# CUDA 6.0


Manuel Ujaldón

Associate Professor, Univ. of Malaga (Spain)

Conjoint Senior Lecturer, Univ. of Newcastle (Australia)

Nvidia CUDA Fellow


NVIDIA

# Acknowledgements

- To the great Nvidia people, for sharing with me ideas, material, figures, presentations, ... Particularly, for this presentation:
  - Mark Ebersole (webinars and slides):
    - CUDA 6.0 overview.
    - Optimizations for Kepler.
  - Mark Harris (SC'13 talk, webinar and "parallel for all" blog):
    - CUDA 6.0 announcements.
    - New hardware features in Maxwell.


2

# Talk contents [49 slides]

# I. The evolution of CUDA

NVIDIA

# The impressive evolution of CUDA

| Year 2008 | Year 2014 |
|---|---|
| **100.000.000** CUDA-capable GPUs | **500.000.000** CUDA-capable GPUs |
| **150.000** CUDA downloads | **2.100.000** CUDA downloads |
| **1** supercomputer | **52** supercomputers |
| **60** university courses | **780** courses |
| **4.000** academic papers | **40.000** academic papers |

- The CUDA software is downloaded once every minute.

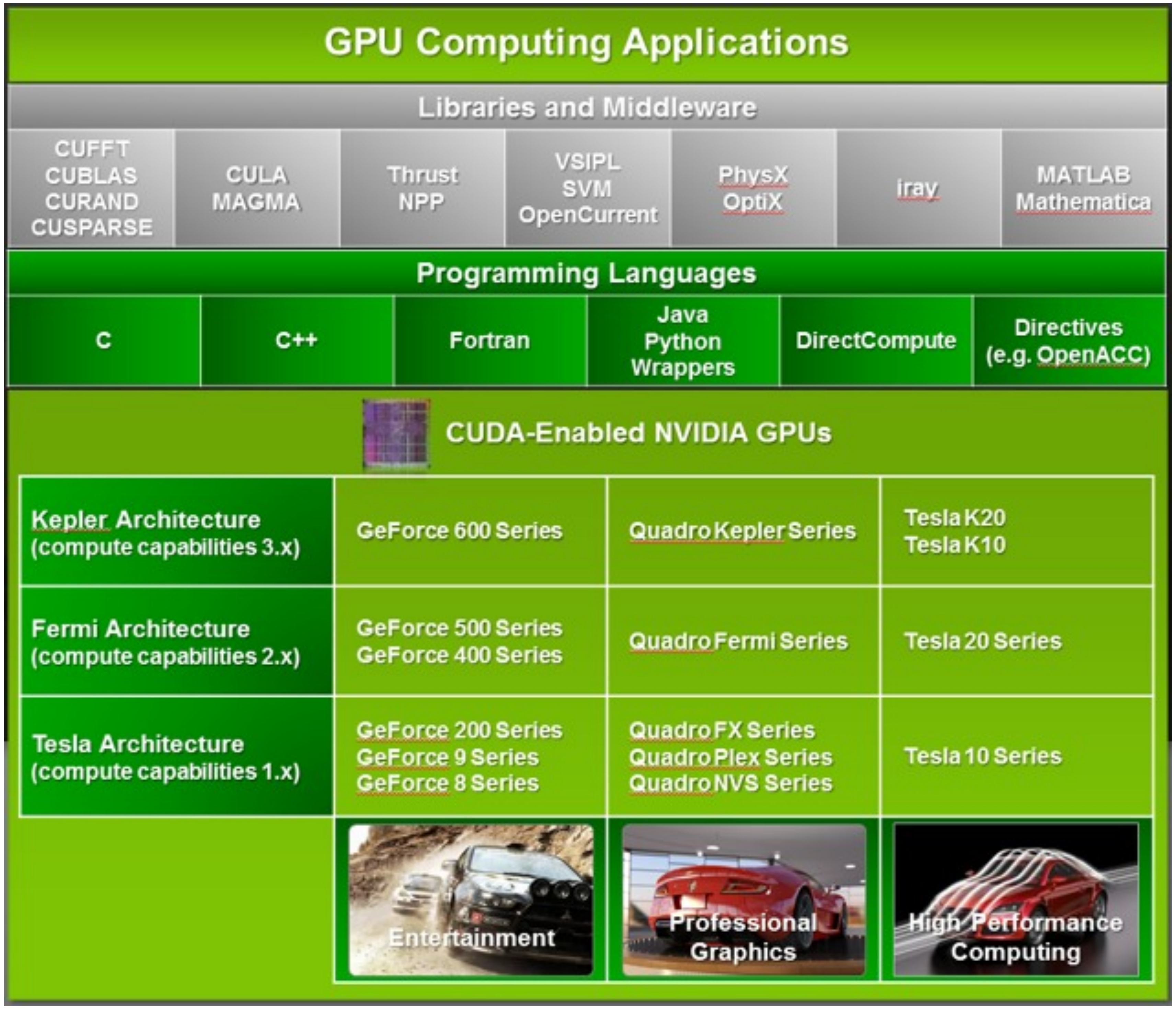
# Worldwide distribution of CUDA university courses

# Summary of GPU evolution

- 2001: First many-cores (vertex and pixel processors).
- 2003: Those processor become programmable (with Cg).
- 2006: Vertex and pixel processors unify.
- 2007: CUDA emerges.
- 2008: Double precision floating-point arithmetic.
- 2010: Operands are IEEE-normalized and memory is ECC.
- 2012: Wider support for irregular computing.
- 2014: The CPU-GPU memory space is unified.
- Still pending: Reliability in clusters and connection to disk.


7

# The CUDA family picture



# CUDA 5 highlights

- Dynamic Parallelism:
  - Spawn new parallel work from within GPU code (from GK110 on).
- GPU Object Linking:
  - Libraries and plug-ins for GPU code.
- New Nsight Eclipse Edition:
  - Develop, Debug, and Optimize... All in one tool!
- GPUDirect:
  - RDMA between GPUs and PCI-express devices.

- CUDA 5.5 is an intermediate step:
  - Smoothes the transition towards CUDA 6.0.

# CUDA 6 highlights

- Unified Memory:
  - CPU and GPU can share data without much programming effort.
- Extended Library Interface (XT) and Drop-in Libraries:
  - Libraries much easier to use.
- GPUDirect RDMA:
  - A key achievement in multi-GPU environments.
- Developer tools:
  - Visual Profiler enhanced with:
    - Side-by-side source and disassembly view showing.
    - New analysis passes (per SM activity level), generates a kernel analysis report.
  - Multi-Process Server (MPS) support in nvprof and cuda-memcheck.
  - Nsight Eclipse Edition supports remote development (x86 and ARM).

# II. CUDA 6.0 support (operating systems and platforms)

# Operating systems

- Windows:
  - XP, Vista, 7, 8, 8.1, Server 2008 R2, Server 2012.
  - Visual Studio 2008, 2010, 2012, 2012 Express.
- Linux:
  - Fedora 19.
  - RHEL & CentOS 5, 6.
  - OpenSUSE 12.3.
  - SUSE SLES 11 SP2, SP3.
  - Ubuntu 12.04 LTS (including ARM cross and native), 13.04.
  - ICC 13.1.
- Mac:
  - OSX 10.8, 10.9.

# Platforms (depending on OS). CUDA 6 Production Release

- https://developer.nvidia.com/cuda-downloads

Windows | Linux | Mac OSX

| Version | | 64-bit | 32-bit |
|---|---|---|---|
| Windows 8.1<br>Windows 7<br>Windows Vista | Notebook | EXE | EXE |
| | Desktop | EXE | EXE |
| Windows XP | Desktop | EXE | EXE |

Getting Started Guide

Windows | Linux | Mac OSX

| Distribution | x86 64-bit | x86 32-bit | ARMv7 |
|---|---|---|---|
| Fedora 19 | RPM RUN | | |
| OpenSUSE 12.3 | RPM RUN | | |
| RHEL 6<br>CentOS 6 | RPM RUN | | |
| RHEL 5<br>CentOS 5 | RUN | | |
| SLES 11 (SP2 & SP3) | RPM RUN | | |
| Ubuntu 13.04 | DEB RUN | RUN | DEB |
| Ubuntu 12.04 | DEB* RUN | DEB RUN | |
| L4T, Linux for Tegra | | LINK | LINK |

Getting Started Guide

Windows | Linux | Mac OSX

| OSX Release | Package |
|---|---|
| 10.8 | PKG |
| 10.9 | |

Getting Started Guide


13

# GPUs for CUDA 6.0

- CUDA Compute Capabilities 3.0 (sm_30, 2012 versions of Kepler like Tesla K10, GK104):
  - Do not support dynamic parallelism nor Hyper-Q.
  - Support unified memory with a separate pool of shared data with auto-migration (a subset of the memory which has many limitations).
- CUDA Compute Capabilities 3.5 (sm_35, 2013 and 2014 versions of Kepler like Tesla K20, K20X and K40, GK110):
  - Support dynamic parallelism and Hyper-Q.
  - Support unified memory, with similar restrictions than CCC 3.0.
- CUDA Compute Capabilities 5.0 (sm_50, 2014 versions of Maxwell like GeForce GTX750Ti, GM107-GM108):
  - Full support of dynamic parallelism, Hyper-Q and unified memory.

# Deprecations

- Things that tend to be obsolete:
  - Still supported.
  - Not recommended.
  - New developments may not work with it.
  - Likely to be dropped in the future.

- Some examples:
  - 32-bit applications on x86 Linux (toolkit & driver).
  - 32-bit applications on Mac (toolkit & driver).
  - G80 platform / sm_10 (toolkit).

# Dropped support

- cuSPARSE “Legacy” API.
- Ubuntu 10.04 LTS (toolkit & driver).
- SUSE Linux Enterprise Server 11 SP1 (toolkit & driver).
- Mac OSX 10.7 (toolkit & driver).

- Mac Models with the MCP79 Chipset (driver)
  - iMac: 20-inch (early ’09), 24-inch (early ’09), 21.5-inch (late ’09).
  - MacBook Pro: 15-inch (late’08), 17-inch (early’09), 17-inch (mid’09), 15-inch (mid ’09), 15-inch 2.53 GHz (mid’09), 13-inch (mid’09).
  - Mac mini: Early ’09, Late ’09.
  - MacBook Air (Late ’08, Mid ’09).


16

# III. Compiling and linking

# CUDA 4.0: Whole-program compilation and linking

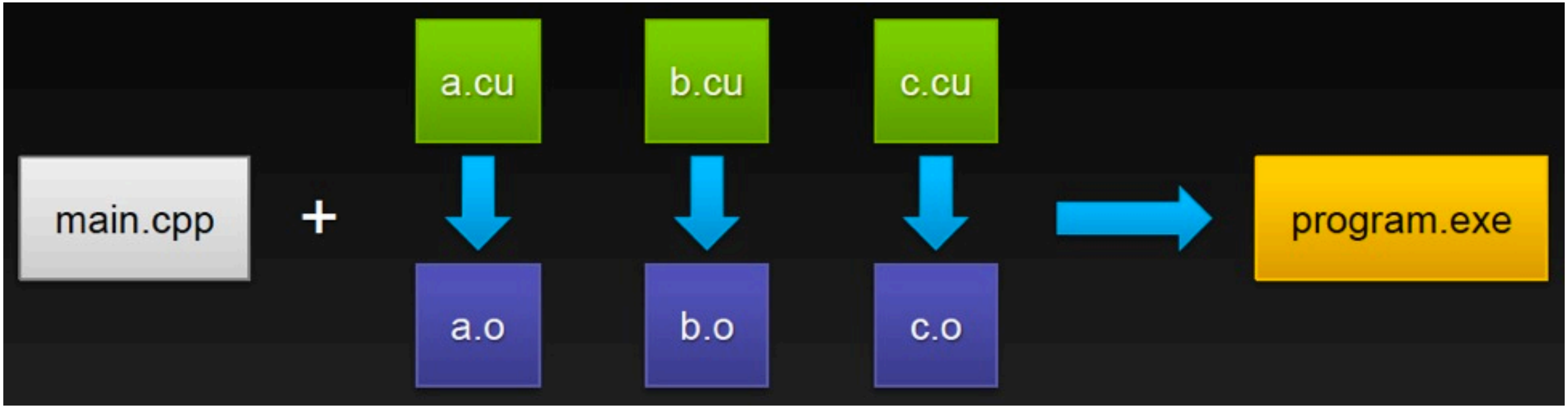
- CUDA 4 required a single source file for a single kernel. It was not possible to link enternal device code.


Include files together to build


18

# CUDA 5.0: Separate Compilation & Linking

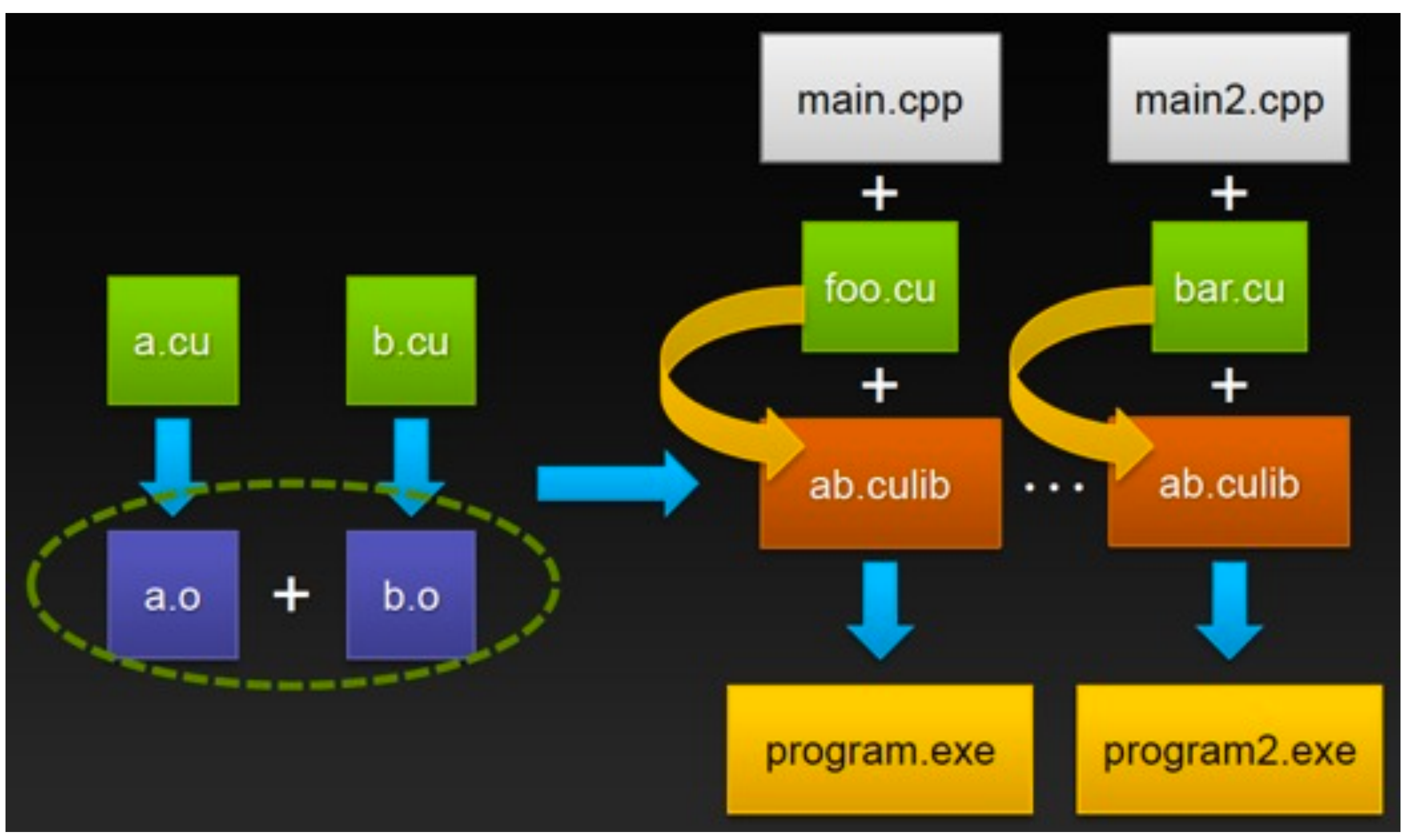
- Now it is possible to compile and link each file separately:
  - That way, we can build multiple object files independently, which can later be linked to build the executable file.

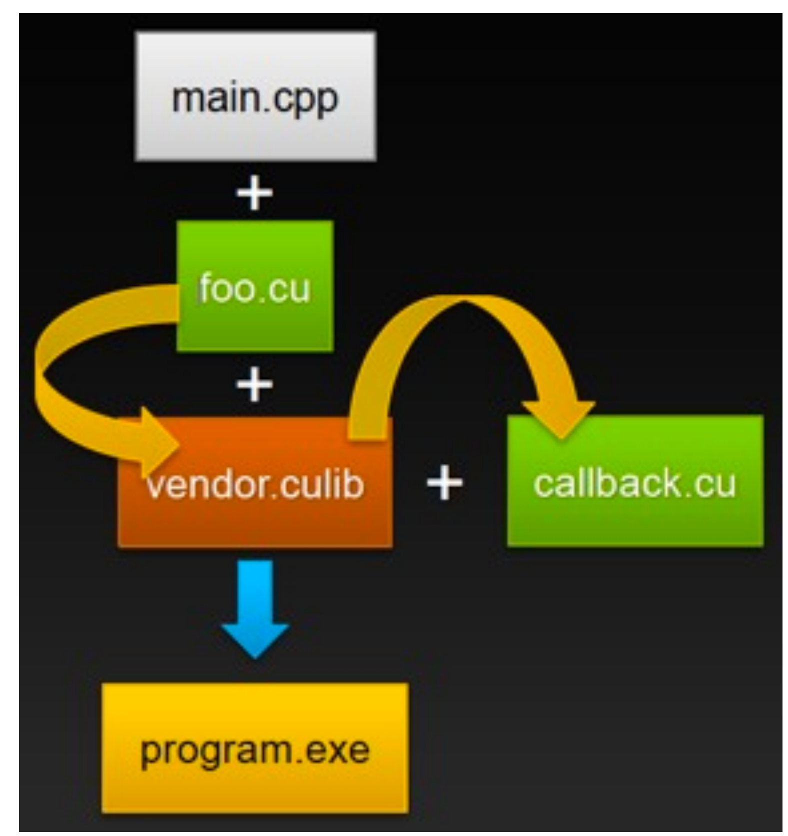


# CUDA 5.0: Separate Compilation & Linking

- We can also combine object files into static libraries, which can be shared from different source files when linking:
  - To facilitate code reuse.
  - To reduce the compilation time.

- This also enables closed-source device libraries to call user-defined device callback functions.



# IV. Dynamic parallelism in CUDA 5 & 6

# Dynamic parallelism allows CUDA 5.0 to improve three primary issues:

Execution

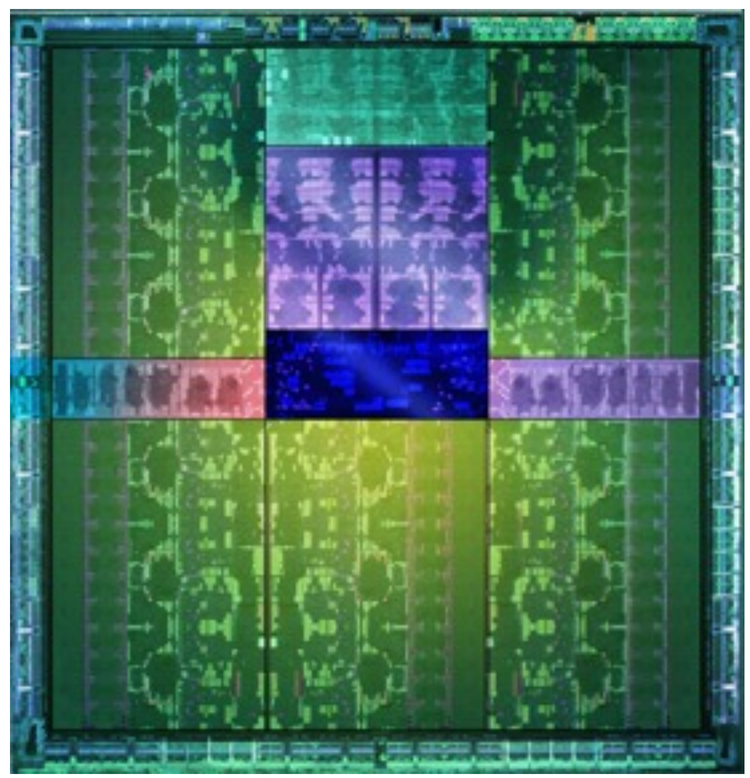
- Data-dependent execution
- Recursive parallel algorithms

Performance

- Dynamic load balancing
- Thread scheduling to help fill the GPU

Programmability

- Library calls from GPU kernels
- Simplify CPU/GPU division

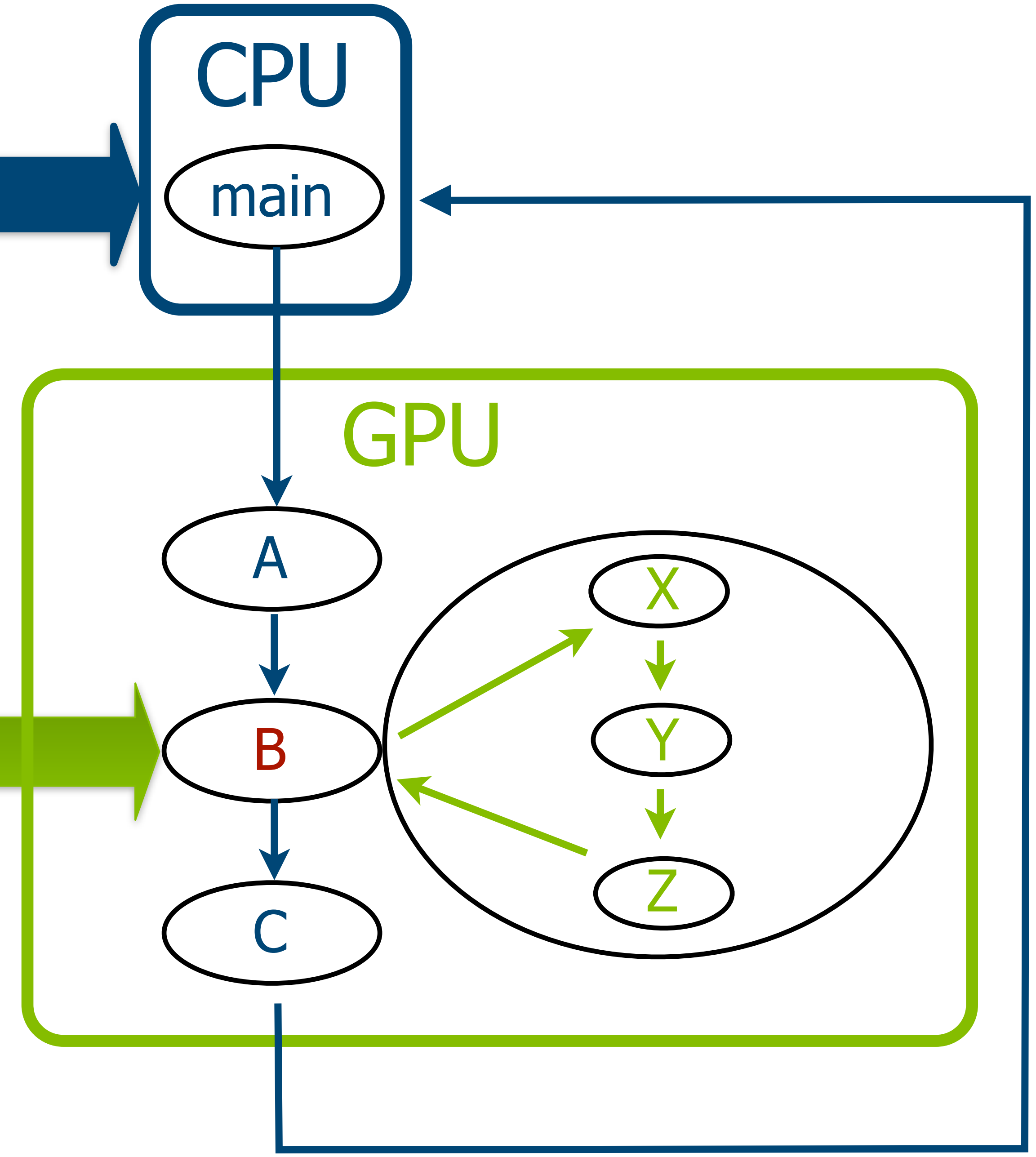



22

# Familiar syntax and programming model

```
int main() {
  float *data;
  setup(data);

  A <<< ... >>> (data);
  B <<< ... >>> (data);
  C <<< ... >>> (data);
  cudaDeviceSynchronize();
  return 0;
}
```

```
__global__ void B(float *data) {
  do_stuff(data);

  X <<< ... >>> (data);
  Y <<< ... >>> (data);
  Z <<< ... >>> (data);
  cudaDeviceSynchronize();

  do_more_stuff(data);
}
```





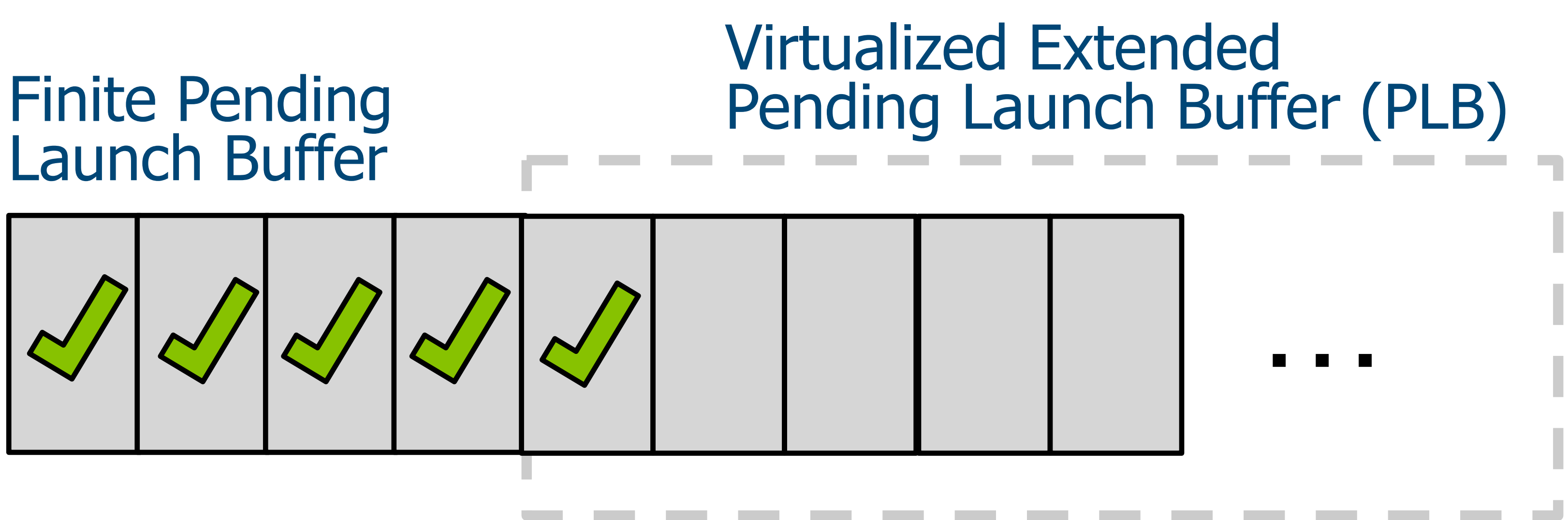

23

# Before CUDA 6.0: Tight limit on Pending Launch Buffer (PLB)

- Applications using dynamic parallelism can launch too many grids and exhaust the pre-allocated pending launch buffer (PLB).
  - Result in launch failures, sometimes intermittent due to scheduling.
  - PLB size tuning can fix the problem, but often involves trial-and-error.

# CUDA 6.0 uses an extended PLB (EPLB)

- EPLB guarantees all launches succeed by using a lower performance virtualized launch buffer, when fast PLB is full.
  - No more launch failures regardless of scheduling.
  - PLB size tuning provides direct performance improvement path.
  - Enabled by default.



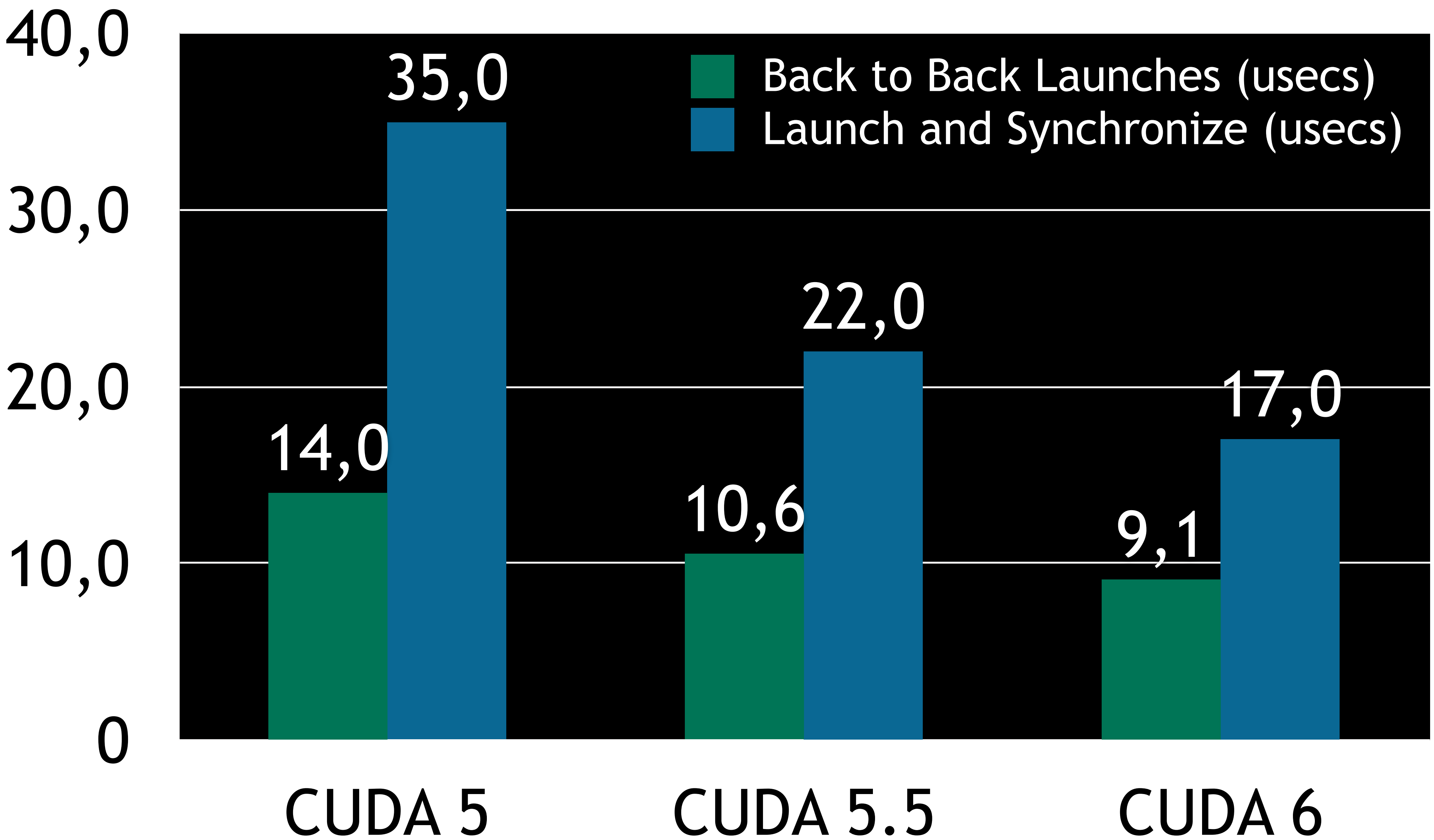


25

# CUDA 6.0: Performance improvements in key use cases

- Kernel launch.
- Repeated launch of the same set of kernels.
- `cudaDeviceSynchronize()`.
- Back-to-back grids in a stream.

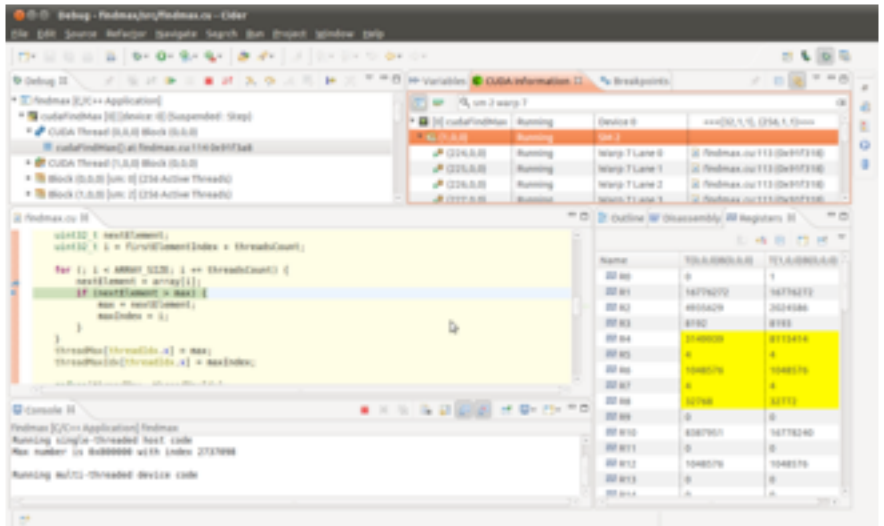
# Performance improvements on dynamic parallelism

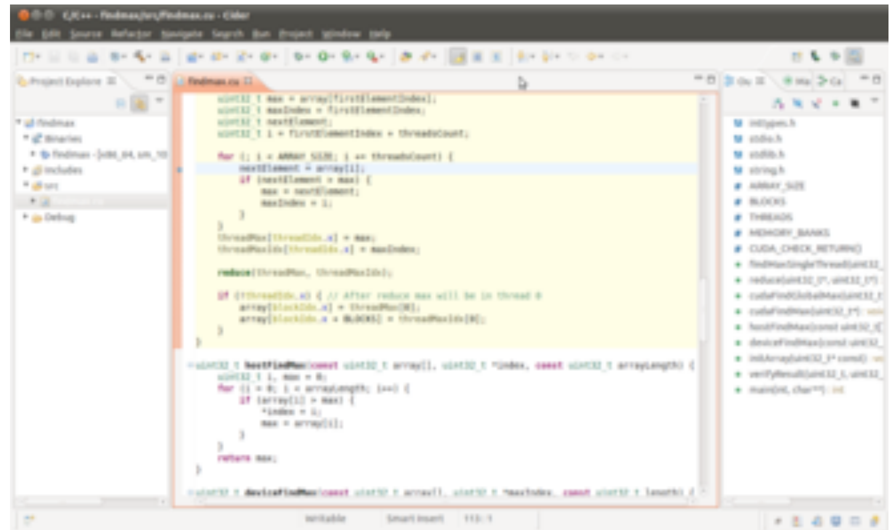


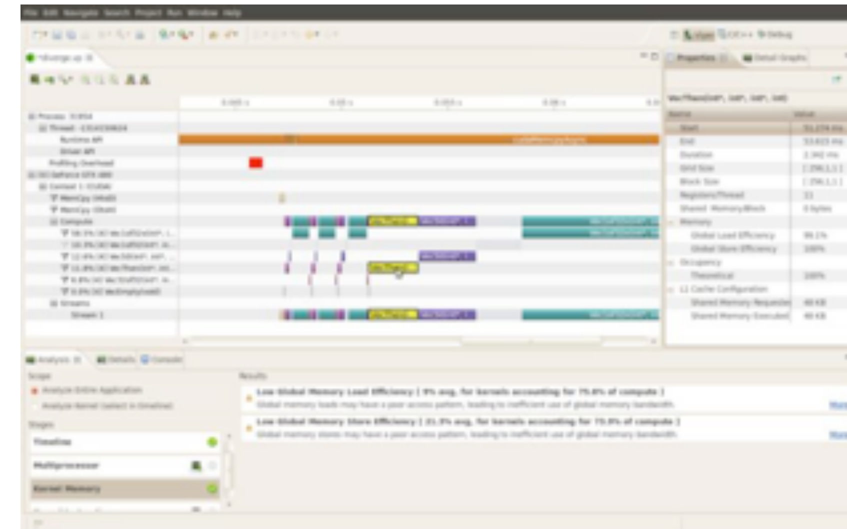
# V. New tools for development, debugging and optimization

NVIDIA

# New features in Nvidia Nsight, Eclipse Edition, also available for Linux and Mac OS





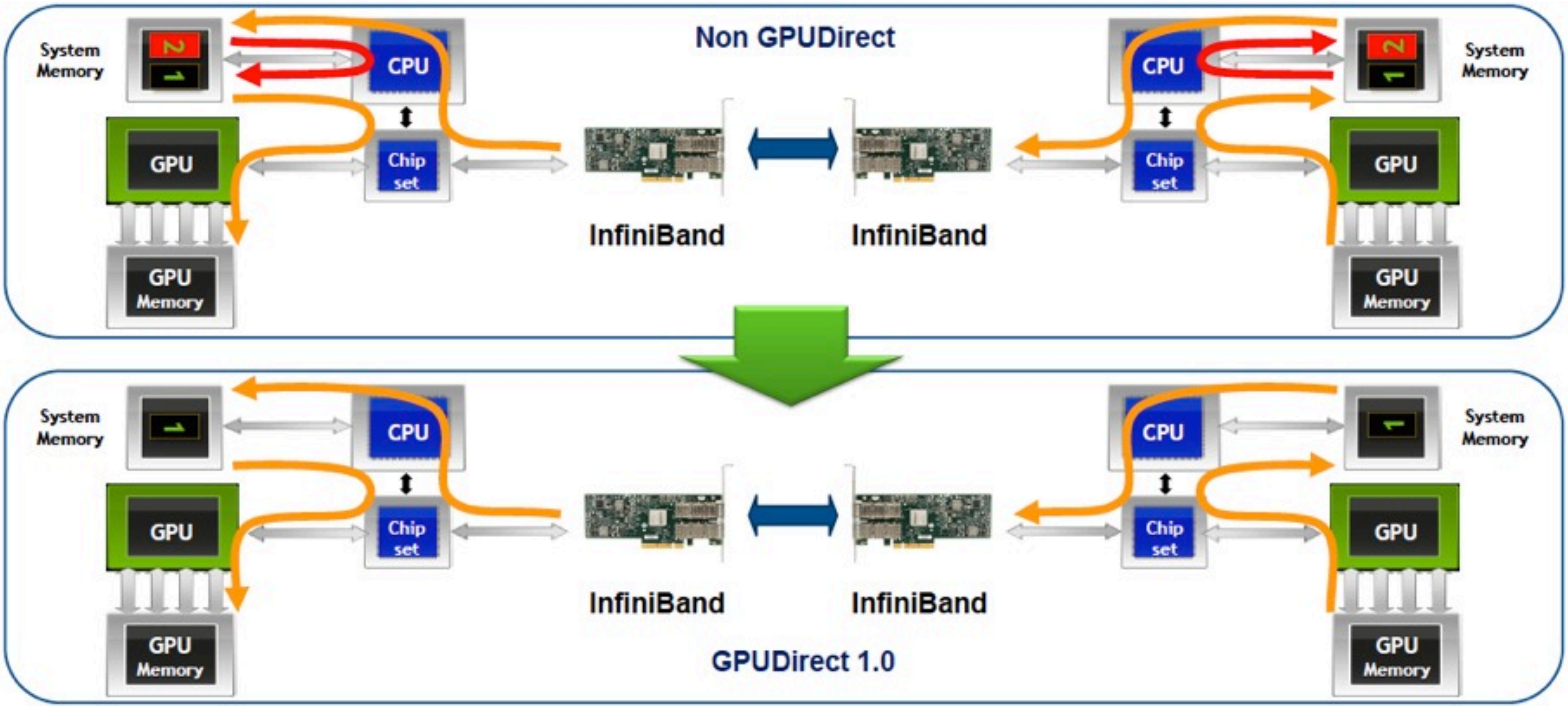


- CUDA-aware editor:
  - Automated CPU to GPU code refactoring.
  - Semantic highlight-ing of CUDA code.
  - Integrated code samples & docs.
- Nsight debugger
  - Simultaneously debugging of CPU and GPU code.
  - Inspect variables across CUDA threads.
  - Use breakpoints & single step debugging.
- Nsight profiler
  - Quickly identifies bottlenecks in source lines and using a unified CPU-GPU trace.
  - Integrated expert system.
  - Fast edit-build-profile optimization cycle.

# VI. GPU Direct

# Communication among GPU memories

- GPU Direct 1.0 was released in Fermi to allow communications among GPUs within CPU clusters.

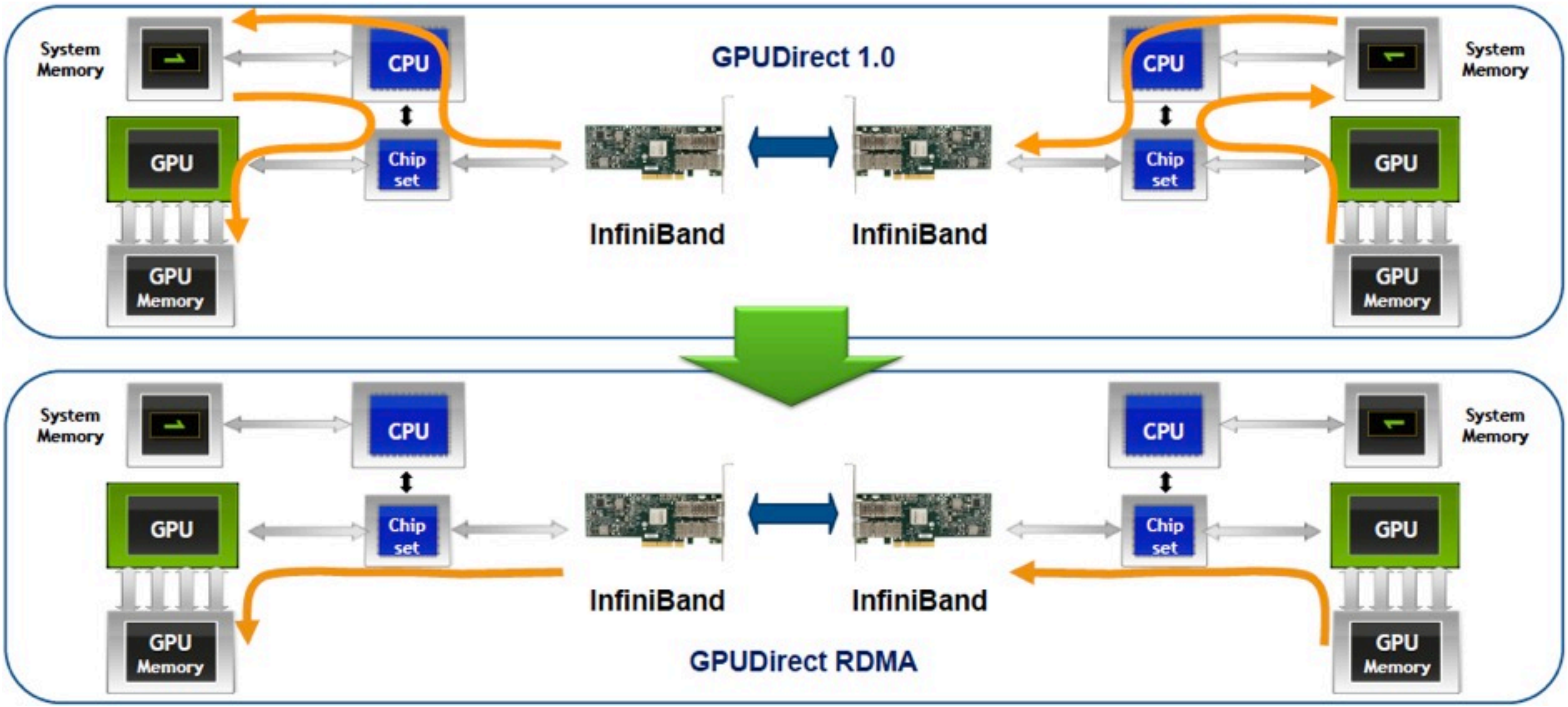



Receiver

Sender

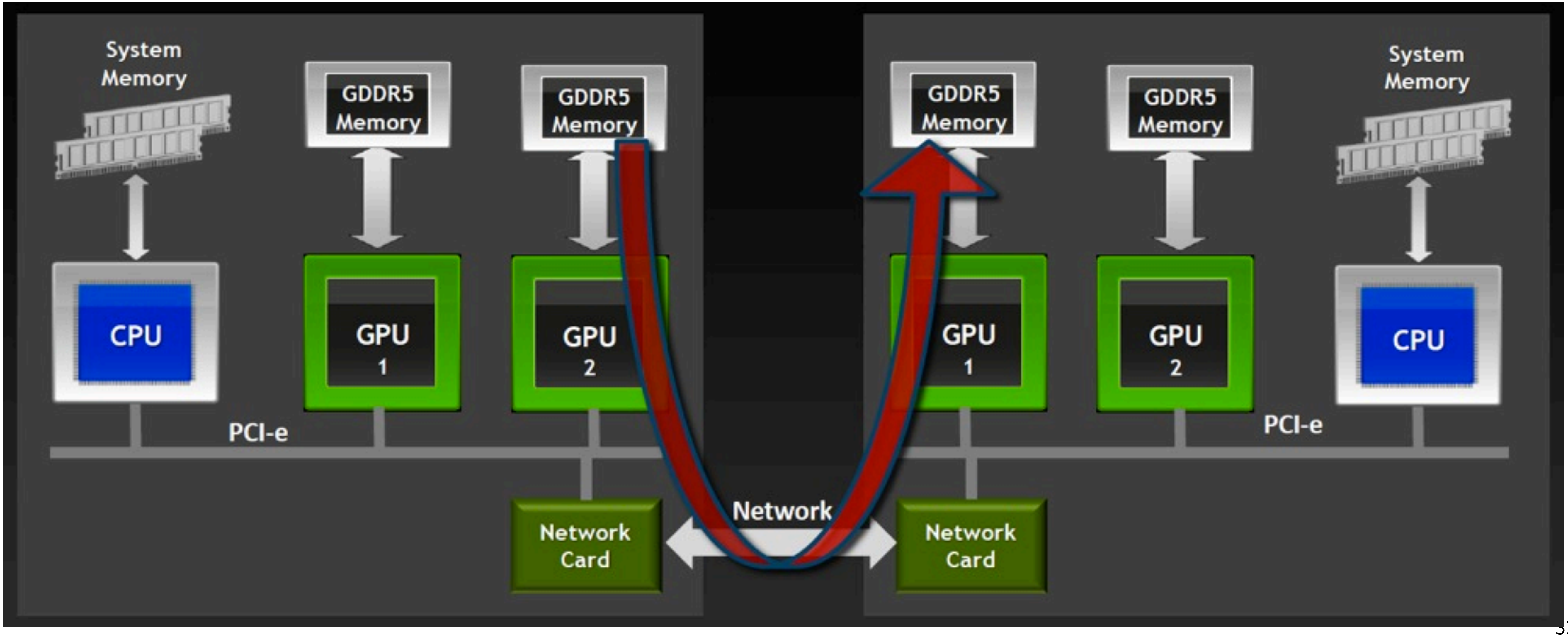
# Kepler + CUDA 5 support GPUDirect-RDMA [Remote Direct Memory Access]

- This allows a more direct transfer between GPUs.
- Usually, the link is PCI-express or InfiniBand.

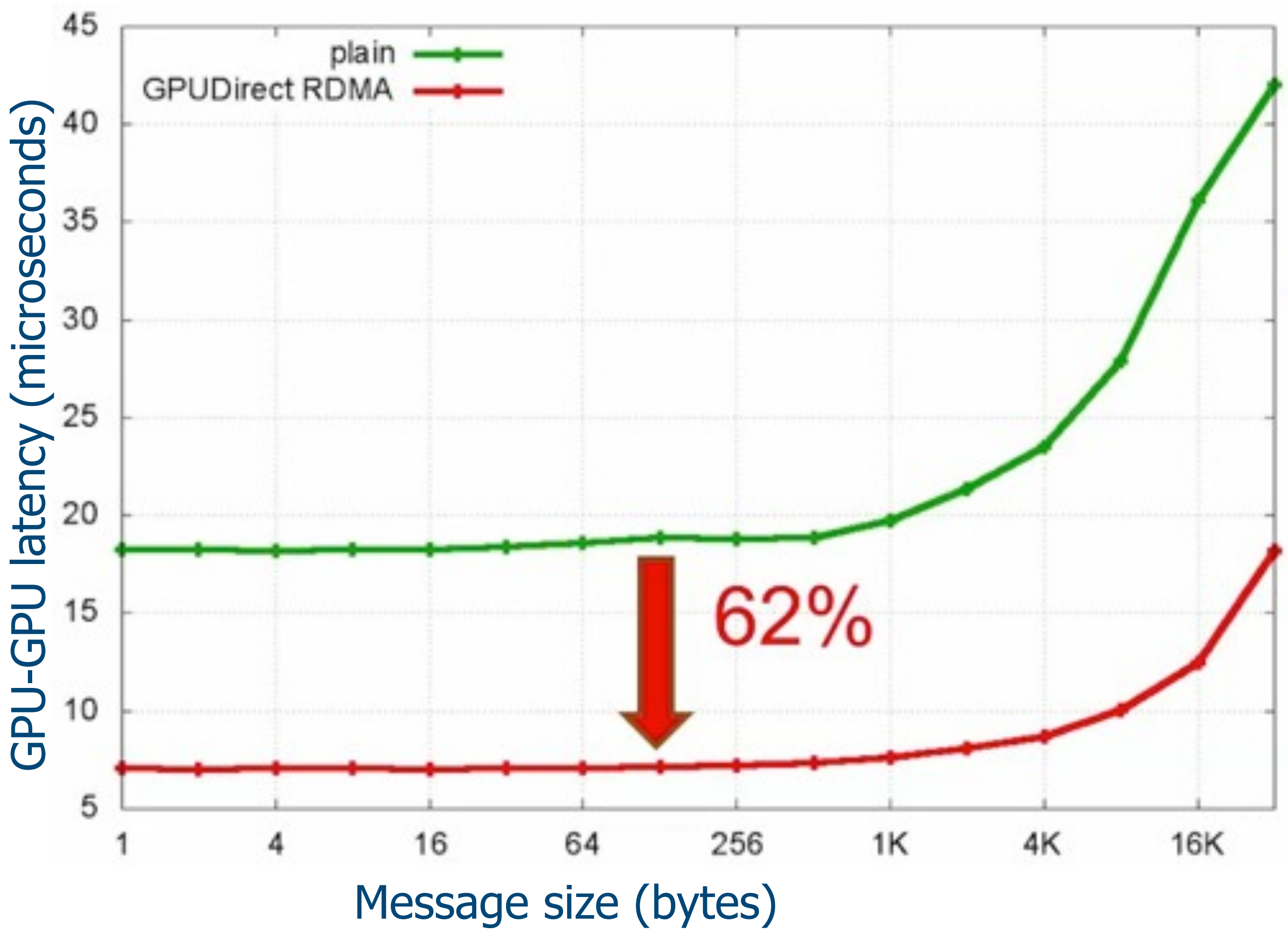


# GPUDirect-RDMA in Maxwell

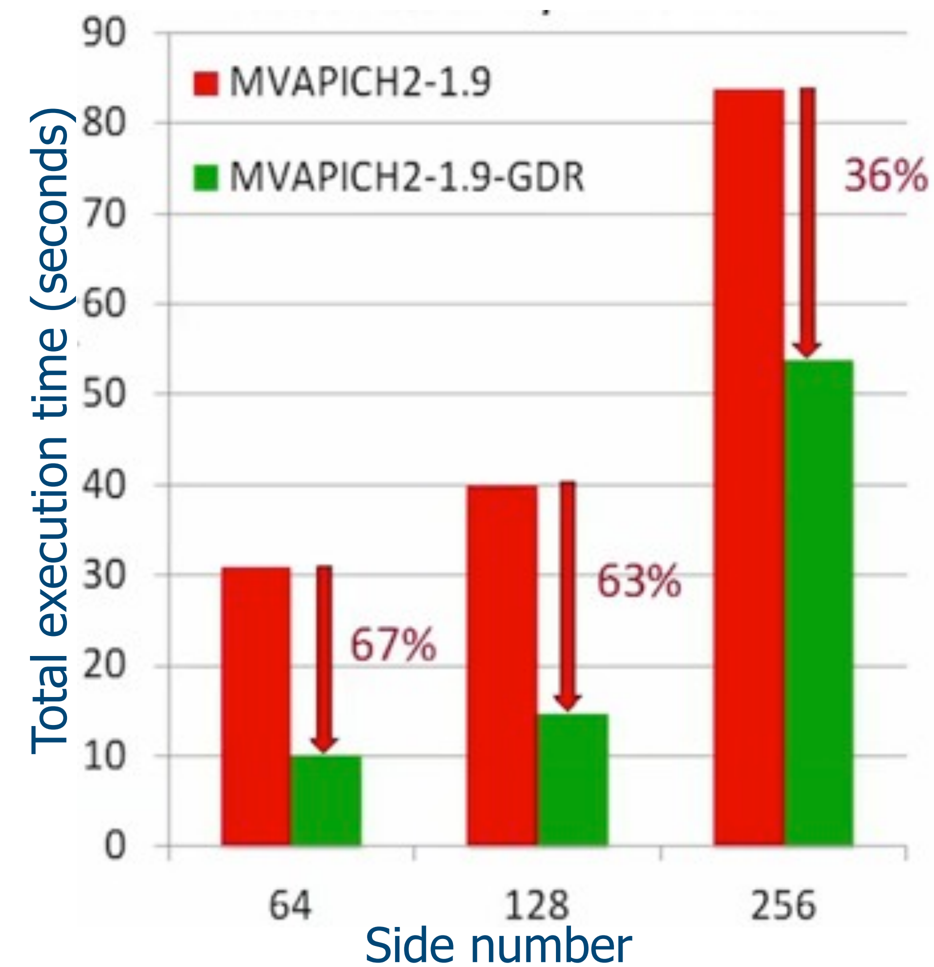
- The situation is more complex in CUDA 6.0 with unified memory.



# Preliminary results using GPUDirect-RDMA (better perf. ahead w. CUDA 6.0 & OpenMPI)



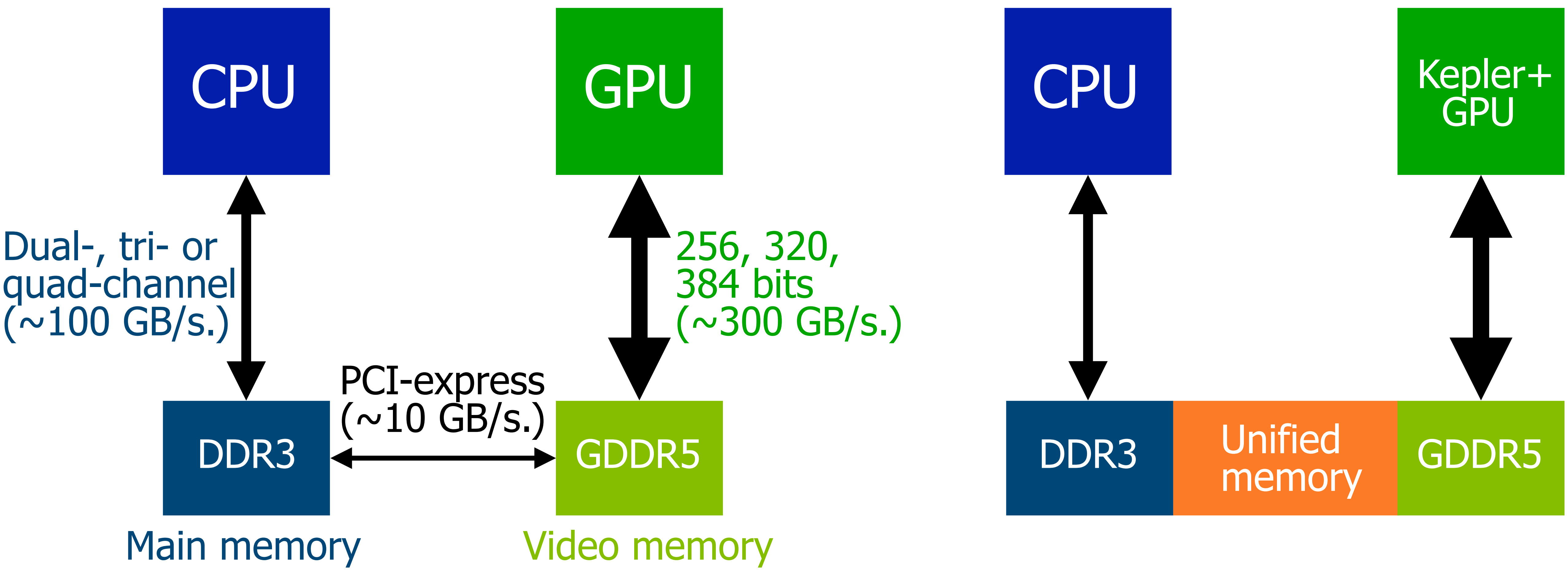

- Inter-node latency using:
  - Tesla K40m GPUs (no GeForces).
  - MPI MVAPICH2 library.
  - ConnectX-3, IVB 3GHz.




- Better MPI Applic. Scaling:
  - Code: HSG (bioinformatics).
  - 2 GPU nodes.
  - 4 MPI processes each node.

# VII. Unified memory

# The idea


CPU
GPU
CPU
Kepler+ GPU
Dual-, tri- or quad-channel (~100 GB/s.)
256, 320, 384 bits (~300 GB/s.)
PCI-express (~10 GB/s.)
DDR3
GDDR5
DDR3
Unified memory
GDDR5
Main memory
Video memory

# Unified memory contributions

- Simpler programming and memory model:
  - Single pointer to data, accessible anywhere.
  - Eliminate need for `cudaMemcpy()`.
  - Greatly simplifies code porting.
- Performance through data locality:
  - Migrate data to accessing processor.
  - Guarantee global coherency.
  - Still allows `cudaMemcpyAsync()` hand tuning.


37

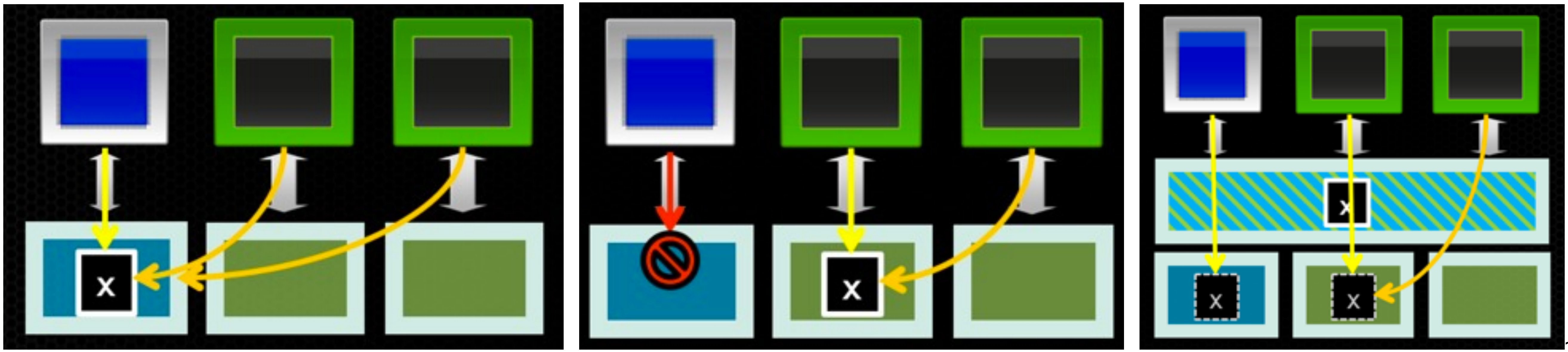
# System requirements

| | Required | Limitations |
|---|---|---|
| GPU | Kepler (GK10x+) or Maxwell (GM10x+) | Limited performance in CCC 3.0 and CCC 3.5 |
| Operating System | 64 bits | |
| Windows | 7 or 8 | WDDM & TCC no XP/Vista |
| Linux | Kernel 2.6.18+ | All CUDA-supported distros, not ARM |
| Linux on ARM | ARM64 | |
| Mac OSX | Not supported in CUDA 6.0 | |


38

# CUDA memory types

| | Zero-Copy (pinned memory) | Unified Virtual Addressing | Unified Memory |
|---|---|---|---|
| CUDA call | cudaMallocHost(&A, 4); | cudaMalloc(&A, 4); | cudaMallocManaged(&A, 4); |
| Allocation fixed in | Main memory (DDR3) | Video memory (GDDR5) | Both |
| Local access for | CPU | Home GPU | CPU and home GPU |
| PIC-e access for | All GPUs | Other GPUs | Other GPUs |
| Other features | Avoid swapping to disk | No CPU access | On access CPU/GPU migration |
| Coherency | At all times | Between GPUs | Only at launch & sync. |
| Full support in | CUDA 2.2 | CUDA 1.0 | CUDA 6.0 |



# Additions to the CUDA API

- New call: **`cudaMallocManaged()`**
  - Drop-in replacement for cudaMalloc() allocates managed memory.
  - Returns pointer accessible from both Host and Device.
- New call: **`cudaStreamAttachMemAsync()`**
  - Manages concurrently in multi-threaded CPU applications.
- New keyword: **`__managed__`**
  - Global variable annotation combines with **`__device__`**.
  - Declares global-scope migratable device variable.
  - Symbol accessible from both GPU and CPU code.



40

# A preliminar example: Sorting the elements from a file

| CPU code in C | GPU code in CUDA 6.0 |
|---|---|

**CPU code in C**

```
void sortfile (FILE *fp, int N) {
  char *data;
  data = (char *) malloc)N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

**GPU code in CUDA 6.0**

```
void sortfile (FILE *fp, int N) {
  char *data;
  cudaMallocManaged(&data, N);

  fread(data, 1, N, fp);

  qsort<<<...>>> (data, N, 1, compare);
  cudaDeviceSynchronize();

  use_data(data);

  cudaFree(data);
}
```

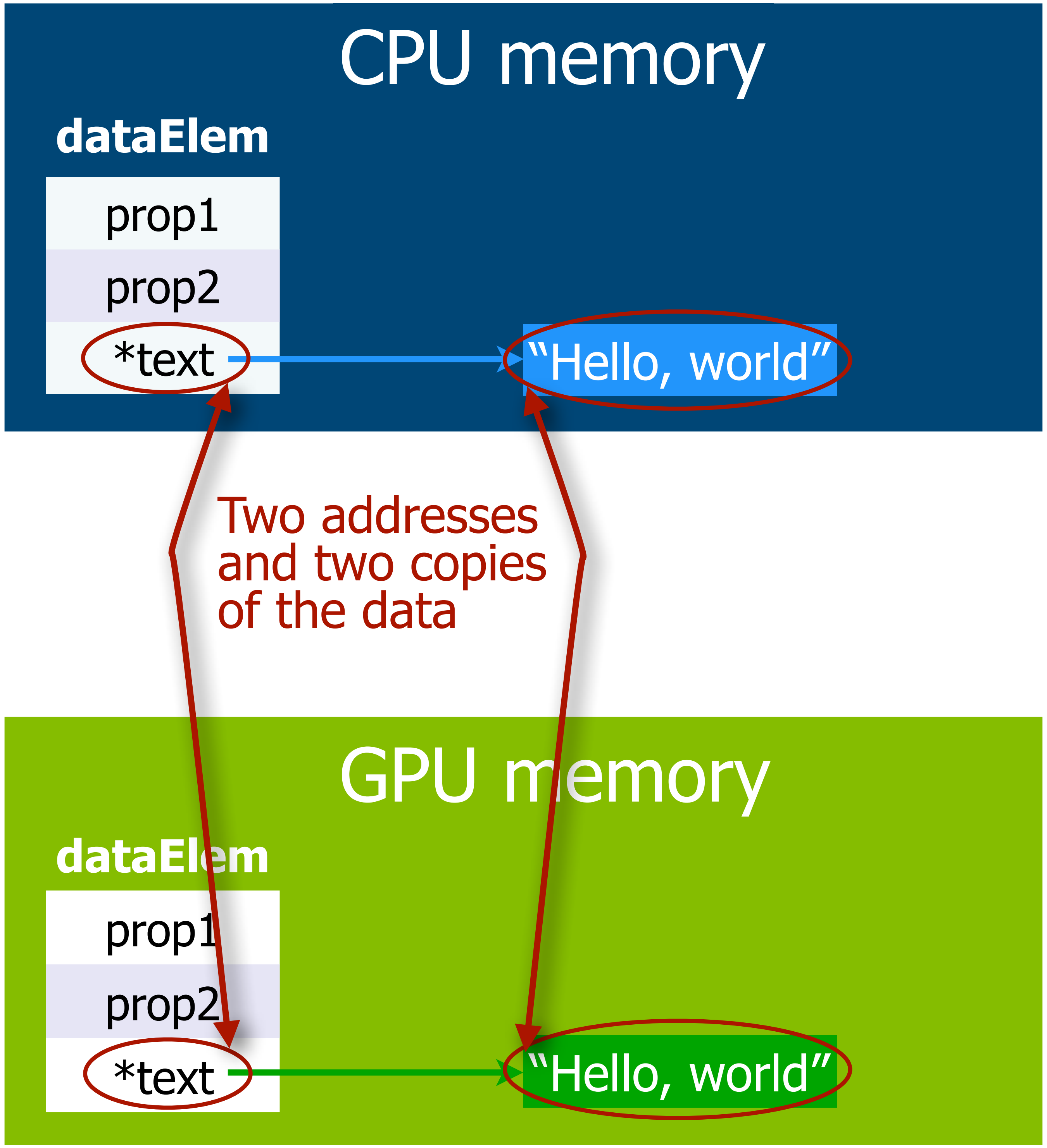


41

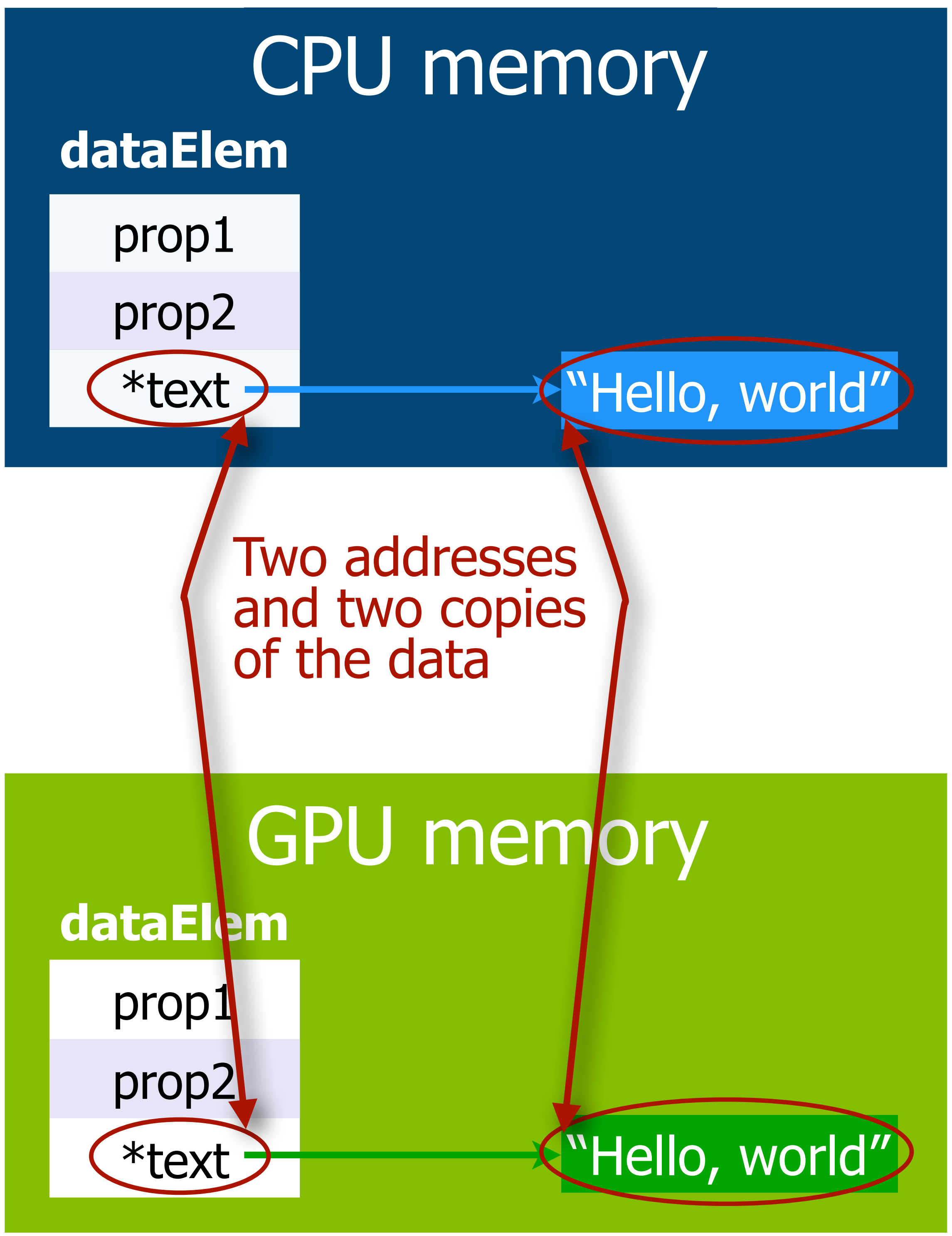
# Before unified memory

```
struct dataElem {
  int prop1;
  int prop2;
  char *text;
}
```

- A "deep copy" is required:
  - We must copy the structure and everything that it points to. This is why C++ invented the copy constructor.
  - CPU and GPU cannot share a copy of the data (coherency). This prevents `memcpy` style comparisons, checksumming and other things.

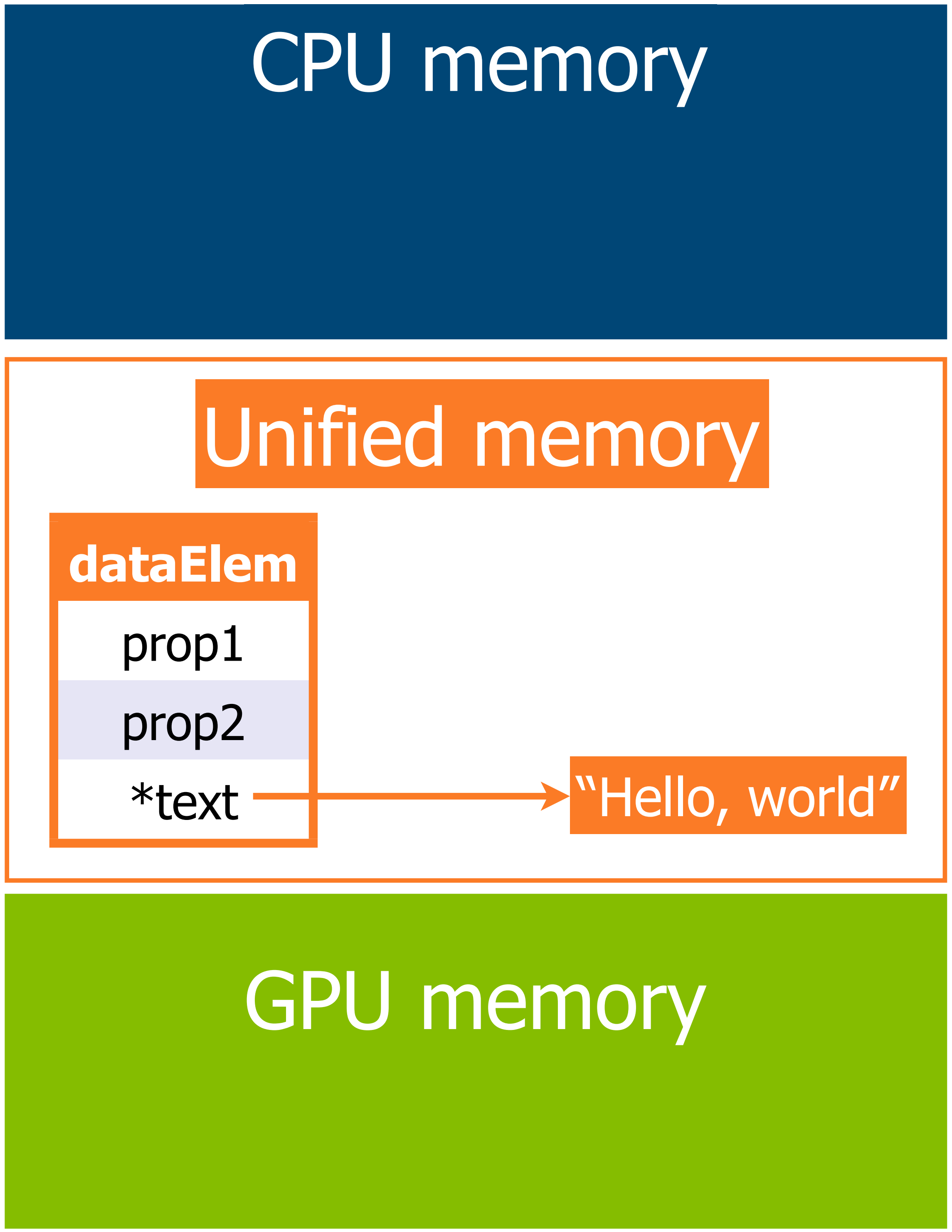


# The code required without unified memory




```
void launch(dataElem *elem) {
  dataElem *g_elem;
  char *g_text;

  int textlen = strlen(elem->text);

  // Allocate storage for struct and text
  cudaMalloc(&g_elem, sizeof(dataElem));
  cudaMalloc(&g_text, textlen);

  // Copy up each piece separately, including
new "text" pointer value
  cudaMemcpy(g_elem, elem, sizeof(dataElem));
  cudaMemcpy(g_text, elem->text, textlen);
  cudaMemcpy(&(g_elem->text), &g_text,
                              sizeof(g_text));

  // Finally we can launch our kernel, but
  // CPU and GPU use different copies of "elem"
  kernel<<< ... >>>(g_elem);
}
```

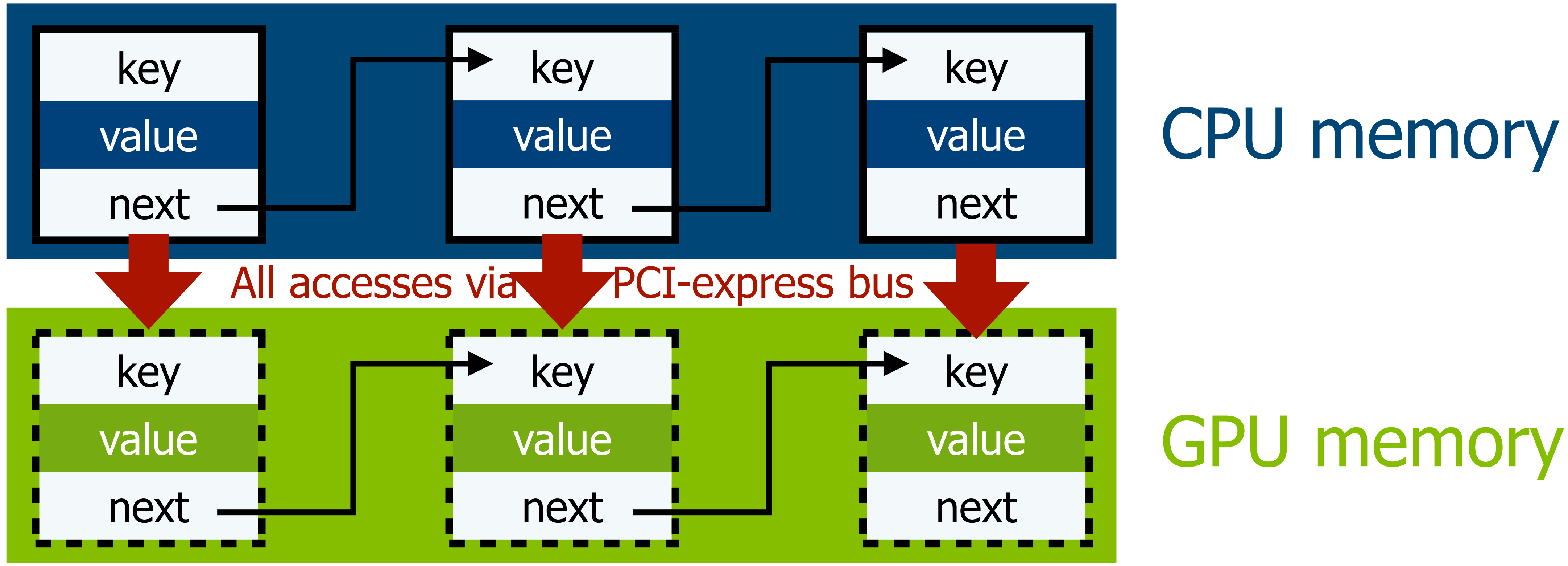
# The code required WITH unified memory




```
void launch(dataElem *elem) {
    kernel<<< ... >>>(elem);
}
```

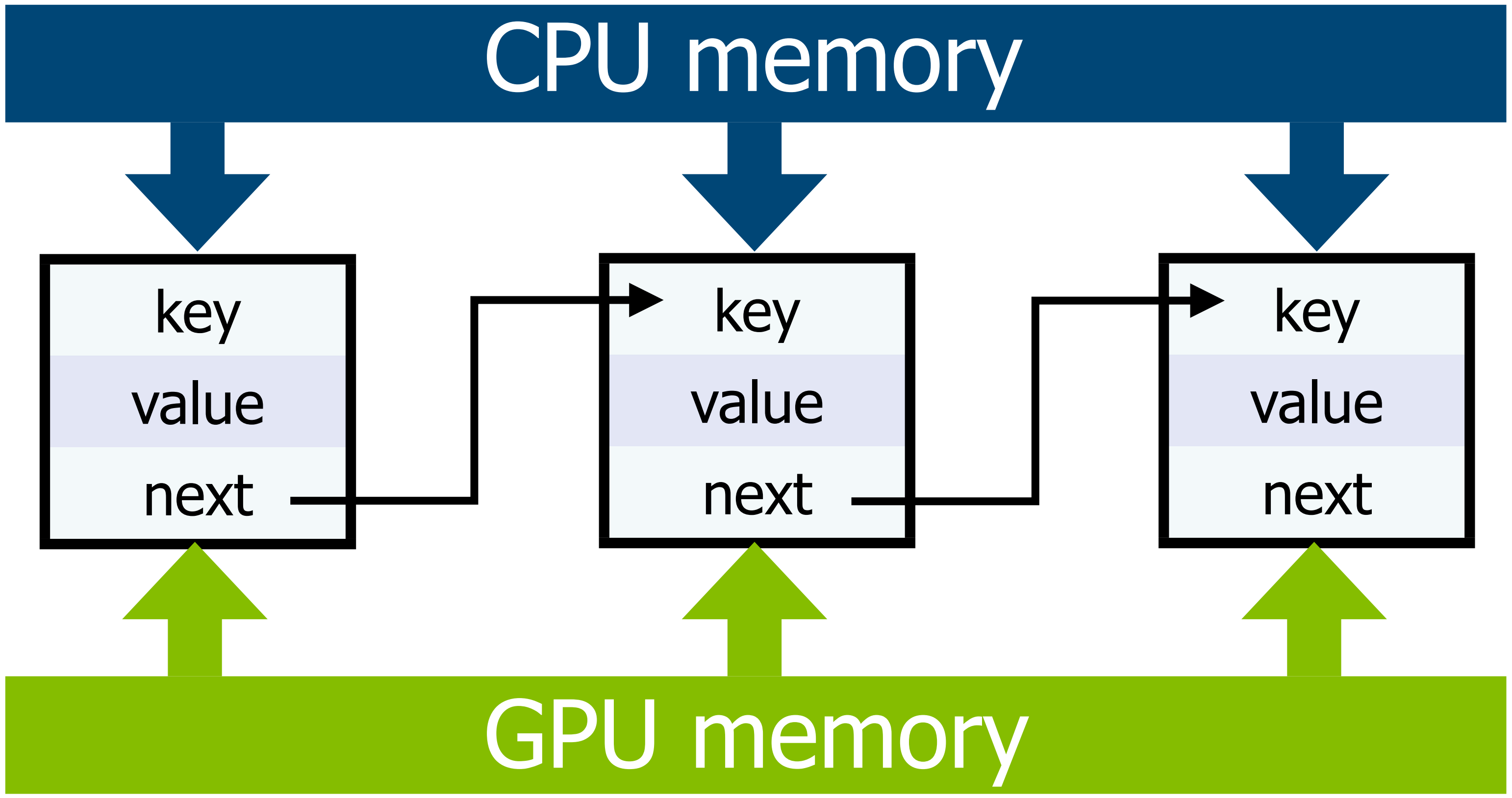
- What remains the same:
  - Data movement.
  - GPU accesses a local copy of text.
- What has changed:
  - Programmer sees a single pointer.
  - CPU and GPU both reference the same object.
  - There is coherence.
- To pass-by-reference vs. pass-by-value you need to use C++.

# An example: Linked lists




- Almost impossible to manage in the original CUDA API.
- The best you can do is use pinned memory:
  - Pointers are global: Just as unified memory pointers.
  - Performance is low: GPU suffers from PCI-e bandwidth.
  - GPU latency is very high, which is particularly important for linked lists because of the intrinsic pointer chasing.

# Linked lists with unified memory

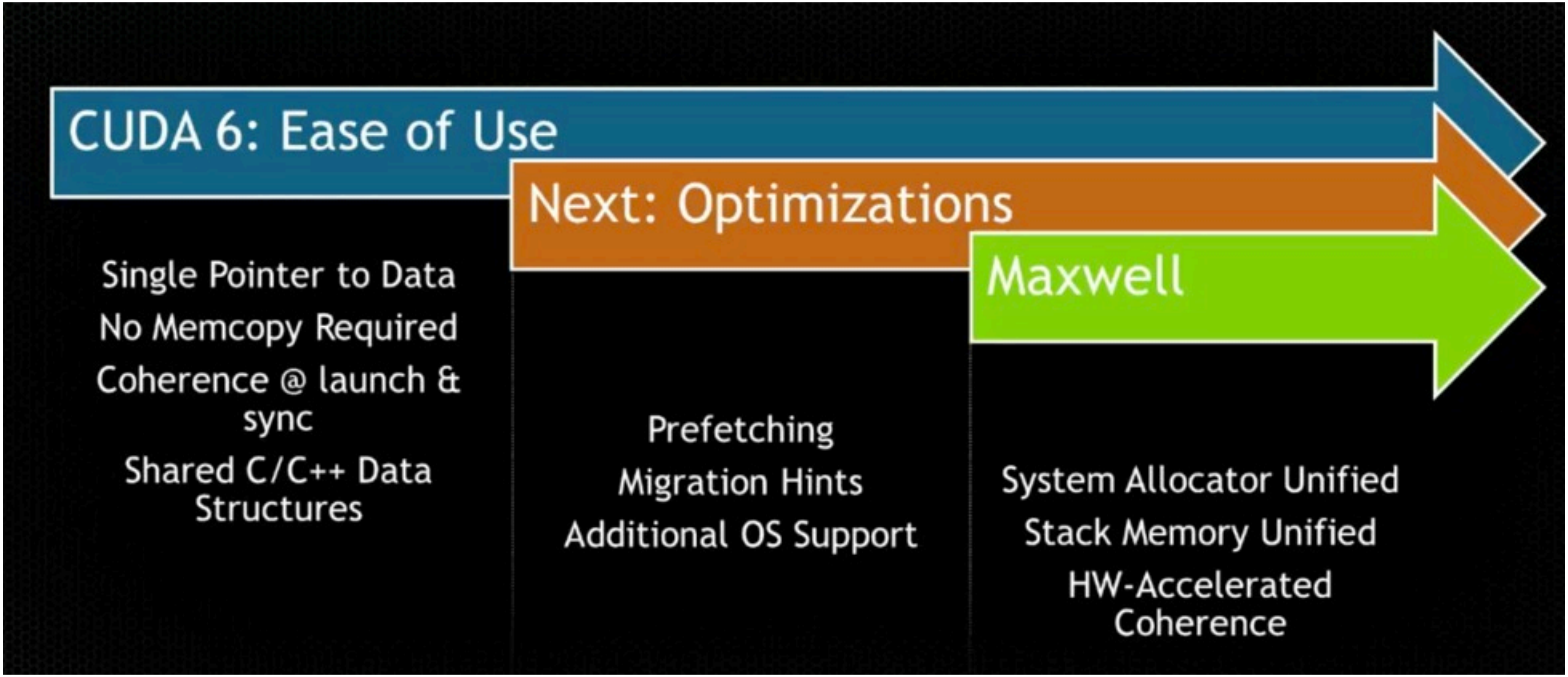



- Can pass list elements between CPU & GPU.
  - No need to move data back and forth between CPU and GPU.
- Can insert and delete elements from CPU or GPU.
  - But program must still ensure no race conditions (data is coherent between CPU & GPU at kernel launch only).

# Unified memory: Summary

- Drop-in replacement for cudaMalloc().
  - cudaMemcpy() now optional.
- Greatly simplifies code porting.
  - Less Host-side memory management.
- Enables shared data structures between CPU & GPU
  - Single pointer to data = no change to data structures.
- Powerful for high-level languages like C++.

# Unified memory: Future developments





48

# VIII. Resources and bibliography

NVIDIA

# CUDA Zone: Basic web resource for a CUDA programmer

**EXPLORE CUDA ZONE**

**WHAT IS CUDA**

Learn more about the CUDA parallel computing platform and programming model.

**GET STARTED - PARALLEL COMPUTING**

Find out about different paths and options for deploying CUDA and GPU Computing in your project

- Languages (C/C++, Python).
- Libraries (cuBLAS, cuFFT).
- Directives (OpenACC).
- Templates (thrust).

**CUDA IN ACTION - RESEARCH & APPS**

Supercomputing is now accessible for every researcher and scientist.
Find latest research, applications and links to how CUDA is transforming the industry

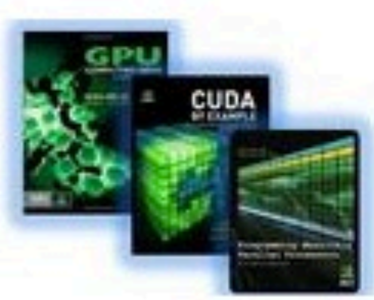
**CUDA TOOLKIT**

The NVIDIA CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications.

- Compiler (NVCC).
- Debugger (GDB).
- Profiler (cudaprof and Visual).
- Development envir. (Nsight).
- Code examples.

**CUDA EDUCATION & TRAINING**

Get educated with online courses, webinars, University Courses and wealth of technical papers & documentation




**CUDA TOOLS & ECOSYSTEM**

Learn more about powerful CUDA tools, libraries, languages, and other development aids available from NVIDIA & partners.

- Eclipse.
- Matlab.
- CUDA Fortran.
- GPUDirect.
- SDK for the LLVM compiler.

- [developer.nvidia.com/cuda-zone]

# CUDA 6 Production Release. Free download for all platforms and users

- [developer.nvidia.com/cuda-downloads]

| Windows | Linux | Mac OSX | | |
|---|---|---|---|---|
| **Version** | | | **64-bit** | **32-bit** |
| Windows 8.1<br>Windows 7<br>Windows Vista | Notebook | | EXE | EXE |
| | Desktop | | EXE | EXE |
| Windows XP | Desktop | | EXE | EXE |

Getting Started Guide

| Windows | Linux | Mac OSX |
|---|---|---|
| **OSX Release** | | **Package** |
| 10.8 | | PKG |
| 10.9 | | |

Getting Started Guide

Windows Linux Mac OSX

| Distribution | x86 64-bit | | x86 32-bit | | ARMv7 |
|---|---|---|---|---|---|
| Fedora 19 | RPM | RUN | | | |
| OpenSUSE 12.3 | RPM | RUN | | | |
| RHEL 6<br>CentOS 6 | RPM | RUN | | | |
| RHEL 5<br>CentOS 5 | | RUN | | | |
| SLES 11 (SP2 & SP3) | RPM | RUN | | | |
| Ubuntu 13.04 | DEB | RUN | | RUN | DEB |
| Ubuntu 12.04 | DEB* | RUN | DEB | RUN | |
| L4T, Linux for Tegra | | | | LINK | LINK |

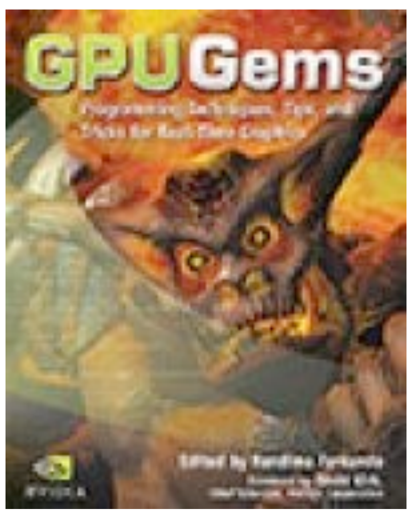
Getting Started Guide



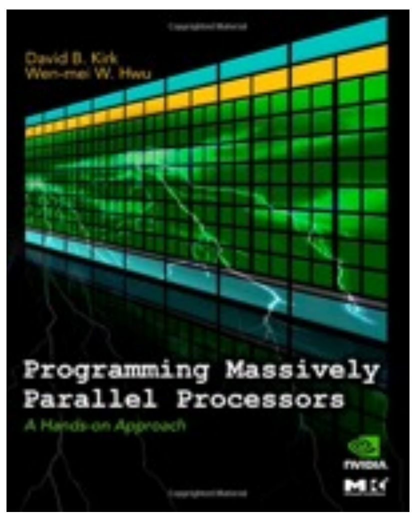
51

# CUDA books: From 2007 to 2013

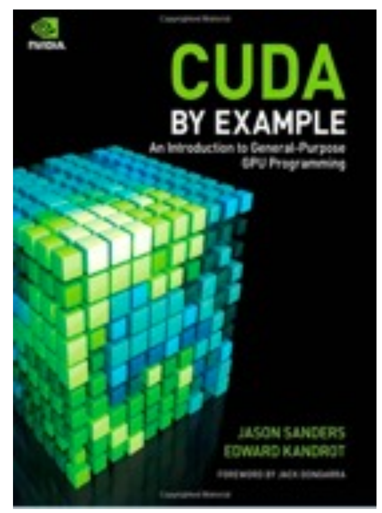
- GPU Gems series [developer.vidia.com/content/GPUGems3/gpugems3_part01.html]
- List of CUDA books in [www.nvidia.com/object/cuda_books.html]



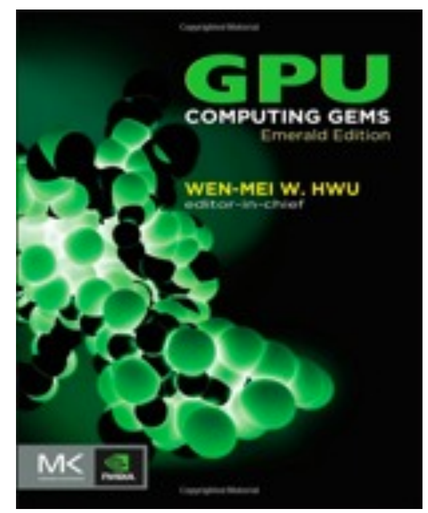
Sep'07



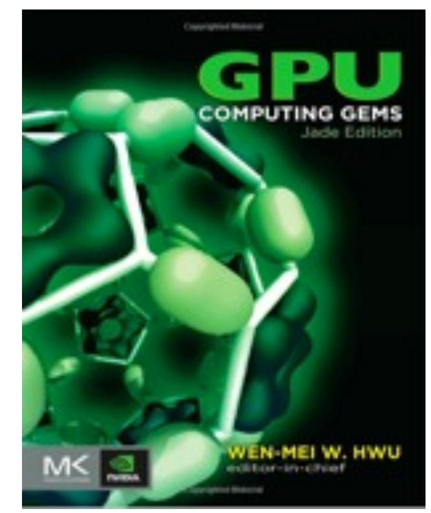
Feb'10



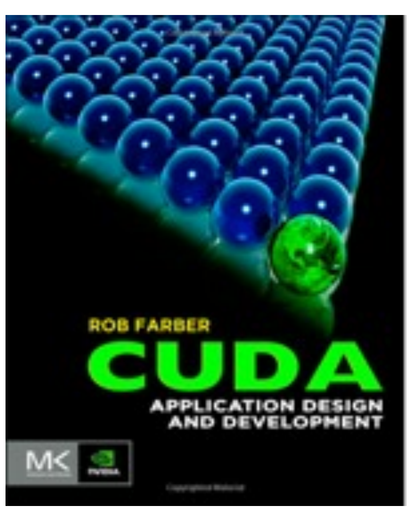
Jul'10



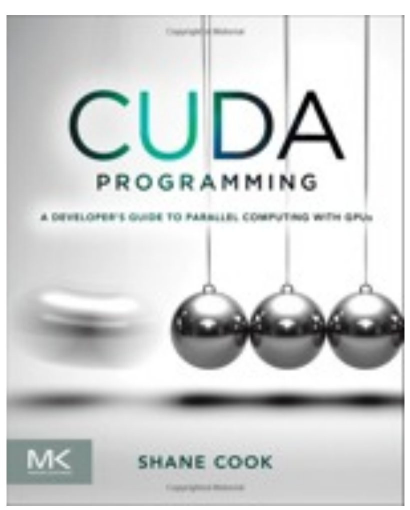
Abr'11



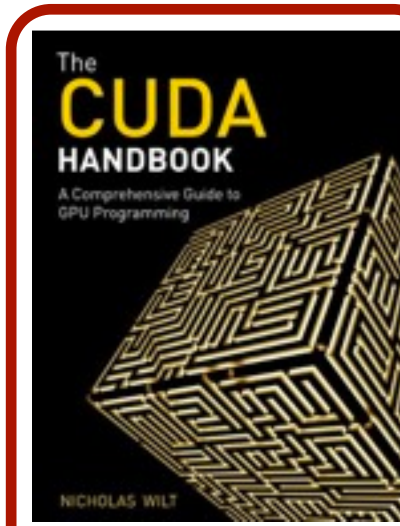
Oct'11



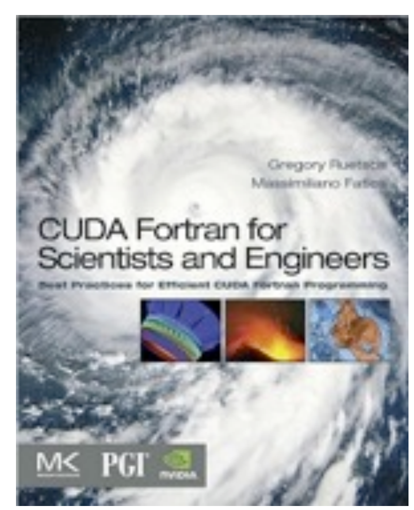
Nov'11



Dic'12



Jun'13



Oct'13

# Guides for developers and more documents

- Getting started with CUDA C: Programmers guide.
  - [docs.nvidia.com/cuda/cuda-c-programming-guide]
- For tough programmers: The best practices guide.
  - [docs.nvidia.com/cuda/cuda-c-best-practices-guide]
- The root web collecting all CUDA-related documents:
  - [docs.nvidia.com/cuda]
- where we can find, additional guides for:
  - Installing CUDA on Linux, MacOS and Windows.
  - Optimize and improve CUDA programs on Kepler platforms.
  - Check the CUDA API syntax (runtime, driver and math).
  - Learn to use libraries like cuBLAS, cuFFT, cuRAND, cuSPARSE, ...
  - Deal with basic tools (compiler, debugger, profiler).


53

# Choices to accelerate your applications on GPUs and material for teaching CUDA

- [developer.nvidia.com/cuda-education-training] (also available from the left lower corner of the CUDA Zone)

## CUDA Education & Training

### Accelerate Your Applications

Learn using step-by-step instructions, video tutorials and code samples.

- Accelerated Computing with C/C++
- Accelerate Applications on GPUs with OpenACC Directives
- Accelerated Numerical Analysis Tools with GPUs
- Drop-in Acceleration on GPUs with Libraries
- GPU Accelerated Computing with Python

### Teaching Resources

Get the latest educational slides, hands-on exercises and access to GPUs for your parallel programming courses.

- Parallel Programming Training Materials
- NVIDIA Research & Academic Programs

Sign up to join the Accelerated Computing Educators Network. This network seeks to provide a collaborative area for those looking to educate others on massively parallel programming. Receive updates on new educational material, access to CUDA Cloud Training Platforms, special events for educators, and an educators focused news letter.

Sign-up Today!

**QUICKLINKS**

- Downloads
- CUDA GPUs
- NVIDIA Nsight Visual Studio Edition
- Get Started - Parallel Computing
- Tools & Ecosystem
- CUDA FAQ

Tweets by @GPUComputing Follow



54

# Courses on-line (free access)

- More than 50.000 registered users from 127 countries over the last 6 months. An opportunity to learn from CUDA masters:
  - Prof. Wen-Mei Hwu (Univ. of Illinois).
  - Prof. John Owens (Univ. of California at Davis).
  - Dr. David Luebke (Nvidia Research).
- There are two basic options:
  - Introduction to parallel programming: [www.udacity.com]
  - Heterogeneous parallel programming: [www.coursera.org]
- If you do not have a CUDA-enabled GPU, you can even request 90 minutes tokens on Amazon EC2 instances (cloud computing):
  - [nvidia.qwiklab.com]
  - Only a supported web browser is required.


55

# Tutorials and webinars

- Presentations recorded at GTC (Graphics Technology Conference):
  - 383 talks from 2013.
  - More than 500 available from 2014.
  - [www.gputechconf.com/gtcnew/on-demand-gtc.php]
- Webinars about GPU computing:
  - List of past talks on video (mp4/wmv) and slides (PDF).
  - List of incoming on-line talks to be enrolled.
  - [developer.nvidia.com/gpu-computing-webinars]
- CUDACasts:
  - [bit.ly/cudacasts]

# Examples of webinars about CUDA 6.0

## GTC Express Webinar Program

GTC Express is a year-round extension of the great content available at GTC. Each month, top developers, scientists, researchers, and industry practitioners present innovative and thought-provoking webinars focused on the GPU-enabled work they're doing and sharing how GPUs are transforming their fields.

Register below for upcoming webinars and explore recordings of previous webinars.

### GTC EXPRESS SCHEDULE AT-A-GLANCE

| Date | Title | Speaker | |
|---|---|---|---|
| June 3, 2014, 9:00 AM PDT | **The Next Steps for Folding@home** | Vijay Pande, Professor, Stanford University | > Register Now |
| May 14, 2014, 10:00 AM PDT | **CUDA 6: Performance Overview** | Jonathan Cohen, Senior Manager, CUDA Libraries and Algorithms, NVIDIA | > Register Now |
| May 13, 2014, 9:00 AM PDT | **An Overview of AMBER 14 - Creating the World's Fastest Molecular Dynamics Software Package** | Ross C. Walker, University of California San Diego, Scott Le Grand, Amazon Web Services, and Adrian Roitberg, University of Florida. | > Register Now |
| May 7, 2014, 10:00 AM PDT | **CUDA 6: Drop-in Performance Optimized Libraries** | NVIDIA DevTech Team | > Register Now |
| May 1, 2014, 10:00 AM PDT | **CUDA 6: Unified Memory** | Mark Ebersole, CUDA Educator, NVIDIA | > Register Now |
| April 23, 2014, 11:00 AM PDT | **CUDA 6 Features Overview** | Ujval Kapasi, CUDA Product Manager, NVIDIA | > Register Now |



57

# Developers

- Sign up as a registered developer:
  - [www.nvidia.com/paralleldeveloper]
  - Access to exclusive developer downloads.
  - Exclusive access to pre-release CUDA installers like CUDA 6.0.
  - Exclusive activities an special offers.
- Meeting point with many other developers:
  - [www.gpucomputing.net]
- GPU news and events:
  - [www.gpgpu.org]
- Technical questions on-line:
  - NVIDIA Developer Forums: [devtalk.nvidia.com]
  - Search or ask on: [stackoverflow.com/tags/cuda]

# Developers (2)

- List of CUDA-enabled GPUs:
  - [developer.nvidia.com/cuda-gpus]
- And a a last tool for tuning code: The CUDA Occupancy Calculator
  - [developer.download.nvidia.com/compute/cuda/CUDA_Occupancy_calculator.xls]

# Future developments

- Nvidia's blog contains articles unveiling future technology to be used within CUDA. It is the most reliable source about what's next (subscription recommended):
  - [devblogs.nvidia.com/parallelforall]
- Some recommended articles:
  - "5 Powerful New Features in CUDA 6", by Mark Harris.
  - "Jetson TK1: Mobile Embedded Supercomputer Takes CUDA Everywhere", by Mark Harris.
  - "NVLINK, Pascal and Stacked Memory: Feeding the Appetite for Big Data", by Denis Foley.
  - "CUDA Pro Tip: Increase Application Performance with NVIDIA GPU Boost", by Mark Harris.
  - "CUDA 6.0 Unified Memory", by Mark Ebersole.

# Thanks!

- You can always reach me in Spain at the Computer Architecture Department of the University of Malaga:
  - e-mail: ujaldon@uma.es
  - Phone: +34 952 13 28 24.
  - Web page: http://manuel.ujaldon.es (english/spanish versions available).
- Or, more specifically on GPUs, visit my web page as Nvidia CUDA Fellow:
  - http://research.nvidia.com/users/manuel-ujaldon


61