

# Introduction to GPU

Shin Kee Chung, Linqing Wen (UWA)  
Kipp Cannon (Caltech)

# What is GPU?

- GPU stands for Graphics Processing Unit
- Originally developed for graphics rendering
- Its development is mainly driven by the gaming market
- CUDA (Compute Unified Device Architecture) is the GPU programming language developed by NVIDIA
- AMD GPU was not tested due to the lack of library support (such as FFT library)

# GPU and CUDA

- Popular
  - Over 100 million CUDA enabled GPU sold
- Easy to program using CUDA
  - C and C++ Integration
  - Sizeable computing libraries
  - CUDA Matlab Plugins
- Cost effective
  - \$400-\$500 can provide teraflops performance

# CUDA Common Library

- CUFFT (CUDA FFT)
- CUBLAS (CUDA Basic Linear Algebra Subprograms)
- CUDA SDK (CUDA Software Development Kit)

[http://www.nvidia.com/object/cuda\\_learn.html](http://www.nvidia.com/object/cuda_learn.html)

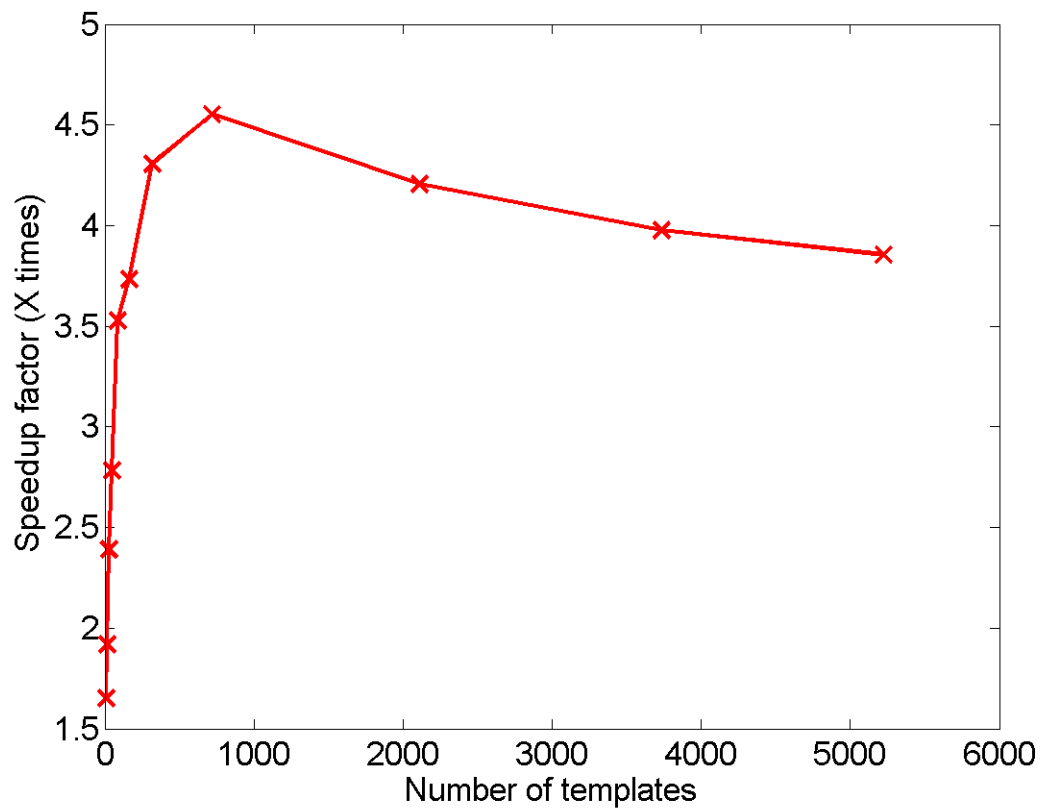
[http://www.nvidia.com/object/cuda\\_sdks.html](http://www.nvidia.com/object/cuda_sdks.html)

# Some CUDA SDK Examples

- Monte Carlo Option Pricing
- FFT Based 2D Convolution
- Eigenvalues
- Matrix Multiplications
- N-body Simulation

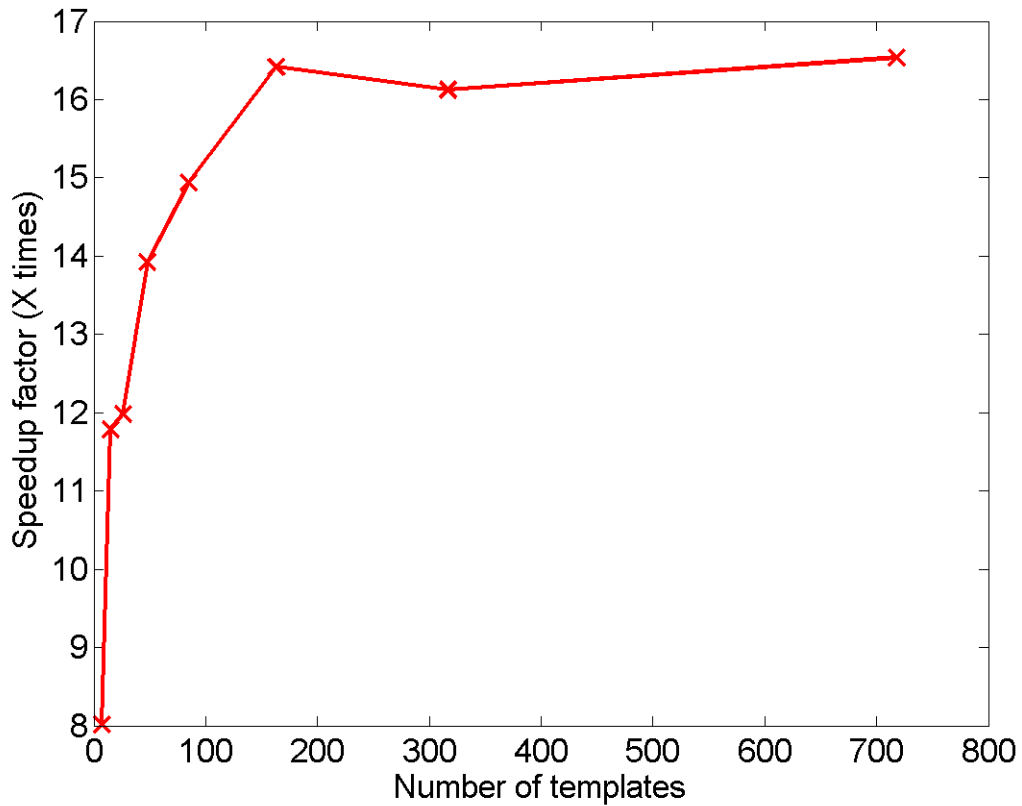
# CUDA in LAL - lalapps\_inspiral

- Developed by UWA-Caltech group
  - Shin Kee Chung, Linqing Wen (UWA), Kipp Cannon (Caltech)
- Results:
  - Replace FFTW in lalapps\_inspiral by CUDA FFT and demonstrated a 4x increase in speed overall using GeForce 8800 Ultra vs. 1 core of a 2.5 GHz Intel Core 2 Quad 9300 CPU.
  - Implement chi-square statistic in lalapps\_inspiral using direct calls to CUDA's multi-FFT routines and data parallelism feature
  - Demonstrated a 16x increase in speed overall in lalapps\_inspiral with the chi-square statistic turned on
- Publication:
  - Chung S. et al. 2009, CQG, to be published, arXiv:0906.4175
  - Chung, S. 2008, Honours thesis, UWA
  - Chung, 2008 SURF report with Wen & Cannon
- On going:
  - CUDA template parallel lalapps\_inspiral



## **lalapps\_inspiral Speedup With CUFFT**

We achieve more than 4 times speedup simply by applying CUFFT in LAL.



### **lalapps\_inspirial Speedup With CUDA Chi-square Implementation**

We modified the original Chi-square in `lalapps_inspirial` to utilize the multi-FFT routines and data parallelism feature of CUDA.

We achieved more than 16 times speedup.



# CUDA in LAL – Current Status

- Enable FFT package's CUDA back-end with

```
$ ./configure --with-cuda={path to cuda} ...
```

replaces all single precision FFTs in LAL/LALApps codes with GPU-accelerated versions.

- Works on
  - Linux machines with CUDA installed.
  - Macs machines with CUDA installed.
- CUDA-based LAL/LALApps tested with
  - GeForce 8500 GT (card purchased by D. Brown for testing)
  - GeForce 8800 Ultra (Shinkee's development box at CIT)
  - GeForce 8800 GTX, GTX 285 (UWA)
  - GeForce 8600M GT on Mac OS X Leopard laptop
  - GTX 295 (RMKI, Hungary)
- “Test” = obtain correct output from `lalapps_inspiral` (fractional error < 0.03% for Shinkee’s tests).

# GPU Low-Latency Inspiral Search

- UWA-Caltech collaboration
- Applied to time-domain low-latency infinite impulse response (IIR) filter
  - Hooper et al. 2009, Budapest LSC Meeting LIGO-G0900770-v2  
<https://dcc.ligo.org/cgi-bin/private/DocDB/ShowDocument?docid=5333>
- Results:
  - 5 times speedup is achieved
  - Tested using GTX 285 and 2.4 GHz Intel Core 2 Quads 6600 CPU (one core)
- Future Work:
  - Apply to stream-based time-domain low-latency pipeline LLOID developed by Caltech LIGO group
    - Insert GPU-enabled IIR filter
    - Replace individual elements in LLOID with GPU-enable components

# GPU @ LSC

- LSC GPU Discussion Group
  - Mailing List
    - [GPU-Discuss@ligo.gwastro.psu.edu](mailto:GPU-Discuss@ligo.gwastro.psu.edu)
    - Still a starting phase in LSC
      - ~30 subscribers, ~15 LSC institutions
        - » UWA, MPI/AEI, Northwestern, Caltech, LLO, RMKI Virgo Group (Hungary), Tsinghua U. (China), PSU, UWM, UNH, RIT, Umass, Cardiff, Birmingham, Columbia, ANU, U. Michigan
  - Wiki page:
    - <https://www.lsc-group.phys.uwm.edu/daswg/wiki/GPUDevelopment>

# GPU @ LSC

- U. of Western Australia (UWA)

- Linqing Wen, Shin Kee Chung\*, Shaun Hooper, David Blair, Amitava Datta
- GPU-enabled inspiral search pipeline (previous slides)
- Resource
  - Tested on Geforce 8800, FX 1700, GTX 285, GTX 295 (on 1 card)
  - International Center for Radio Astronomy Research (ICRAR), WASP center at UWA, CSIRO GPU cluster (200 GPUs)

[http://www.gravity.uwa.edu.au/gpu\\_implementation.html](http://www.gravity.uwa.edu.au/gpu_implementation.html)

- Caltech

- Kipp Cannon,
- GPU-enabled inspiral search pipeline and LLOID (previous slides)
- Support from the LIGO group

# GPU @ LSC

- AEI-Hannover: Einstein @ Home
  - Bruce Allen, Reinhard Prix, Oliver Bock, Bernd Machenschalk, Carsten Aulbert et al.
  - Hardware:
    - 6 machines with two Tesla C1060 each (3 Intel, 3 AMD, all Debian Lenny x64)
    - 1 machine with two GTX 285 (AMD, running XP32, Vista32 and Debian x64)
    - 1 machine with one dual-GPU GTX 295 (Intel, running XP64, Vista64 and Debian x64)
    - The ATLAS cluster is going to be extended by ~120 Tesla cards later this year or early next year (available to LSC members, same resource policy as for ATLAS itself)
  - GPU development
    - hierarchical search for continuous GWs in S5 data (development)
    - the search for binary pulsars in Arecibo radio data (improvement, public beta).

# GPU @ LSC

- **Northwestern**
  - Vicky Kalogera et al.
  - 30-GPU cluster : “happy to make them available”
  - benchmarking the SPINspiral MCMC code
- **RMKI Virgo Group (Hungary)**
  - Debreczeni Gergely et al.
  - Testing on NVIDIA GeForce GTX 295.
  - Submitted proposals for GPU clusters to OTKA (“Hungarian NSF”)
- **University of New Hampshire / NIKHEF**
  - Maurik Holtrop, Jo van den Brand
  - Hardware: two GTX280 and one 8800 GT, Tesla cluster (future)
  - Application: CW all sky search using quadratic filters.

# GPU @ LSC

- Tsinghua U. (China)
  - Junwei Cao et al.
  - Hardware
    - GeForce 9800GTX
  - Substantial non-LSC related GPU research has been done
    - implemented a collection of algorithms using CUDA
      - Confirm that CUDA is a good platform for computation-intense applications while performs not very well for control-intense algorithms
    - investigate different GPU programming models
      - Confirm that only some GPU programming models are good extension of CUDA
    - designed and implemented a parallel Viterbi sequence finding algorithm on GPU
      - More than one order magnitude is achieved
    - optimization GPU for finance application
      - 100 fold speedup achieved
    - MapReduce framework (MARS) on GPU
  - Currently planning to apply GPU to burst search using the Omega pipeline

# GPU @ LSC

- UWM

- Patrick Brady, Adam Mercer et al
- Adam helps build system patches, committed LAL CUDA FFT and working on committing GPU-enabled chi-square test
- plan to deploy a substantial GPU testbed at UWM over the next 6-12 months to enable larger scale prototyping of GPU enabled codes for gravitational-wave astronomy.

- LLO

- Dwayne Giardina , Rupal Amin et al
- built a workstation at LLO with a NVIDIA Tesla C1060
- Compare a Matlab scripts with and without CUDA plugins



# Acknowledgements

- Alan Weinstein, Patrick Brady and Sam Finn for support
- Inputs from Adam Mercer, Oliver Bock, Jun-wei Cao, Debreczni Gergely, Vivien Raymond, Reinhard Prix, and Maurik Holtrop
- GPU-Discuss community

# Appendix

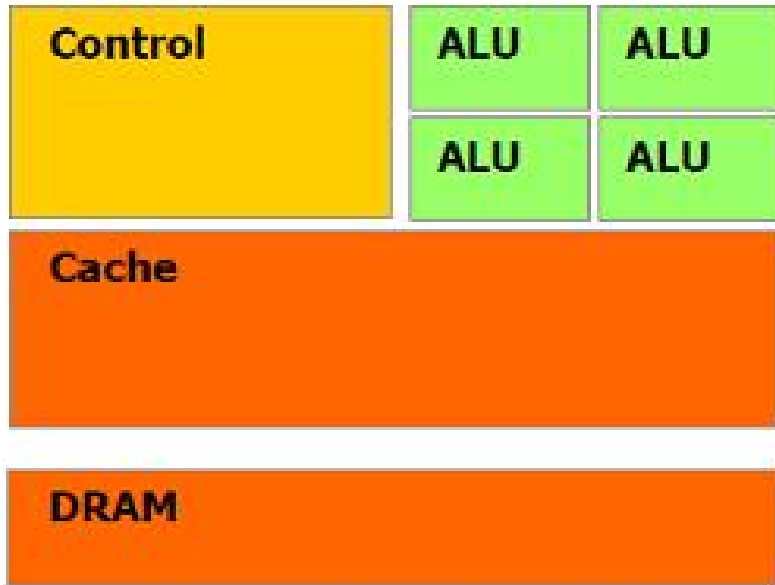
# GPU vs CPU

## GPU

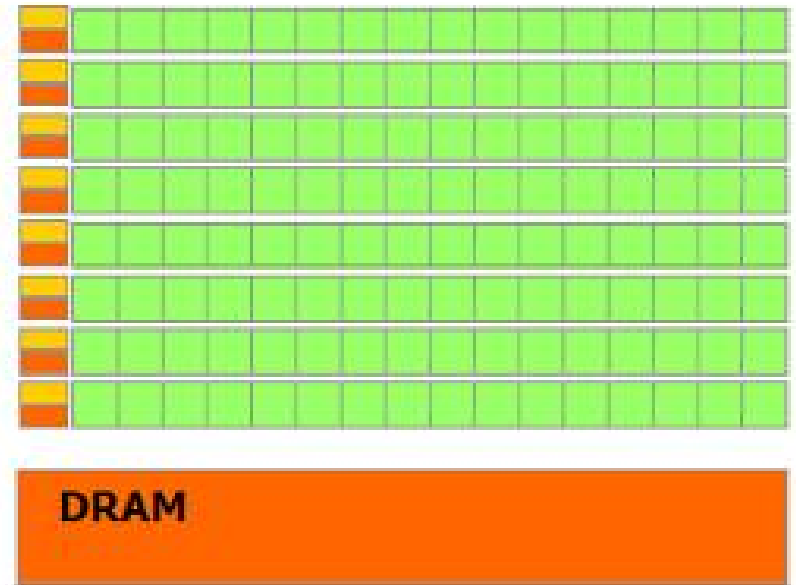
- Its architecture adopts parallel programming naturally
- Less efficient in flow control
- Small cache size limits the speed of memory access

## CPU

- Originally designed to execute commands sequentially
- Very efficient in flow control
- Huge cache size speed up its memory access



**CPU**



**GPU**

## Comparison of CPU and GPU hardware

Diagram taken from the NVIDIA CUDA Programming Guide.

# Graphics Cards Specification

GPU Card	Peak Performance	Memory Size	Memory Bandwidth	Number of Processing Cores	Max Power Consumption	Price (USD)
GTX 285 (Single GPU)	1062.72 GFLOPs	1 GB	159.0 GB/sec	240	204W	~ \$400
GTX 295 (Dual GPU)	1788.48 GFLOPs	1792 MB (896 MB each)	223.8 GB/sec	480 (240 each)	289W	~ \$550
Supercomputer with up to 4 Tesla C1060	Up to 4 TFLOPs	4 GB each	102 GB/sec each	240 each	187.8W each	~ \$8000

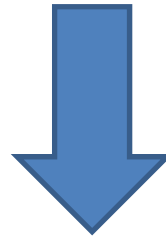
# CUDA Quick Start Guide

- After installing CUDA following some simple instructions from NVIDIA website:
  - Write a simple “hello world” C program, and save it as .cu extension
  - Compile it using nvcc

```
nvcc -o program_name program_name.cu
```
  - This is already a CUDA program (although it is still executed in CPU)
  - Then we can start putting in kernel functions (that runs in GPU) s

# Example: Simple Number Inverter

1	0	0	1	0
---	---	---	---	---




0	1	1	0	1
---	---	---	---	---

# Example: Simple Number Inverter

## C Program

```
int *run(int *data, int length)
{
    int i;

    for( i = 0; i < length; i++ )
    {
        data[i] = 1 - data[i];
    }
    return data;
}
```



Calculation  
done in a loop



# Example: Simple Number Inverter

## CUDA Program, multi-threaded execution

Kernel Function

```
__global__ void invert(int *d_data, int length)
{
    // Getting the thread id, block id and number of threads per block
    int tx = threadIdx.x;
    int bx = blockIdx.x;
    int numThreads = blockDim.x;

    // Inverting the element accessed by each thread
    d_data[bx * numThreads + tx] = 1 - d_data[bx * numThreads + tx];
}
```

No loop is used



Host Function

```
int *run(int *data, int length)
{
    ...
    // Perform inversion, assuming that total threads = array length
    invert<<< blocks, threads >>>( d_data, length );
    // Then copy d_data to data
    ...
    return data;
}
```

# GPU in Science: example

## N-body Simulation

- Pre-CUDA:
  - Portegies Zwart et al. 2007  
New Astron., 12, 641
- CUDA:
  - Belleman et al. 2008 New  
Astron., 13, 103
- GeForce 8800GTX with  
CUDA runs at about the  
same speed as GRAPE-6Af  
for  $N \geq 10^6$

## Radio Astronomy

- GPU enabled correlator
  - Harris et al. 2008  
Experimental Astron., 22, 129
- > 100x achieved
- International Center for  
Radio Astronomy Research  
(ICRAR), UWA and Curtin

# Possible Future Application

- lalapps inspiral is unable to scale up to Adv.LIGO analysis requirements; a re-work of the internal data management is required.
- One project to investigate solutions is “gstlal”, a project to combine LAL with GStreamer, a Free stream-based multi-media signal processing framework.
- Prototype application is LLOID (a Low-Latency Online Inspiral Data analysis application) - a stream-based version of lalapps inspiral.
  - allows for very long, low-frequency, templates
  - provides sub-template latency
  - allows matched-filtering across gaps in data
  - multi-threaded to take advantage of multi-core CPUs

# Possible Future Application

- gstreamer/gstlal provides collection of “elements” that are chained together in a graph to construct the analysis pipeline.
  - highly-modular
  - example elements: resampler, tee, adder, FIR filter, IIR filter, mixer.
  - individual elements are easily replaced with alternate implementations - e.g., GPU-based versions.
- GPU use speculative:
  - thread contention for GPU is potentially a problem.
  - data rate on PCI bus is potentially a problem.
  - increasing on-device processing addresses PCI bandwidth. Possible?
  - gstreamer supports concept of special on-device data buffers; possibly allows smart management of GPU RAM.
  - or maybe data rate on PCI bus will be fine.
  - many unknowns.