Welcome to

Programming Massively Parallel Processors (PMPP)

Prof. Dr.-Ing. Michael Goesele
Dr. Stefan Guthe
Dominik Wodniok

Graphics, Capture and Massively Parallel Computing (GCC)
TU Darmstadt

Preliminaries

TECHNISCHE UNIVERSITÄT DARMSTADT

- lecturers
  - Prof. Dr.-Ing. Michael Goesele
    goesele@cs.tu-darmstadt.de
    office: Room 2.7, Rundeturmstrasse 12 (Building S3|19)
  - Dr. Stefan Guthe
    stefan.guthe@gris.informatik.tu-darmstadt.de
    office: Room 2.10, Rundeturmstrasse 12 (Building S3|19)
  - Dominik Wodniok
    dominik.wodniok@gris.informatik.tu-darmstadt.de
    office: Room 2.1, Rundeturmstrasse 12 (Building S3|19)

Preliminaries

- tutors
  - Daniel Thul
    daniel.thul@gris.informatik.tu-darmstadt.de
  - Max von Bülow
    max.von.buelow@gris.informatik.tu-darmstadt.de

Preliminaries

- “Integrierte Lehrveranstaltung” (IV4)
  - Integrierte Lehrveranstaltungen bieten dem Lehrenden die Möglichkeit je nach Erfordernis des Stoffes zwischen verschiedenen Lehrformen wie Vorlesung, Übung, Multimedia-/Teleteaching usw. frei hin und her zu schalten. Z.B. besteht auch die Möglichkeit, dass die Studierenden zuerst einen Text lesen und anschließend darüber diskutiert wird.
    [Studienordnung Informatik, TU Darmstadt, FB Informatik]

Course Concept

- lectures
  - theoretical and practical introduction into the topic of programming massively parallel processors (e.g., including algorithms, architectural aspects)
- exercises
  - practical programming exercises based on NVIDIAs CUDA framework (on Linux)
- final projects
  - Topics for the final project will be proposed and co-advised by researchers from different fields. Final projects will be group projects, typically with 2 students per group.
- potentially other elements

➔ it is important that you can attend both weekly slots (but not all slots will be used)

Preliminaries

- class meetings
  - Monday 11:40 – 13:10
    ~~Room 074, Fraunhofer IGD, Building S3|05~~
    Room S103/223, Altes Hauptgebäude (with some exceptions)
  - Tuesday 11:40 – 13:10
    Room 074, Fraunhofer IGD, Building S3|05
    subject to changes

Preliminaries

- prerequisites
  - solid programming experience in C/C++
  - basic algorithms and data structures
  - no knowledge in CUDA required

- course web page(s)
  - http://www.gris.tu-darmstadt.de/teaching/courses/ws1516/pmpp/index.en.htm
    → mostly static content

  - lecture slides and additional information will be available in the HRZ Moodle
    https://moodle.tu-darmstadt.de/course/view.php?id=6733

Final Projects

- possibility to choose among different projects
- topics from several areas
  - details TBA

- each project will have one or two supervisors
  - one from the specific field
  - one for CUDA related questions

➔ learn massively parallel programming not from toy examples but from real problems

Grades, Final Exam

TECHNISCHE UNIVERSITÄT DARMSTADT

- written final exam
  - details and date TBA

- exercises/final projects
  - bonus system with up to one full grade improvement
  - in total 10 points (2 for exercises, 8 for final project)
  - algorithm if X BP achieved
    - X < 4 BP → no change
    - 4 BP ≤ X < 6 BP → 1/3 grade improvement
    - 6 BP ≤ X < 8 BP → 2/3 grade improvement
    - 8 BP ≤ X → 1 grade improvement

(Preliminary) Course Schedule

TECHNISCHE UNIVERSITÄT DARMSTADT

12.10.2015	Introduction to PMPP
13.10.2015	Lecture CUDA Programming 1
19.10.2015	Lecture CUDA Programming 2
20.10.2015	Lecture CUDA Programming 3
26.10.2015	Introduction Final Projects, Exercise 1 assigned
27.10.2015	Questions and Answers (Q&A)
2.11.2015	Lecture, Final Projects assigned, Ex. 1 due, Ex. 2 assigned
3.11.2015	Questions and Answers (Q&A)
9.11.2015	Lecture, Exercise 2 due
10.11.2015	Lecture
16.11.2015	Questions and Answers (Q&A)
17.11.2015	Questions and Answers (Q&A)
23.11.2015	1st Status Presentation Final Projects
24.11.2015	1st Status Presentation Final Projects (continued)
30.11.2015	
1.12.2015	

(Preliminary) Course Schedule

TECHNISCHE UNIVERSITÄT DARMSTADT

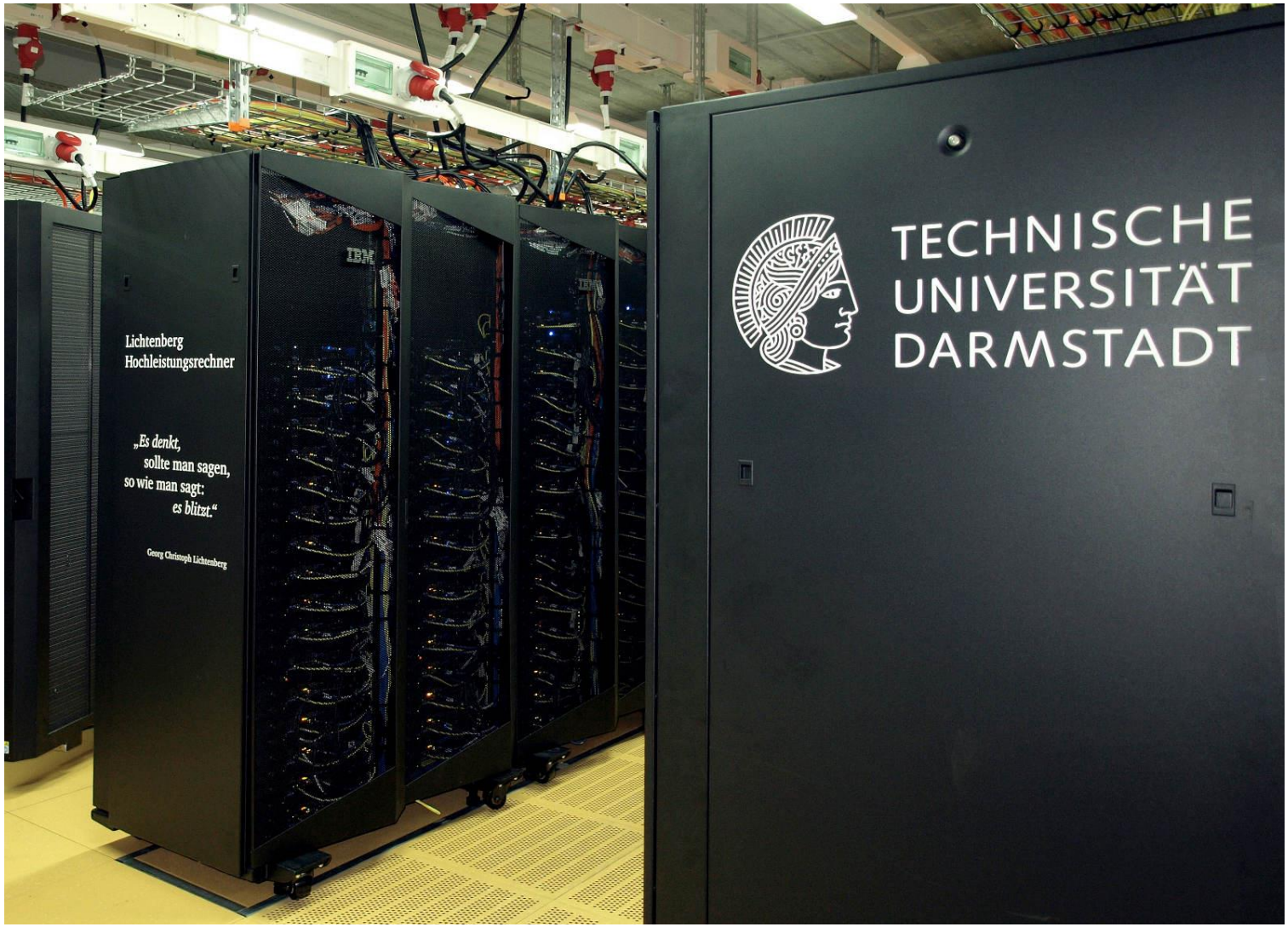
7.12.2015	
8.12.2015	
14.12.2015	
15.12.2015	
	Christmas break
11.1.2016	2nd Status Presentation Final Projects
12.1.2016	2nd Status Presentation Final Projects (continued)
18.1.2016	
19.1.2016	
25.1.2016	
26.1.2016	
1.2.2016	
2.2.2016	
8.2.2016	Final Presentation Final Projects
9.2.2016	Final Presentation Final Projects (continued)

Hardware and Accounts

TECHNISCHE UNIVERSITÄT DARMSTADT

- all exercises and projects will be solved using NVIDIA CUDA on Linux systems
  - all exercises and projects will run on the HHLR
  - register in TuCAN ASAP so that we can create your accounts
  - fill in and sign account form
- also possible but not really recommended to run CUDA on your own system
  - no support provided
  - all exercises and projects must run on HHLR for grading

HHLR


IBM
Lichtenberg
Hochleistungsrechner
„Es denkt,
sollte man sagen,
so wie man sagt:
es blitzt.“
Georg Christoph Lichtenberg
TECHNISCHE
UNIVERSITÄT
DARMSTADT

HHLR Account

TECHNISCHE UNIVERSITÄT DARMSTADT

Nutzungsordung des Hochleistungsrechners der TU Darmstadt Nutzung durch Studierende im Rahmen einer Lehrveranstaltung

TECHNISCHE UNIVERSITÄT DARMSTADT
HRZ Hochschulrechenzentrum

1. Präambel

Diese Nutzungsordnung legt fest, nach welchen Regeln der Hochleistungsrechner von Studierenden im Rahmen einer Lehrveranstaltung der TU Darmstadt benutzt werden darf.

Der Hochleistungsrechner steht überdies den Wissenschaftlerinnen und Wissenschaftlern der TU Darmstadt und den Wissenschaftlerinnen und Wissenschaftlern anderer Universitäten zur Verfügung. Wissenschaftliches Rechnen ist gestattet, sofern die eingesetzte Software dies erlaubt. Jegliche rein kommerzielle Nutzung ist untersagt. Bestandteil dieser Nutzungsordnung sind die Bestimmungen in der Allgemeinen Benutzungsordnung für die Informationsverarbeitungs- und Kommunikations-Infrastruktur [1] der TU Darmstadt.

Ein Verstoß gegen diese Nutzungsordnung kann zum Entzug der Nutzungsberechtigung führen.

Alle ausgefüllten Nutzungsanträge (dieses Formular) müssen vom Veranstalter gemeinsam mit dem gesonderten Nutzungsantrag zur Lehrveranstaltung beim HRZ eingereicht werden. Für Fragen stehen wir Ihnen gerne auch per E-Mail: hhlr@hrz.tu-darmstadt.de zur Verfügung.

2. Aufgaben des Hochschulrechenzentrums der TU Darmstadt

2.a. Betrieb des Hochleistungsrechners

Der Hochleistungsrechner wird vom Hochschulrechenzentrum (HRZ) der Technischen Universität Darmstadt betrieben. Zu den Aufgaben des HRZ gehören unter anderem die Zuteilung von Rechenzeit, Hilfestellung bei der Softwareinstallation und das Einrichten von Nutzerkonten.

Die Zuteilung der Rechenzeit erfolgt nach festgelegten Regeln (siehe Abschnitt 3.c) ohne Ausgleichsanspruch. Für den Fall, dass einem Nutzer/einer Nutzerin Rechenzeit entgeht (z.B. wegen einer Systemauszeit oder durch Abbruch eines Rechenjobs auf Grund eines Fehlers), gibt es keinen Anspruch auf Ausgleich. Im Zweifelsfall entscheidet der Leiter des HRZ der TU Darmstadt.

2.b. Speicherung von Login-Daten

Das HRZ speichert ausschließlich Login-Daten der Nutzer/-innen. Diese sind: a) Personen- und Projekt-Informationen (Abschnitt 5) und b) die Betriebsdaten (z.B. verbrauchte Ressourcen oder Login-Zeiten). Diese Daten dienen allein der internen Verwaltung und sind für das Management des Hochleistungsrechners notwendig; die Zweckbindung des § 13 Abs. 5 HDSG ist zu beachten.

2.c. Speicherung von Arbeitsdaten *Bitte BETA-Hinweis unten beachten!*

Ein Ziel ist ausschließlich die Weiterbildung im Rahmen der unter 5 angegebenen Veranstaltung, deshalb ist die Speicherung und Verarbeitung von privaten Daten des Nutzers (z.B. persönliche Bilder oder E-Mails etc.) auf der Infrastruktur des HPC-Clusters untersagt.

Für die Arbeitsdaten stehen dem/der Nutzer/-in der Bereich des Home-Verzeichnisses „/home_na/<TU-ID>" zur Verfügung. Eine Sicherung der Daten im Home (/home_na/<TU-ID>) wird für Accounts im Rahmen der Lehrveranstaltung nicht durchgeführt. Der Speicherplatz des Home-Bereichs ist für jeden Nutzer/ jede Nutzerin begrenzt.

Sollte der/die Nutzer/-in bereits ein reguläres Nutzer-Konto (z.B. im Rahmen seiner HiWi-Tätigkeit) besitzen, bleibt dieses Home-Verzeichnis (/home/<TU-ID>) unangetastet. Für die hier beantragte Veranstaltung wurde ein gesondertes Home-Verzeichnis unter „/home_na/<TU-ID>" eingrichtet.

2.d. Ende der Nutzungsberechtigung

Beim Auslaufen oder beim Entzug der Nutzungsberechtigung schließt das HRZ das Nutzerkonto. Nach Beendigung der Nutzungsberechtigung des Antragstellers/der Antragstellerin werden die Nutzer-Daten (/home_na/<TU-ID>) unwiderruflich gelöscht.

3. Aufgaben der Nutzer-/innen

3.a. Allgemeine Nutzungsbedingungen

Das HPC-Cluster des Hochschulrechenzentrums der TU Darmstadt steht den Studierenden ausschließlich für Arbeiten im Rahmen der beantragten Lehrveranstaltung zur Verfügung. Eine anderweitige Nutzung, z.B. kommerzieller Art, ist untersagt.

3.b. Umgang mit dem Nutzerkonto

Für die Nutzung des Hochleistungsrechners wird dem/der Antragsteller/-in ein persönliches Nutzerkonto eingerichtet. Das Nutzerkonto gehört zum ID-System der TU Darmstadt.

Das Passwort setzt der/die Nutzer/-in nach Maßgabe der Sicherheitsrichtlinien aus der Allgemeinen Benutzerordnung [1] und den Richtlinien unter www.hhlr.tu-darmstadt.de [2]. Das betrifft insbesondere die Sicherheit bei der Auswahl des Passwortes sowie die regelmäßige Änderung. **Die Weitergabe des Nutzerkontos (z.B. durch Passwortweitergabe oder fremde SSH-Keys) an andere Personen ist grundsätzlich untersagt.** Der/die Besitzer/-in des Nutzerkontos haftet für Schäden, die durch unsachgemäßen Umgang mit dem Nutzerkonto (z.B. Passwortweitergabe) angerichtet werden.

3.c. Faire Nutzung

Es ist untersagt, zum Beispiel durch das bewusste Ausnutzen von Systemfehlern, sich a) unerlaubten Zugang zu Daten anderer Nutzer/-innen zu verschaffen, b) sich am „fair-queuing" vorbei mehr Rechenzeit zuteilen oder c) den Speicherplatz für nicht zur Nutzung des Systems notwendige Zwecke zu missbrauchen. Das kann zum Entzug der Nutzungsberechtigung führen.

Die genauen Regeln für die Rechenzeitverteilung nach dem Prinzip des „fair-queuing" werden auf der Webseite des Hochleistungsrechners [2] bekanntgegeben.

3.d. Lizenzbedingungen

Der/die Nutzer/-in darf lizenzpflichtige Software ausschließlich unter Einhaltung der Lizenzbedingungen einsetzen, insbesondere sind ggf. Beschränkungen hinsichtlich rein wissenschaftlicher Anwendung zu beachten. Der/die Nutzer/-in ist für die Prüfung und Einhaltung selbst verantwortlich.

4. Referenzen

[1] Allgemeine Benutzungsordnung für die Informationsverarbeitungs- und Kommunikations-Infrastruktur
http://www.hrz.tu-darmstadt.de/itsicherheit/regelwerke/allgemeinebenutzerordnung.de.jsp

[2] Hauptseite des hessischen Hochleistungsrechners an der TU Darmstadt
http://www.hhlr.tu-darmstadt.de/

*Achtung BETA: Die Adresse des Homeverzeichnisses kann sich noch ändern. Ein Update wird ggf in der Vorlesung bekanntgegeben.

5. Antrag auf Nutzung des Hochleistungsrechners im Rahmen einer Lehrveranstaltung

Informationen über den/die Antragsteller/-in

Dieses Formular finden Sie online auch unter „Nutzerantrag für Studierende (Lehrveranstaltung)" auf unserer Webseite http://www.hhlr.tu-darmstadt.de/hhlr/lichtenberg/zugang/lichtenberg_zugang.de.jsp.

Nachname, Vorname:	
E-Mail:	
TU-ID (auch Externe):	
Universität/wiss.Einrichtung:	TU-Darmstadt
Titel der Lehrveranstaltung:	Grundlagen der Informatik 3 (GdI 3)
Semester / Jahr:	Wintersemester 2013/14

Die maximale Laufzeit der Nutzungsberechtigung richtet sich nach dem Nutzungsantrag für die Lehrveranstaltung (aber max. 6 Monate).

Ich bestätige hiermit die Richtigkeit meiner Angaben und verpflichte mich zur Einhaltung dieser Nutzungsordnung. Ich bin mit der Verarbeitung meiner personenbezogenen Daten (nach Abschnitt 2.b) einverstanden.

....

Ort, Datum Unterschrift

(Wird vom HRZ ausgefüllt)

Wurde genehmigt und eingerichtet: ja/nein

Laufzeit des Nutzerkontos: *- siehe Hauptformular des/der Lehrenden -*

....

Datum Unterschrift

Anmerkungen:

Preliminaries

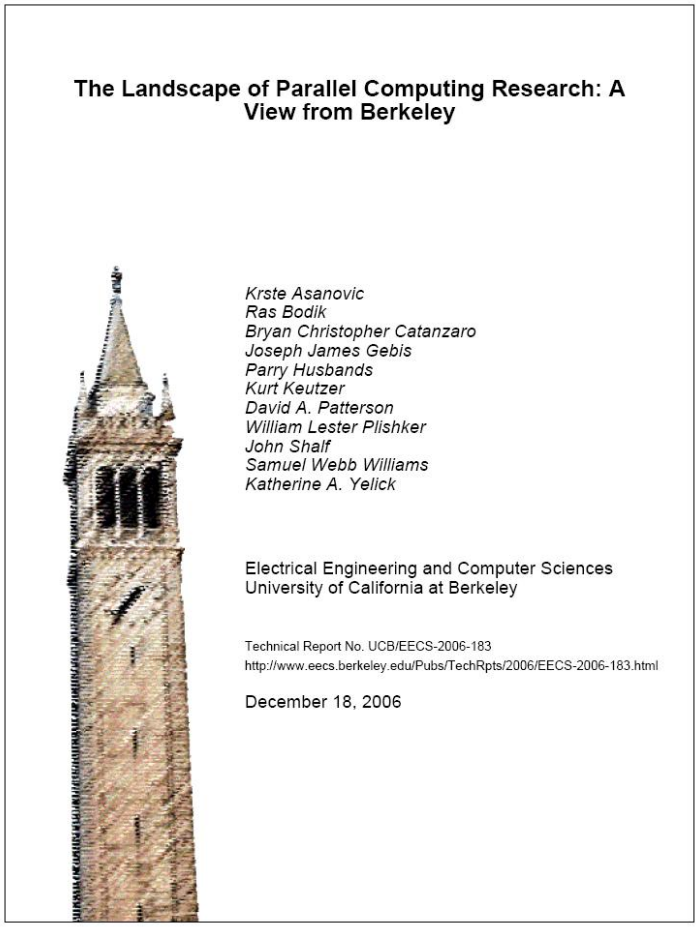
- literature and web references (see also web page)
  - CUDA Programming Guide and material from NVIDIA website
  - CUDA forums at NVIDIA

  - D. Kirk, W, Hwu: Programming Massively Parallel Processors
  - T. Mattson, B. Sanders, B. Massingill: Patterns for Parallel Programming, Addison Wesley
  - J. Keller, C. Kessler, J. Larsson Traeff: Practical PRAM Programming, Wiley-Interscience
    (out-of-print but available as ebook)
  - Hubert Nguyen: GPU Gems 3, Addison Wesley
  - publications in the field

- will be updated during the lecture

Why PMPP?

- discussion inspired by The Landscape of Parallel Computing Research: A View from Berkeley by Asanivic et al., December 18, 2006 Technical Report UCB/EECS-2006-183



The Landscape of Parallel Computing Research: A View from Berkeley

Krste Asanovic
Ras Bodik
Bryan Christopher Catanzaro
Joseph James Gebis
Parry Husbands
Kurt Keutzer
David A. Patterson
William Lester Plishker
John Shalf
Samuel Webb Williams
Katherine A. Yelick

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-183
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

December 18, 2006

Why PMPP?

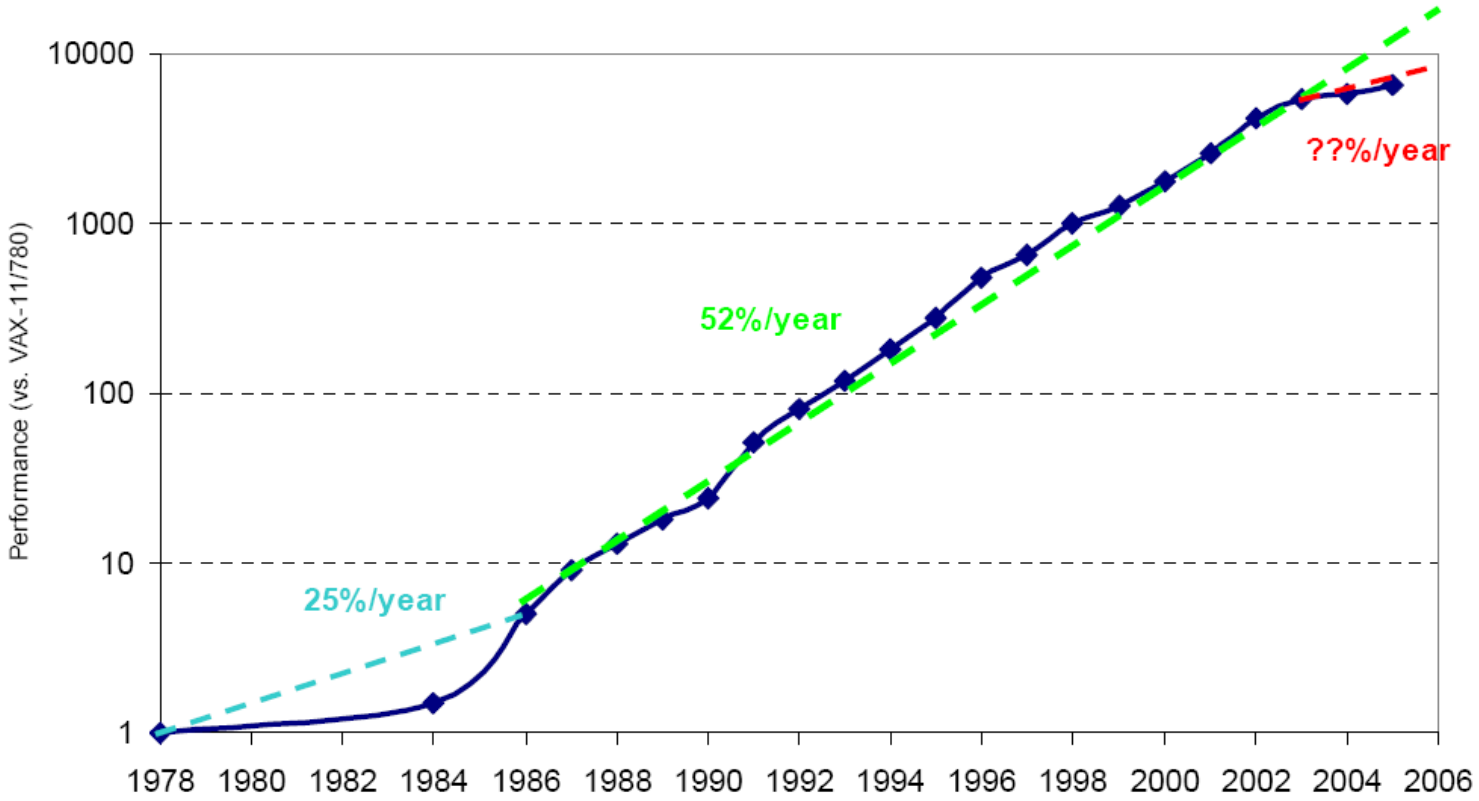


Figure 2. Processor performance improvement between 1978 and 2006 using integer SPEC [SPEC 2006] programs. RISCs helped inspire performance to improve by 52% per year between 1986 and 2002, which was much faster than the VAX minicomputer improved between 1978 and 1986. Since 2002, performance has improved less than 20% per year. By 2006, processors will be a factor of three slower than if progress had continued at 52% per year. This figure is Figure 1.1 in [Hennessy and Patterson 2007].

from [Asanovic et al. 2006]

Why PMPP?

- historically, single processor performance doubled every 18 months (1986-2002)
- shift to multi-core architecture with 2, 4, 8, … cores
- prediction that many-core systems with 100s or 1000s or processors will dominate in the future

➔ some conventional wisdoms (CW) do no longer apply (see [Asanovic et al. 2006])

Old and New CWs

- Old CW #1: Power is free, but transistors are expensive.

- New CW #1: “Power Wall”: Power is expensive, but transistors are “free”. That is, we can put more transistors on a chip than we have power to turn on.

Old and New CWs

- Old CW #7: Multiply is slow, but load and store is fast.

- New CW #7: “Memory Wall”: Load and store is slow, but multiply is fast. Modern microprocessors can take 200 clocks to access Dynamic Random Access Memory (DRAM), but even floating-point multiplies may take only four clock cycles.

Old and New CWs

- Old CW #8: “ILP Wall”: We can reveal more instruction-level parallelism (ILP) via compilers and architecture innovation. Examples from the past include branch prediction, out-of-order execution, speculation, and Very Long Instruction Word systems.

- New CW #8 : There are diminishing returns on finding more ILP. [Hennessy and Patterson 2007]

Old and New CWs

- Old CW #9: Uniprocessor performance doubles every 18 months.

- New CW #9: Power Wall + Memory Wall + ILP Wall = Brick Wall. […] In 2006, performance is a factor of three below the traditional doubling every 18 months that we enjoyed between 1986 and 2002. The doubling of uniprocessor performance may now take 5 years.

Old and New CWs

- Old CW #10: Don't bother parallelizing your application, as you can just wait a little while and run it on a much faster sequential computer.

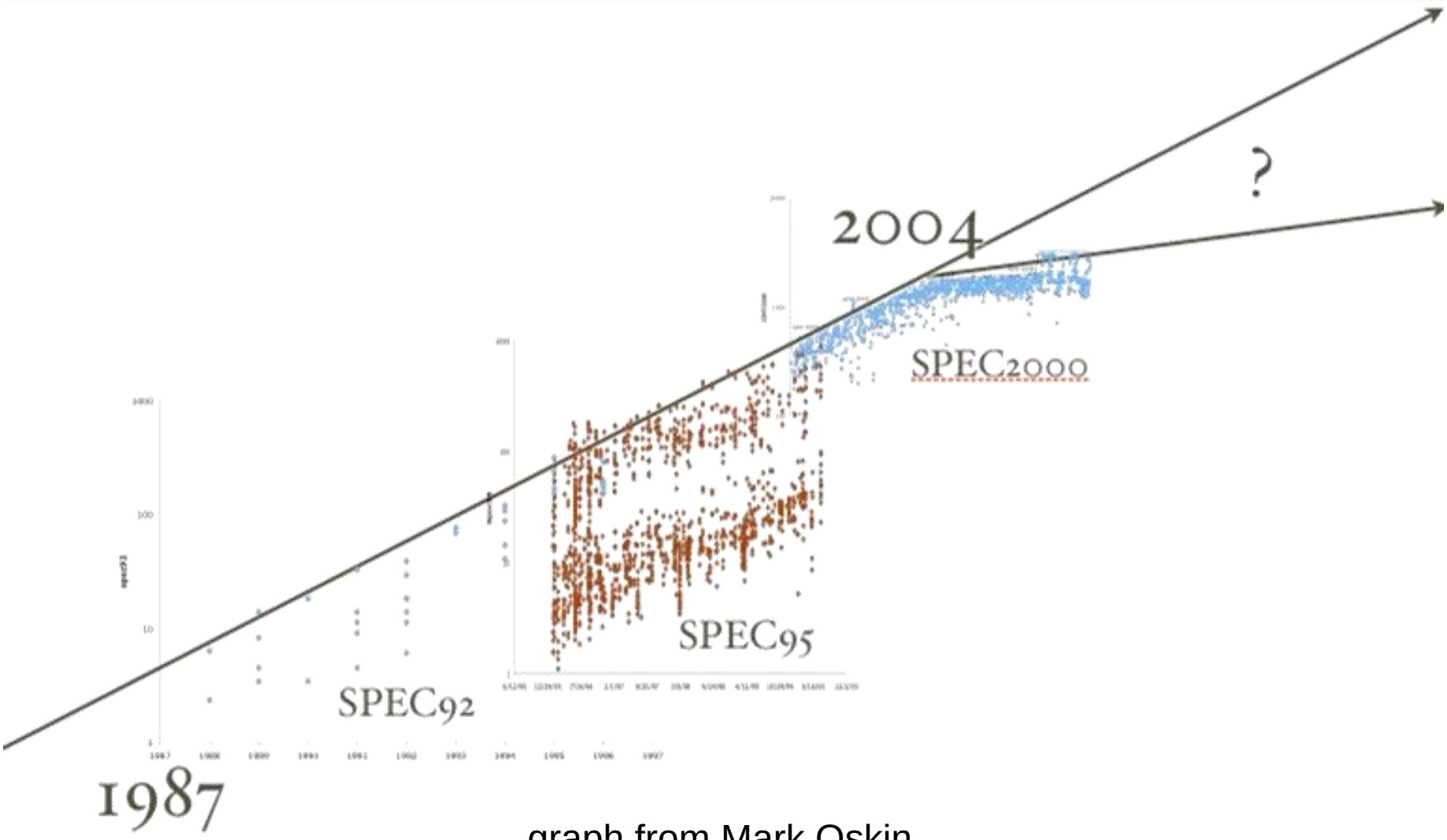
- New CW #10: It will be a very long wait for a faster sequential computer (see above).

Old and New CWs

- Old CW #11: Increasing clock frequency is the primary method of improving processor performance.

- New CW #11: Increasing parallelism is the primary method of improving processor performance. […]

Why PMPP?



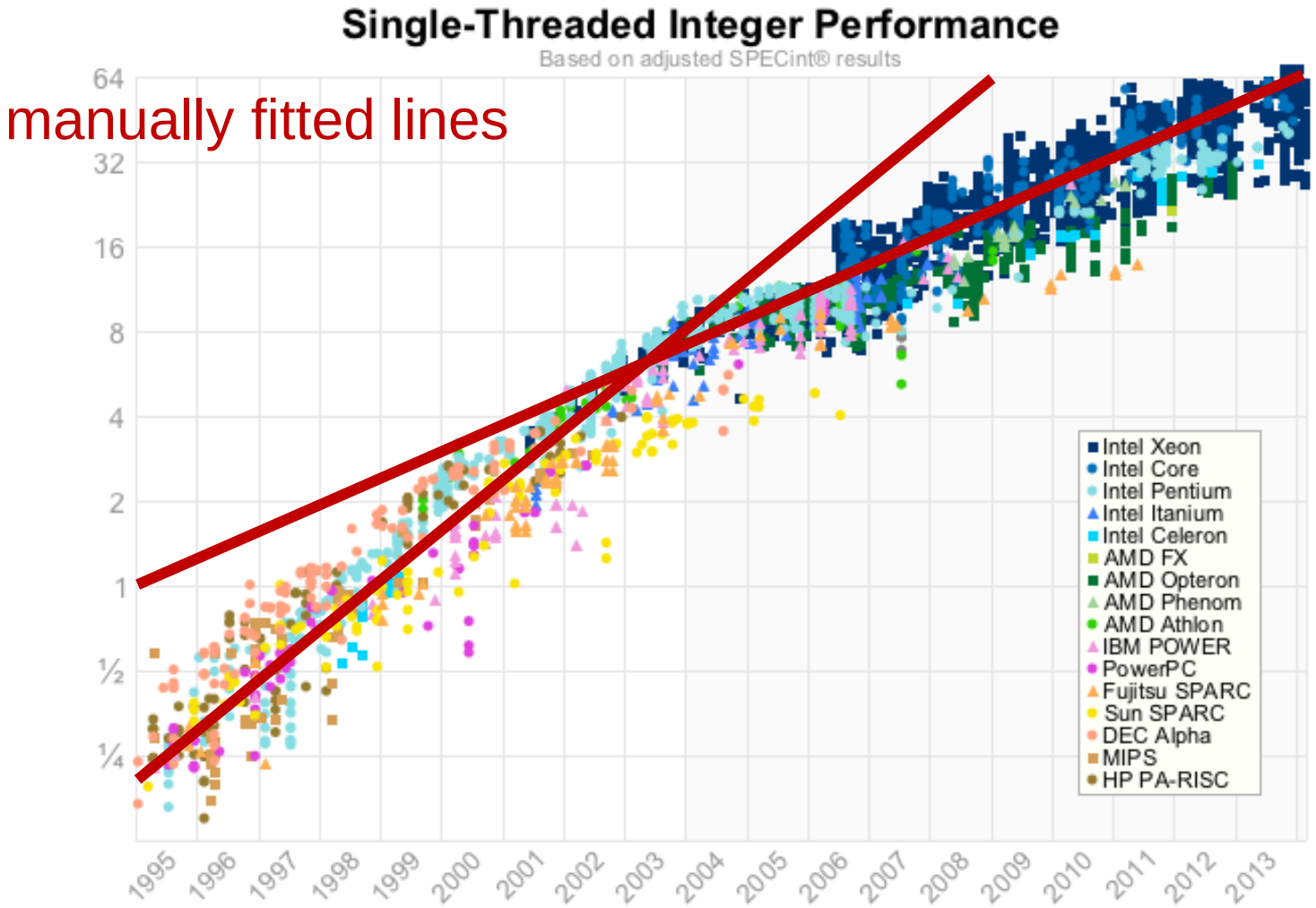
graph from Mark Oskin

Why PMPP?

Why you should take this course even if you are not an architect: See graph. Your world is now very different. If you can't program parallel machines (and to first order, no one can), then you'll be unable to utilize the new machines coming down the pike.

Mark Oskin about a course at UW with the informal title "Will parallelism save the world or are we doomed?"

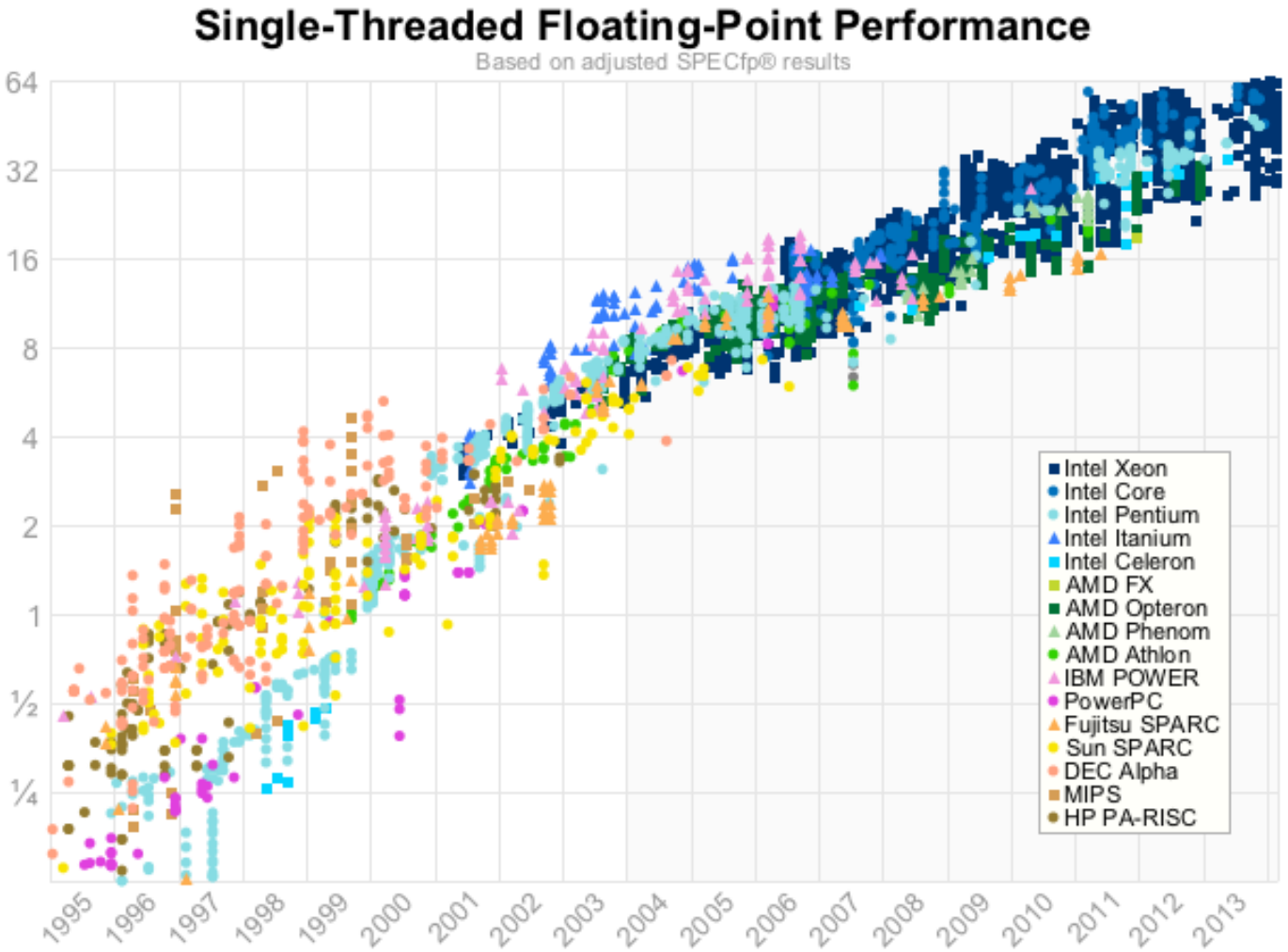
Why PMPP?



graph from Jeff Preshing, updated 02/2014 by Henk Poley

http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/

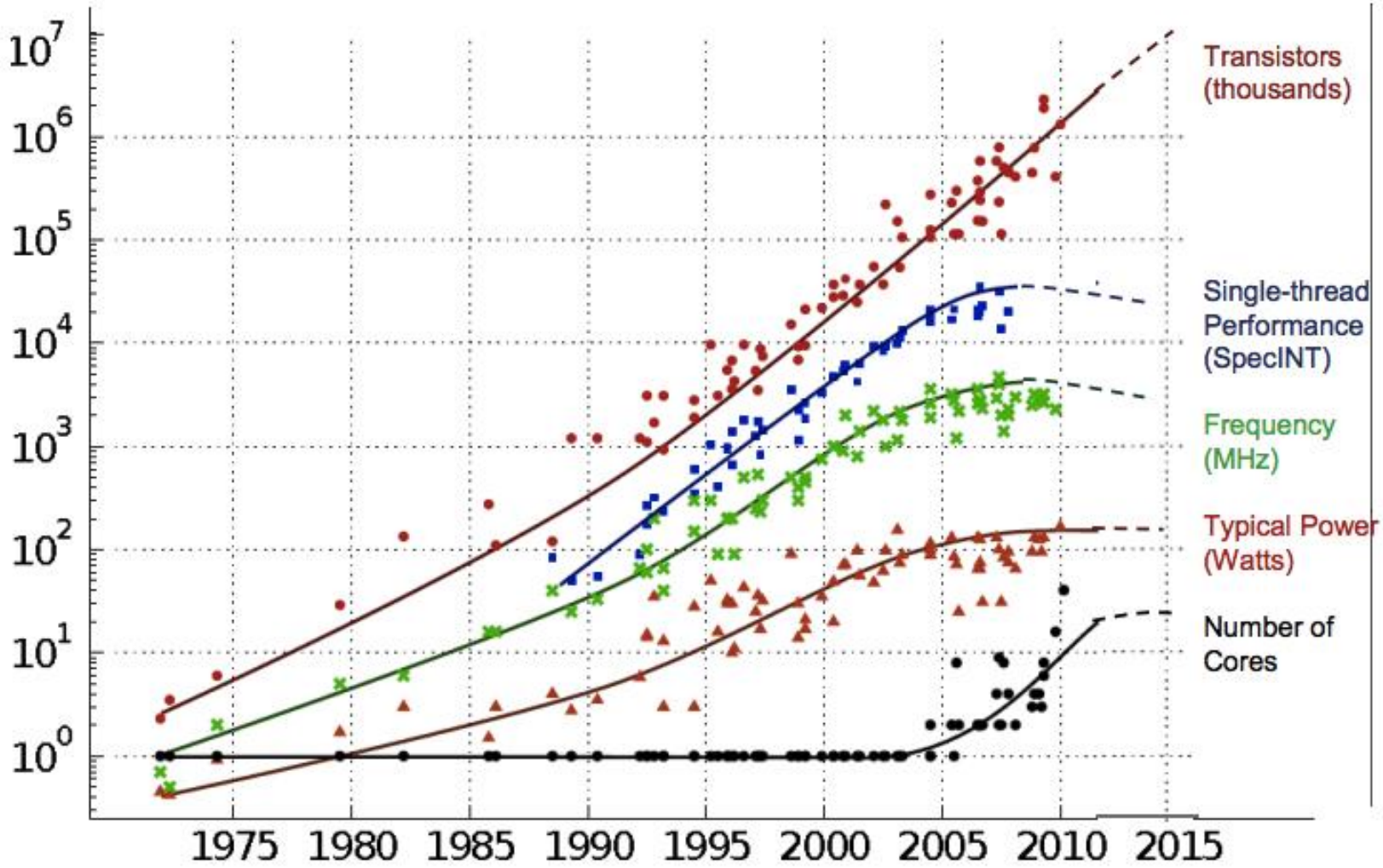
Why PMPP?



graph from Jeff Preshing, updated 02/2014 by Henk Poley

http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/

Why PMPP?



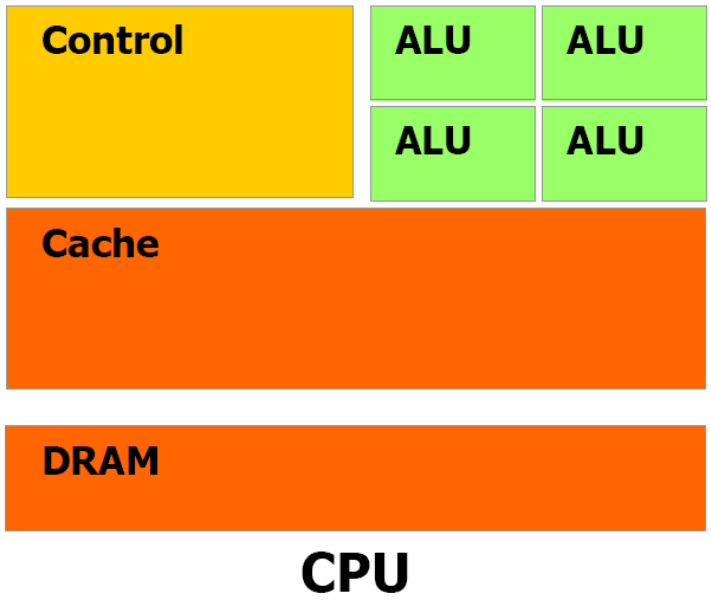
Chuck Moore, AMD Corporate Fellow and CTO of Technology Group

"DATA PROCESSING IN EXASCALE-CLASS COMPUTER SYSTEMS"
The Salishan Conference on High Speed Computing, 2011

Example System

TECHNISCHE UNIVERSITÄT DARMSTADT

- NVIDIA GPU with CUDA
  - data-parallel compute device
  - shift from fixed graphics pipeline to flexible setup with general purpose processors
  - multiple SIMD multi-processors with on-chip shared memory on a single GPU



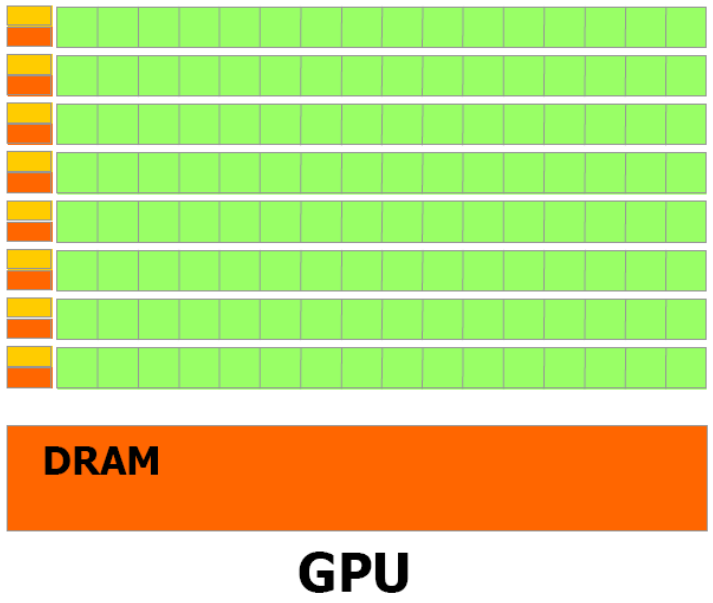


image source: NVIDIA

Example System

- NVIDIA GPU with CUDA capability
  - massive economies of scale
  - massively parallel
- two versions
  - “graphics series”: GeForce/Quadro
  - “compute series:” Tesla

image source: NVIDIA

HHLR Cluster

TECHNISCHE UNIVERSITÄT DARMSTADT

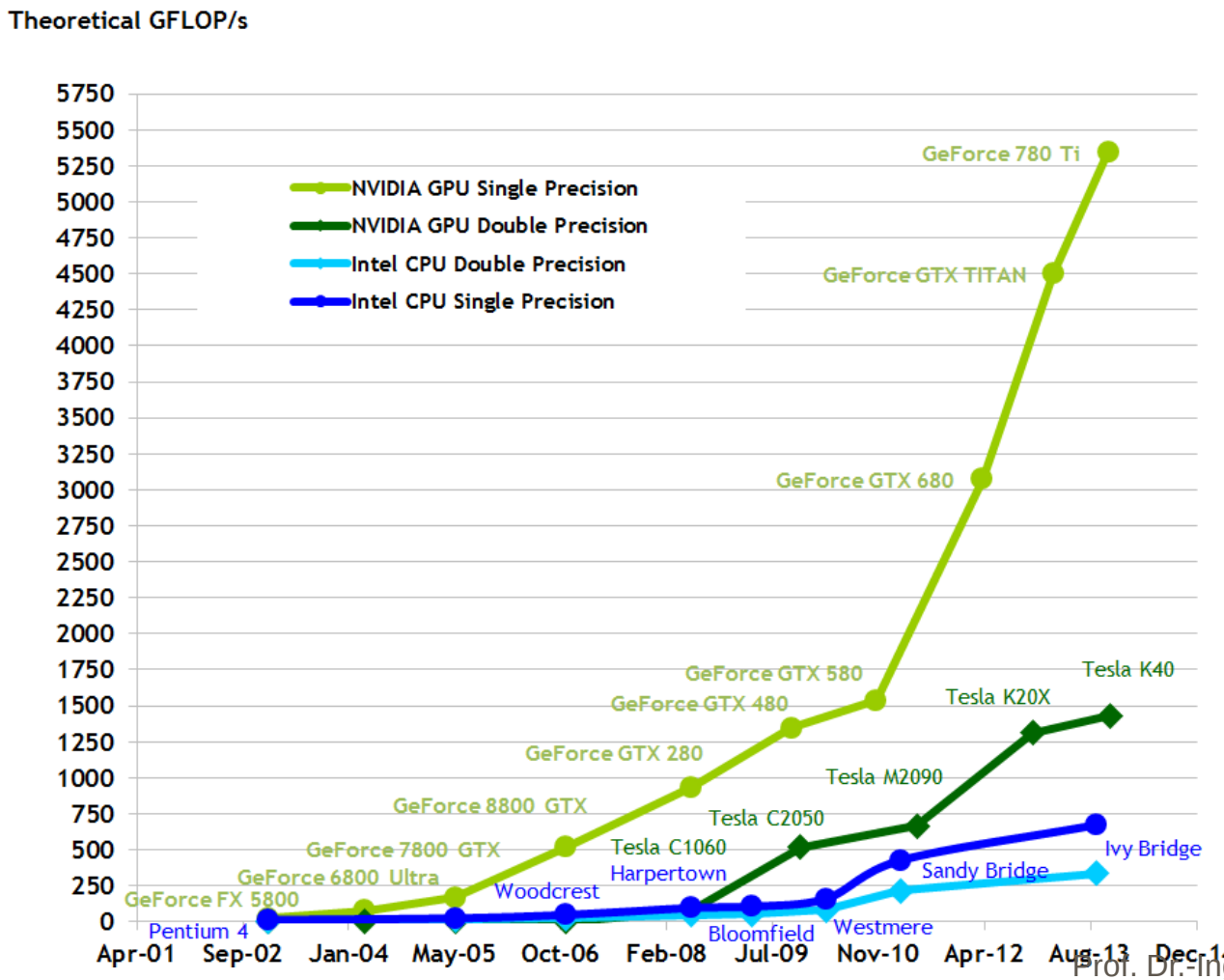
- 44 Nodes (IBM System x iDataPlex dx360 M4) with
  - 2x Intel Sandy Bridge Processors (each 8 Cores at 2.6 GHz)
  - Infiniband FDR-10 Interconnect
  - 32 GByte (Node-) Main Memory
  - 2x 500 GByte Hard disks

  - 2x NVIDIA Kepler (K20X), each
    - Peak Performance: 1.31 TFlop/s (Double Precision)
    - Linpack Performance: 1TFlop/s per card
    - 6 GByte GDDR5 Memory
    - Memory Performance: 250GByte/s

- 2 additional nodes with 2x NVIDIA K40m each
  - 1.43 TFlop/s peak(Double Precision)

Comparing GPU and CPU

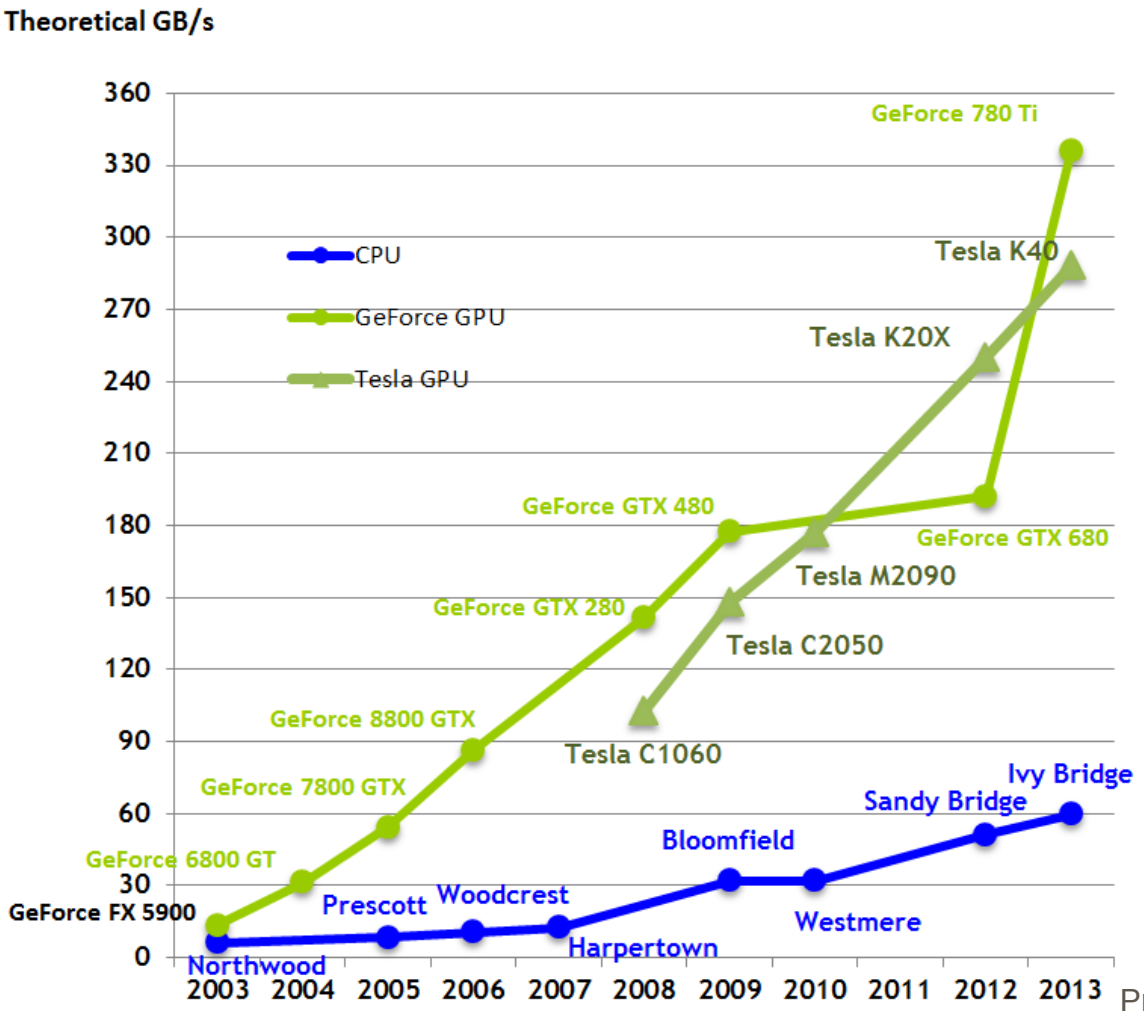
- floating point operations per second



source: NVIDIA

Comparing GPU and CPU

- memory bandwidth

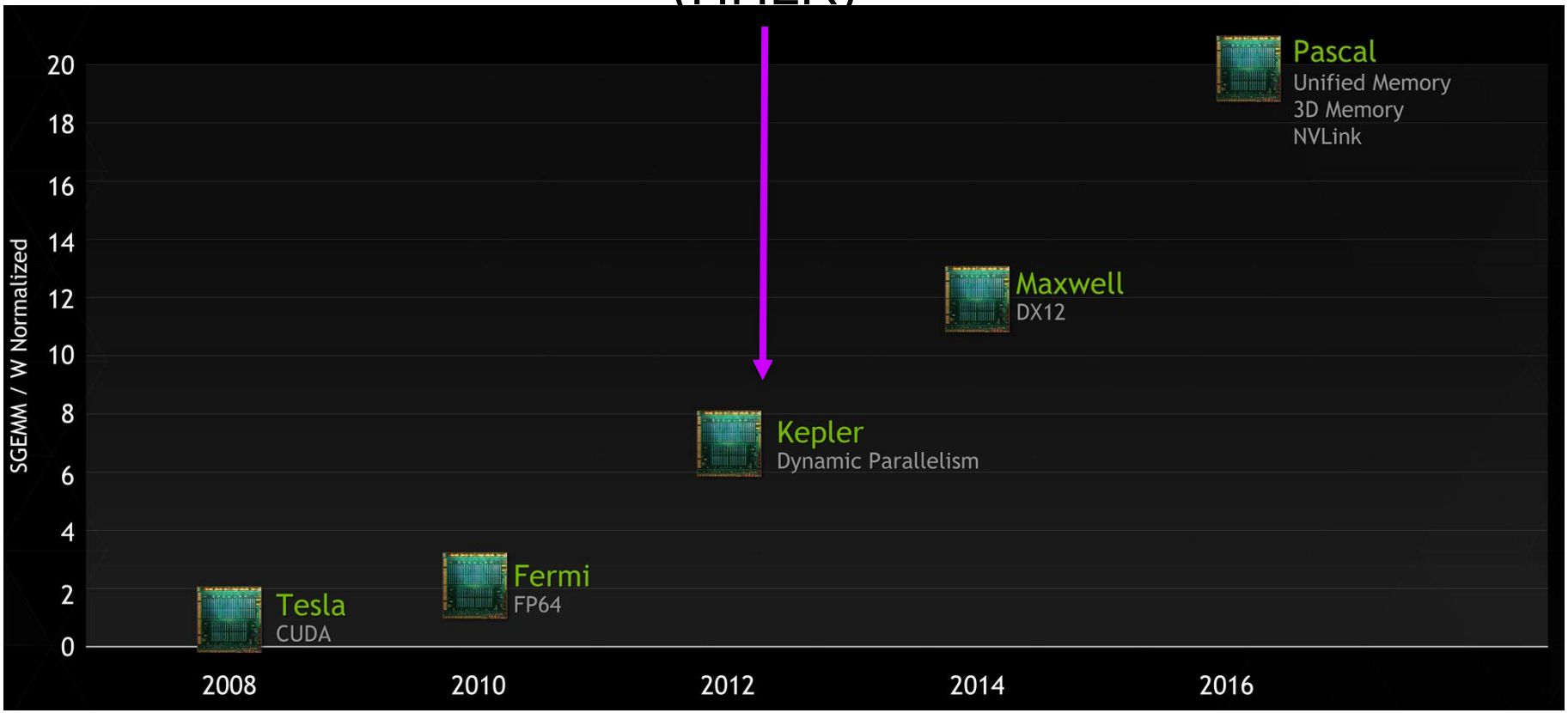


source: NVIDIA

GPU Performance Growth

TECHNISCHE UNIVERSITÄT DARMSTADT

Tesla K20x, K40m
(HHLR)



SGEMM - perform one of the matrix-matrix operations

C := alpha*op(A)*op(B) + beta*C

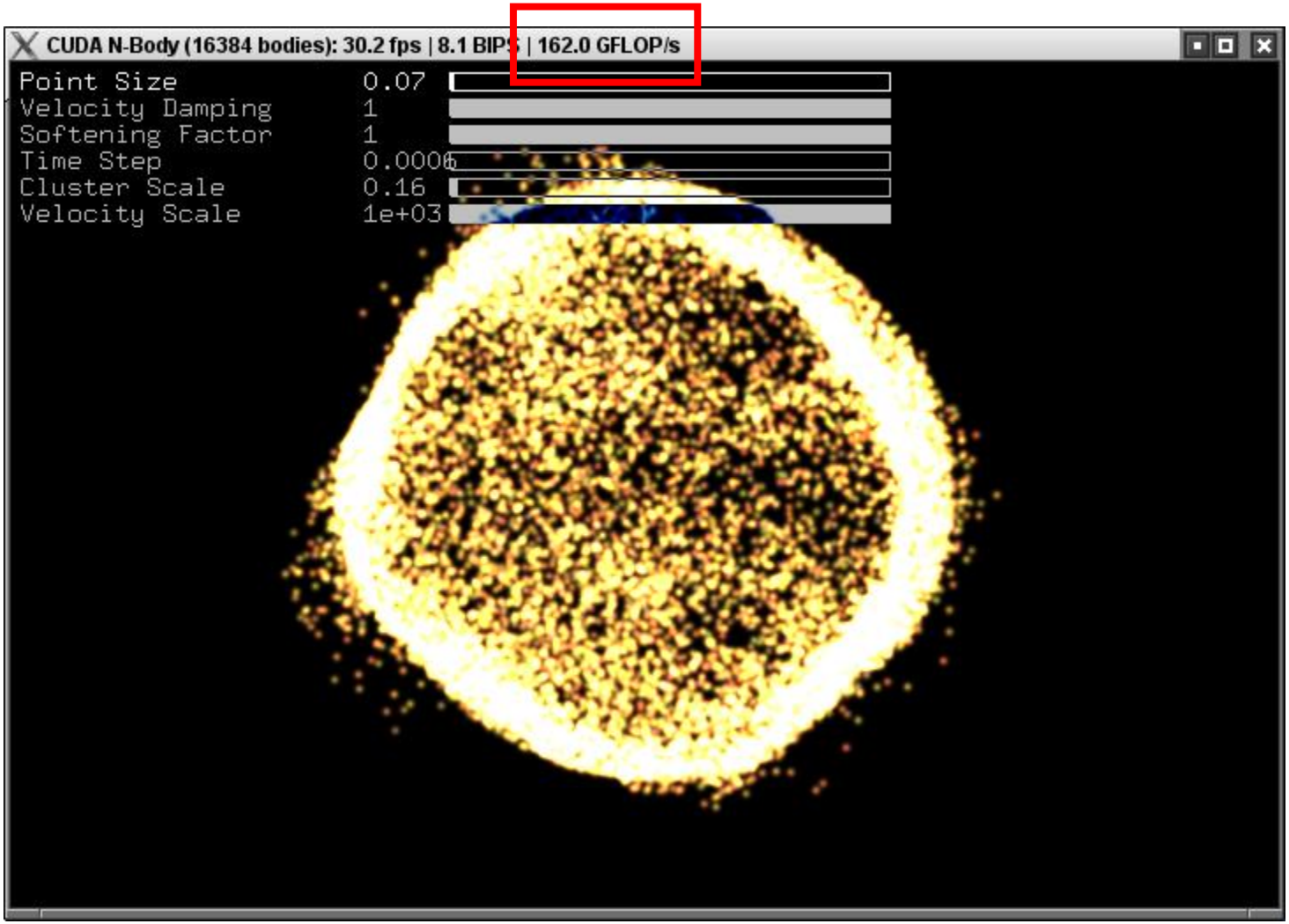
image source: NVIDIA

Maxwell

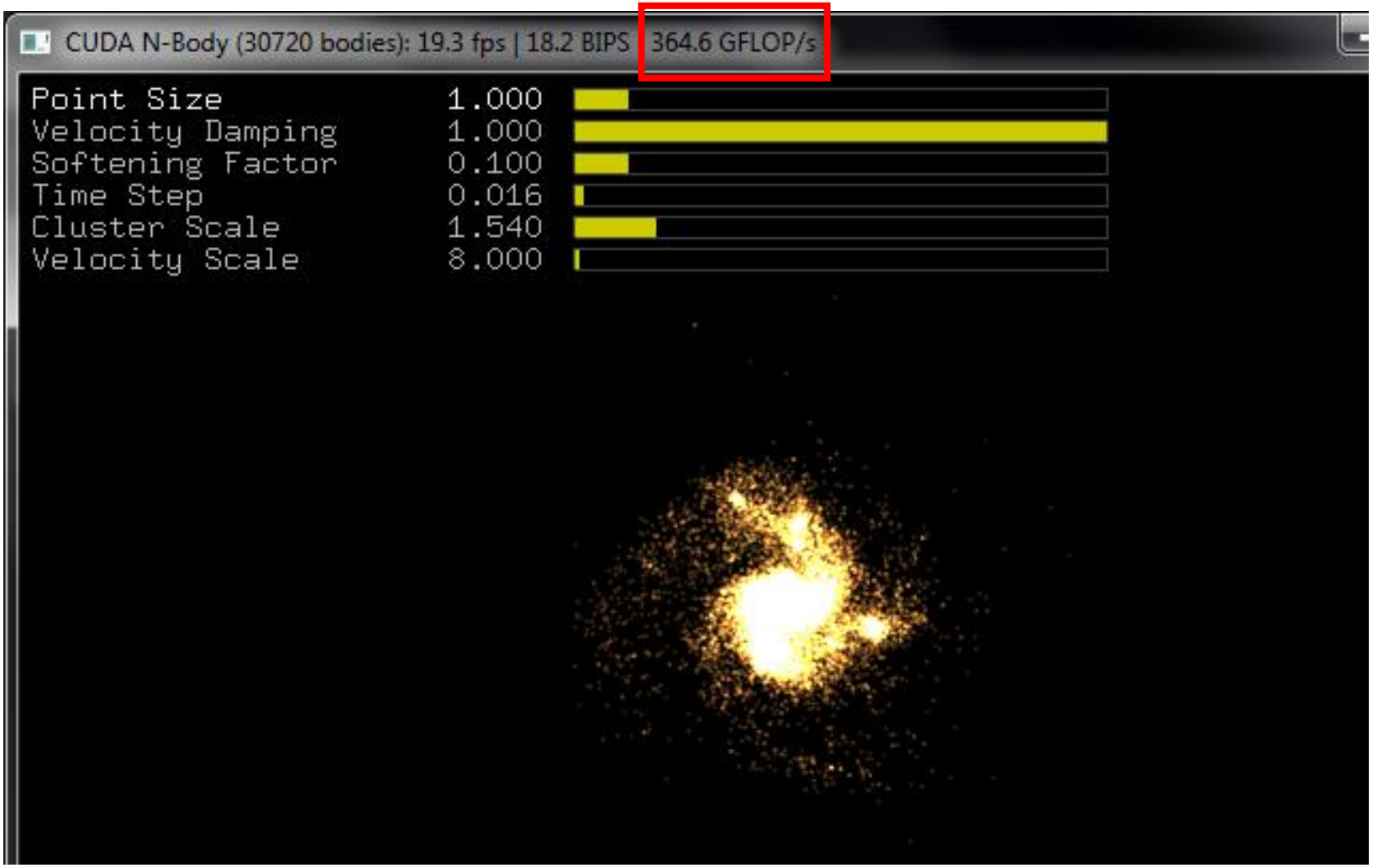
- GPU-Generation released September 2014
- New features for HPC
  - Improved power efficiency – est. 50% more GFlops / Watt
  - Redistribution of warp schedulers for fine-grained resource allocation
  - Compute Capability 5.2
  - 64 kB shared memory per multiprocessor
  - 34 kB cache for read-only and constant memory
- Might be added to HHLR in the future

source: NVIDIA

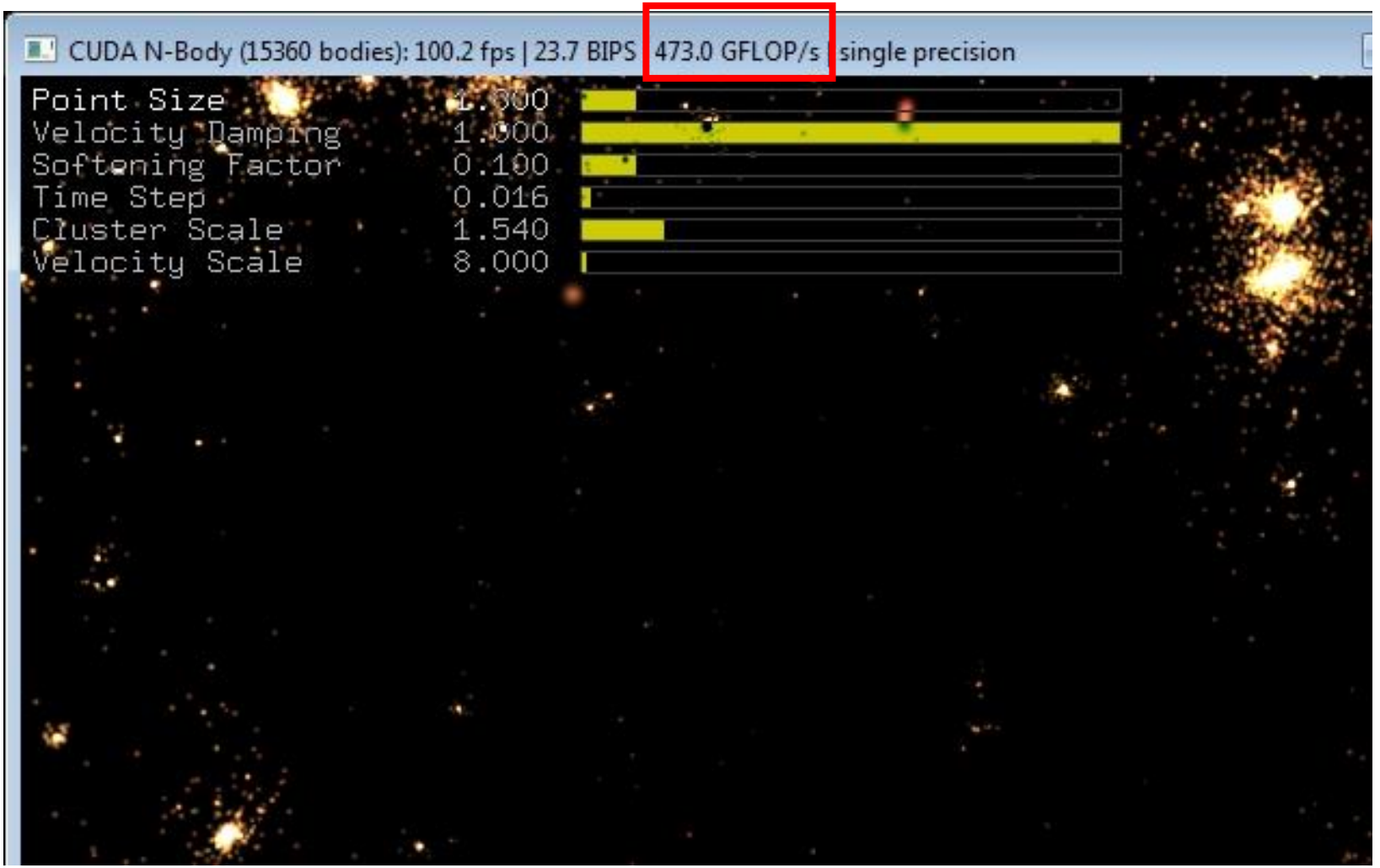
CUDA Example: N-Body Simulation



CUDA Example: N-Body Simulation

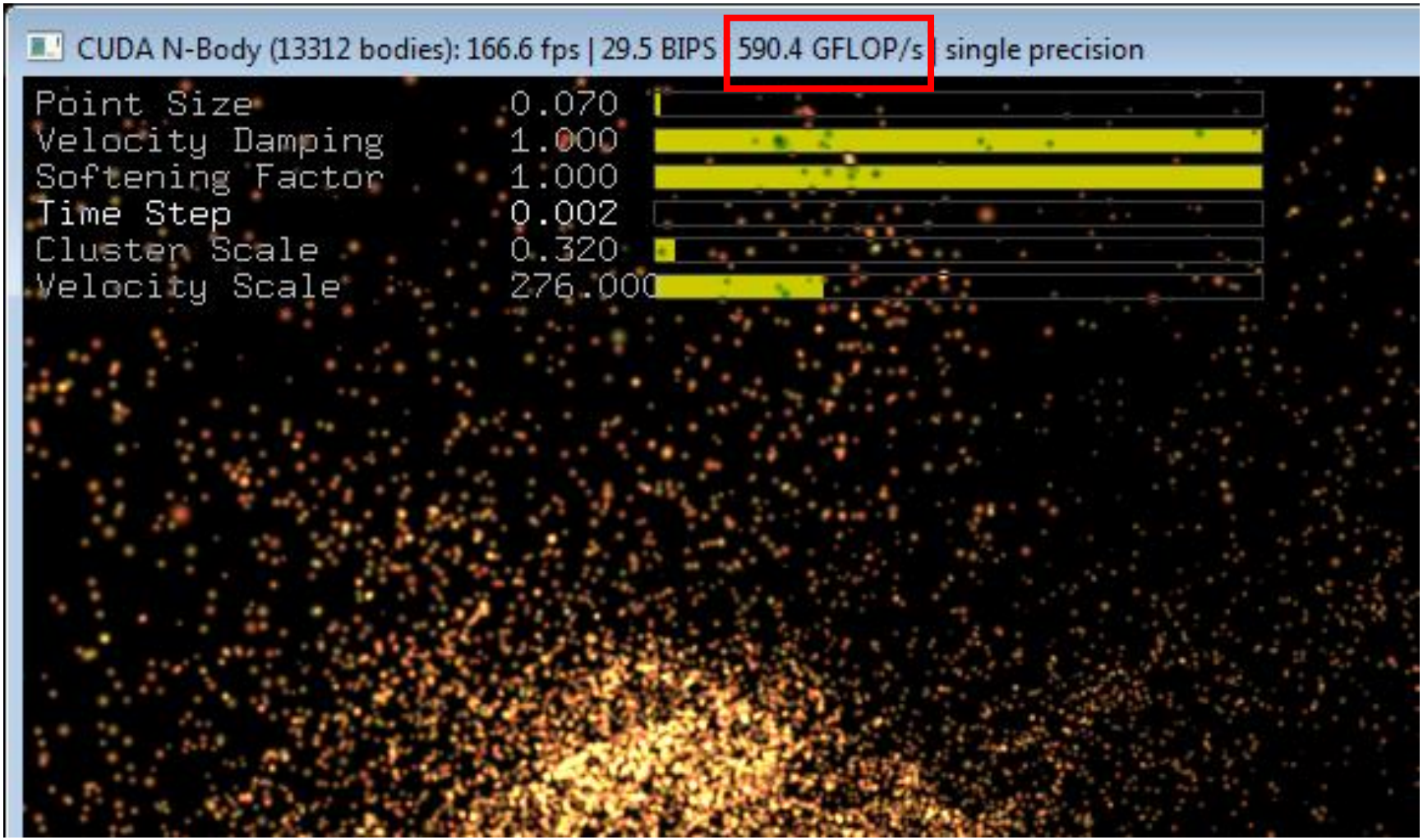


CUDA Example: N-Body Simulation



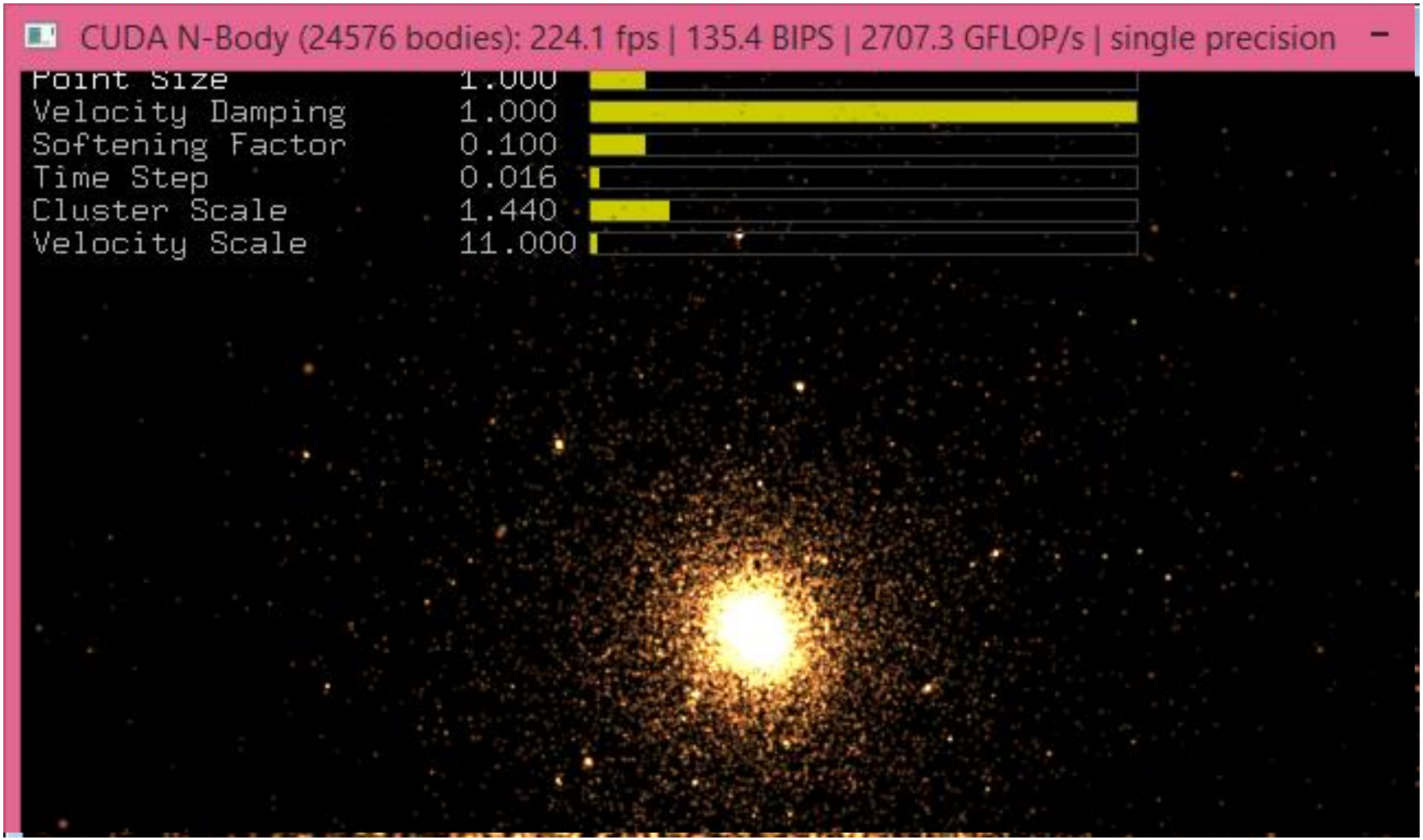
April 2011, GTX 480

CUDA Example: N-Body Simulation



April 2013, K20C

CUDA Example: N-Body Simulation



October 2015, Titan X

Faster is not “Just Faster”

TECHNISCHE UNIVERSITÄT DARMSTADT

- 2-3X faster is “just faster”
  - do a little more, wait a little less
  - doesn’t change how you work
- 5-10x faster is “significant”
  - worth upgrading
  - worth re-writing (parts of) the application
- 100x (or more!) faster is “fundamentally different”
  - worth considering a new platform
  - worth re-architecting the application
  - makes new applications possible
  - drives “time to discovery”
  - creates fundamental changes in science

source: NVIDIA

Debunking this Speed-Up?

- some discussions about achievable speed-up
  - depends a lot on the implementations used for CPU and GPU
  - raw floating point performance as a rough guideline
  - problem must be matched to architecture

Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU

Victor W Lee†, Changkyu Kim†, Jatin Chhugani†, Michael Deisher†, Daehyun Kim†, Anthony D. Nguyen†, Nadathur Satish†, Mikhail Smelyanskiy†, Srinivas Chennupaty*, Per Hammarlund*, Ronak Singhal* and Pradeep Dubey†

victor.w.lee@intel.com

†Throughput Computing Lab, Intel Corporation

*Intel Architecture Group, Intel Corporation

two decades earlier:

Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers. D. H. Bailey. Supercomputing Review, Aug. 1991, 54-55.

Course Goals

- efficient programming of massively parallel (MP) systems
  - naïve parallel approaches
  - advanced algorithms
  - using NVIDIA GPU with CUDA as example
- requires
  - architecture basics
  - parallel algorithms suitable for the hardware
  - practical programming experience
- understanding the (real) limitations