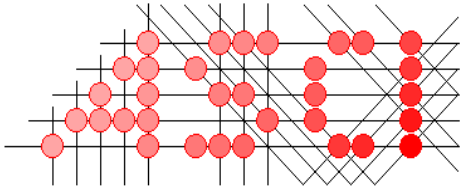
DAS-5: Harnessing the Diversity of Complex e-Infrastructures

Henri Bal

Vrije Universiteit Amsterdam

distributed
asci
supercomputer



Advanced School for Computing and Imaging

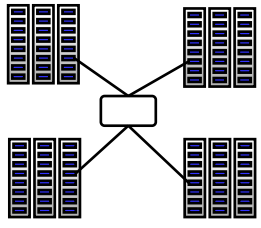
vrije Universiteit

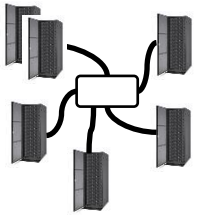
High-Performance Distributed Computing

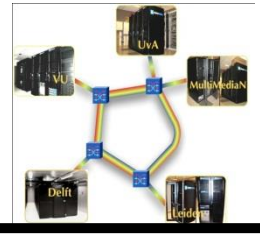
- **Distributed systems continue to change**
  - **Clusters, grids, clouds, heterogeneous systems**
- **Distributed applications continue to change**
- **Distributed programming continues to be notoriously difficult**
- **I build infrastructures and study programming systems and applications hand-in-hand**

My background

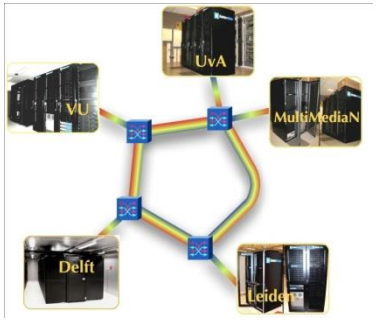
- Cluster computing
  - Zoo (1994), Orca
- Wide-area computing
  - DAS-1 (1997), Albatross
- Grid computing
  - DAS-2 (2002), Manta, Satin
- eScience & optical grids
  - DAS-3 (2006), Ibis
- Many-core accelerators
  - DAS-4 (2010), MCL, Glasswing
  - DAS-5 (2015), Cashmere







Agenda



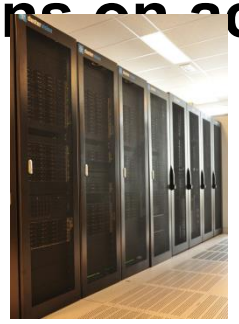
I. DAS (1997-2015)
- Unique aspects, the 5 DAS generations, organization
- Earlier results and impact [IEEE Computer 2015]

II. Current research on DAS studying *diversity*
- A methodology for programming many-core accelerators
- Big data (MapReduce) applications on accelerators


DAS-1

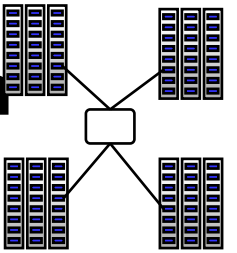

DAS-2


DAS-3


DAS-4


DAS-5

What is DAS?

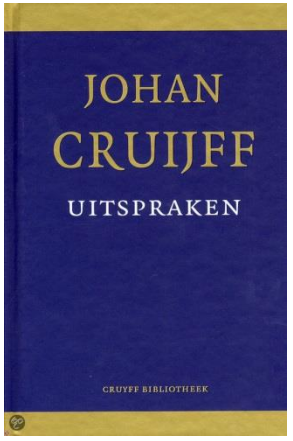


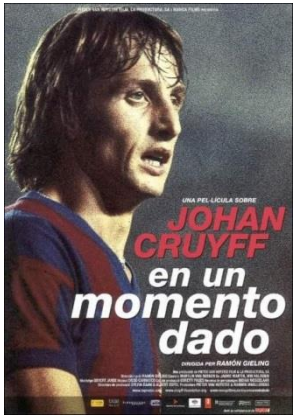
- Distributed common infrastructure for Dutch Computer Science
  - Distributed: multiple (4-6) clusters at different locations
  - Common: single formal owner (ASCI), single design team
    - Users have access to entire system
  - Dedicated to CS experiments (like Grid'5000)
    - Interactive (distri
    - Able to modify/br
  - Dutch: small scal

vrije Universiteit

About SIZE

- Only ~200 nodes in total per DAS generation
- Less than 1.5 M€ total funding per generation
- Johan Cruyff:
  - "Ieder nadeel heb zijn voordeel"
  - Every disadvantage has its advantage

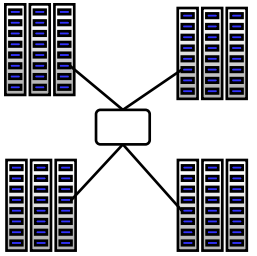




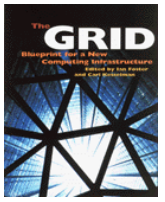
Small is beautiful

- **We have superior wide-area latencies**
  - **“The Netherlands is a 2×3 msec country” (Cees de Laat, Univ. of Amsterdam)**
- **Able to build each DAS generation from scratch**
  - **Coherent distributed system with clear vision**
- **Despite the small scale we achieved:**
  - **3 CCGrid SCALE awards, numerous TRECVID awards**
  - **>100 completed PhD theses**

DAS generations: visions

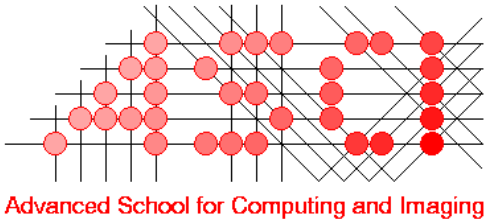


- DAS-1: Wide-area computing (1997)
  - Homogeneous hardware and software
- DAS-2: Grid computing (2002)
  - Globus middleware



- DAS-3: Optical Grids (2006)
  - Dedicated 10 Gb/s optical links between all sites
- DAS-4: Clouds, diversity, green IT (2010)
  - Hardware virtualization, accelerators, energy measurements
- DAS-5: Harnessing diversity, data-explosion (June 2015)
  - Wide variety of accelerators, larger memories and disks

vrije Universiteit



ASCI (1995)

- Research schools (Dutch product from 1990s), aims:
  - Stimulate top research & collaboration
  - Provide Ph.D. education (courses)
- ASCI: Advanced School for Computing and Imaging
  - About 100 staff & 100 Ph.D. Students
  - 16 PhD level courses
  - Annual conference

Organization

- ASCI steering committee for overall design
  - Chaired by Andy Tanenbaum (DAS-1) and Henri Bal (DAS-2 – DAS-5)
  - Representatives from all sites: Dick Epema, Cees de Laat, Cees Snoek, Frank Seinstra, John Romein, Harry Wijshoff
- Small system administration group coordinated by VU (Kees Verstoep)
  - Simple homogeneous setup reduces admin overhead

vrije Universiteit

TU Delft Delft University of Technology

MultimediaN

vrije Universiteit

Universiteit van Amsterdam

Universiteit Leiden

ASTRON

Historical example (DAS-1)

- **Change OS globally from BSDI Unix to Linux**
  - **Under directorship of Andy Tanenbaum**

The New York Times

World

Belgian King, Unable to Sign Abortion Law, Takes Day Off

By PAUL L. MONTGOMERY, Special to The New York Times
Published: April 05, 1990

The civilian Government temporarily suspended King Baudouin I from power today after he declared that he could not, in good conscience as a Roman Catholic, sign a new law permitting abortion.

After declaring the King "unable to govern," the Cabinet assumed the King's powers and promulgated the abortion law, which was published in the official gazette. And it called the Chamber of Deputies and the Senate back from Easter vacation for a special session on Thursday.

The lawmakers will be asked to vote on the proposition that the 59-year-old King is once again able to govern. Approval is expected.

Seen as Acceptable Way Out

There was uproar in some political groups that have traditionally been opposed to the monarchy. But television and press comment tonight generally supported the apparent solution to the King's problem as an acceptable outcome.

Belgium has 25 political parties divided among speakers of French and Flemish and four self-governing regions. Solutions that give each participant a taste of victory in complicated circumstances are called " a la Belge" - in the Belgian manner.

Belgium, a country of 10 million people, sometimes puts itself forward at international gatherings as a good model to follow when big powers and unyielding ideologies cannot find common ground.

Today's events were the first time since the creation of Belgium and its constitutional monarchy in 1830 that the special powers of the Government had been used in such a way.

Netherlands Organisation for Scientific Research

Financing

- NWO ``middle-sized equipment'' program
  - Max 1 M€, very tough competition, but scored 5-out-of-5
  - 25% matching by participating sites

    - Going Dutch for ¼ th
- Extra funding by VU and (DAS-5) COMMIT + NLeSC
- SURFnet (GigaPort) provides wide-area networks

GigaPort SURF NET netherlands eScience center COMMIT/

vrije Universiteit

Steering Committee algorithm

FOR i **IN** 1 .. 5 **DO**

Develop vision for DAS[i]
NWO/M proposal by 1 September [4 months]
Receive outcome (accept) [6 months]
Detailed spec / EU tender [4-6 months]
Selection; order system; delivery [6 months]

Research_system := DAS[i];
Education_system := DAS[i-1] (if i>1)
Throw away DAS[i-2] (if i>2)
Wait (2 or 3 years)

DONE

vrije Universiteit

Output of the algorithm

	DAS-1	DAS-2	DAS-3	DAS-4	DAS-5
CPU	Pentium Pro	Dual Pentium3	Dual Opteron	Dual 4-core Xeon	Dual 8-core Xeon
LAN	Myrinet	Myrinet	Myrinet 10G	QDR Infini Band	FDR Infini Band
# CPU cores	200	400	792	1600	3168
1-way latency MPI (μs)	21.7	11.2	2.7	1.9	1.2
Max. throughput (MB/s)	75	160	950	2700	6000
Wide-area network	6 Mb/s ATM	1 Gb/s Internet	10 Gb/s Light paths	10 Gb/s Light paths	10 Gb/s Light paths

Earlier results

- **VU: programming distributed systems**
  - **Clusters, wide area, grid, optical, cloud, accelerators**
- **Delft: resource management [CCGrid'2012 keynote]**
- **MultimediaN: multimedia knowledge discovery**
- **Amsterdam: wide-area networking, clouds, energy**
- **Leiden: data mining, astrophysics [CCGrid'2013 keynote]**
- **ASTRON: accelerators, signal processing**

vrije Universiteit

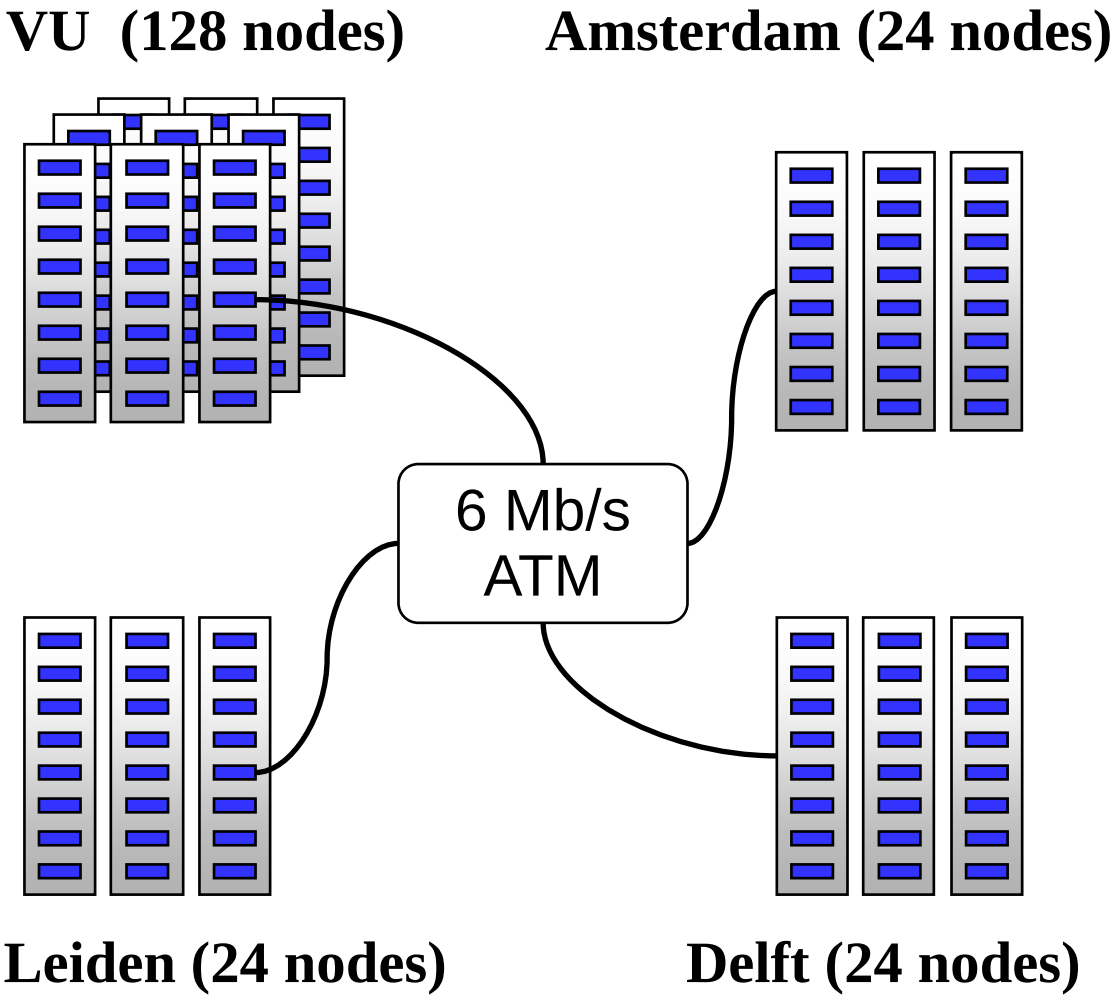
Universiteit van Amsterdam

Universiteit Leiden

ASTRON

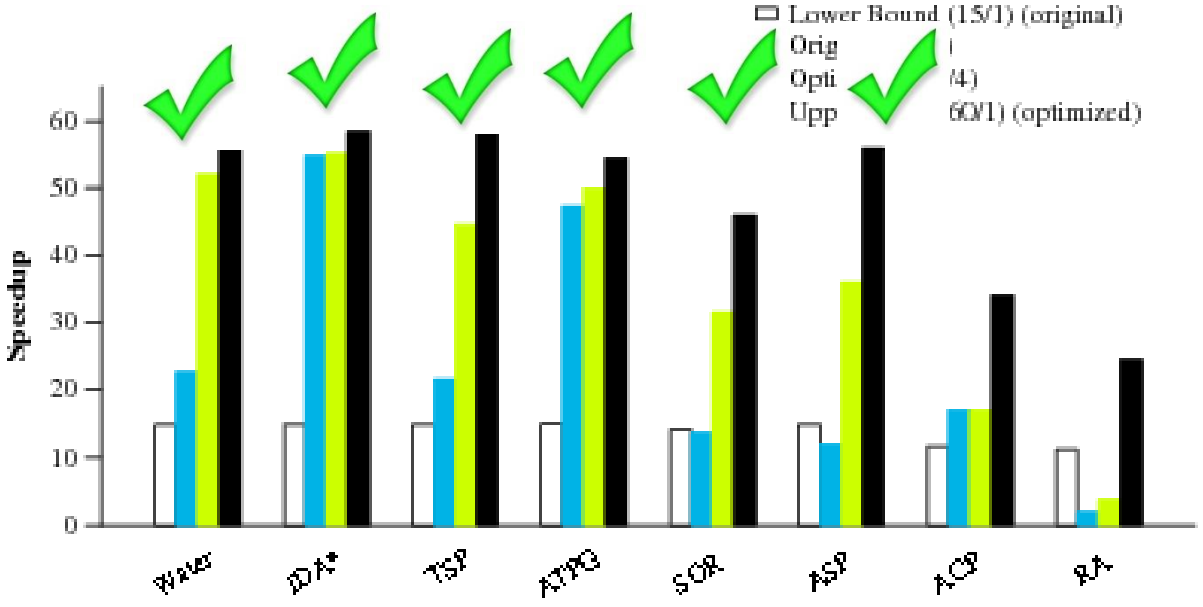
DAS-1 (1997-2002)
A homogeneous wide-area system

200 MHz Pentium Pro
Myrinet interconnect
BSDI → Redhat Linux
Built by Parsytec



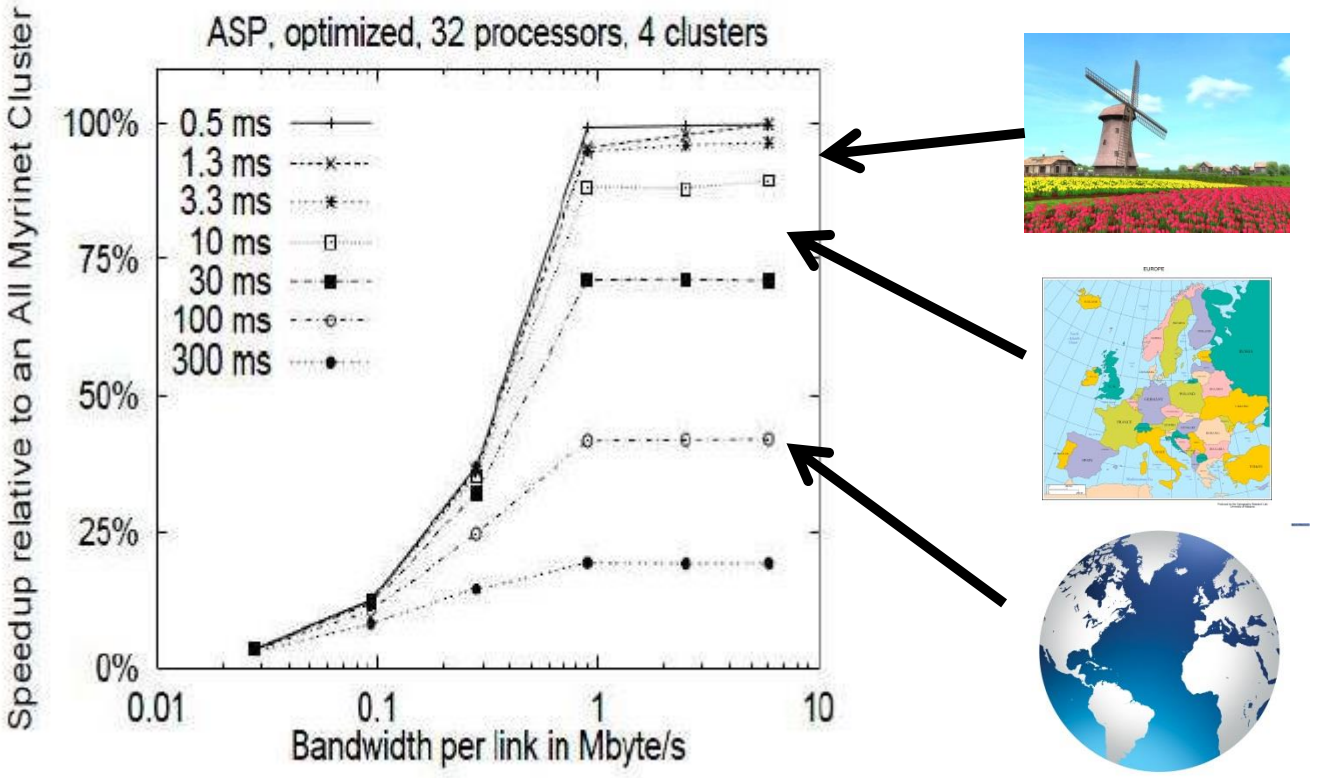
Albatross project

- Optimize algorithms for wide-area systems
  - Exploit hierarchical structure ➔ locality optimizations
- Compare:
  - 1 small cluster (15 nodes)
  - 1 big cluster (60 nodes)
  - wide-area system (4×15 nodes)



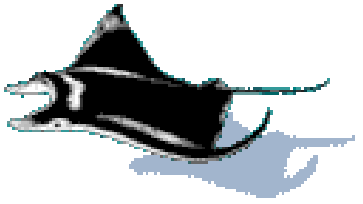
Sensitivity to wide-area latency and bandwidth

- **Used local ATM links + delay loops to simulate various latencies and bandwidths [HPCA'99]**



Wide-area programming systems

- **Manta:**
  - **High-performance Java [TOPLAS 2001]**

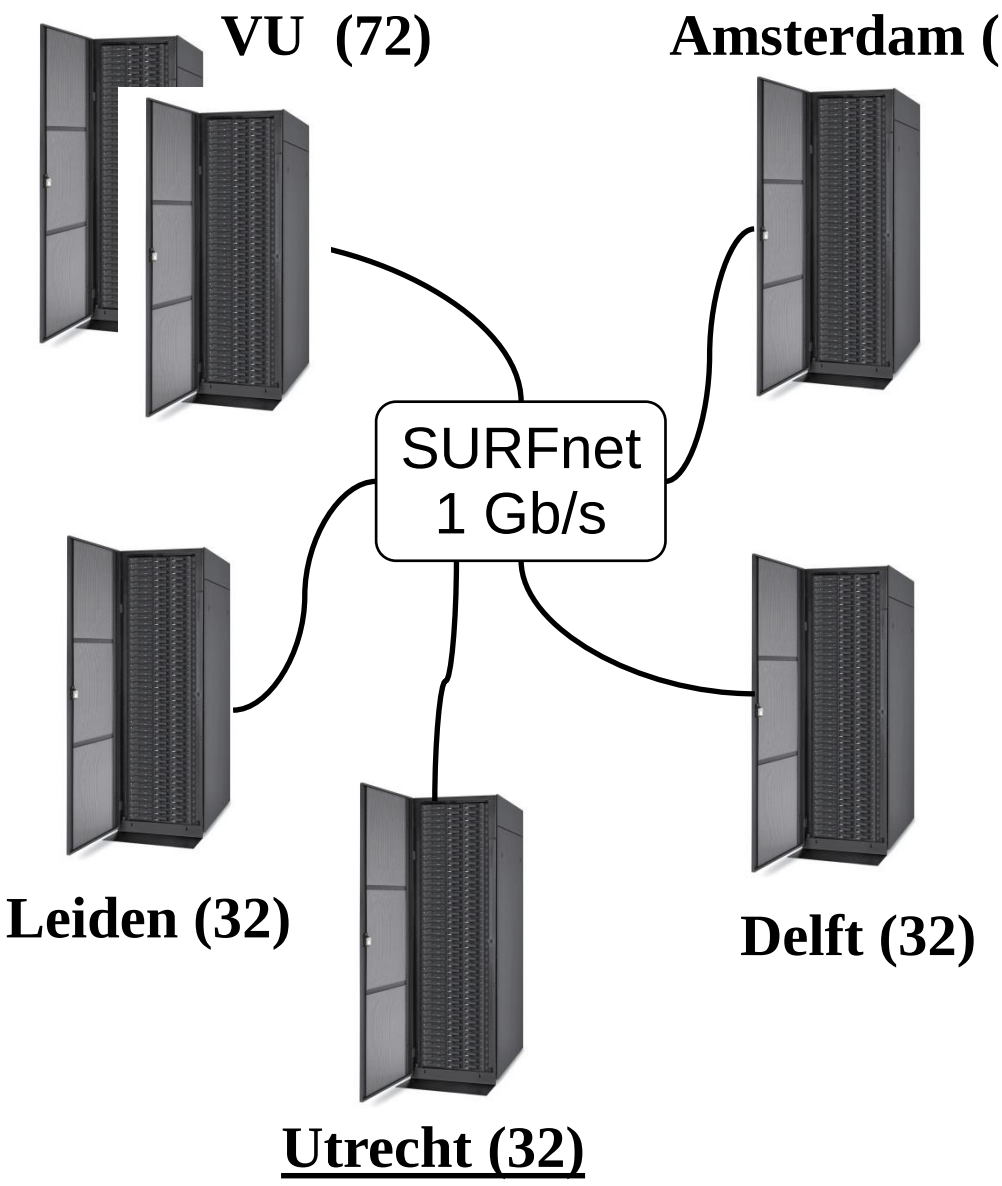


- **MagPIe (Thilo Kielmann):**
  - **MPI's collective operations optimized for hierarchical wide-area systems [PPoPP'99]**

- **KOALA (TU Delft):**
  - **Multi-cluster scheduler with support for co-allocation**

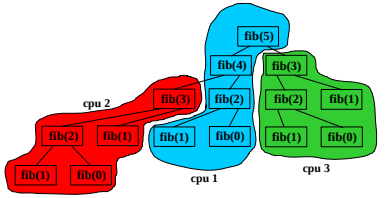
DAS-2 (2002-2006)
a Computer Science Grid

two 1 GHz Pentium-3s
Myrinet interconnect
Redhat Enterprise Linux
Globus 3.2
PBS → Sun Grid Engine
Built by IBM



vrije Universiteit

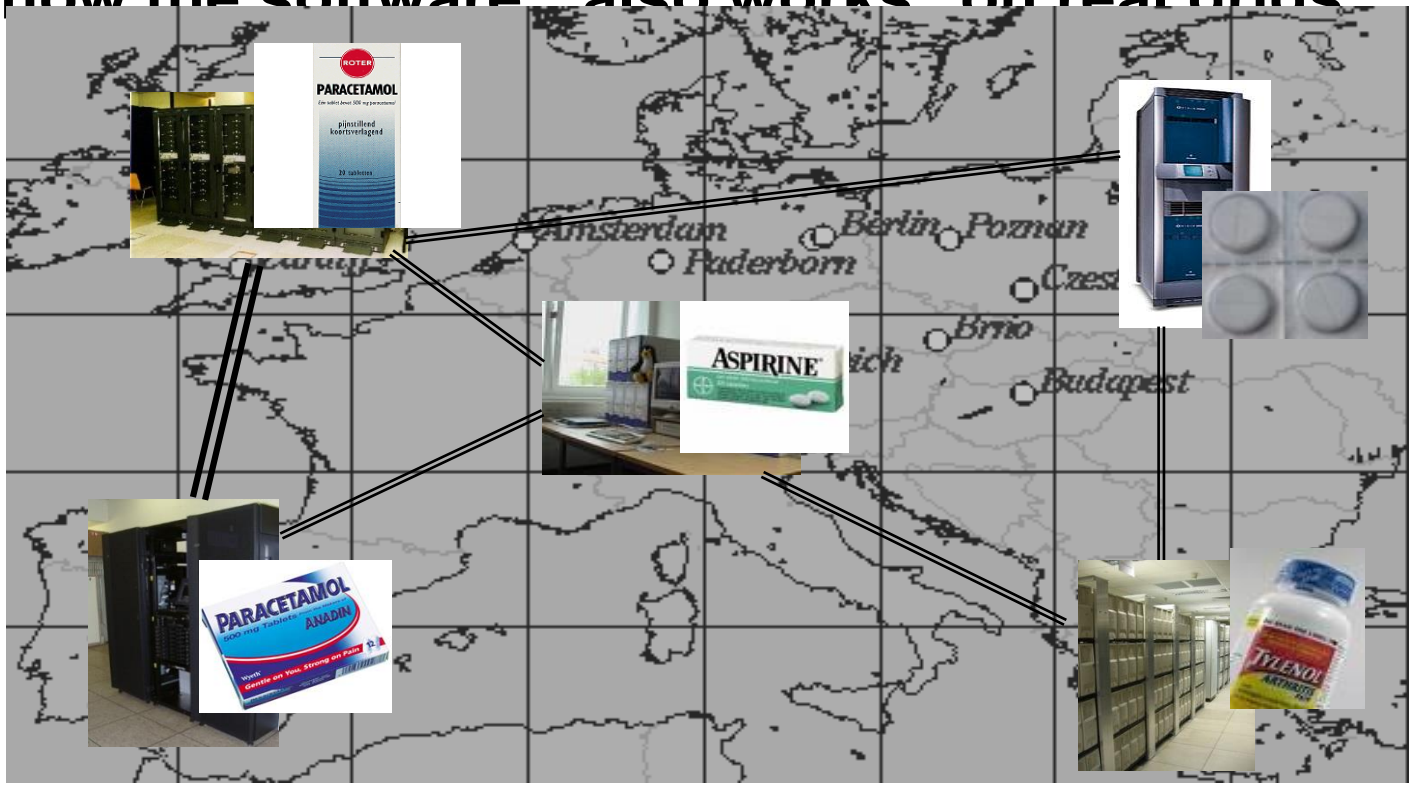
Grid programming systems



- **Satin (Rob van Nieuwpoort):**
  - **Transparent divide-and-conquer parallelism for grids**
    - **Hierarchical computational model fits grids [TOPLAS 2010]**
- **Ibis: Java-centric grid computing [IEEE Computer 2010]**
  - **Efficient and transparent ``Jungle computing''**
- **JavaGAT:**
  - **Middleware-independent API for grid applications [SC'07]**

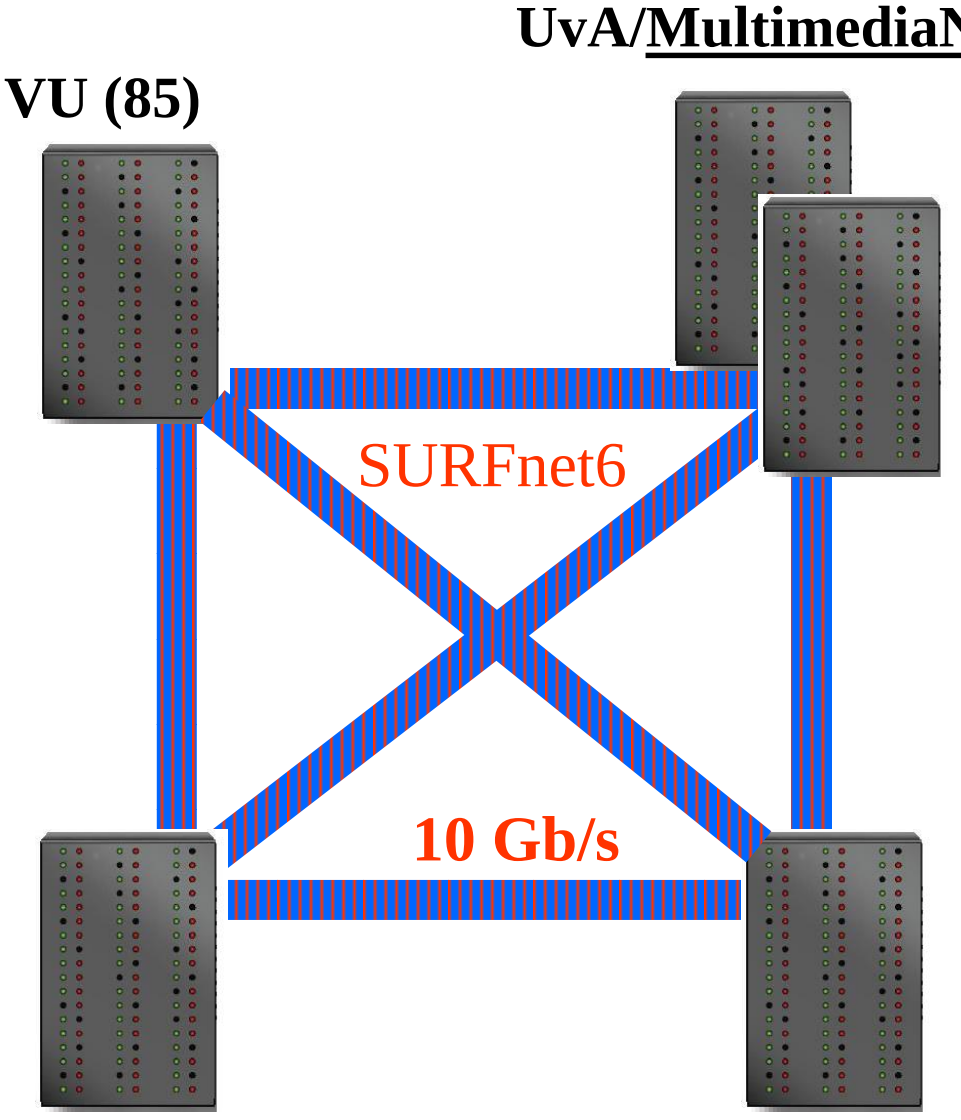
Grid experiments

- Do clean performance measurements on DAS
- Combine DAS with EU grids to test heterogeneity
  - Show the software ``also works'' on real grids



DAS-3 (2006-2010)
An optical grid

- Dual AMD Opterons
- 2.2-2.6 GHz
  - Single/dual core nodes
- Myrinet-10G
- Scientific Linux 4
- Globus, SGE
- Built by ClusterVision

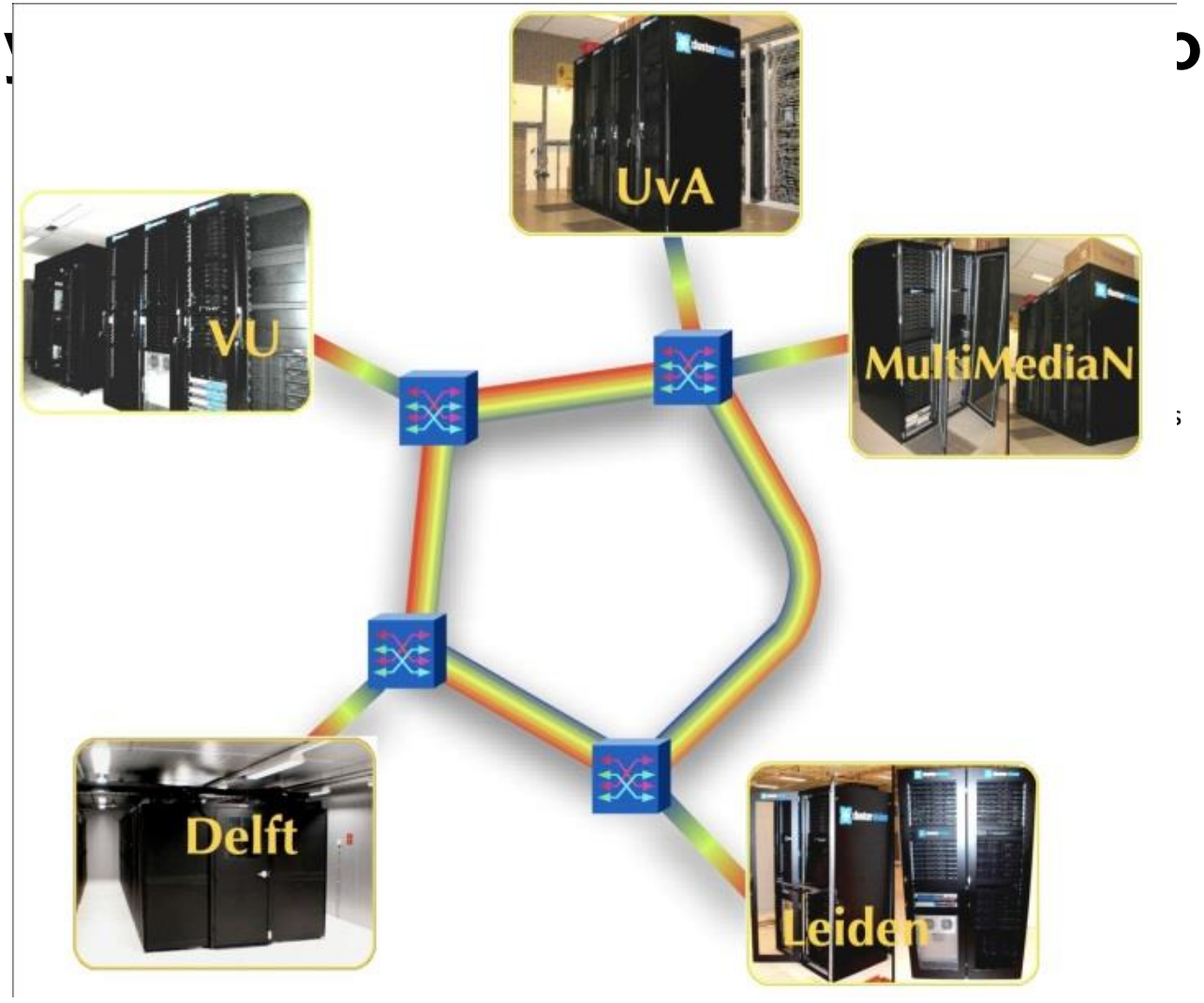


vrije Universiteit

GigaPort

StarPlane

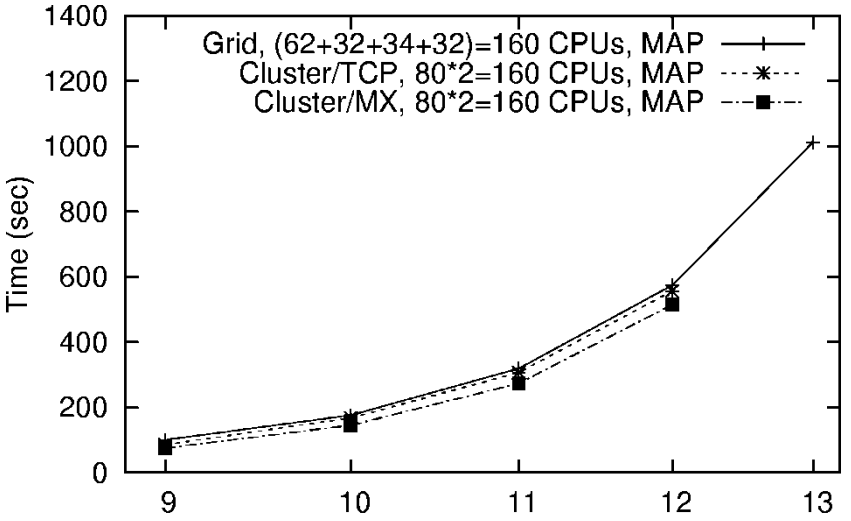
- Multiple dedicated 10G light paths between sites
- Idea: dy ogy

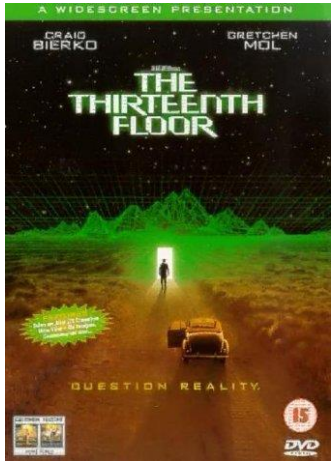


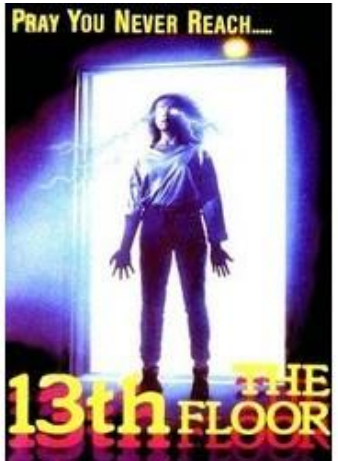
vrije Universiteit

Distributed Model Checking

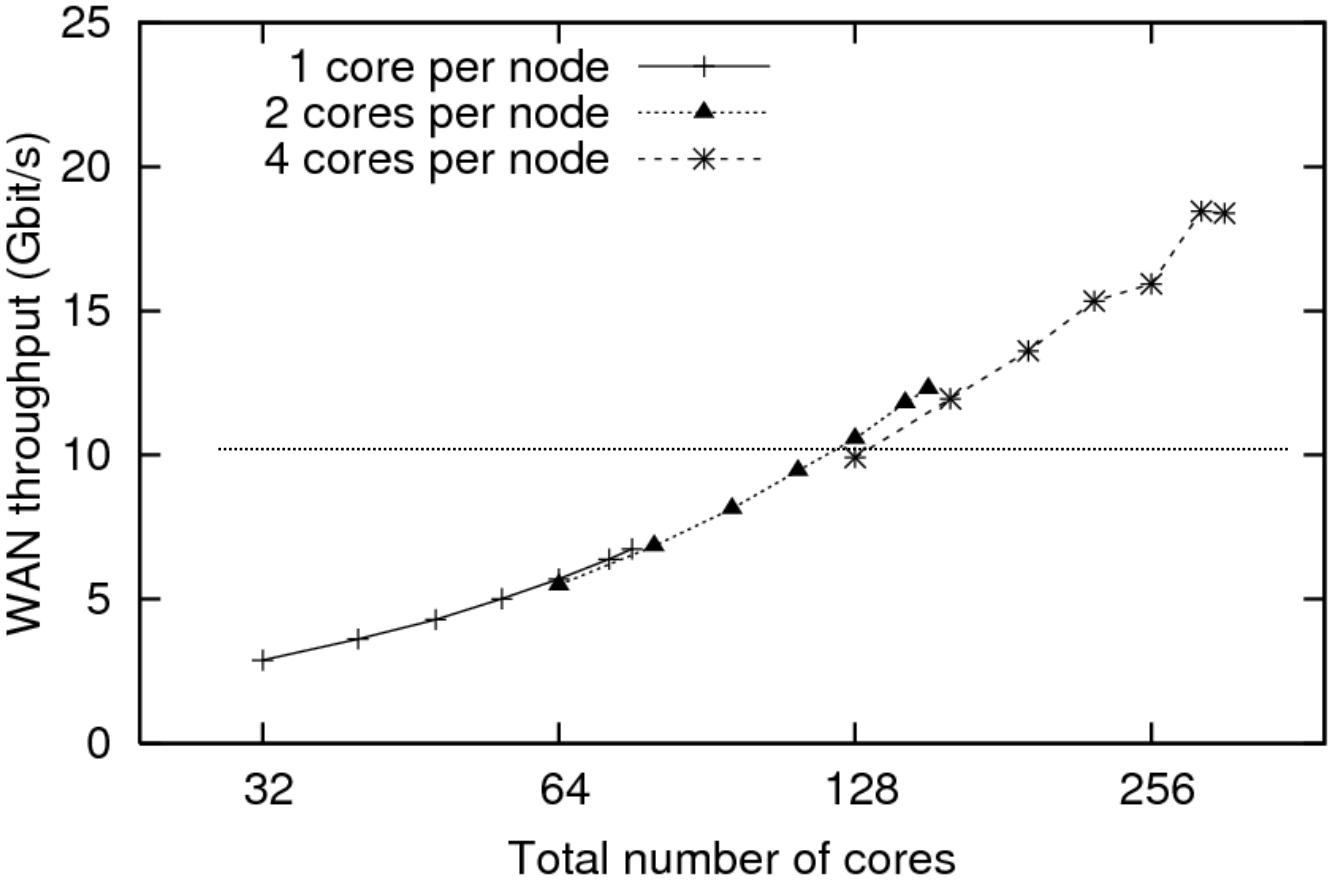
- **Huge state spaces, bulk asynchronous transfers**
- **Can efficiently run DiVinE model checker on wide-area DAS-3, use up to 1 TB memory [IPDPS'09]**

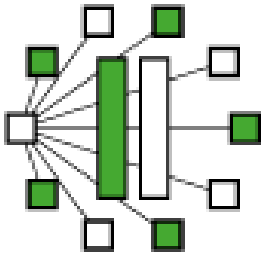






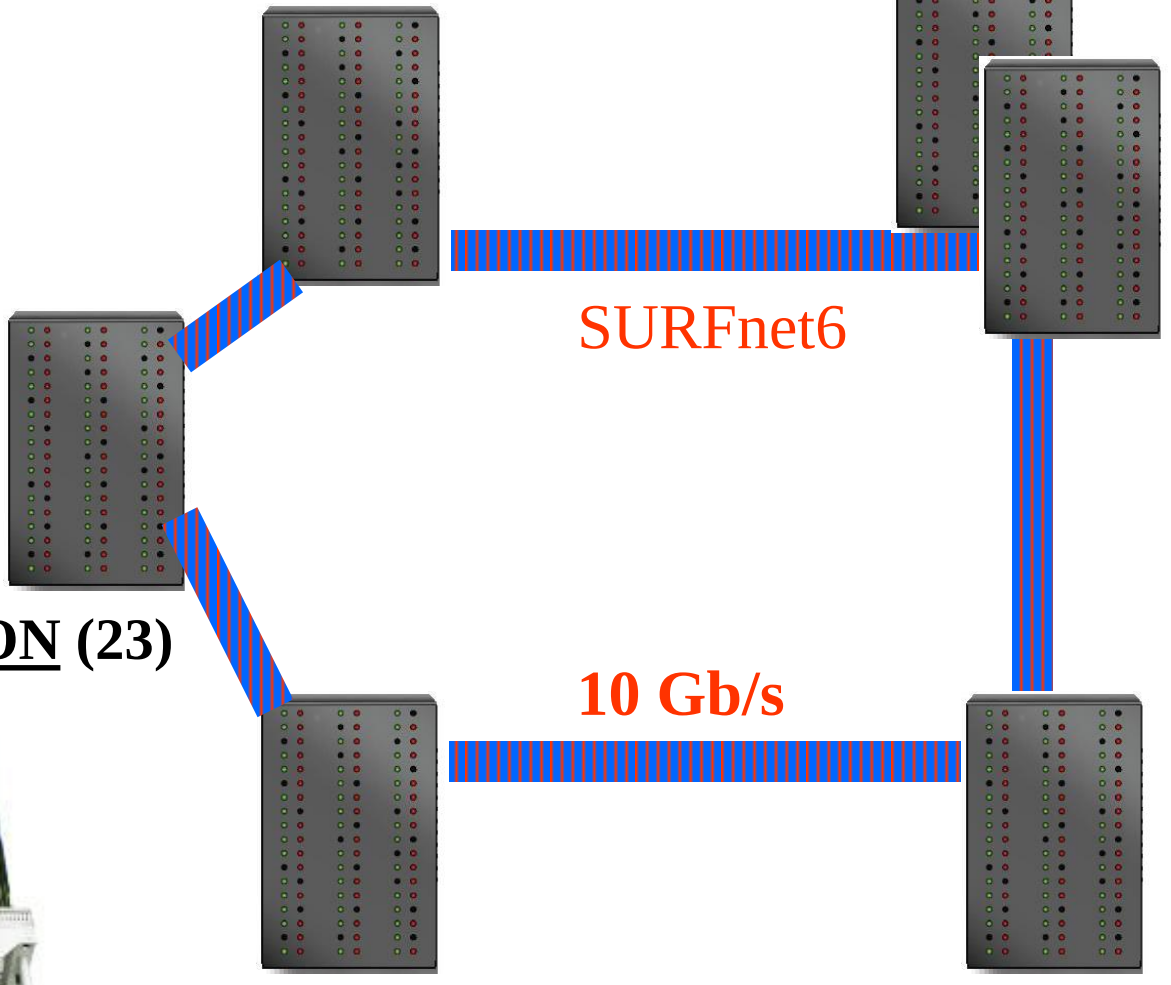
Required wide-area bandwidth



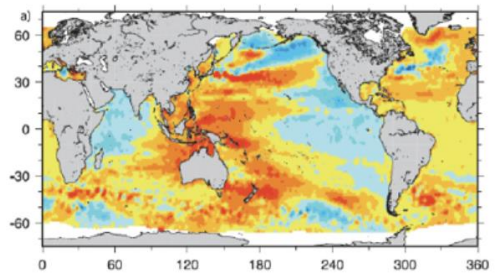


DAS-4 (2011) Testbed for Clouds, diversity, green IT

Dual quad-core Xeon E5620
Infiniband
Various accelerators
Scientific Linux
Bright Cluster Manager
Built by ClusterVision



vrije Universiteit



Global Climate Modeling

- Understand future local sea level changes
- Needs high-resolution simulations
- Combine two approaches:
  - Distributed computing (multiple resources)
  - GPUs

netherlands eScience center

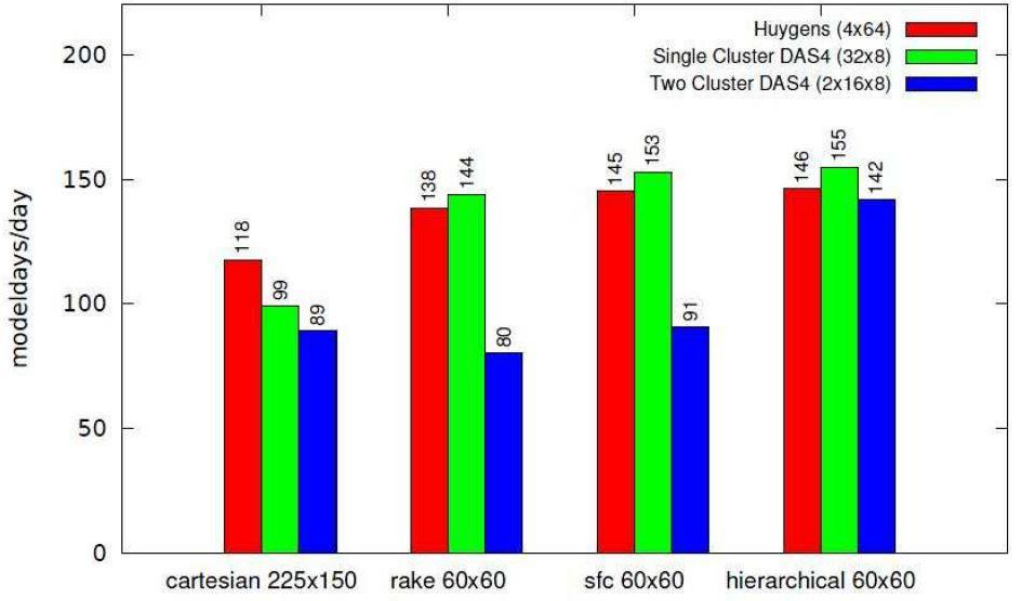
IMAU Institute for Marine and Atmospheric research Utrecht

COMMIT/

vrije Universiteit

Distributed Computing

- **Use Ibis to couple different simulation models**
  - **Land, ice, ocean, atmosphere**
- **Wide-area optimizations similar to Albatross project (16 years ago), like hierarchical load balancing**



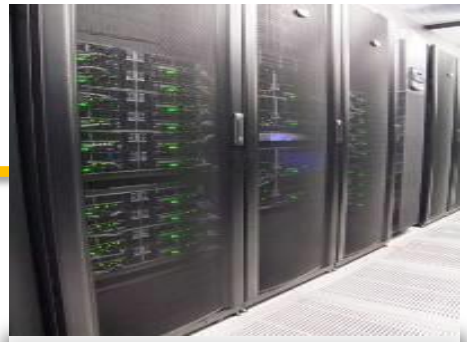
Enlighten Your Research Global award

#7 STAMPEDE (USA)

10G

CARTESIUS (NLD)

10G



EMERALD (UK)

KRAKEN (USA)

10G

SUPERMUC (GER) #10

IMAU Institute for Marine and Atmospheric research Utrecht

LMU Ludwig-Maximilians-Universität München

OXFORD e-Research CENTRE

UNIVERSITY OF OXFORD

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

RUTGERS

RSMAS ROSENSTIEL SCHOOL OF MARINE AND ATMOSPHERIC SCIENCE UNIVERSITY OF MIAMI

VU UNIVERSITY AMSTERDAM

UNIVERSITY OF MIAMI

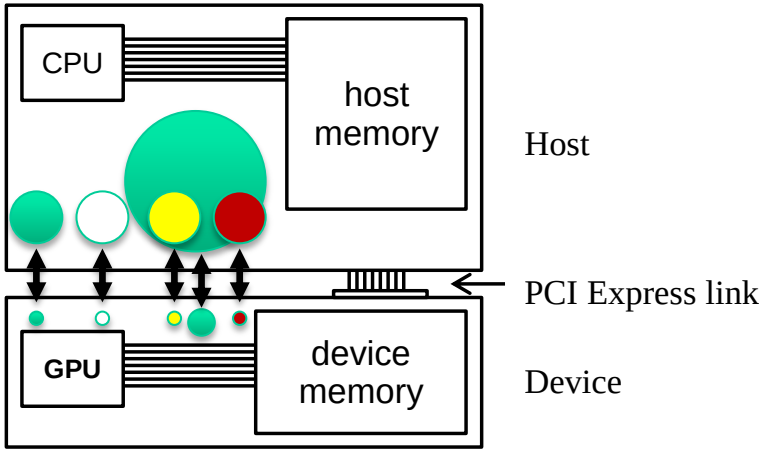
Universiteit Utrecht

netherlands eScience center

vrije Universiteit

GPU Computing

- Offload expensive kernels for Parallel Ocean Program (POP) from CPU to GPU
  - Many different kernels, fairly easy to port to GPUs
  - Execution time becomes virtually 0
- New bottleneck: moving data between CPU & GPU



Problem

- **Problem:**
  - **Which method will be most efficient for a given GPU kernel? Implementing all can be a large effort**
- **Solution:**
  - **Created a performance model that identifies the best implementation:**
    - **What implementation strategy for overlapping computation and communication is best for my program?**

Ben van Werkhoven, Jason Maassen, Frank Seinstra & Henri Bal: Performance models for CPU-GPU data transfers, CCGrid2014

DAS-5 (June 2015): Harnessing diversity, data-explosion

- Same 5 sites as DAS-4
- Two new (co-funding) participants
  - Netherlands e-Science Center (NLeSC)
  - COMMIT/ : a public-private research community

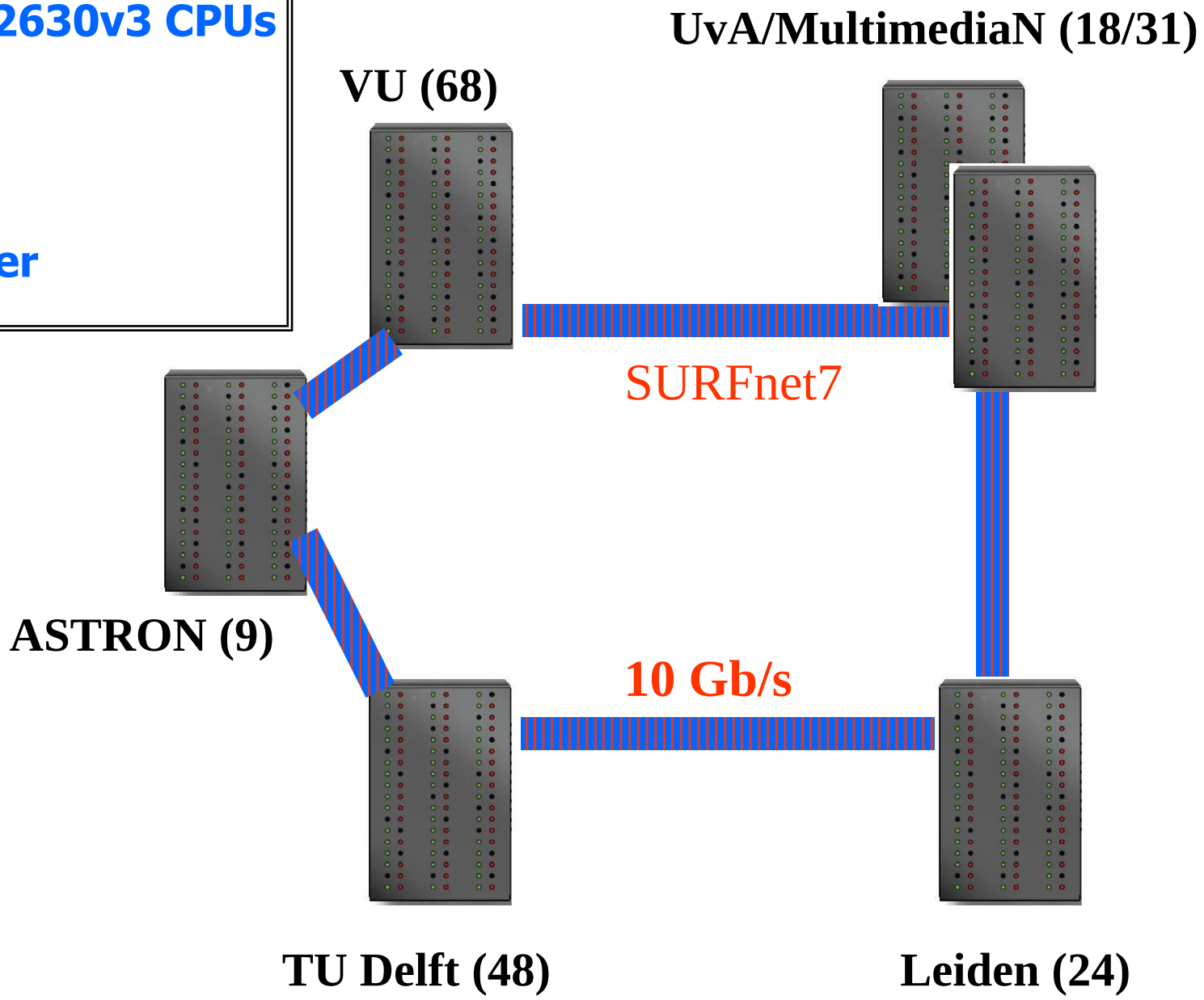
netherlands eScience center

COMMIT/

das 5

DAS-5

- Dual 8-core Intel E5-2630v3 CPUs
- FDR InfiniBand
- OpenFlow switches
- Various accelerators
- CentOS Linux
- Bright Cluster Manager
- Built by ClusterVision



Highlights of DAS users

- **Awards**
- **Grants**
- **Top-publications**
- **New user-communities**

Awards

- 3 CCGrid SCALE awards
  - 2008: Ibis: ``Grids as promised''
  - 2010: WebPIE: web-scale reasoning
  - 2014: BitTorrent analysis

TU Delft Delft University of Technology

- Video and image retrieval

Multimedia N

Universiteit van Amsterdam

  - 5 TRECVID awards, ImageCLEF, ImageNet, Pascal VOC classification, AAAI 2007 most visionary research award

- Euro-Par 2014 achievement award

vrije Universiteit

Grants

- **Externally funded PhD/postdoc projects using DAS:**

DAS-3 proposal	20
DAS-4 proposal	30
DAS-5 proposal	50

- **Major incentive for VL-e (20 M€ funding)**
  - **Virtual Laboratory for e-Science (2003-2009)**
- **Strong participation in COMMIT/**
- **100 completed PhD theses**

Recent DAS papers

- Prebaked uVMs: Scalable, Instant VM Startup for IaaS Clouds *(ICDCS'15)*
- Cashmere: Heterogeneous Many-Core Computing *(IPDPS'15)*
- Glasswing: MapReduce on Accelerators *(HPDC'14 / SC'14)*
- Performance models for CPU-GPU data transfers *(CCGrid'14)*
- Auto-Tuning Dedispersion for Many-Core Accelerators *(IPDPS'14)*
- How Well do Graph-Processing Platforms Perform? *(IPDPS'14)*
- Balanced resource allocations across multiple dynamic MapReduce clusters *(SIGMETRICS '14)*
- Squirrel: Virtual Machine Deployment *(SC'13 + HPDC'14)*

Top papers

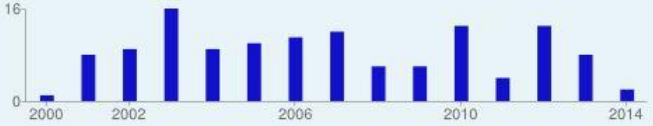
IEEE Computer	5
Comm. ACM	2
IEEE TPDS	7
ACM TOPLAS	3
ACM TOCS	4
Nature	2

SIGOPS 2000 paper

« Back to list | Edit | Export | Delete

Title	The distributed ASCI supercomputer project [PDF] from vu.nl
Authors	Henri Bal, Raoul Bhoedjang, Rutger Hofman, Ceriel Jacobs, Thilo Kielmann, Jason Maassen, Rob Van Nieuwpoort, John Romein, Luc Renambot, Tim Rühl, Ronald Veldema, Kees Verstoep, Aline Baggio, Gerco Ballintijn, Ihor Kuz, Guillaume Pierre, Maarten Van Steen, Andy Tanenbaum, Gerben Doornbos, Desmond Germans, Hans Spoelder, Evert-Jan Baerends, Stan Van Gisbergen, Hamideh Afsermanesh, Dick Van Albada, Adam Belloum, David Dubbeldam, Zeger Hendrikse, Bob Hertzberger, Alfons Hoekstra, Kamil Iskra, Drona Kandhai, Dennis Koelma, Frank Van Der Linden, Benno Overeinder, Peter Sloot, Piero Spinnato, Dick Epema, Arjan Van Gemund, Pieter Jonker, Andrei Radulescu, Cees Van Reeuwijk, Henk Sips, Peter Knijnenburg, Michael Lew, Floris Sluiter, Lex Wolters, Hans Blom, Cees de Laat, Aad van der Steen
Publication date	2000/10/1
Journal name	ACM SIGOPS Operating Systems Review
Volume	34
Issue	4
Pages	76-96
Publisher	ACM
Description	Abstract The Distributed ASCI Supercomputer (DAS) is a homogeneous wide-area distributed system consisting of four cluster computers at different locations. DAS has been used for research on communication software, parallel languages and programming systems, schedulers, parallel applications, and distributed applications. The paper gives a preview of the most interesting research results obtained so far in the DAS project.

50 authors

Total citations	Cited by 130
Citations per year	
Scholar articles	The distributed ASCI supercomputer project H Bal, R Bhoedjang, R Hofman, C Jacobs, T Kielmann… - ACM SIGOPS Operating Systems Review, 2000 Cited by 130 - Related articles - All 17 versions

130 citations

« Back to list | Edit | Export | Delete

Forthcoming paper (2015)

The Distributed ASCI Supercomputer: a Long-term Computer Science Research Infrastructure

Henri Bal VU University, Amsterdam
Dick Epema Delft University of Technology
Cees de Laat University of Amsterdam
Rob van Nieuwpoort Netherlands eScience Center
John Romein ASTRON
Frank Seinstra Netherlands eScience Center
Cees Snoek University of Amsterdam
Harry Wijshoff University of Leiden

New user-communities

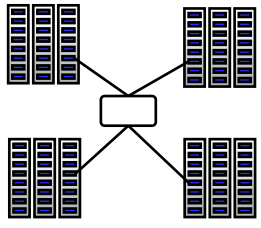
- DAS is totally unattractive for applications research
  - It comes nowhere near the TOP-500
  - It cannot be used for production runs during day-time
- Still, DAS keeps attracting new communities
  - DAS-3: multimedia; DAS-4: astronomy; DAS-5: eScience
- Reason: DAS is useful as stepping stone
  - Easy and fast experiments with parallel algorithms
  - Distributed experiments
  - ...nts w... re (acc... networks)

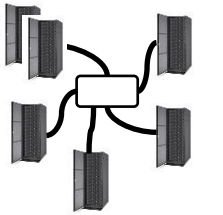
MultimediaN ASTRON netherlands eScience center

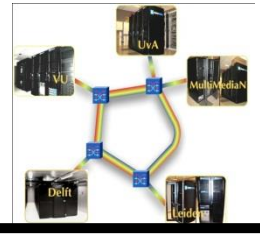
vrije Universiteit

My background

- Cluster computing
  - Zoo (1994), Orca
- Wide-area computing
  - DAS-1 (1997), Albatross
- Grid computing
  - DAS-2 (2002), Manta, Satin
- eScience & optical grids
  - DAS-3 (2006), Ibis
- Many-core accelerators
  - DAS-4 (2010), MCL, Glasswing
  - DAS-5 (2015), Cashmere







Part II: DAS-5 - Harnessing the Diversity of Complex e-Infrastructures

- **e-Infrastructures are becoming more and more distributed and diverse**
  - **Variety of accelerators**
  - **Variety of software-defined networks**
- **Many applications need to harness their power to address big data processing needs**
- **Handling the complexity and diversity of distributed infrastructures thus is a key research challenge**
- **We have >50 PhD/Postdoc projects in this area**

Research agenda for DAS-5

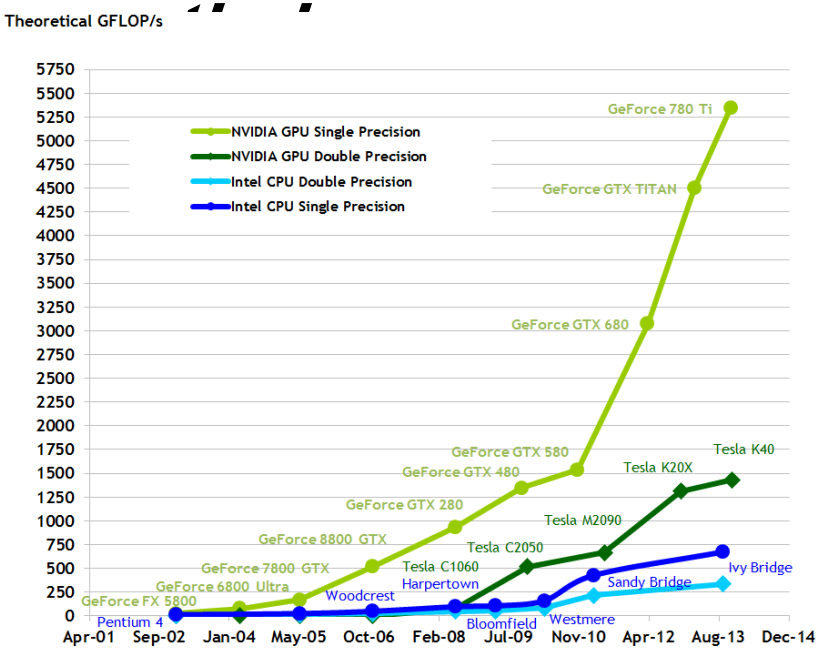
- e-Infrastructure management
  - Programmable networks, resource management, GreenIT
- Harnessing Diversity
  - Programming systems for accelerators
  - Numerous eScience applications
- Interacting with big data
  - Semantic web, sensor networks, eHealth, life sciences
- Multimedia and games
  - Multimedia content analysis
- Astronomy

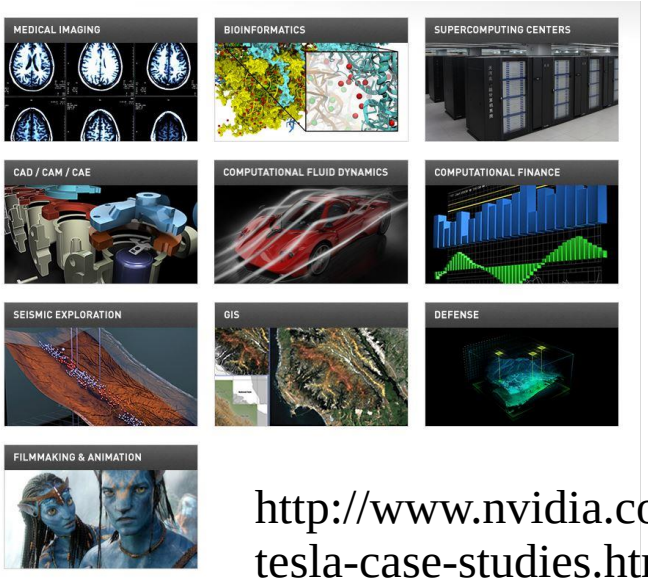
Agenda for Part II
Distributed computing + accelerators

- **GPU programming**
- **MCL: A methodology for programming accelerators**
- **Cashmere: Heterogeneous Many-Core Computing**
- **Glasswing: MapReduce on accelerators**

Graphics Processing Units

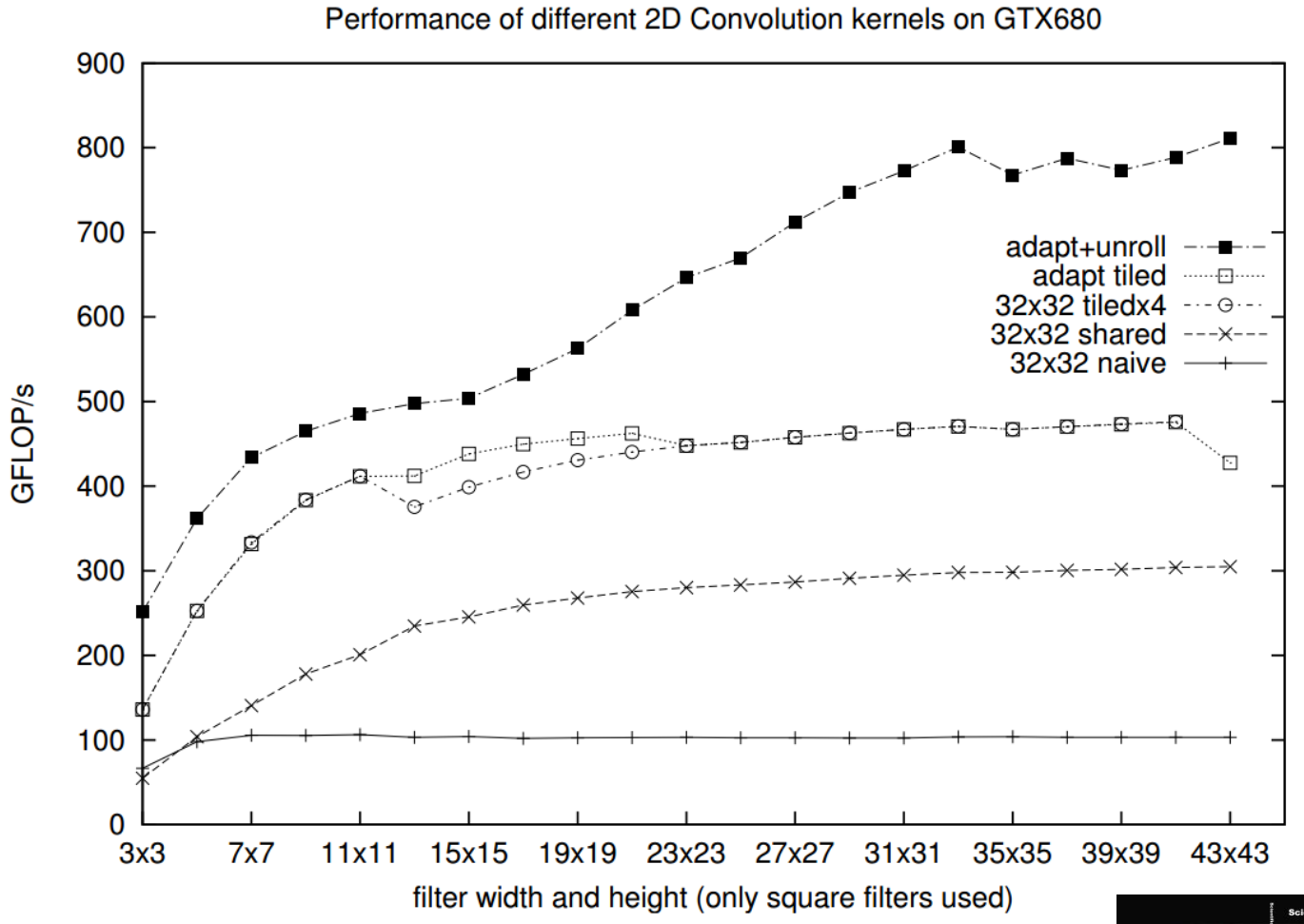
- GPUs and other accelerators take top-500 by storm
- Many application success stories
- *But GPUs are very difficult to program and*





http://www.nvidia.com/object/tesla-case-studies.html

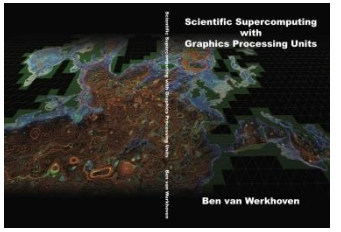
Example: convolution



Fully optimized

Naive

- **About half a Ph.D. thesis**



Parallel Programming Lab course

- **Lab course for MSc students (next to lectures)**
- **CUDA:**
  - **Simple image processing application on 1 nod**
- **MPI:**
  - **Parallel all pairs shortest path algorithms**
- **CUDA: 40-52 % passed over last 2 years**
- **MPI: 80 % passed**

Questions

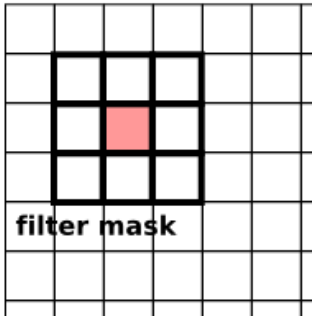
- ~~Why are accelerators so difficult to program?~~
- ~~Can we develop a better methodology for programming accelerators?~~
- ~~Can we exploit heterogeneous (diverse) clusters with different types of accelerators?~~
- ~~Can we handle big (out-of-core) data?~~

Example: Convolution operations

```
 1 for (y=0; y<I_h; y++) {
 2   for (x=0; x<I_w; x++) {
 3     sum = 0;
 4     for (j=0; j<F_h; j++) {
 5       for (i=0; i<F_w; i++) {
 6         sum += S[y+j][x+i] * F[j][i];
 7       }
 8     }
 9     I[y][x] = sum / (F_w * F_h);
10   }
11 }
```

```
 1 x = threadIdx.x+blockIdx.x*B_w;
 2 y = threadIdx.y+blockIdx.y*B_h;
 3 sum = 0;
 4 for (j=0; j<F_h; j++) {
 5   for (i=0; i<F_w; i++) {
 6     sum += S[y+j][x+i] * F[j][i];
 7   }
 8 }
 9 I[y][x] = sum / (F_w * F_h);
10
11
```

Image I of size I_w by I_h
Filter F of size F_w by F_h
Thread block of size B_w by B_h

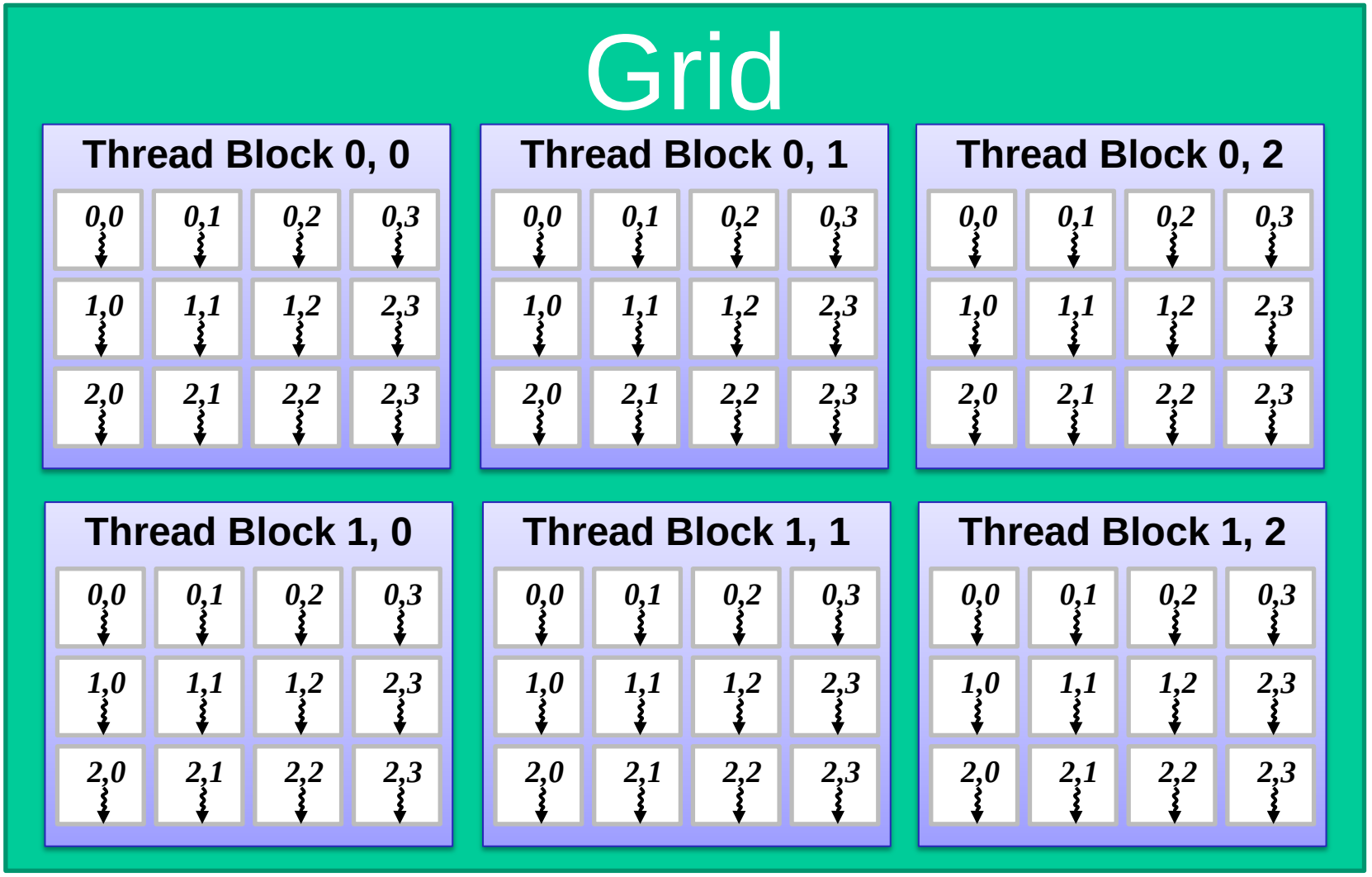


Naïve CUDA kernel:
1 thread per output pixel

Does 2 arithmetic operations and 2 loads (8 bytes)

Arithmetic Intensity (AI) = 0.25

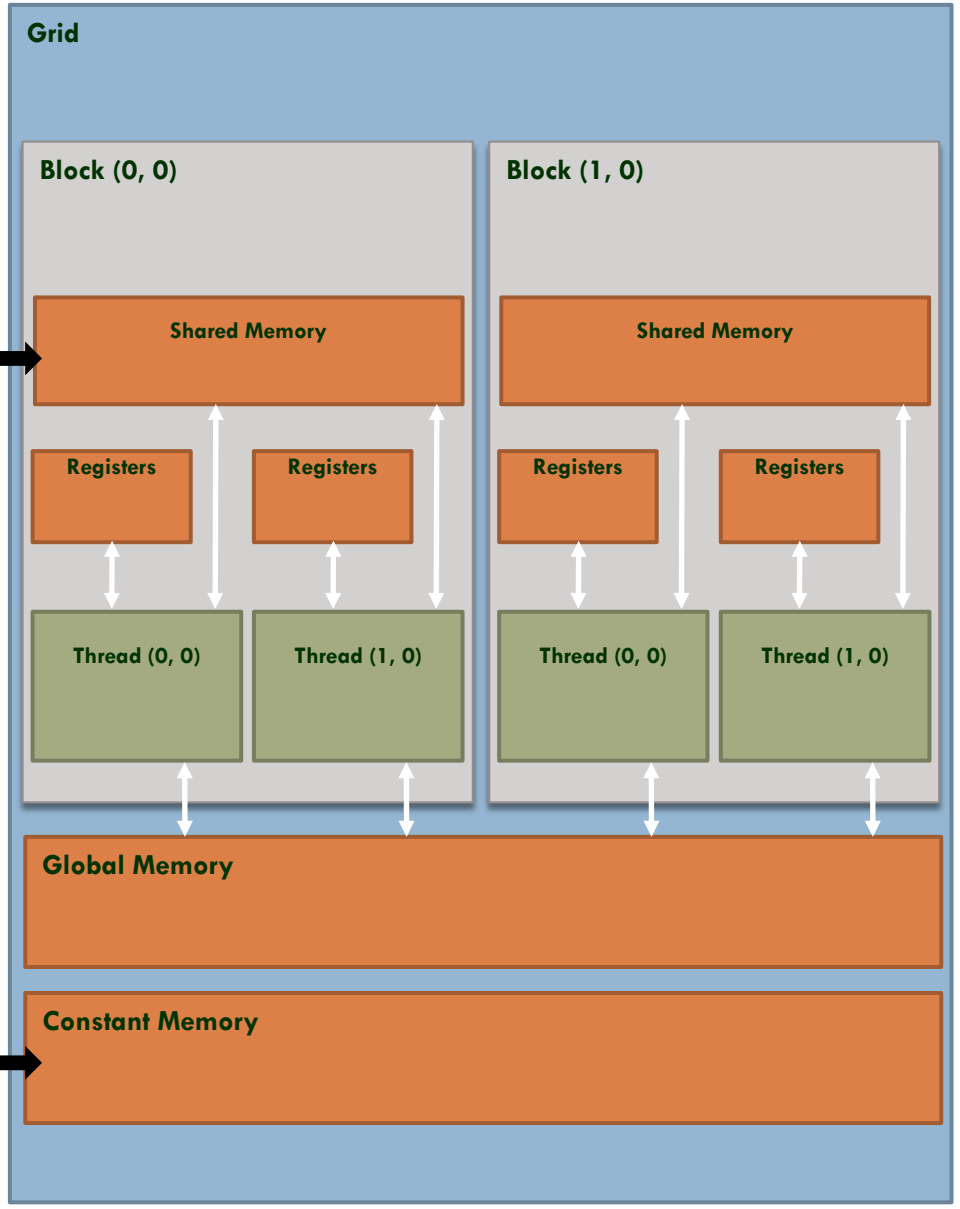
Hierarchy of concurrent threads



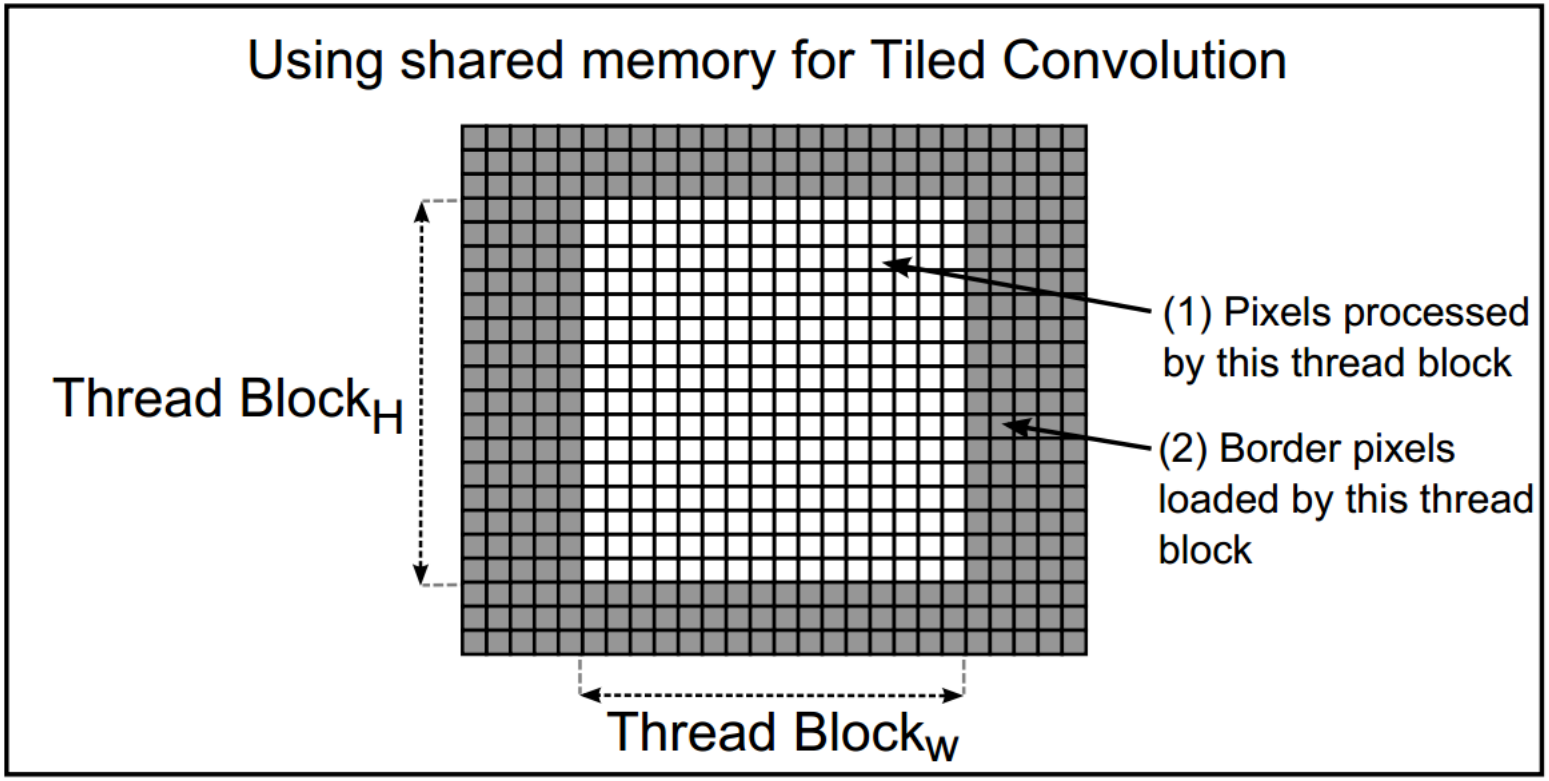
Memory optimizations for tiled convolution

Threads within a block cooperatively load entire area they need into a small (e.g. 96KB) *shared memory*

Filter (small) goes into *constant memory*



Tiled convolution



16x16 thread block processing an 11x 7 filter

- **Arithmetic Intensity:**

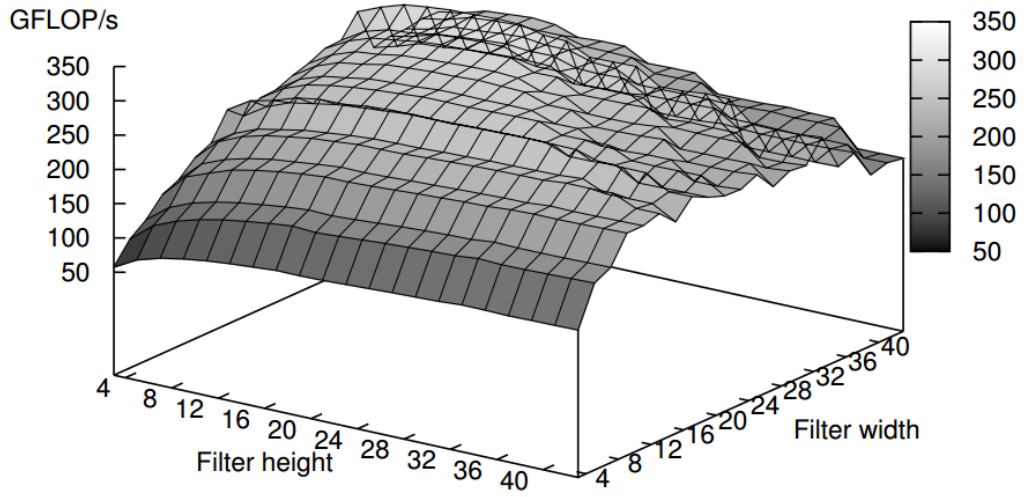
$$AI = \frac{2 \times F_w \times F_h \times B_w \times B_h}{(F_w - 1 + B_w) \times (F_h - 1 + B_h) \times 4}.$$

Analysis

- If filter size increases:
  - Arithmetic Intensity increases:
    - Kernel shifts from memory-bandwidth bound to compute-bound

$$AI = \frac{2 \times F_w \times F_h \times B_w \times B_h}{(F_w - 1 + B_w) \times (F_h - 1 + B_h) \times 4}.$$

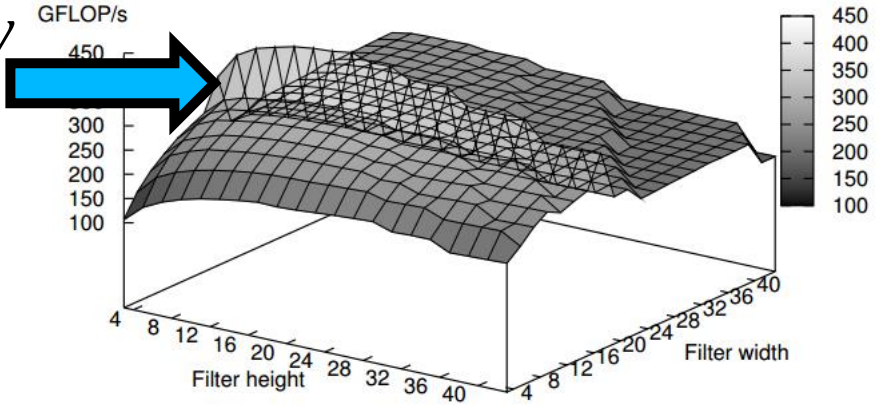
  - Amount of shared memory needed increases → fewer t on each SM



Tiling

- **Each thread block computes 1xN tiles in horizontal direction**
  - **+ Increases amount of work per thread**
  - **+ Saves loading overlapping borders**
  - **+ Saves redundant instructions**
  - **- More shared memory, fewer concurrent thread blocks**

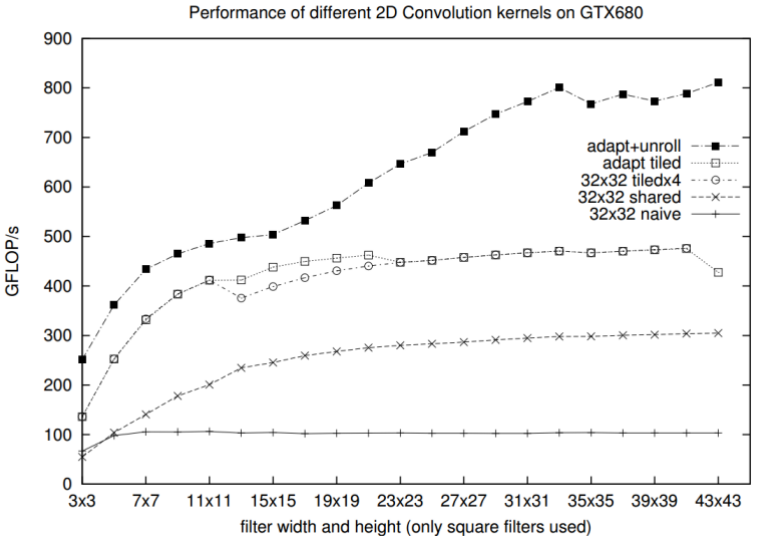
No shared memory bank conflicts



Adaptive tiling

- **Tiling factor is selected at runtime depending on the input data and the resource limitations of the device**
  - **Highest possible tiling factor that fits within the shared memory available (depending on filter size)**
- **Plus loop unrolling, mem optimal configuration**

Ph.D. thesis Ben van Werkhoven, Oct. 2014 + FGCS journal, 2014



Why is GPU programming hard?

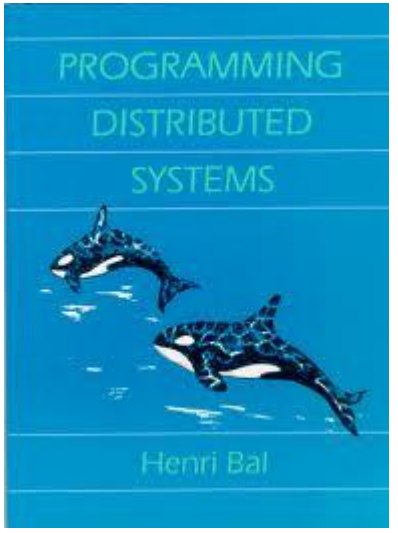
- **Mapping algorithm to architecture is difficult, especially as the architecture is difficult:**
  - **Many levels of parallelism**
  - **Limited resources (registers, shared memory)**
    - **Less of everything than CPU (except parallelism), especially *per thread*, makes problem-partitioning difficult**
  - **Everything must be in balance to obtain performance**
- **Portability**
  - **Optimizations are architecture-dependent, and the architectures change frequently**

Why is GPU programming hard?

- **Many crucial high-impact optimizations needed:**
  - **Data reuse**
    - **Use shared memory efficiently**
    - **Limited by #registers per thread, shared memory per thread block**
  - **Memory access patterns**
    - **Shared memory bank conflicts, global memory coalescing**
  - **Instruction stream optimization**
    - **Control flow divergence, loop unrolling**
  - **Moving data to/from the GPU**

Why is GPU programming hard?

- **Bottom line: tension between**
  - **control over hardware to achieve performance**
  - **higher abstraction level to ease programming**
- **Programmers need understandable performance**
- **Old problem in Computer Science, but now in extreme form**

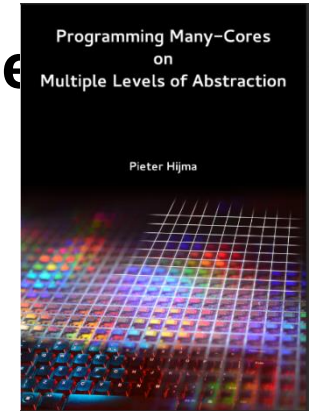


(1989)

Programming methodology: stepwise refinement for performance

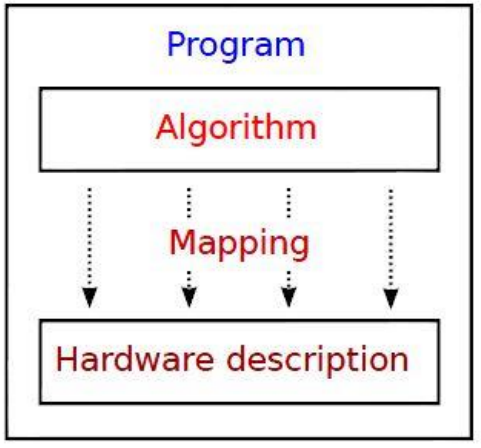
- **Methodology:**
  - **Programmers can work on *multiple levels of abstraction***
  - **Integrate *hardware descriptions* into programming model**
  - ***Performance feedback* from compiler, based on hardware description and kernel**
  - **Cooperation between compiler and programme**

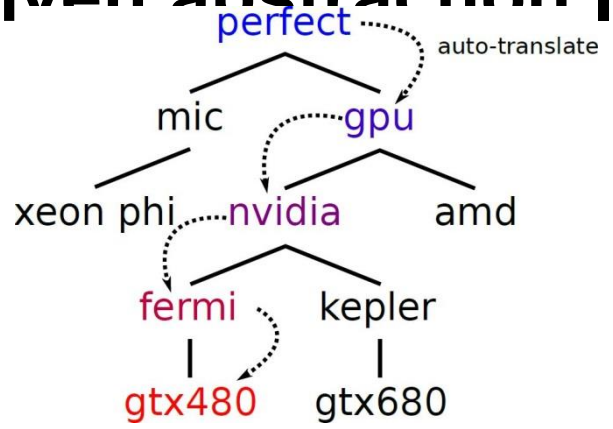
Pieter Hijma, Rob van Nieuwpoort, Ceriel Jacobs, Henri Bal,
Concurrency & Computation: Practice & Experience 2015
+ Ph.D. thesis June 2015



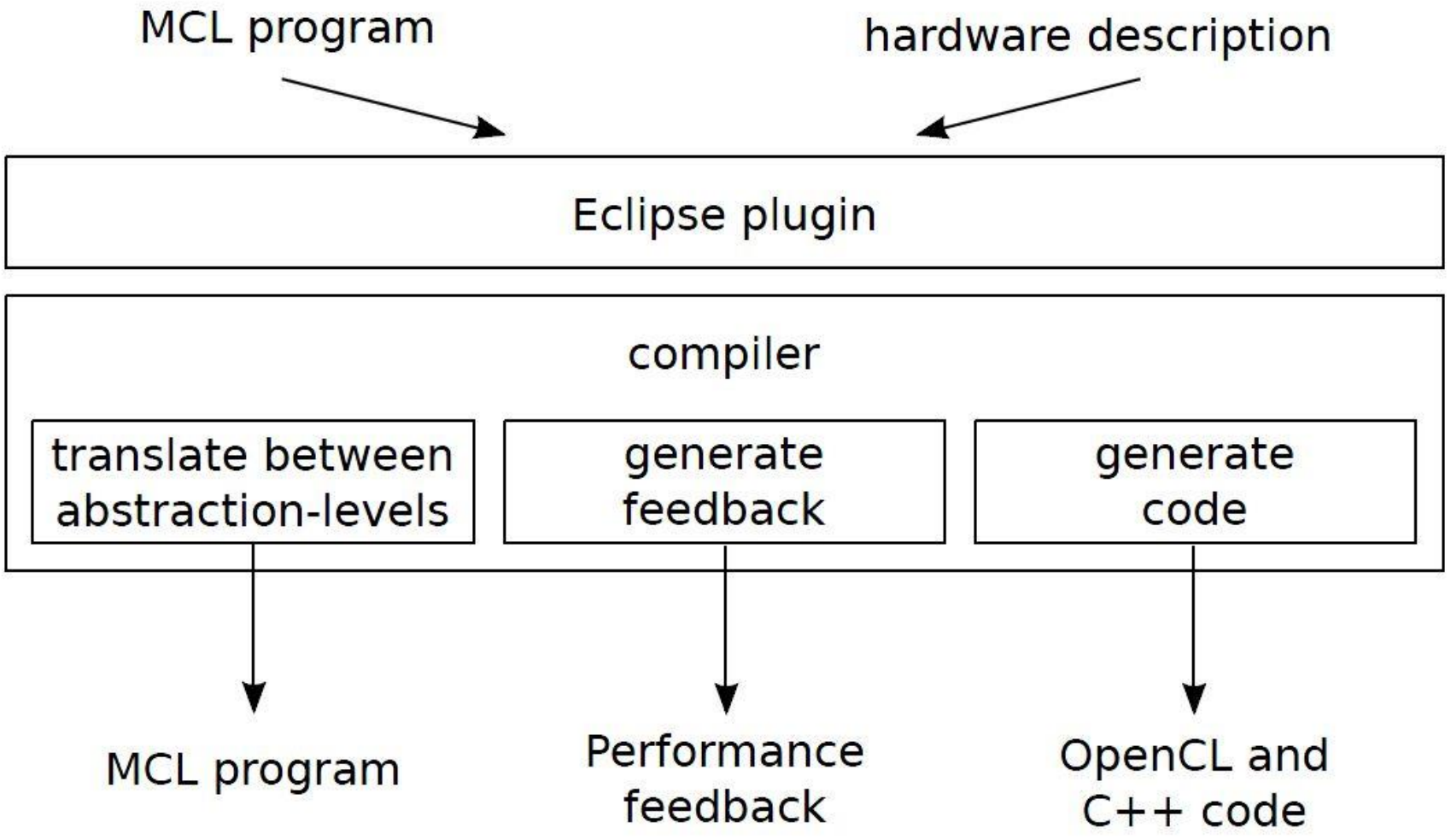
MCL: Many-Core Levels

- MCL program is an algorithm mapped to hardware
- Start at a suitable abstraction level
  - E.g. idealized accelerator, NVIDIA Kepler GPU, Xeon Phi
- MCL compiler guides programmer which opt[illegible] pply on given abstraction level or [illegible] per levels





MCL ecosystem



Powered by Rascal

```
parallelism hierarchy {
    memory_space main {
    }
    par_group threads {
        nr_units = unlimited;
        par_unit thread {
}   }   }

device perfect {
    mem;
    ic;
    cores;
}

memory mem {
    space(main);
    capacity = unlimited B;
}
```

```
interconnect ic {
    connects(mem, cores.core[*]);
    latency = 1 cycle;
    bandwidth = unlimited bits/s;
}

execution_group cores
    nr_units = unlimited;
    execution_unit core {
        slots(thread, 1);
        instructions ops {
            instruction((+), 1 cycle);
            instruction((/), 1 cycle);
            instruction((-), 1 cycle);
            instruction((*), 1 cycle);
}   }   }
```

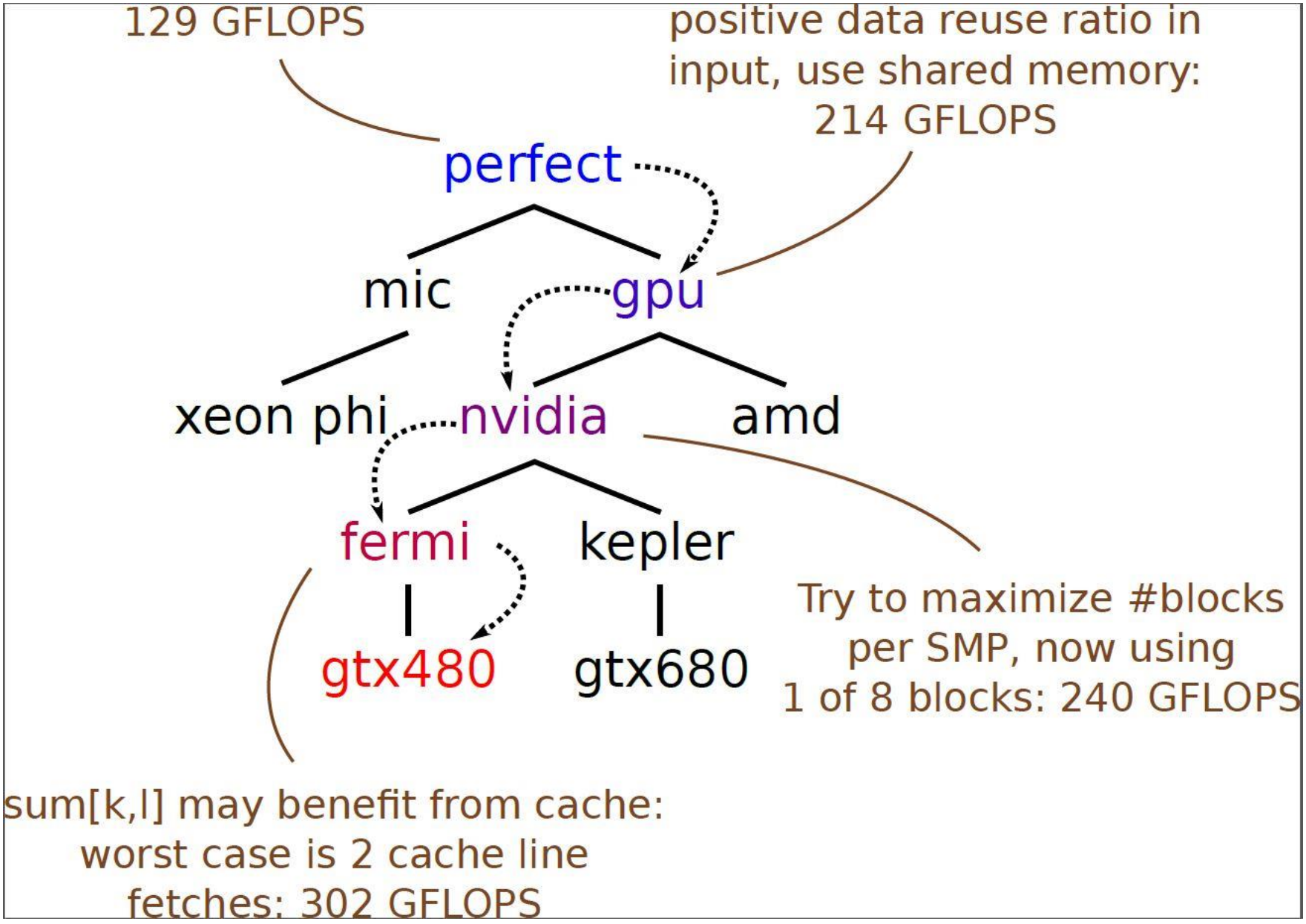
```
import perfect;

perfect void convolve(int h, int w, int fh, int fw,
    main float [h, w] output,
    main float [h + fh/2*2, w + fw/2*2] input,
    main float [fh, fw] filter) {

  foreach (int i in h threads ) {
    foreach(int j in w threads ) {

      float  sum =0.0;
      for (int y = 0; y < fh; y++) {
        for (int x = 0; x < fw; x++) {
          sum += filter[y, x] * input[i + y, j + x];
        }
      }
      output[i, j] = sum / (fh * fw);
 } } }
```

Compiler feedback

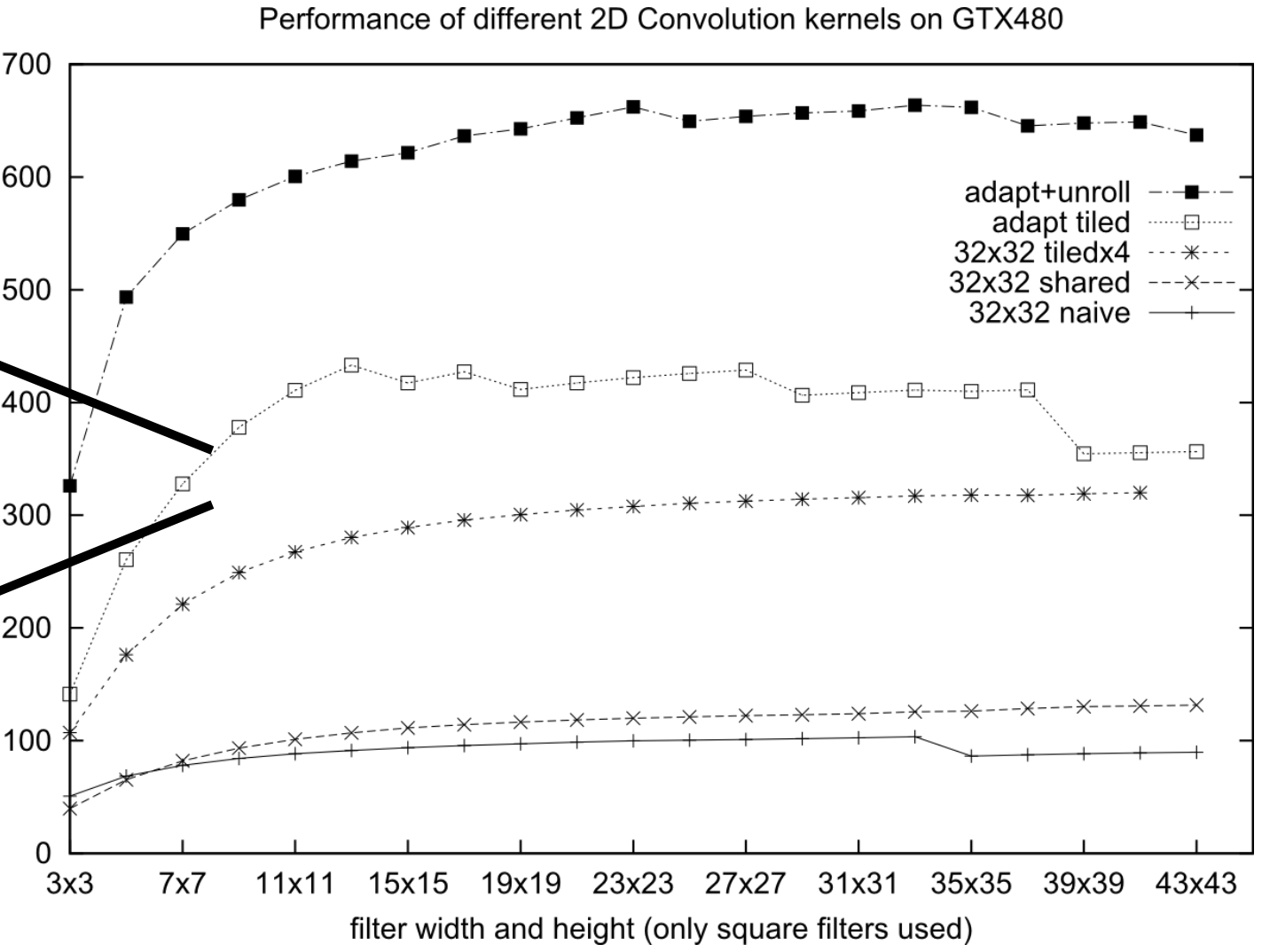


Performance (GTX480, 9×9 filters)

380 GFLOPS

MCL:
302 GFLOPS

Compiler +



vrije Universiteit

Performance evaluation

Application	Naive	MCL	Other implementations
NVIDIA GTX480 GPU			
matmul	89.0 GFLOPS	564 GFLOPS	892 GFLOPS
convolution	129 GFLOPS	302 GFLOPS	380 GFLOPS
histogram	6.94 GB/s	80.0 GB/s	80 GB/s
gesummv*	1.25 GFLOPS	51.9 GFLOPS	2.98 GFLOPS
blackScholes	297 GFLOPS	432 GFLOPS	237 GFLOPS
sparse matmul	1.01 GFLOPS	3.88 GFLOPS	8.9 GFLOPS
Intel Xeon Phi (5110P)			
matmul	39.9 GFLOPS	488 GFLOPS	695 GFLOPS
convolution	66.4 GFLOPS	171 GFLOPS	n/a
histogram**	0.331 GB/s	11.9 GB/s	0.3 GB/s
gesummv	0.84 GFLOPS	55.4 GFLOPS	n/a
blackScholes	66 GFLOPS	115 GFLOPS	n/a
sparse matmul**	3.93 GFLOPS	5.28 GFLOPS	13 GFLOPS

Compared to known, fully optimized versions
(* measured on a C2050, ** using a different input).

Lessons learned from MCL

- **The stepwise-refinement for performance methodology supports a structured approach**
  - **Gradually expose programmers to more hardware details**
- **It enables to develop optimized many-core kernels**
- **Key ideas: stepwise refinement + multiple abstraction levels + compiler feedback**

Heterogeneous many-core clusters

- New GPUs become available frequently, but older-generation GPUs often still are fast enough
  - Clusters become heterogeneous and contain different types of accelerators
- VU DAS-4 cluster
  - NVIDIA GTX480 GPUs (22)
  - NVIDIA K20 GPUs (8)
  - Intel Xeon Phi (2)
  - NVIDIA C2050 (2), Titan, GTX680 GPU
  - AMD HD7970 GPU

Heterogeneity in TOP-500 (list of Nov. 2014)

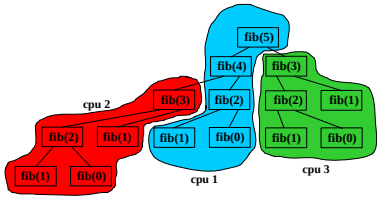
name	**ranking**	**conguration**
Quartetto	49	K20, K20X, Xeon Phi 5110P
Lomonosov	58	2070, PowerXCell 8i
HYDRA	77	K20X, Xeon Phi
SuperMIC	88	Xeon Phi 7110P, K20X
Palmetto2	89	K20m, M2075, M2070
Armstrong	103	Xeon Phi 5120D, K40
Loewe-CSC	179	HD5870, FirePro S10000
Inspur	310	K20m, Xeon Phi 5110P
Tsubame 2.5	392	K20X, S1070, S2070
El Gato	465	K20, K20X, Xeon Phi 5110P

Cashmere

- **Programming system for heterogeneous many-core clusters**
- **Integrates two systems**
  - **Satin: divide-and-conquer Grid programming system**
  - **MCL: allows kernels to be written and optimized for each type of hardware**
- **Cashmere does integration, application logic, mapping, and load balancing for multiple GPUs/node**

Pieter Hijma, Rob van Nieuwpoort, Ceriel Jacobs, Henri Bal, 29th IEEE Int. Parallel & Distributed Processing Symposium (IPDPS 2015)

Satin



- **Model**
  - **Divide-and-conquer programming model, inspired by Cilk**
  - **Targets hierarchical distributed systems (e.g., grids)**
- **Advantages**
  - **Load-balancing (cluster-aware random work-stealing)**
  - **Latency hiding, especially on wide-area links**
  - **Good scalability on wide-area systems**

Cashmere programming model

- **Additional levels of parallelism**
  - **Parallelism on many-core device**
  - **Multiple many-core devices per compute node**
  - **Overlap in communication and computation**
- **Goal:**
  - **Use these additional levels of parallelism without disruptive changes to the Satin programming model**

Cashmere skeleton

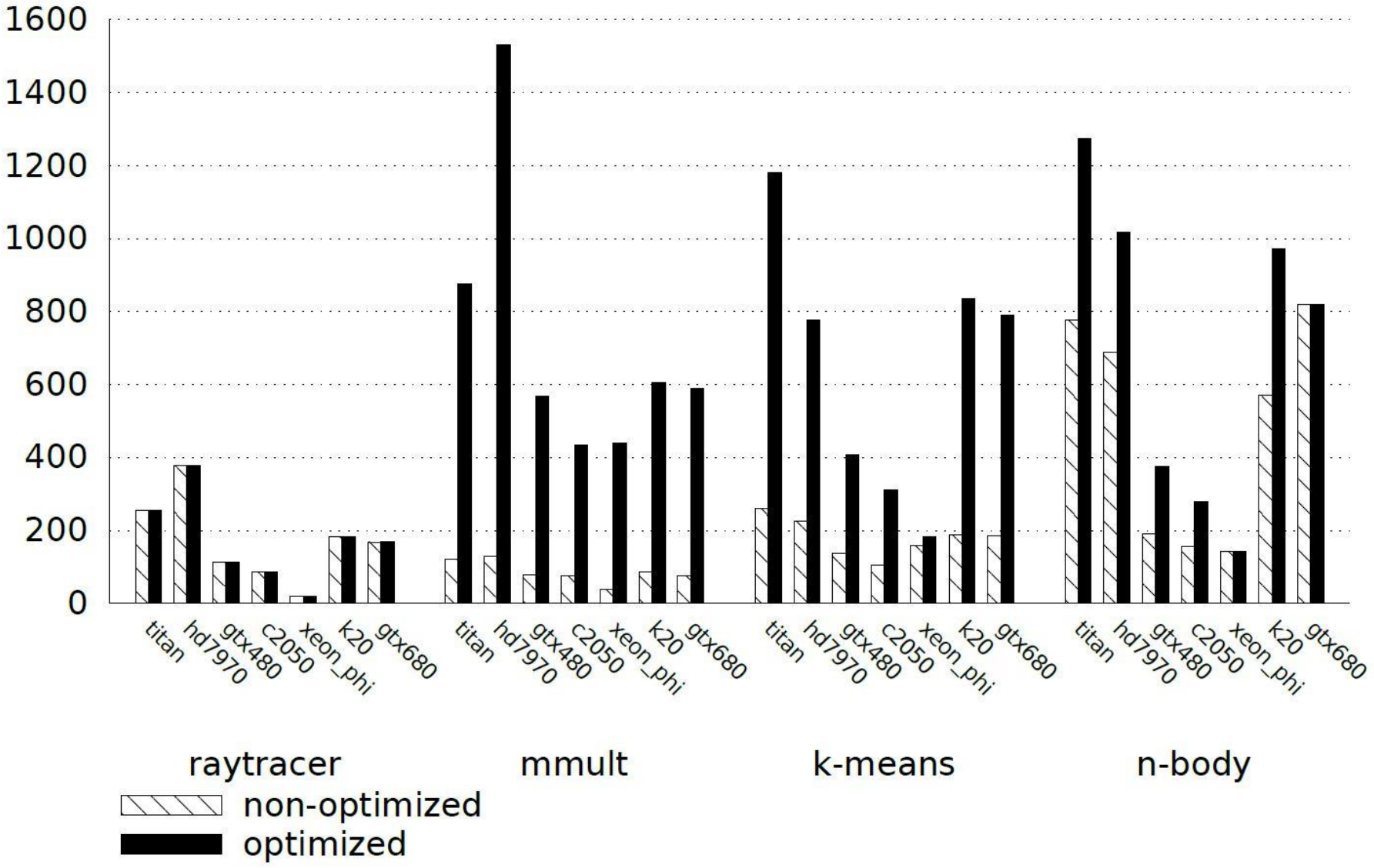
```
1 spawnable f(a) {
2   if (small_enough_for_leaf(a)) {
3     return do_leaf_computation(a)
4   }
5   else if (small_enough_for_many_core(a)) {
6     Cashmere.enableManyCore()
7   }
8
9   r1 = f(make_smaller(a)) //asynchronous
10  r2 = f(make_smaller(a)) //asynchronous
11  sync
12
13  return combine(r1, r2)
14 }
```

Evaluation

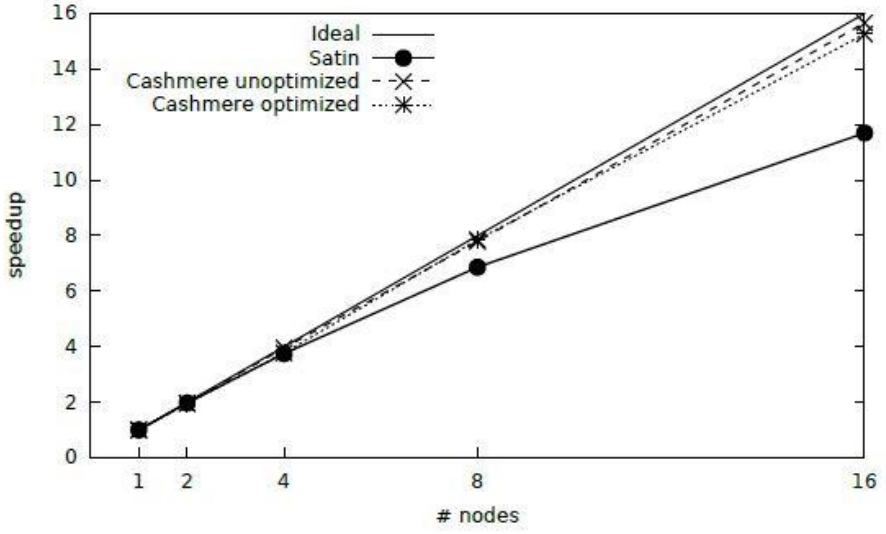
- **Methodology**
  - **Measure kernel performance**
  - **Measure scalability on homogeneous cluster**
  - **Compare efficiency homogeneous/heterogenous**
- **Application classes**

Application	Type	Computation	communication
Raytracer	Irregular	heavy	light
Matmul	Regular	heavy	heavy
k-means	Iterative	moderate	light
n-body	Iterative	heavy	moderate

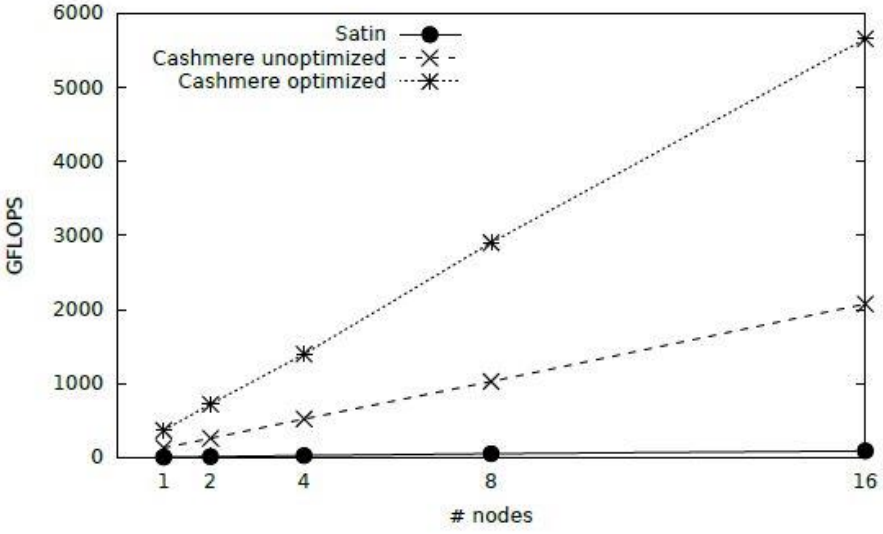
Kernel performance (GFLOP/s)



K-Means on a homogeneous GTX480 cluster



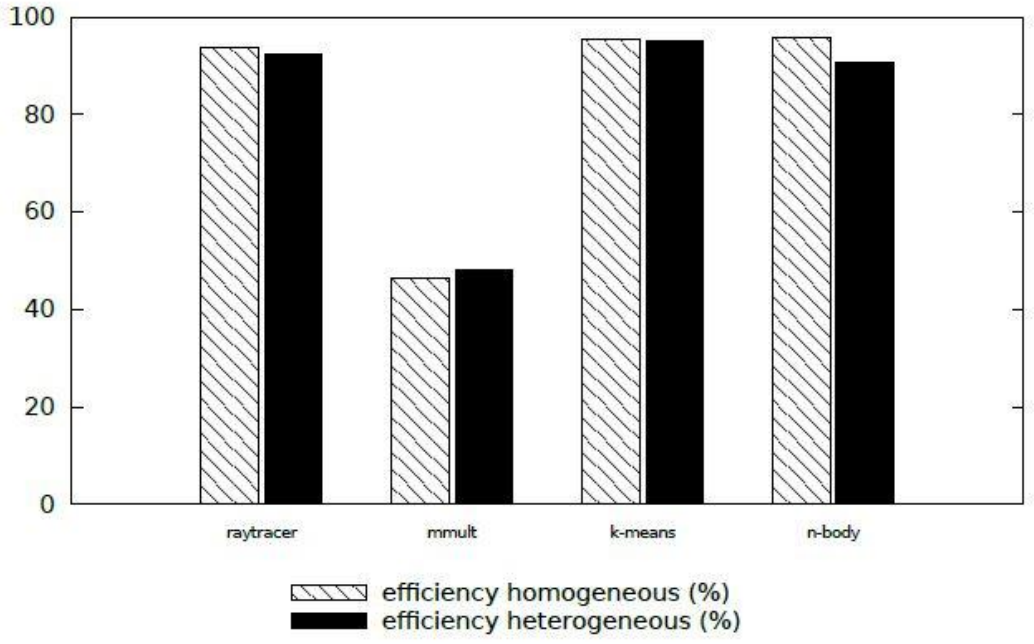
scalability



absolute performance

Heterogeneous performance

application	performance (GFLOPS)	configuration
raytracer	1883	10 gtx480, 2 c2050, 1 gtx680, 1 titan, 1 hd7970
matmul	3927	10 gtx480, 2 c2050, 1 gtx680, 1 titan, 1 hd7970
k-means	10644	10 gtx480, 2 c2050, 1 gtx680, 1 titan, 1 hd7970, 7 k20, 1 xeon_phi
n-body	7002	10 gtx480, 2 c2050, 1 gtx680, 1 titan, 1 hd7970



Homogeneous:
efficiency on 16 GTX480

Heterogeneous:
efficiency over total combined hardware

Lessons learned on Cashmere

- **Seamless integration of many-cores in Satin**
- **MCL enabled writing/optimizing variety of kernels**
- **High performance and automatic load balancing even when the many-core devices differ widely**
- **Efficiency >90% in 3 out of 4 applications in heterogeneous executions**

Future work on MCL/Cashmere

- Improve MCL compiler analysis
- Include performance prediction models
- Managing different versions at different levels
- Real-world applications
  - Digital forensics
  - Climate modeling
  - Natural language processing
  - High-energy physics (LHC)
- Extend Cashmere model beyond divide&conquer
- Extend implementation to wide-area systems
- Extend to out-of-core (big data) applications

Other approaches that deal with performance vs abstraction

- **Domain specific languages**
- **Patterns, skeletons, frameworks**
- **Berkeley Dwarfs**

List of Dwarfs

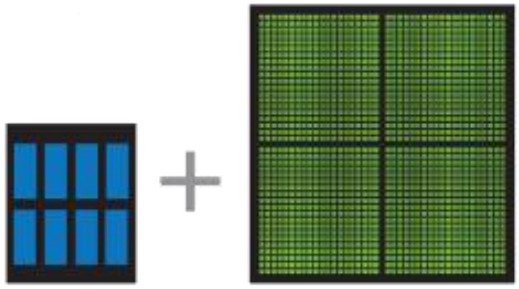
1. Dense Linear Algebra
2. Sparse Linear Algebra
3. Spectral Methods
4. N-Body Methods
5. Structured Grids
6. Unstructured Grids
7. MapReduce
8. Combinational Logic
9. Graph Traversal
10. Dynamic Programming
11. Backtrack and Branch-and-Bound
12. Graphical Models
13. Finite State Machines

MapReduce

- **Big Data revolution**
- **Designed for cheap commodity hardware**
- **Scales horizontally**
- **Coarse-grained parallelism**
- **MapReduce on modern hardware?**

Glasswing: Rethinking MapReduce

- **Use accelerators (OpenCL) as mainstream feature**
- **Massive out-of-core data sets**
- **Scale vertically & horizontally**
- **Maintain MapReduce abstraction**



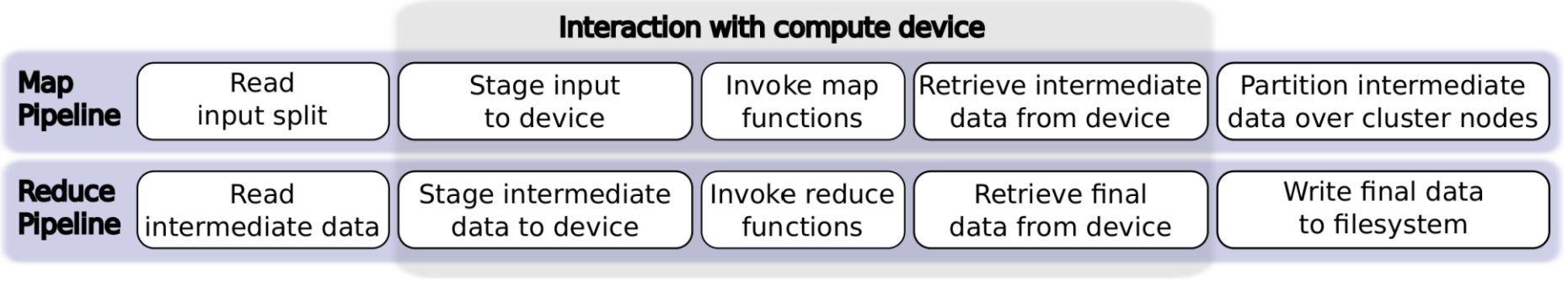
Ismail El Helw, Rutger Hofman, Henri Bal [HPDC'2014, SC'2014]

Related work

	Out of Core	Compute Device	Cluster
Phoenix	×	CPU-only	×
Tiled-MapReduce	×	NUMA CPU	×
Mars	×	GPU-only	×
Ji et al.	×	GPU-only	×
MapCG	×	CPU/GPU	×
Chen et al.	×	GPU-only	×
GPMR	×	GPU-only	✓
Chen et al.	×	AMD Fusion	×
Merge	×	Any	×
HadoopCL	✓	APARAPI	✓
Glasswing	✓	OpenCL enabled	✓

Glasswing Pipeline

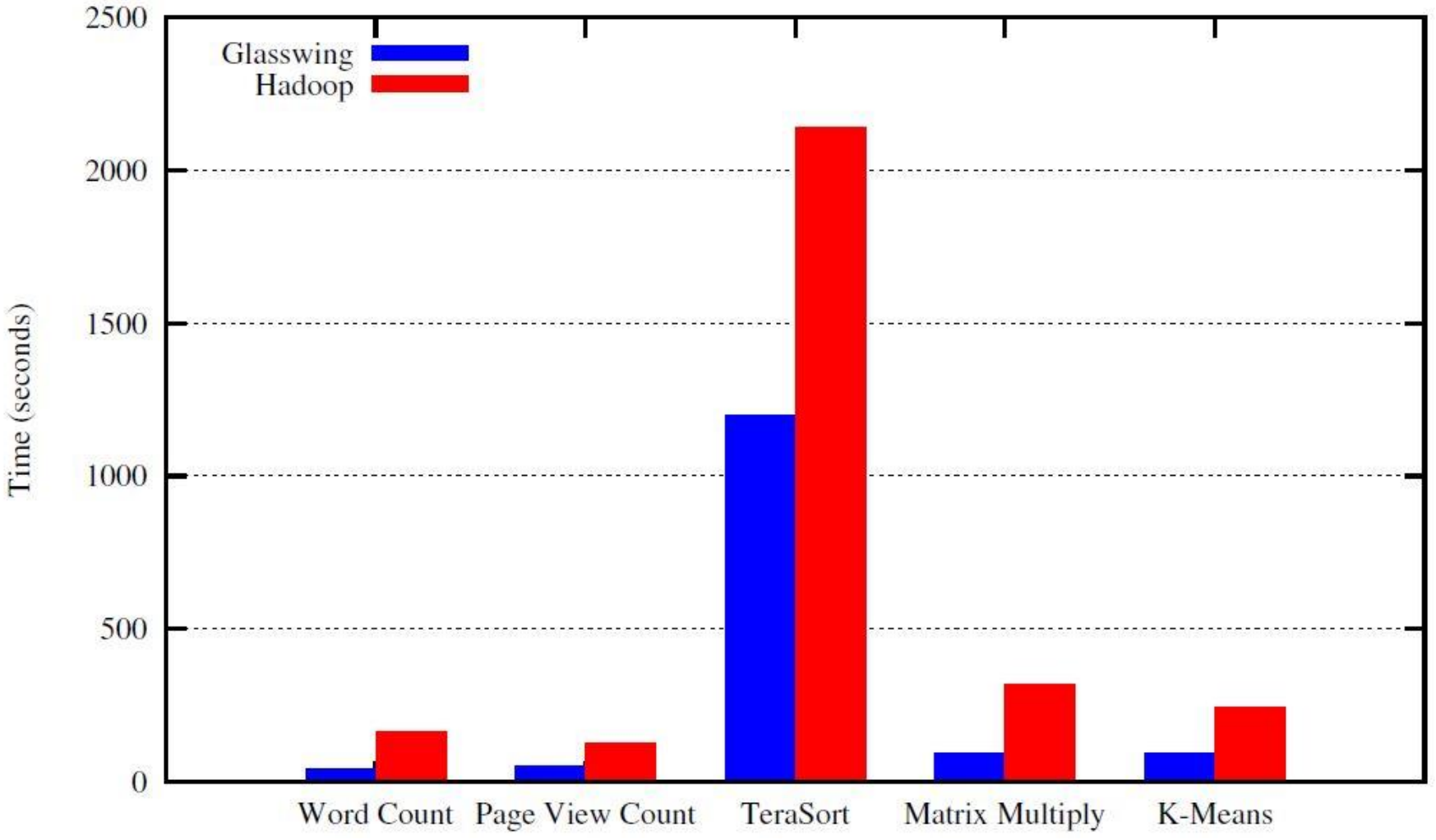
- **Overlaps computation, communication & disk access**
- **Supports multiple buffering levels**



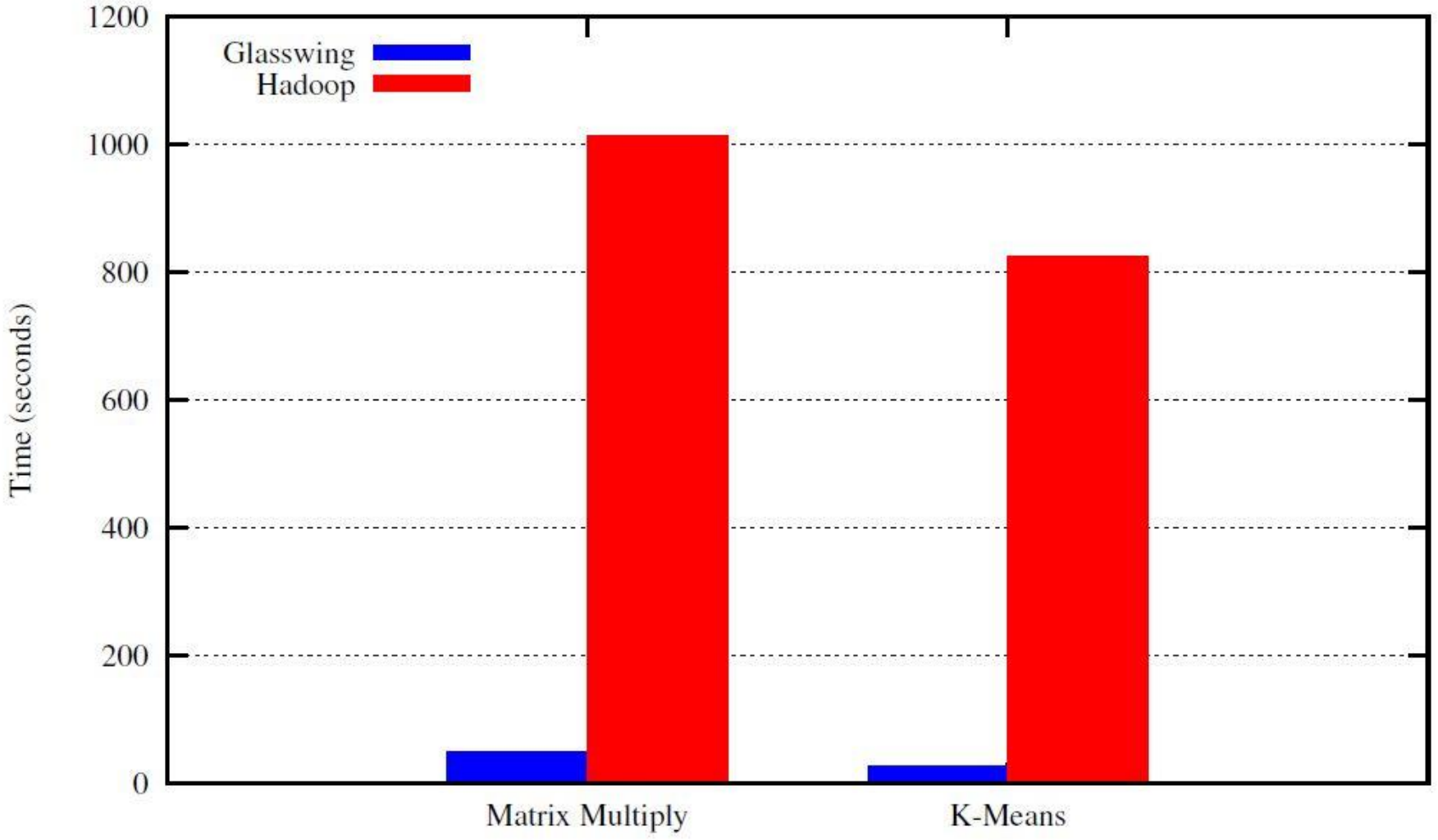
GPU optimizations

- Glasswing framework does:
  - Memory management
  - Some shared memory optimizations
  - Data movement, data staging
- Programmer:
  - Focusses on the map and reduce kernels (using OpenCL)
  - Can do kernel optimizations if needed
    - Coalescing, memory banks, etc.

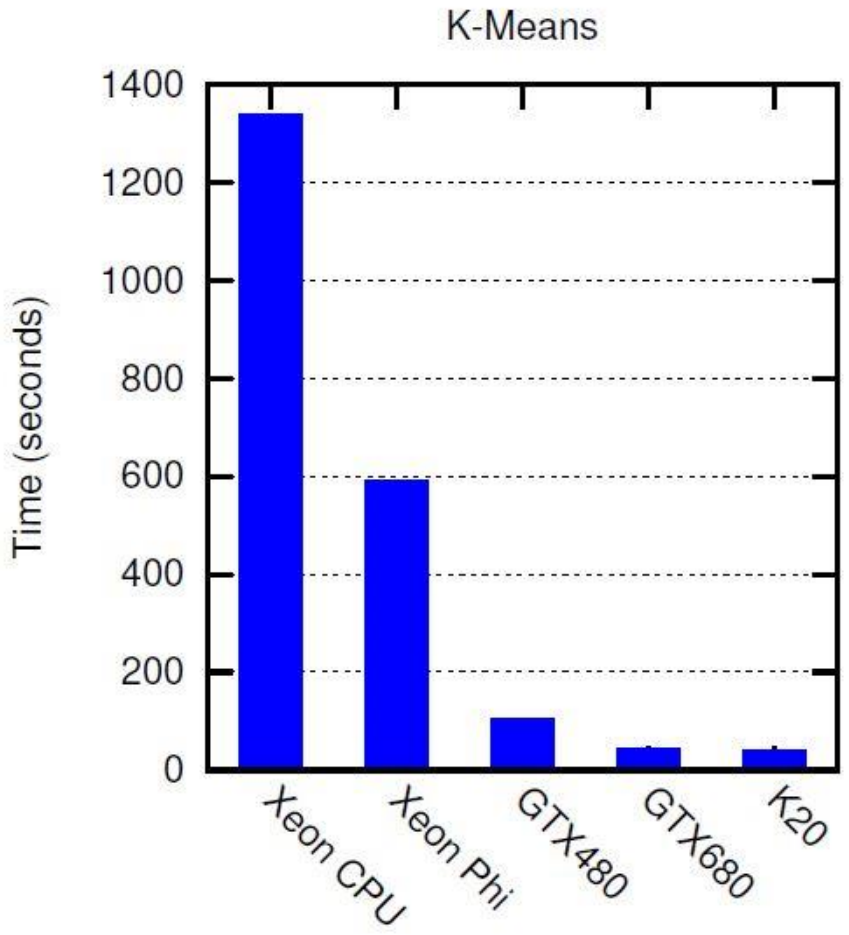
Glasswing vs. Hadoop
64-node CPU Infiniband cluster

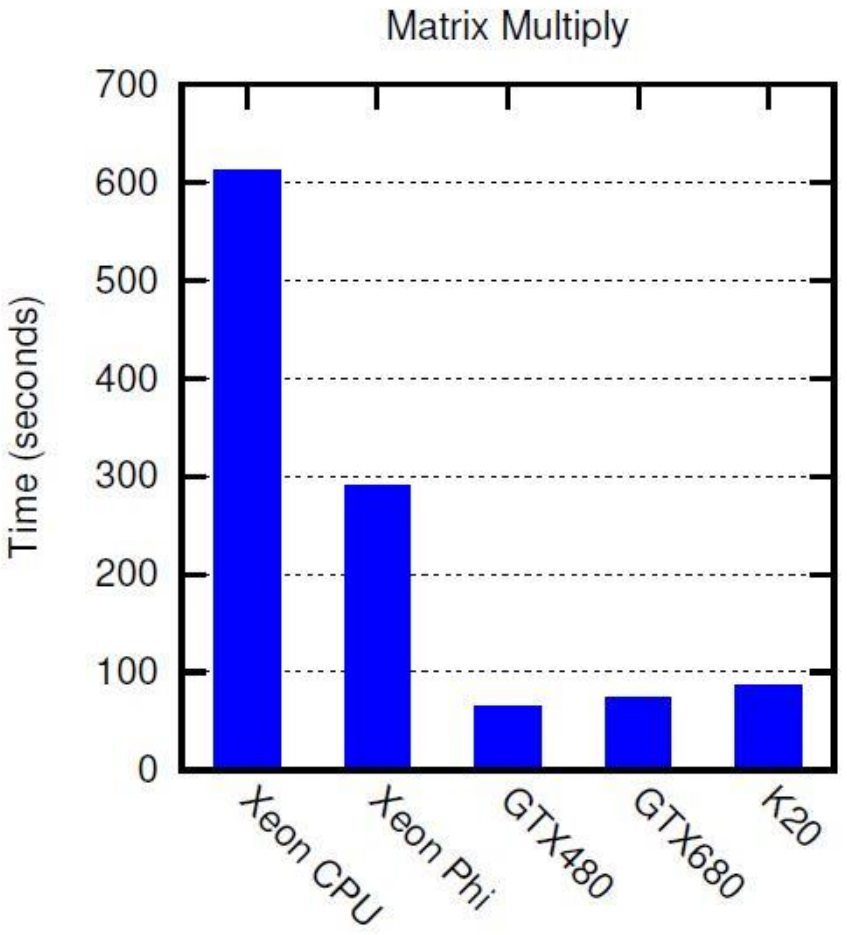


Glasswing vs. Hadoop
16-Node GTX480 GPU Cluster



Compute Device Comparison





Lessons learned on Glasswing

- **MapReduce framework combining coarse-grained and fine-grained parallelism**
- **Scales vertically & horizontally**
- **Handles out-of-core data, sticks with MapReduce model**
- **Overlaps kernel executions with memory transfers, network communication and disk access**
- **Outperforms Hadoop by 1.2 – 4x on CPUs and 20 – 30x on GPUs**

Discussion on programming accelerators

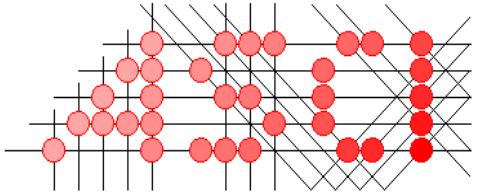
- **Accelerator programming and optimization currently is too time-consuming for many applications**
  - **GPU programs need many complex optimizations to obtain high performance**
- **Many (eScience) applications do need performance of modern accelerators**
  - **Next in line: SKA, high-energy physics, forensics, water management, ecology, natural language processing, …**
- **How to deal with the tension between abstraction-level and control?**
  - **New programming methodologies that allow a choice**

netherlands eScience center

vrije Universiteit

Discussion on Infrastructure

- **Computer Science needs its own infrastructure for interactive experiments**
  - **DAS was instrumental for numerous projects**
- **Being organized helps**
  - **ASCI is a (distributed) communit**



- **Having a vision for each generation helps**
  - **Updated every 4-5 years, in line with research agendas**
- **Other issues:**
  - **Expiration date?**
  - **Size?**

Expiration date

- Need to stick with same hardware for 4-5 years
  - Also cannot afford expensive high-end processors
- Reviewers sometimes complain that the current system is out-of-date (after > 3 years)
  - Especially in early years (clock speeds increased fast)
- DAS-4/DAS-5: accelerators added during the project

Does size matter?

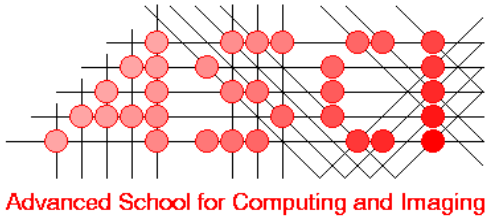
- **Reviewers seldom reject our papers for small size**
  - **``This paper appears <u>in-line with experiment sizes in related SC research</u> work, if <u>not up to scale with current large operational supercomputers</u>.”**
- **We sometimes do larger-scale runs in clouds or on production supercomputers**

Interacting with applications

- **Used DAS as *stepping stone* for applications**
  - **Small experiments, no productions runs**
- **Applications really helped the CS research**
  - **DAS-3: multimedia → Ibis applications, awards**
  - **DAS-4: astronomy → many new GPU projects**
  - **DAS-5: eScience Center → EYR-G award, GPU work**
- **Many plans for new collaborations**
  - **Data science**
  - **Business analytics**

Conclusions

- **DAS has had a major impact on Dutch Computer Science for almost two decades**
- **It aims at *coherence* + modest system size**
- **Highly effective and small investment**
  - **E.g. It avoids fragmented systems management**
- **Central organization and coherence are key**



Acknowledgements

DAS Steering Group:
Dick Epema
Cees de Laat
Cees Snoek
Frank Seinstra
John Romein
Harry Wijshoff

System management:
Kees Verstoep et al.

Hundreds of users

Support:
TUD/GIS
Stratix
ASCI office

DAS grandfathers:
Andy Tanenbaum
Bob Hertzberger
Henk Sips

More information:
http://www.cs.vu.nl/das5/

Funding:

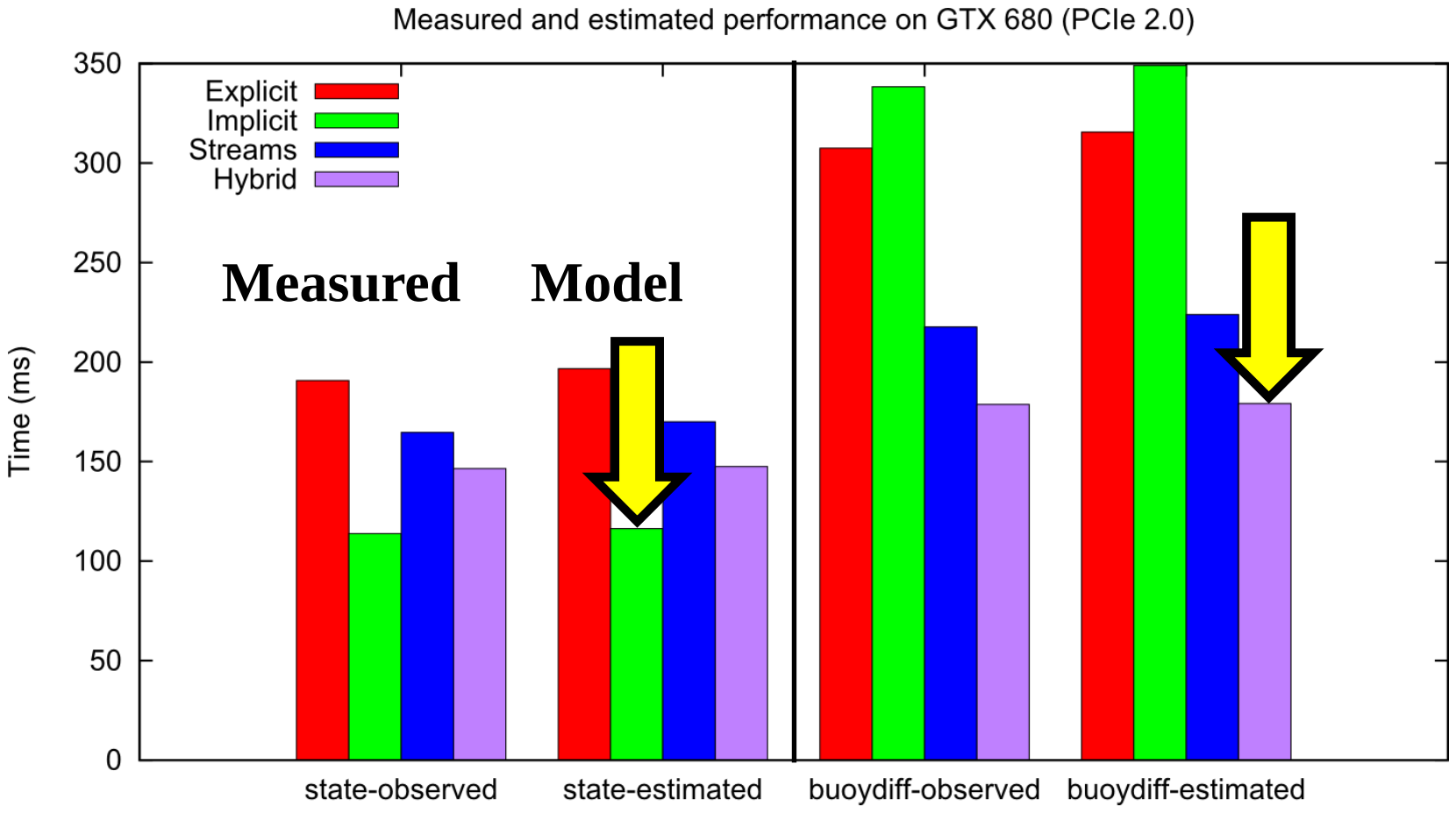
NWO Netherlands Organisation for Scientific Research
vl·e
SURF NET
MultimediaN
netherlands eScience center
COMMIT/

vrije Universiteit

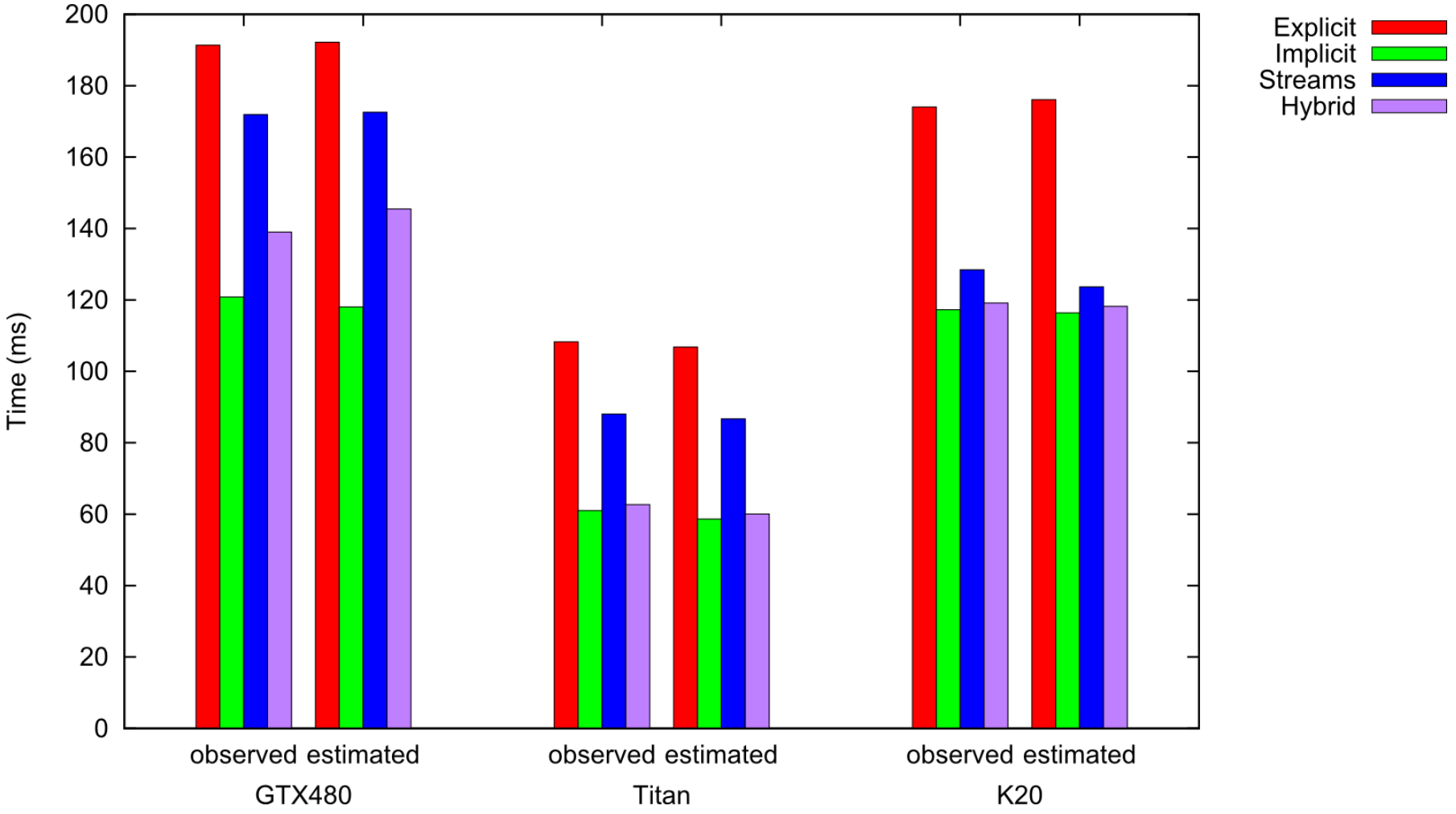
Different methods for CPU-GPU communication

- **Memory copies (explicit)**
  - **No overlap with GPU computation**
- **Device-mapped host memory (implicit)**
  - **Allows fine-grained overlap between computation and communication in either direction**
- **CUDA Streams or OpenCL command-queues**
  - **Allows overlap between computation and communication in different streams**
- **Any combination of the above**

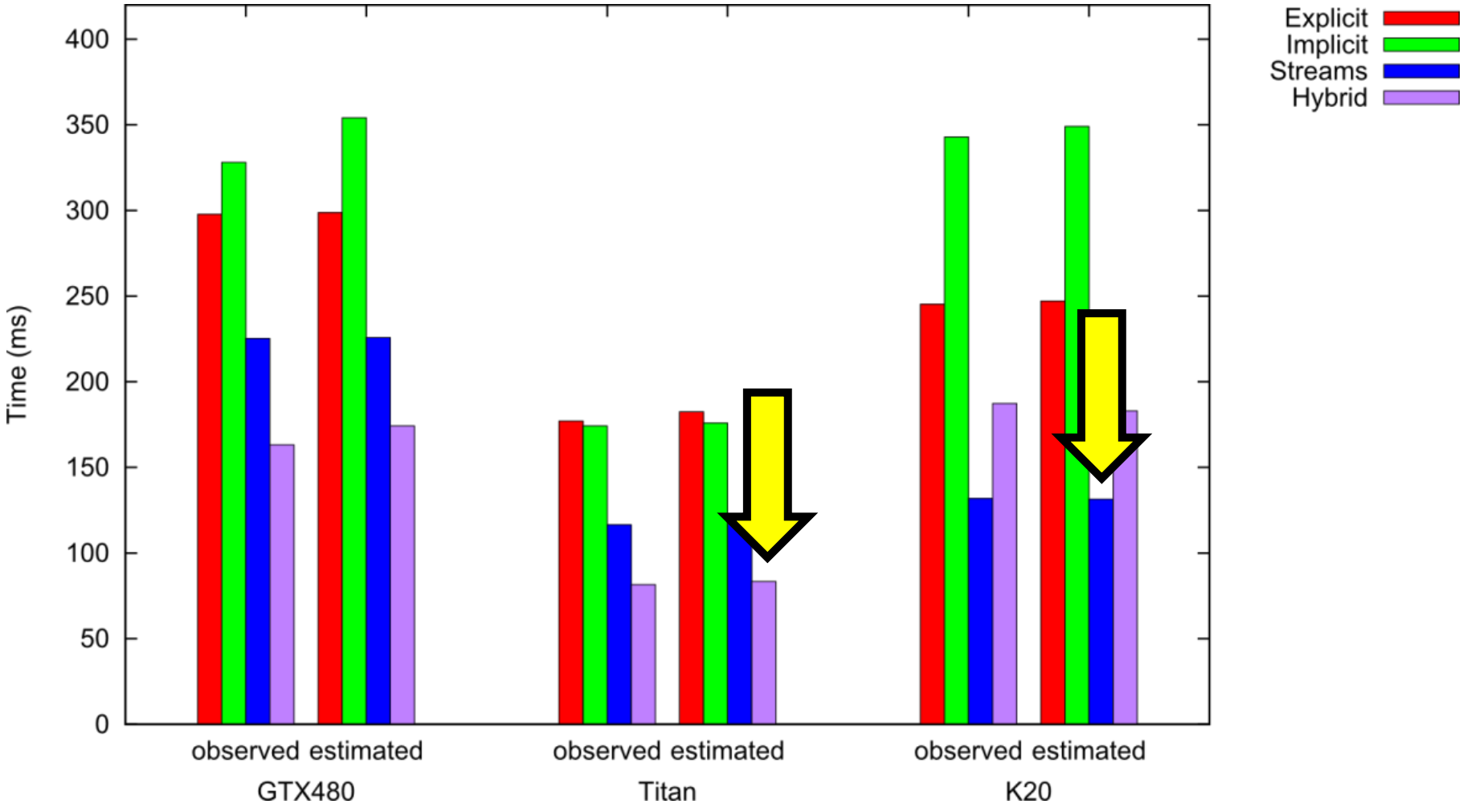
Example result



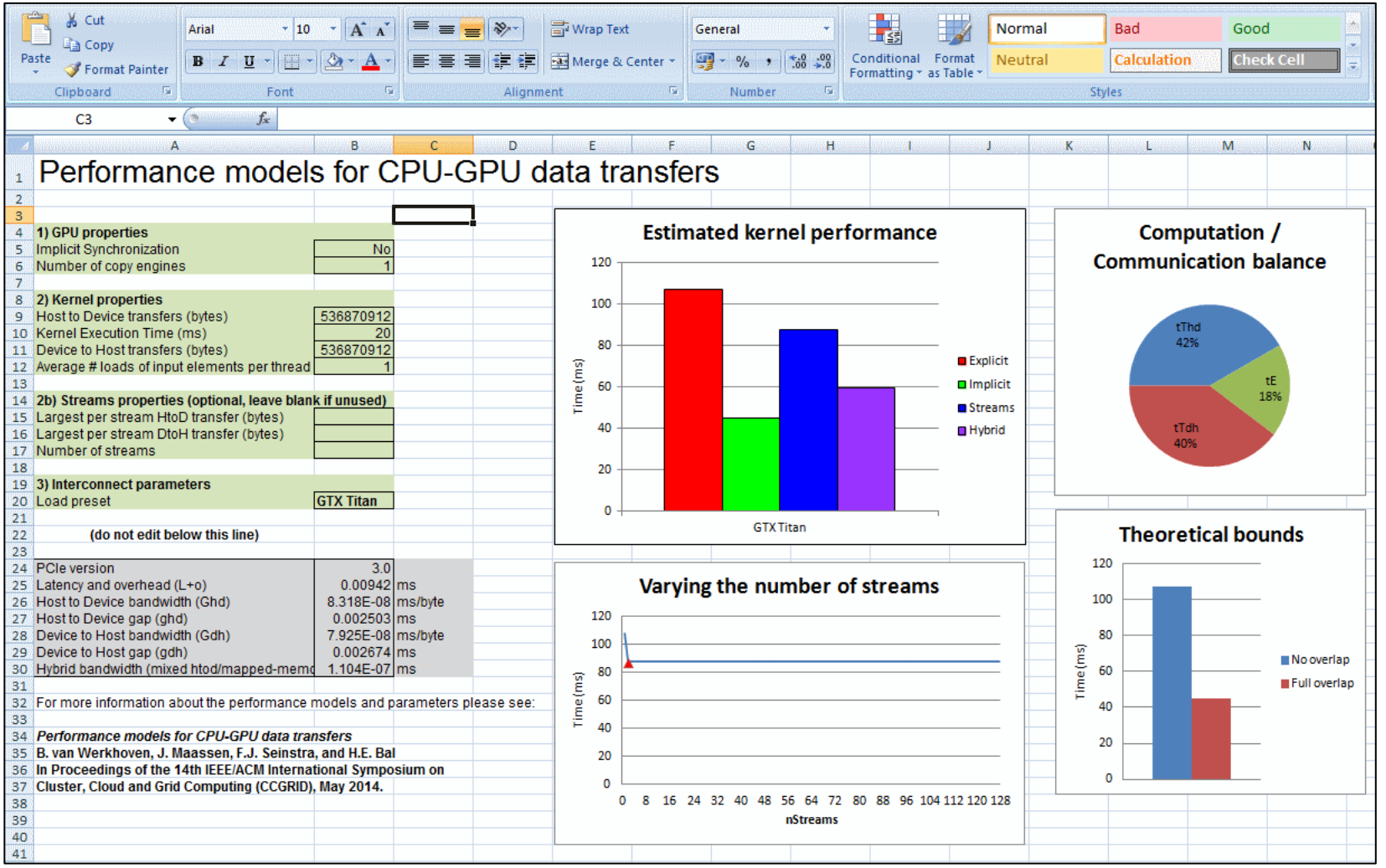
Different GPUs (state kernel)



Different GPUs (buoydiff)



Comes with spreadsheet



Performance models for CPU-GPU data transfers

1) GPU properties	
Implicit Synchronization	No
Number of copy engines	1
2) Kernel properties	
Host to Device transfers (bytes)	536870912
Kernel Execution Time (ms)	20
Device to Host transfers (bytes)	536870912
Average # loads of input elements per thread	1
2b) Streams properties (optional, leave blank if unused)	
Largest per stream HtoD transfer (bytes)	
Largest per stream DtoH transfer (bytes)	
Number of streams	
3) Interconnect parameters	
Load preset	GTX Titan
(do not edit below this line)	

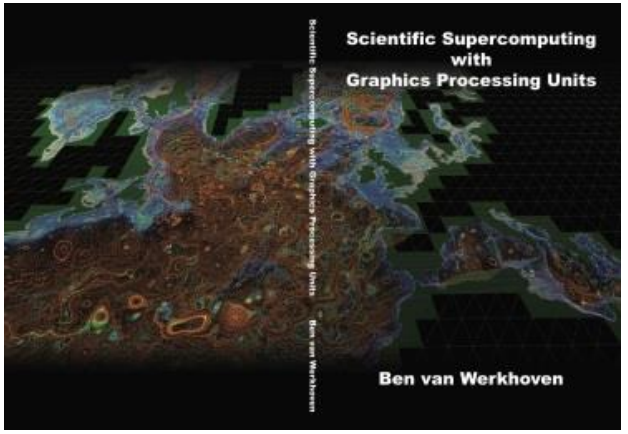
PCIe version	3.0	
Latency and overhead (L+o)	0.00942	ms
Host to Device bandwidth (Ghd)	8.318E-08	ms/byte
Host to Device gap (ghd)	0.002503	ms
Device to Host bandwidth (Gdh)	7.925E-08	ms/byte
Device to Host gap (gdh)	0.002674	ms
Hybrid bandwidth (mixed htod/mapped-memc	1.104E-07	ms

For more information about the performance models and parameters please see:

Performance models for CPU-GPU data transfers
B. van Werkhoven, J. Maassen, F.J. Seinstra, and H.E. Bal
In Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), May 2014.

Lessons learned

- **Everything must be in balance to obtain high performance**
  - **Subtle interactions between resource limits**
- **Runtime decision system (adaptive tiling), in combination with standard optimizations**
  - **Loop unrolling, memory bank conflicts**



Why is GPU programming hard?

- **Portability**
  - **Optimizations are architecture-dependent, and the architectures change frequently**
  - **Optimizations are often input dependent**
- **Finding the right parameters settings is difficult**
- **Need better performance models**
  - **Like Roofline and our I/O model**

Performance K-Means