaster
Timothy G. Mattson • James Fung • Dan Ginsburg
Foreword by Professor Pat Hanrahan, Stanford University

Grading

- 60% Weekly homework
- 40% Final project

Smile! You're on camera

Lecture video will be posted soon after each class.

Outline

Key Realization

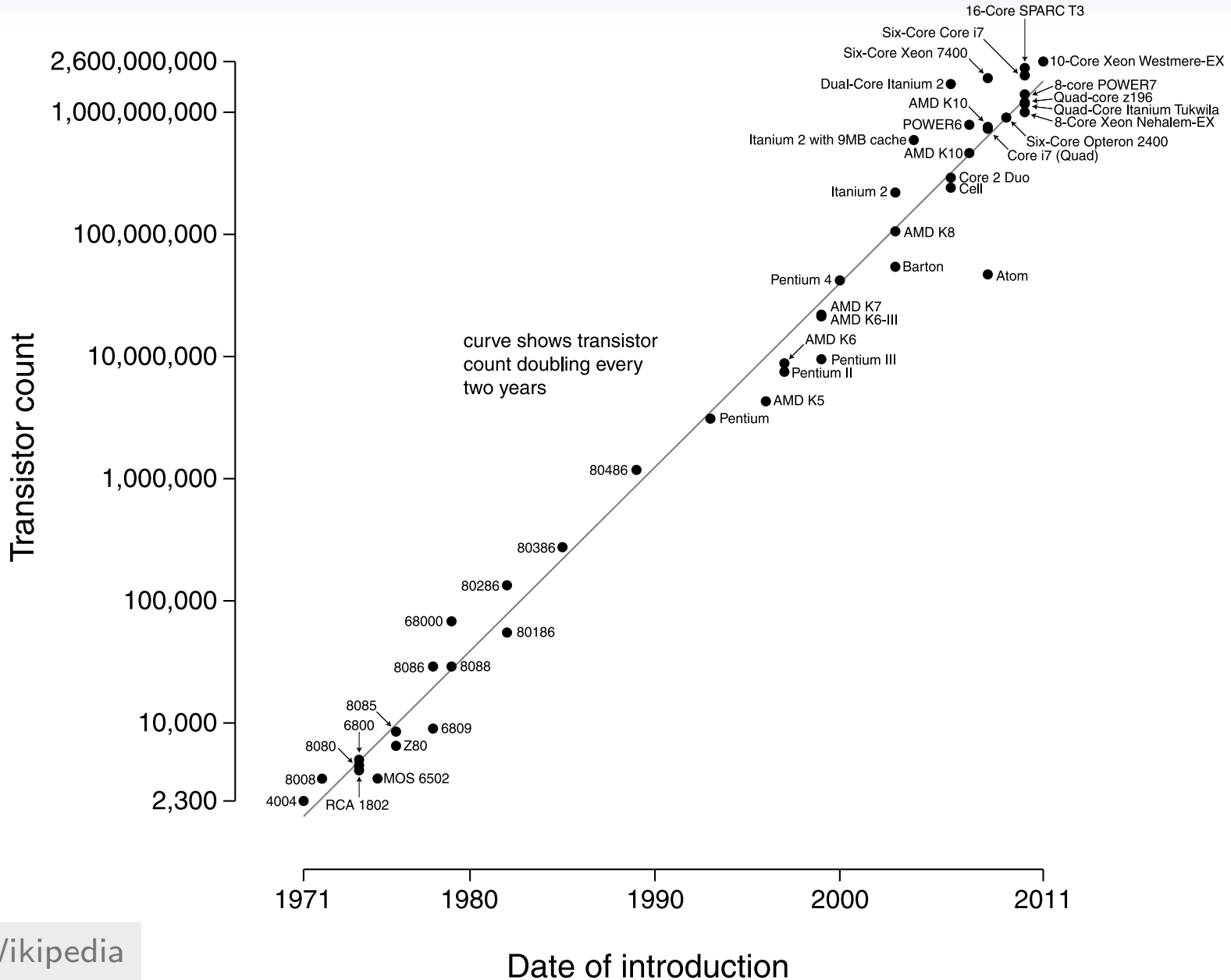
My program is taking too long.

Key Realization

My program is taking too long.

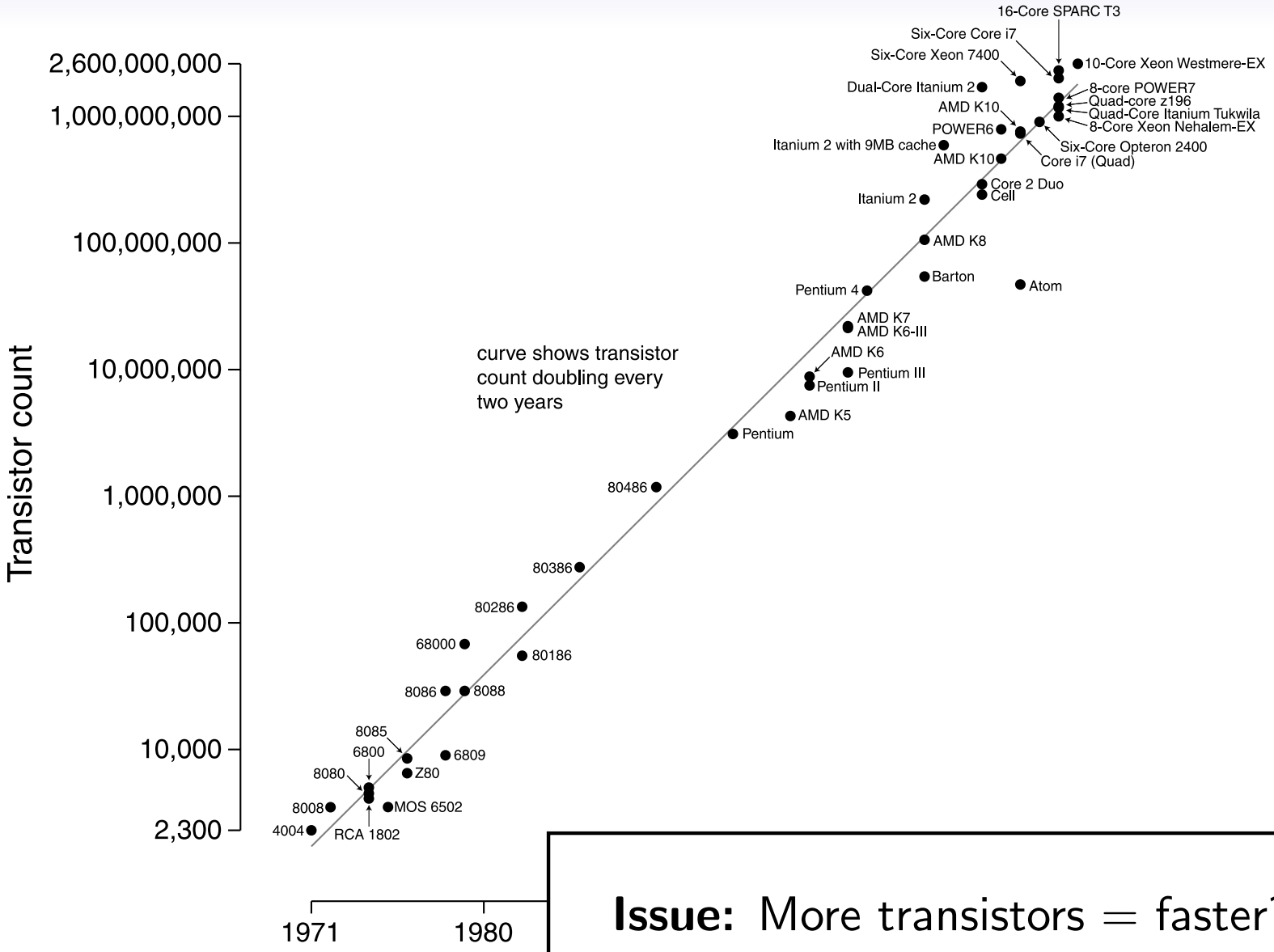
Maybe it'll get faster if I wait long enough?

Moore's law



Wikipedia

Moore's law



Issue: More transistors = faster?

Wikipedia

So what does it mean?

So what does it mean?

So what *does* it mean?

$$\frac{\text{Work}}{s} = \text{Clock Frequency} \times \text{Work/Clock}$$

So what *does* it mean?

$$\frac{\text{Work}}{s} = \textbf{Clock Frequency} \times \text{Work/Clock}$$

Dennard scaling of MOSFETs

Parameter	**Factor**
Dimension	$1/\kappa$
Voltage	$1/\kappa$
Current	$1/\kappa$
Capacitance	$1/\kappa$
Delay Time	$1/\kappa$
Power dissipation/circuit	$1/\kappa^2$
Power density	1

[Dennard et al. '74, via Bohr '07]

Dennard scaling of MOSFETs

Parameter	**Factor**
Dimension	$1/\kappa$
Voltage	$1/\kappa$
Current	$1/\kappa$
Capacitance	$1/\kappa$
Delay Time	$1/\kappa$
Power dissipation/circuit	$1/\kappa^2$
Power density	1

[Dennard et al. '74, via Bohr '07]

Frequency = Delay time^{-1}

Dennard scaling of MOSFETs

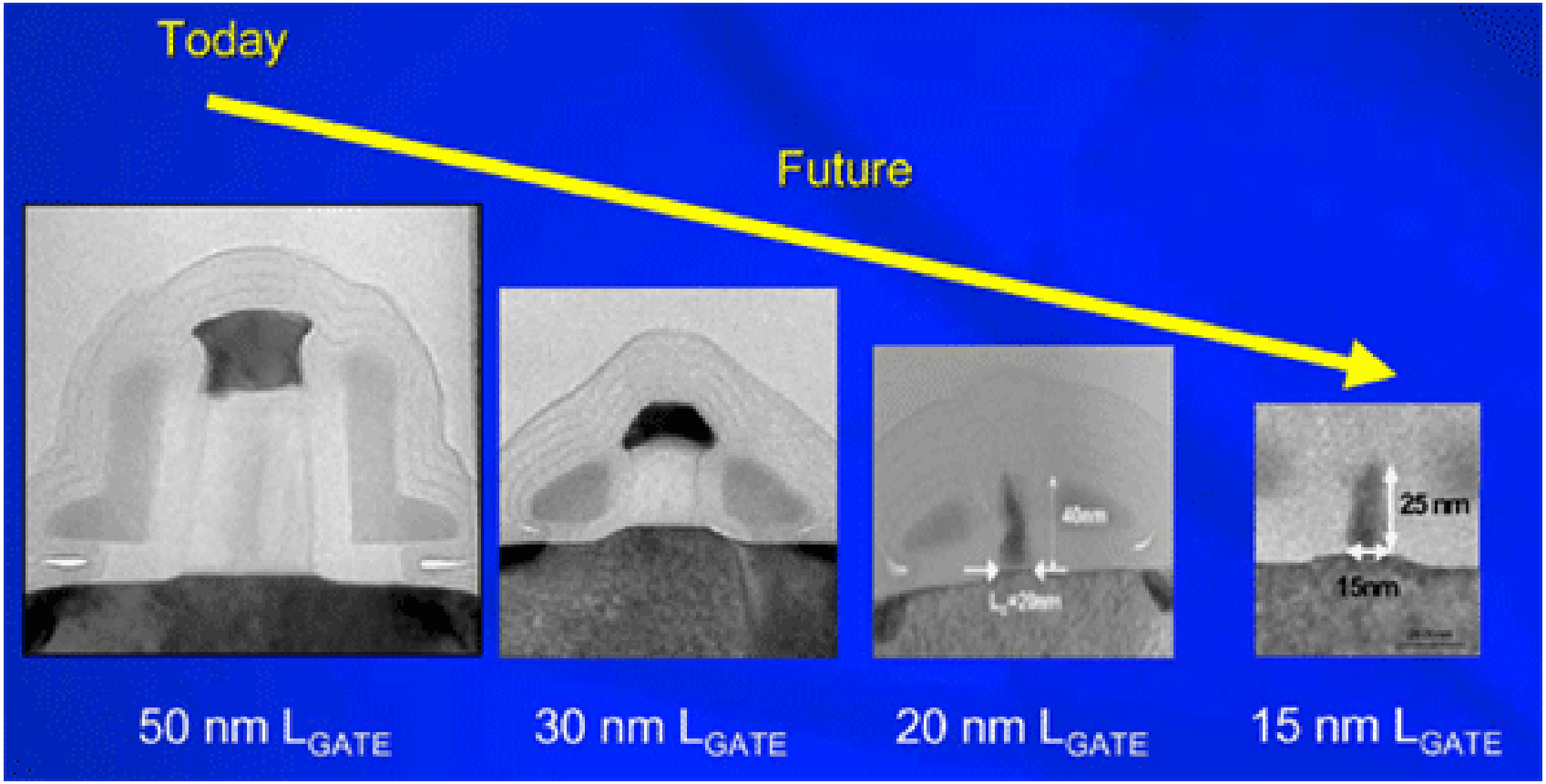
Parameter	**Factor**
Dimension	$1/\kappa$
Voltage	$1/\kappa$
Current	$1/\kappa$
Capacitance	$1/\kappa$
Delay Time	$1/\kappa$
Power dissipation/circuit	$1/\kappa^2$
Power density	1

[Dennard et

Frequency $=$ Delay time^{-1}

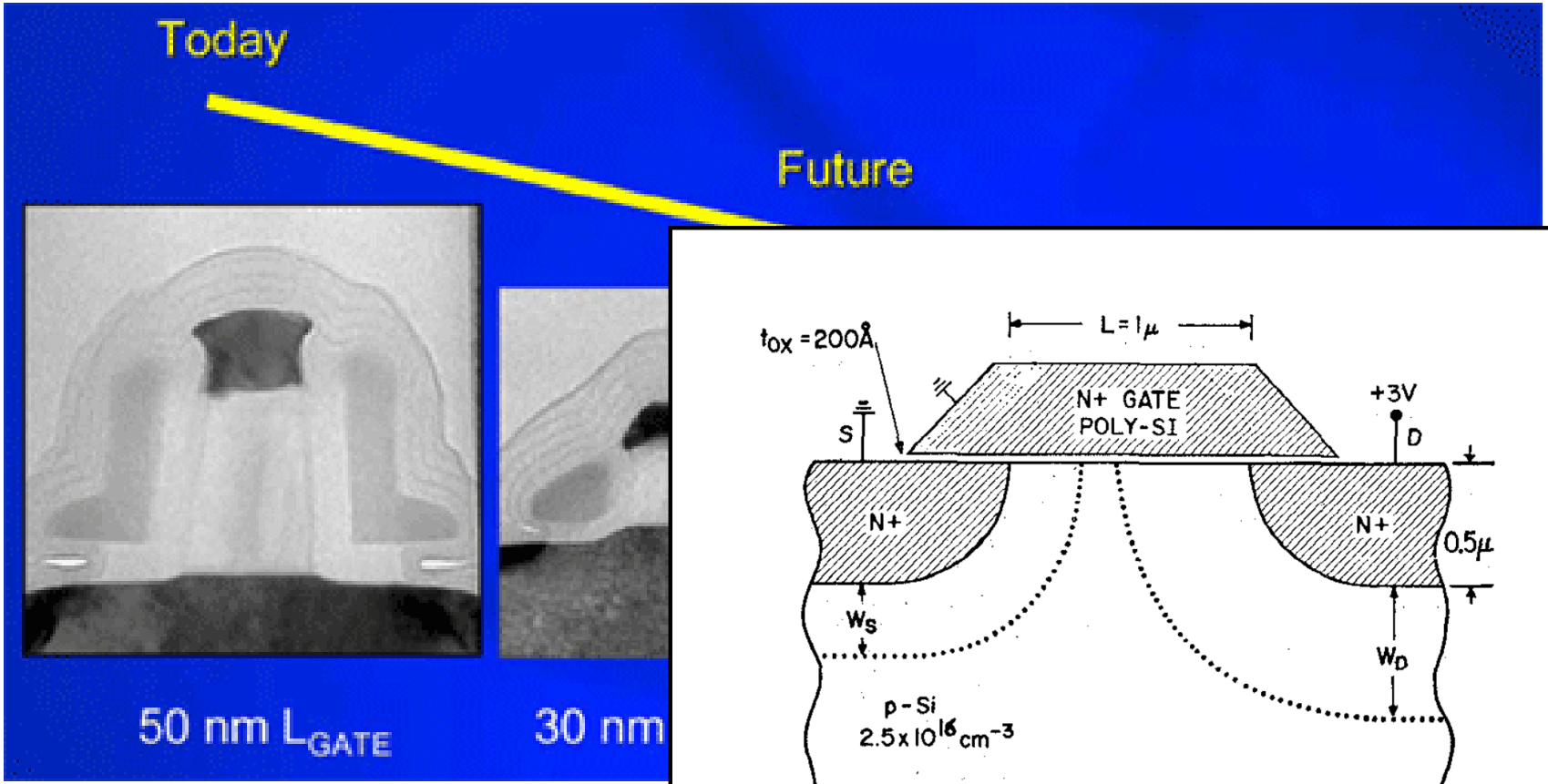
'New' problem at small scale: Sub-threshold leakage (due to low voltage, small structure)

MOSFETs



Intel Corporation

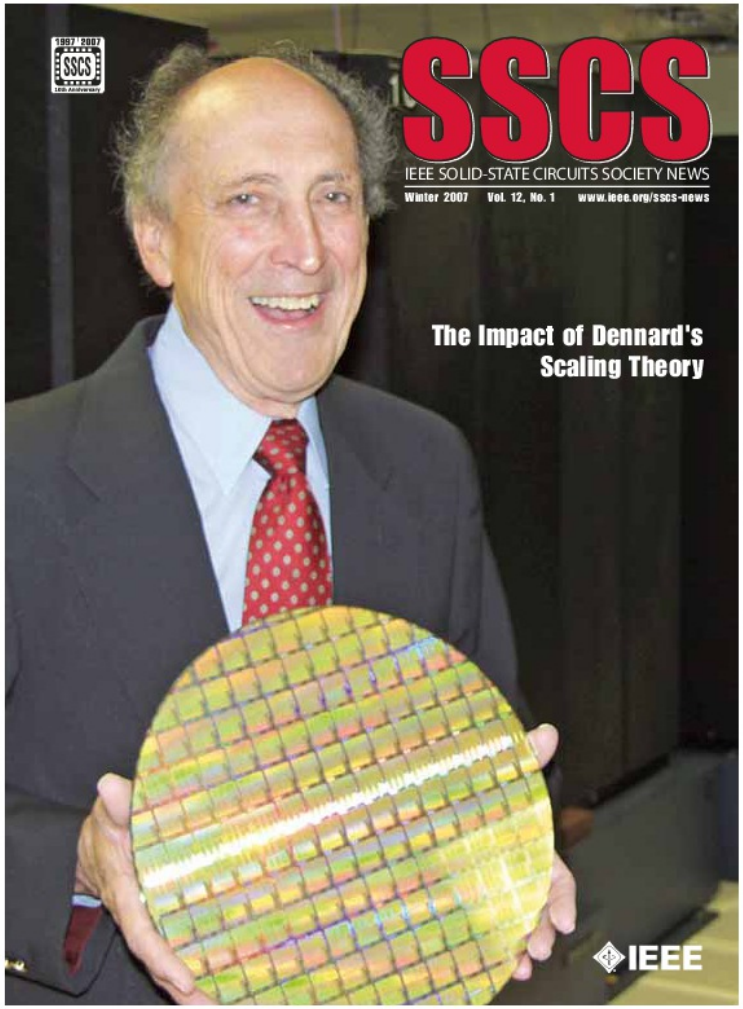
MOSFETs



[Dennard et al. '74]

Intel Corporation

Robert Dennard



So what *does* it mean?

$$\frac{\text{Work}}{s} = \text{Clock Frequency} \times \text{Work/Clock}$$

So what *does* it mean?

$$\frac{\text{Work}}{s} = \cancel{\text{Clock Frequency}} \times \text{Work/Clock}$$

So what *does* it mean?

$$\frac{\text{Work}}{s} = \cancel{\text{Clock Frequency}} \times \textbf{Work/Clock}$$

Instructions per clock: Intel

CPU	IPC	Year
Pentium 1	1.1	1993
Pentium MMX	1.2	1996
Pentium 3	1.9	1999
Pentium 4 (Willamette)	1.5	2003
Pentium 4 (Northwood)	1.6	2003
Pentium 4 (Prescott)	1.8	2003
Pentium 4 (Gallatin)	1.9	2003
Pentium D	2	2005
Pentium M	2.5	2003
Core 2	3	2006

Charlie Brej, `http://brej.org/blog/?p=15`

Instructions per clock: AMD

CPU	IPC	Year
K6 II	1.1	1998
K6 III	1.3	1999
Athlon B	1.9	1999
Athlon XP	2	2001
Athlon 64	2.3	2003
Athlon 64 X2	2.5	2005

Charlie Brej, `http://brej.org/blog/?p=15`

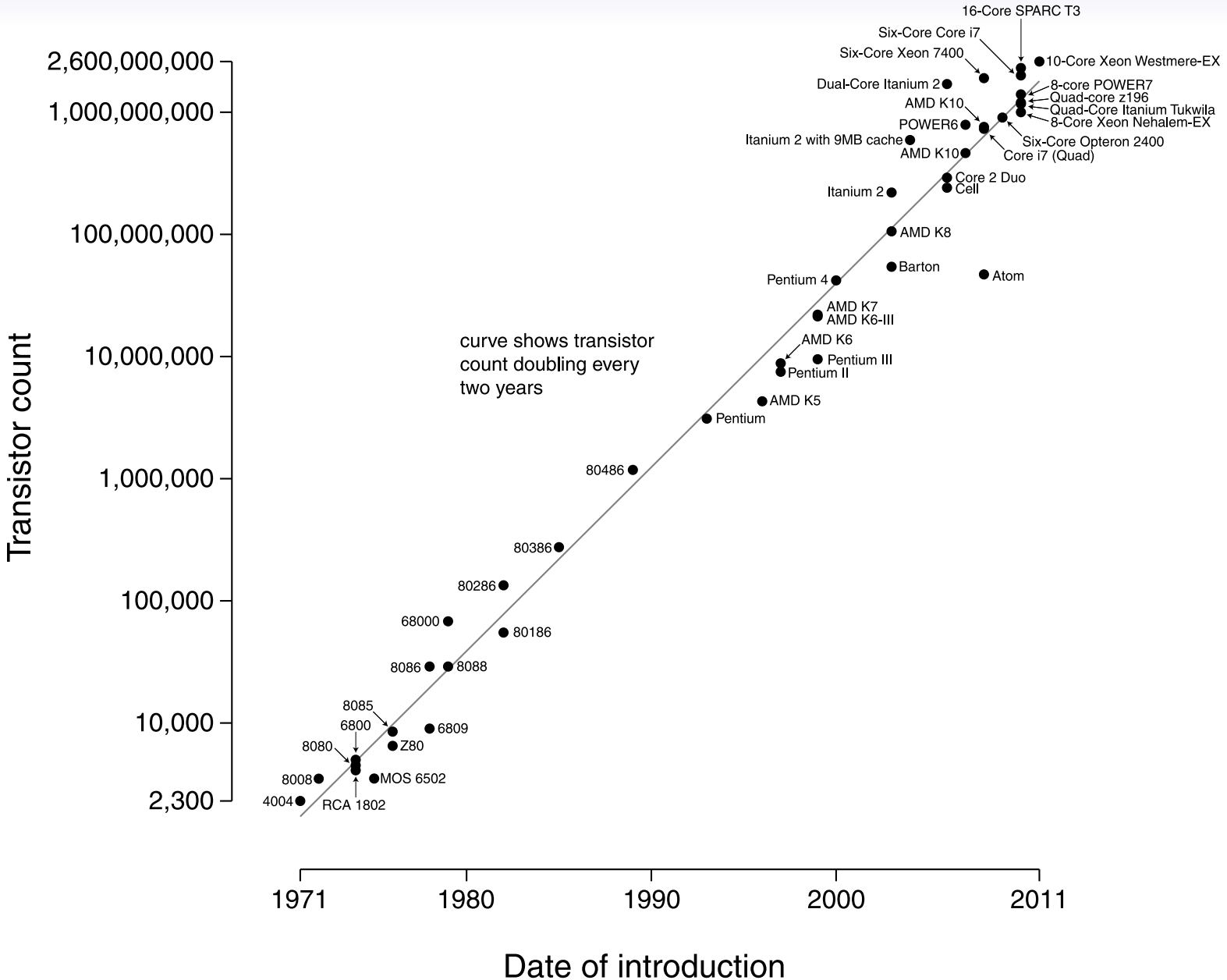
Instructions per clock: AMD

CPU	IPC	Year
K6 II	1.1	1998
K6 III	1.3	1999
Athlon B	1.9	1999
Athlon XP	2	2001
Athlon 64	2.3	2003
Athlon 64 X2	2.5	2005

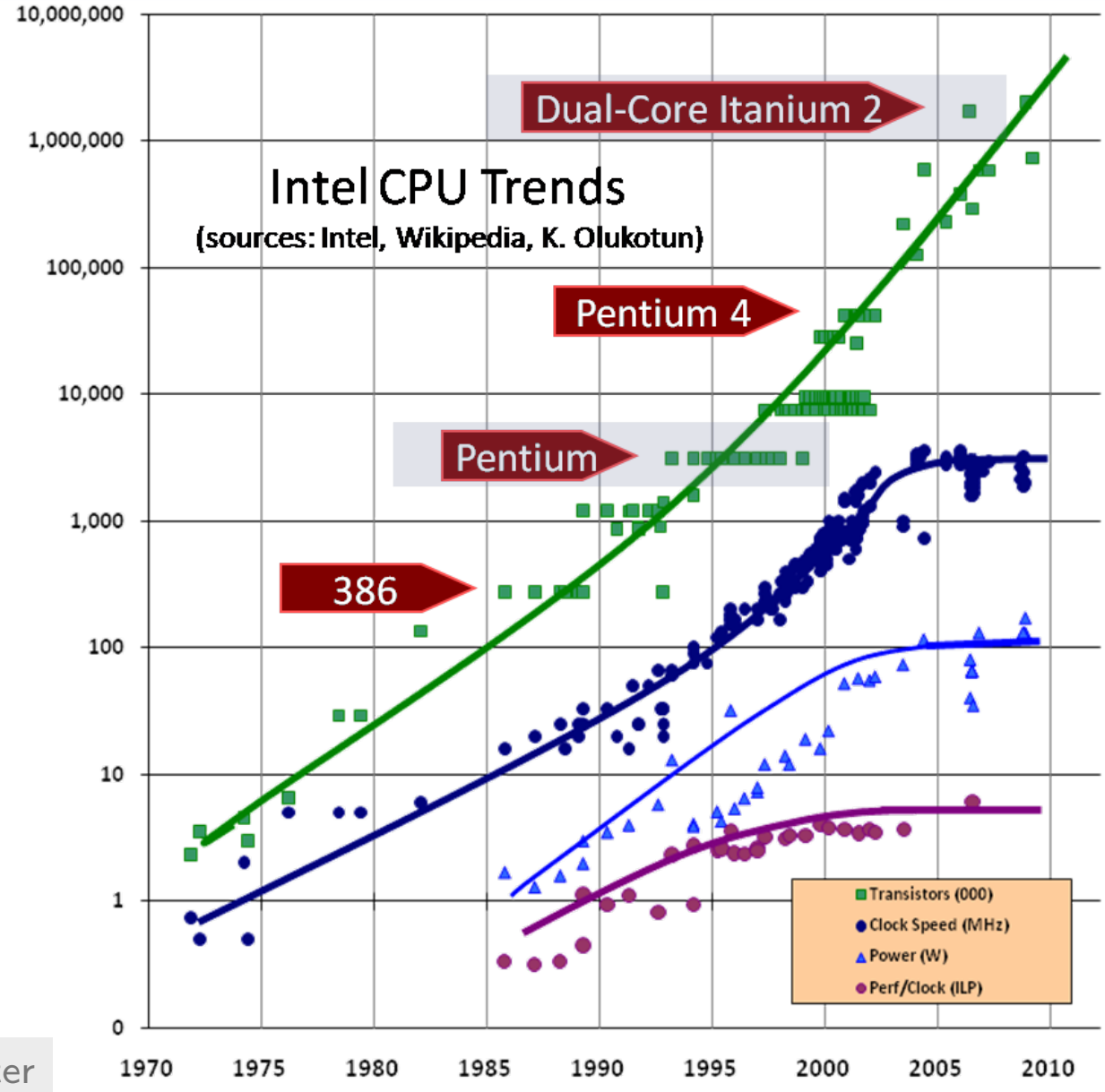
Charlie Brej, `http://brej.org/bl`

A failure of the programming model!

Processor Evolution



Processor Evolution



Herb Sutter

Parallel Programming

High-performance computing *is* parallel computing. (...)

Parallel programming is ...

- *inevitable* (if you'd like maximal throughput)
- *hard*

Problem: People don't think 'that way'.

"Automatic parallelization" has largely been a failure.

→ People have to be taught to think that way.

Parallel Programming

High-performance computing *is* parallel computing. (. . .)

Parallel programming is . . .

- *inevitable* (if you'd like maximal throughput)
- *hard*

Problem: People don't think 'that way'.

"Automatic parallelization" has largely been a failure.

→ People have to be taught to think that way. **:)**

Parallel Programming

High-performance computing *is* parallel computing. (...)

Parallel programming is ...

- *inevitable* (if you'd like maximal throughput)
- *hard*

Problem: People don't think 'that way'.

"Automatic parallelization" has largely been a failure.

→ People have to be taught to think that way.

Parallel Programming

High-performance computing *is* parallel computing. (. . .)

Parallel programming is . . .

- *inevitable* (if you'd like maximal throughput)
- *hard*

Problem: People don't think 'that way'

"Automatic parallelization" ha

→ People have to be taught t

Bad news: Parallelism might not even be our worst problem.

Don't just need to compute, also need to transmit information (to memory, say)

More bad news from Dr. Dennard

Parameter	**Factor**
Dimension	$1/\kappa$
Line Resistance	κ
Voltage drop	κ
Response time	1
Current density	κ

[Dennard et al. '74, via Bohr '07]

- The above scaling law is for on-chip interconnects.
- Off-chip: Similar consideration.
  Current $\sim$ Power *vs.* response time

More bad news from Dr. Dennard

Parameter	**Factor**
Dimension	$1/\kappa$
Line Resistance	κ
Voltage drop	κ
Response time	1
Current density	κ

[Dennard et a

- The above scaling law is f
- Off-chip: Similar consider
  Current $\sim$ Power *vs.* re

Getting information from

- processor to memory
- one computer to the next

is

- slow (in *latency*)
- power-hungry

Summary

Main problems for this class:

1. Express parallelism
2. Express communication/synchronization
3. Analyze, understand run time
   Both theoretically and practically (by measurement)

1 and 2 are **language issues!**

Summary

Main problems for this class:

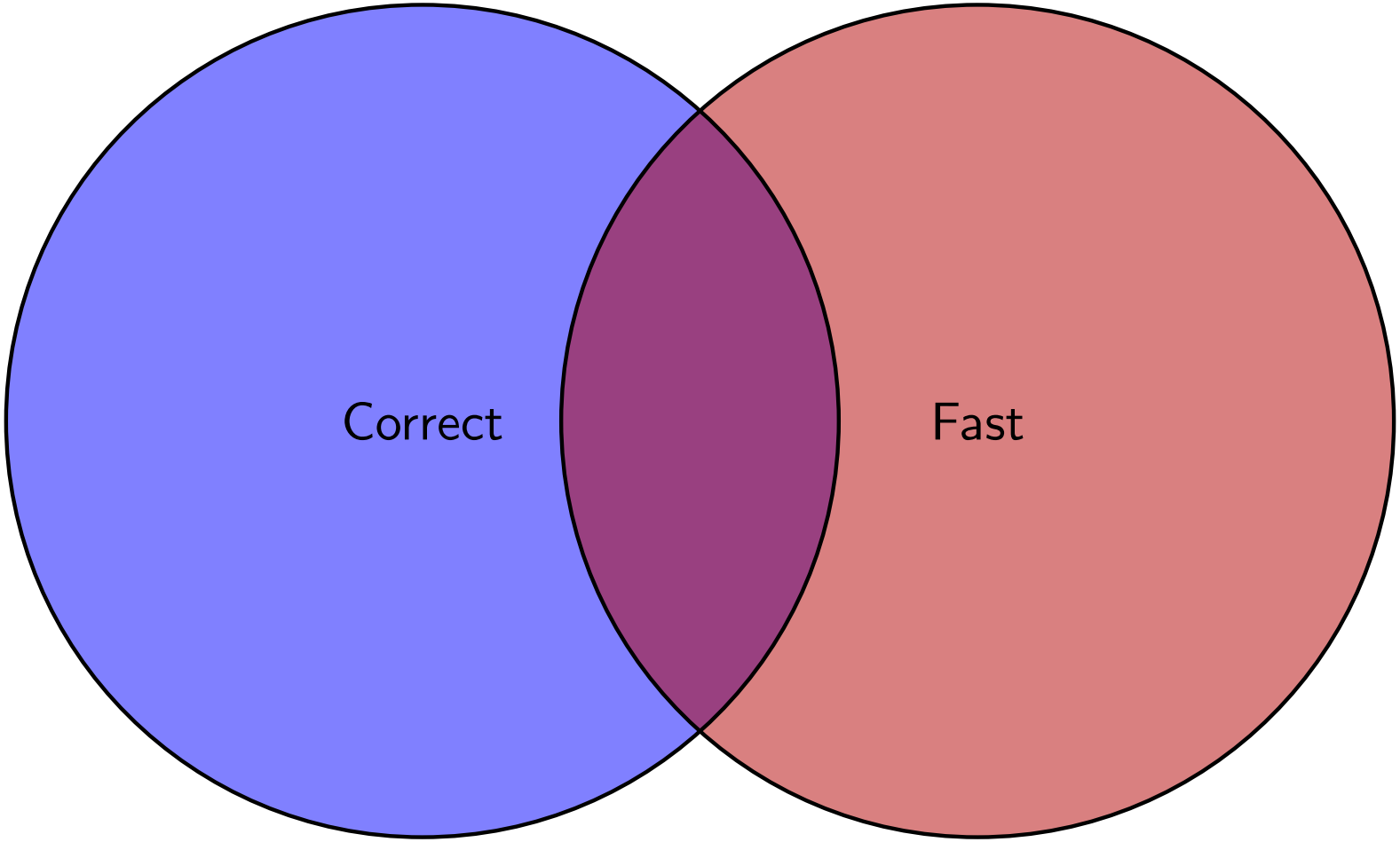
1. Express parallelism
2. Express communication/synchronization
3. Analyze, understand run time
   Both theoretically and practically (by measurement)

1 and 2 are **language issues!**

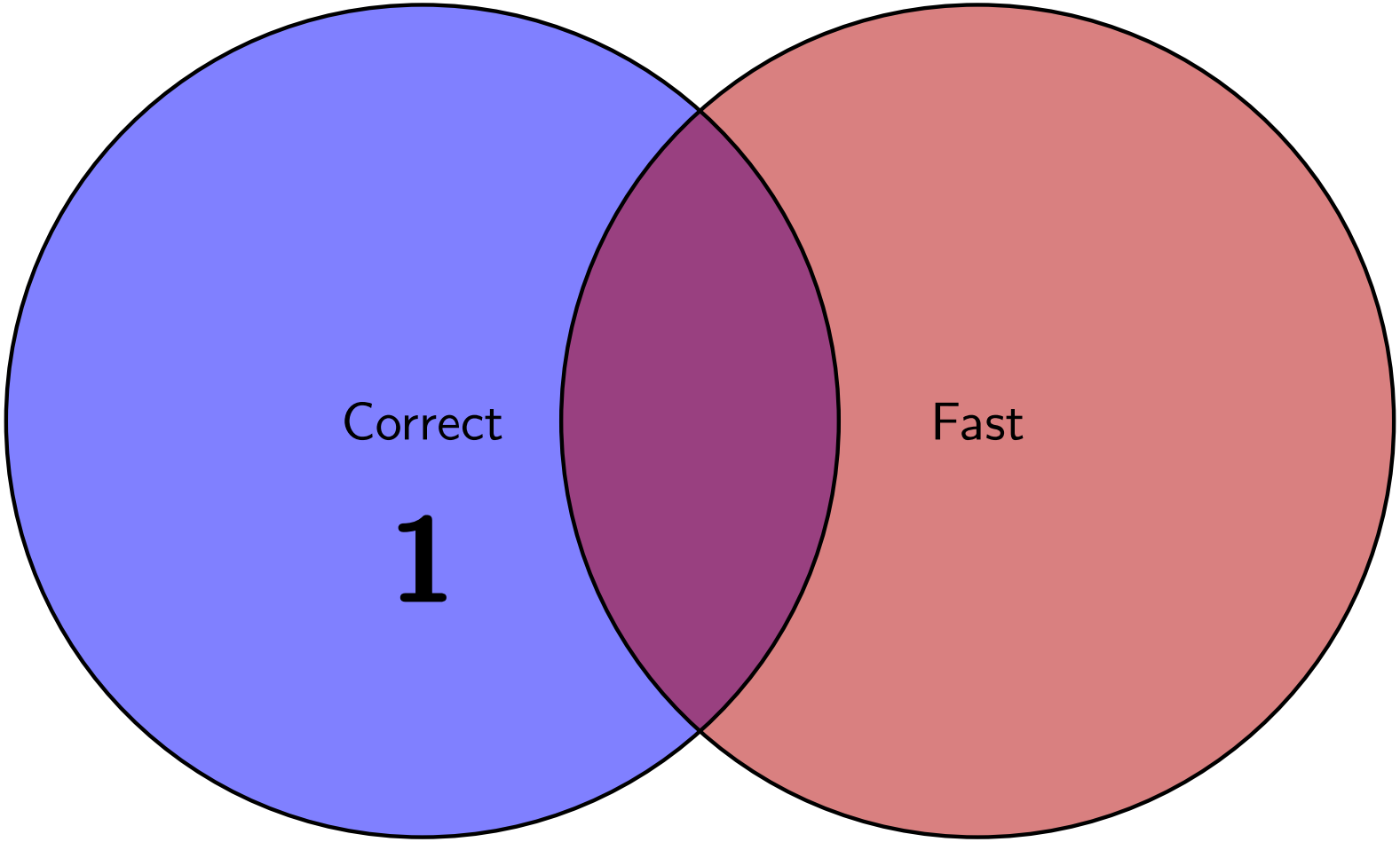
A little bit of terminology:

- Speedup, Efficiency, Scaling
- "Amdahl's law":
  Speed up 10% of your program by a factor of 10?

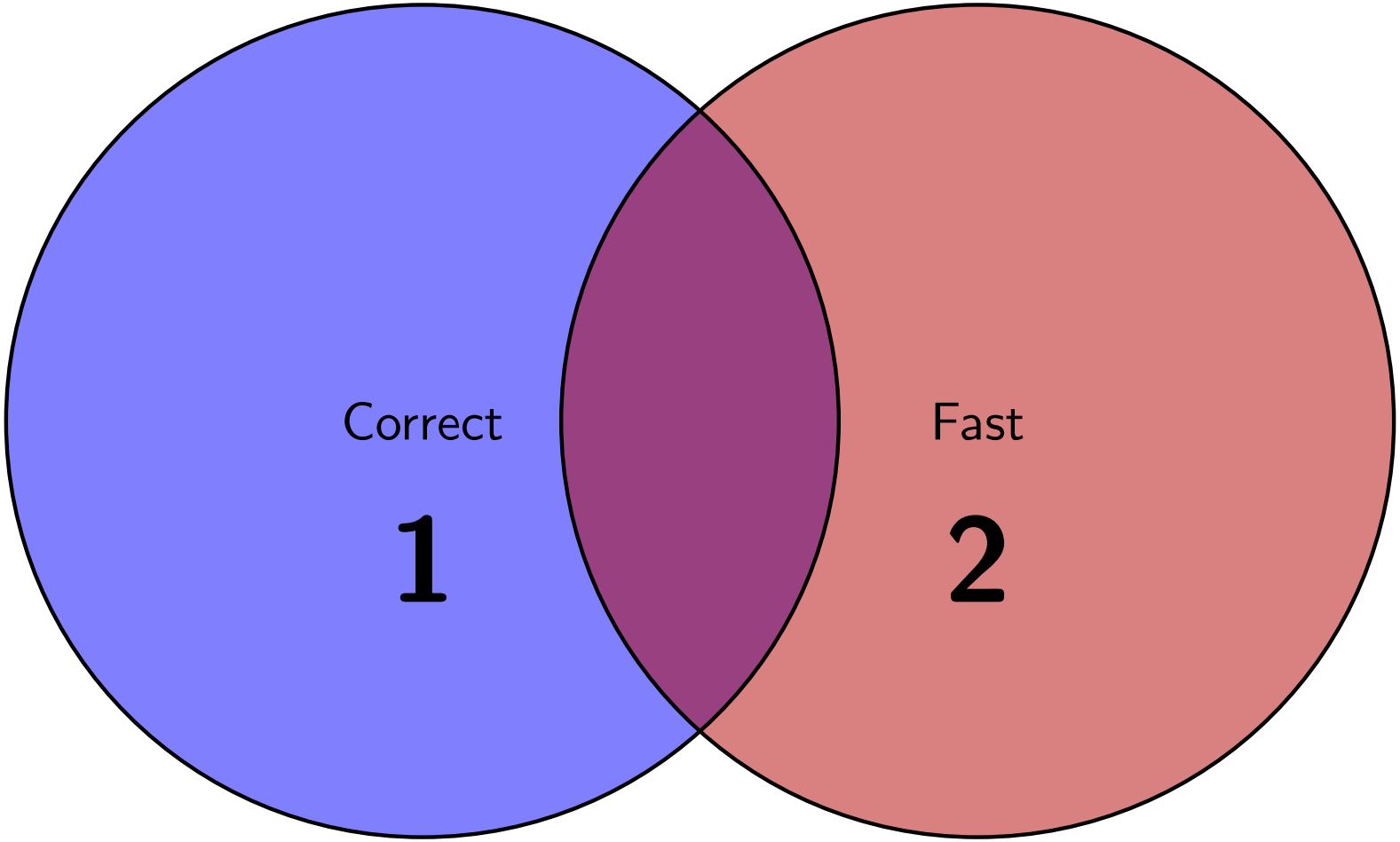
Parallelism as a Language Question



Parallelism as a Language Question



Parallelism as a Language Question



Outline

Demo time!

Outline

HPC as a Spectator Sport

Rank	Rmax Rpeak (Pflops)	Name	Computer design Processor type, interconnect	Vendor	Site Country, year	Operating system
1	16.324 20.132	*Sequoia*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Lawrence Livermore National Laboratory United States, 2011	Linux (RHEL and CNL)
2	10.510 11.280	*K computer*	**RIKEN** SPARC64 VIIIfx, Tofu	Fujitsu	RIKEN Japan, 2011	Linux
3	8.162 10.066	*Mira*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Argonne National Laboratory United States, 2012	Linux
4	2.897 3.185	*SuperMUC*	**iDataPlex DX360M4** Xeon E5–2680, Infiniband	IBM	Leibniz-Rechenzentrum Germany, 2012	Linux
5	2.566 4.701	*Tianhe-1A*	**NUDT YH Cluster** Xeon 5670 + Tesla 2050, Arch[4]	NUDT	National Supercomputing Center of Tianjin China, 2010	Linux
6	1.941 2.627	*Jaguar*	**Cray XT5** Opteron 6274 + Tesla 2090, Cray Gemini	Cray	Oak Ridge National Laboratory United States, 2009	Linux (CLE)
7	1.725 2.097	*Fermi*	**Blue Gene/Q** PowerPC A2, Custom	IBM	CINECA Italy, 2012	Linux
8	1.380 1.677	*JuQUEEN*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Forschungszentrum Jülich Germany, 2012	Linux
9	1.359 1.667	*Curie*	**Bullx B510** Xeon E5–2680, Infiniband	Bull	TGCC at CEA, and GENCI France, 2012	Linux (bullx)
10	1.271 2.984	*Nebulae*	**TC3600 Blade** Xeon 5650 + Tesla 2050, InfiniBand	Dawning	National Supercomputing Center in Shenzhen|NSCS China, 2010	Linux

`http://top500.org`

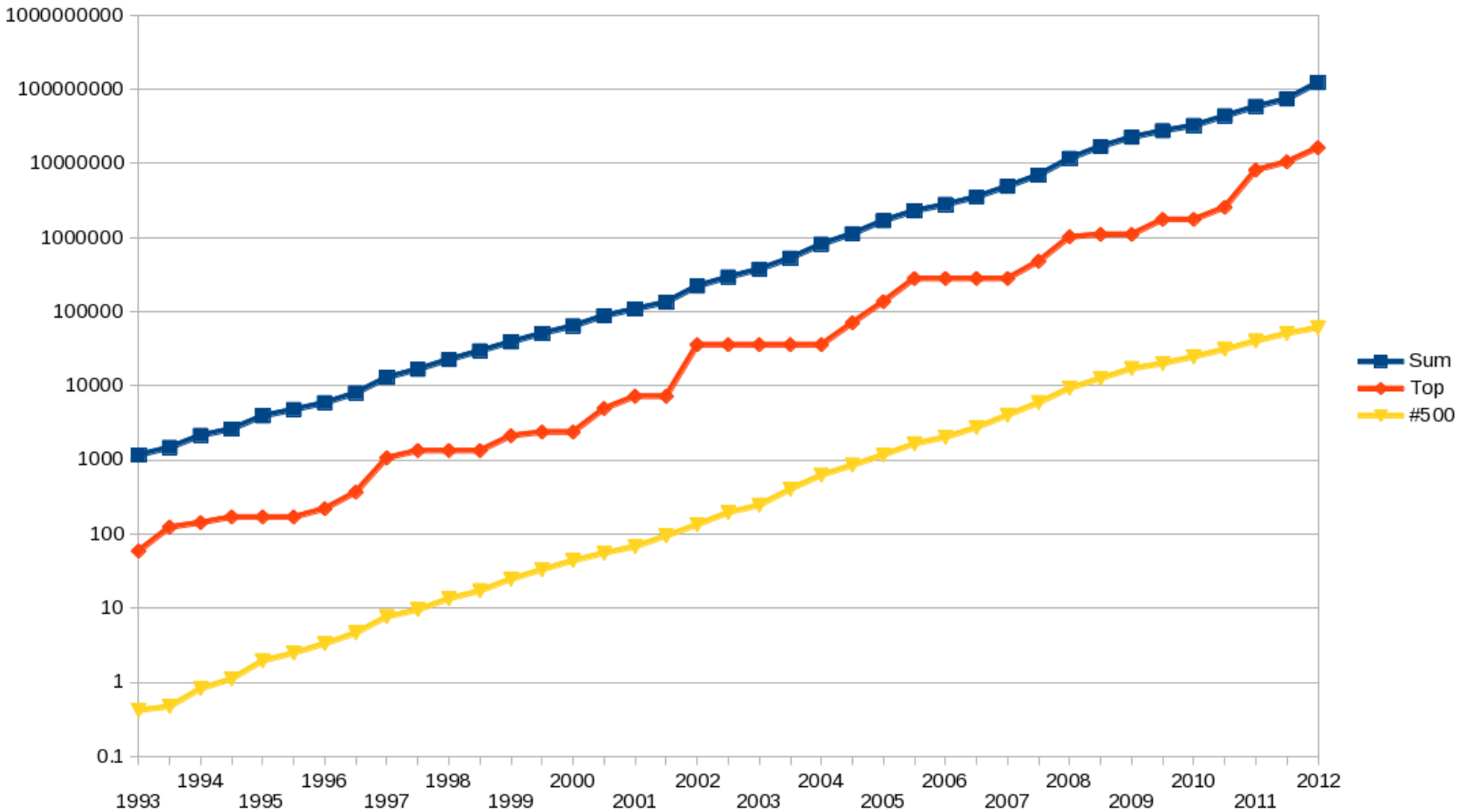
HPC as a Spectator Sport

Rank	Rmax Rpeak (Pflops)	Name	Computer design Processor type, interconnect	Vendor	Site Country, year	Operating system
1	16.324 20.132	*Sequoia*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Lawrence Livermore National Laboratory United States, 2011	Linux (RHEL and CNL)
2	10.510 11.280	*K computer*	**RIKEN** SPARC64 VIIIfx, Tofu	Fujitsu	RIKEN Japan, 2011	Linux
3	8.162 10.066	*Mira*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Argonne National Laboratory United States, 2012	Linux
4	2.897 3.185	*SuperMUC*	**iDataPlex DX360M4** Xeon E5–2680, Infiniband	IBM	Leibniz-Rechenzentrum Germany, 2012	Linux
5	2.566 4.701	*Tianhe-1A*	**NUDT YH Cluster** Xeon 5670 + Tesla 2050, Arch[4]	NUDT	National Supercomputing Center of Tianjin China, 2010	Linux
6	1.941 2.627	*Jaguar*	**Cray XT5** Opteron 6274 + Tesla 2090, Cray Gemini	Cray	Oak Ridge National Laboratory United States, 2009	Linux (CLE)
7	1.725 2.097	*Fermi*	**Blue Gene/Q** PowerPC A2, Custom	IBM	CINECA Italy, 2012	Linux
8	1.380 1.677	*JuQUEEN*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Forschungszentrum Jülich Germany, 2012	Linux
9	1.359 1.667	*Curie*	**Bullx B510** Xeon E5–2680, Infiniband	Bull	TGCC at CEA, and GENCI France, 2012	Linux (bullx)
10	1.271 2.984	*Nebulae*	**TC3600 Blade** Xeon 5650 + T InfiniBand		National Supercomputing Center in	

`http://top500.org`

Know your gigas, teras, petas, and exas.

HPC as a Spectator Sport



`http://top500.org`

Questions?

?

Image Credits

- Notebook: sxc.hu/abeall
- Question mark: sxc.hu/svilen001
- Camera: sxc.hu/Kolobsek
- Gordon Moore: Wikipedia