CUDA Assignment, Code Examples and Scaling your App

Abhijit Bendale (abendale@vast.ucds.edu)
04/08/2014
Today’s Agenda

- Discussion on CUDA Assignment
- Code example
  - Detailed look at multiple examples from CUDA SDK
- Scaling up your application
  - Amazon EC2, Amazon S3, Auto-Scale
  - Map-Reduce, Apache Hadoop
CUDA Assignment: Implement Canny Edge Detection
Steps for Canny Edge Detection: Noise Reduction

\[
B = \frac{1}{159} \begin{bmatrix}
2 & 4 & 5 & 4 & 2 \\
4 & 9 & 12 & 9 & 4 \\
5 & 12 & 15 & 12 & 5 \\
4 & 9 & 12 & 9 & 4 \\
2 & 4 & 5 & 4 & 2
\end{bmatrix} \ast A.
\]

Noise Reduction

A = Image
**Sobel Filter**

\[
G_x = \begin{bmatrix}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{bmatrix} \ast A \quad \text{and} \quad G_y = \begin{bmatrix}
+1 & +2 & +1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix} \ast A
\]

**Gradient in X direction**

\[
G = \sqrt{G_x^2 + G_y^2}
\]

\[
\Theta = \text{atan2}(G_y, G_x)
\]

Round up angles to (0, 45, 90, 135 degrees angle)
Non-Max Supression

Angle = 90
Check top/bottom gradients. In this case 7 is greater. Hence an edge

Angle = 90
Check top/bottom gradients. In this case 5 is not greater than top/bottom. Hence not an edge
E.g. output of Non-Max Suppression

(a) Gradient values
(b) Edges after non-maximum suppression
Hysteresis (Double) Thresholding

Use high and low threshold to determine ideal thickness for edge lines. This is a bit empirical process. It is possible that you won't be able to perform high level of parallelization for this step. Give it a try.
Code base given to you

- Computation of Sobel Edge Filter

- Files
  - SobelFilter.cpp → overall I/O is handled here
  - SobelFilter_kernels.cu → Splitting of image in chunks and computing sobel filter on these chunks
    - sobelFilter() is the global wrapper function
    - ComputeSobel() is where horizontal and vertical filters are defined. __device__ function. Only convolution happens on GPU.
    - SobelShared() splits the image into multiple chunks
  - SobelFilter_kernels.h → header file
  - Makefile → library linking

- Feel free to rename the files, create more files etc.

- Note the different function scope identifiers: __global__, __device__,

- Works on both Linux (command line: type “make”) and Windows (Visual Studio). Open visual studio project file
About the code

- Make sure you understand data types used:
  - Pixel is just an unsigned char (standard way of representing intensity values in images)
  - Image is organized as texture: which is a 2D vector in C++ (texture<unsigned char, 2> tex)
  - setupTexture allocates memory in the device
  - Contains 2 ways to implement:
    - SobelTex() → Doesn’t use shared memory
    - SobelShared() → uses shared memory

- Feel free to get inspiration from existing open-source canny edge detection code.

- Finally.. All the best for your assignment..!
Convolution Separable

Goal: Given a filter kernel, compute convolution with matrix/image
A separable filter can be divided into 2 consecutive filter operations. They offer flexibility in implementation and reduce mathematical complexity.

Apply row filter and column filter separately.
3_Imaging/convolutionSeparable

- main.cpp: main program, allocating host and device memory, generating input data, issuing CUDA computations
- convolutionSeparable.cu: CUDA convolution kernels (contains row and column kernels)
- convolutionSeparable_gold.cpp: reference CPU separable convolution implementation, which is used to validate results from CUDA
Seperable Convolution

3_Imaging/convolutionSeparable/main.cpp

```cpp
for (int i = -1; i < iterations; i++)
{
    convolutionRowsGPU(
        d_Buffer,
        d_Input,
        imageW,
        imageH
    );

    convolutionColumnsGPU(
        d_Output,
        d_Buffer,
        imageW,
        imageH
    );
}
```

**Output of Rows**

**Becomes input of columns**

```cpp
__global__ void convolutionRowsKernel(
    float *d_Dst,
    float *d_Src,
    int imageW,
    int imageH,
    int pitch
)

__global__ void convolutionColumnsKernel(
    float *d_Dst,
    float *d_Src,
    int imageW,
    int imageH,
    int pitch
)
```

convolutionSeparable.cu
Histogram

Goal: Given an image, compute its histogram

![Histogram Chart]

Figure 1: An example of an image histogram

```java
for(int i = 0; i < BIN_COUNT; i++)
    result[i] = 0;

for(int i = 0; i < dataN; i++)
    result[data[i]]++;
```

Listing 1. Histogram calculation on a single-threaded device. (pseudocode)
Parallelizing Histograms

- Subdivision of input data array between execution threads
- Processing of the sub-arrays by each dedicated execution thread and sorting the result into a certain number of sub-histograms
- Merge sub histograms into a single histogram
extern "C" void histogram64(
    uint *d_Histogram,
    void *d_Data,
    uint byteCount
)
{

    const uint histogramCount = iDivUp(byteCount, HISTOGRAM64_THREADBLOCK_SIZE * iSnapDown(255, sizeof(data_t)));

    assert(byteCount % sizeof(data_t) == 0);
    assert(histogramCount <= MAX_PARTIAL_HISTOGRAM64_COUNT);

    histogram64Kernel<<<histogramCount, HISTOGRAM64_THREADBLOCK_SIZE>>>(
        d_PartialHistograms,
        (data_t *)d_Data,
        byteCount / sizeof(data_t)
    );
    getLastCudaError("histogram64Kernel() execution failed\n");

    mergeHistogram64Kernel<<<HISTOGRAM64_BIN_COUNT, MERGE THREADBLOCK_SIZE>>>(
        d_Histogram,
        d_PartialHistograms,
        histogramCount
    );
    getLastCudaError("mergeHistogram64() execution failed\n");
}
__global__ void mergeHistogram64Kernel(
    uint *d_Histogram,
    uint *d_PartialHistograms,
    uint histogramCount
)
{
    __shared__ uint data[MERGE_THREADSBLOCK_SIZE];

    uint sum = 0;

    for (uint i = threadIdx.x; i < histogramCount; i += MERGE_THREADSBLOCK_SIZE)
    {
        sum += d_PartialHistograms[blockIdx.x + i * HISTOGRAM64_BIN_COUNT];
    }

    data[threadIdx.x] = sum;

    for (uint stride = MERGE_THREADSBLOCK_SIZE / 2; stride > 0; stride >>= 1)
    {
        __syncthreads();

        if (threadIdx.x < stride)
        {
            data[threadIdx.x] += data[threadIdx.x + stride];
        }
    }

    if (threadIdx.x == 0)
    {
        d_Histogram[blockIdx.x] = data[0];
    }
}
Scaling up your application
Agenda

- Amazon EC2: Scaling Computation, Auto-Scaling
- Amazon S3: Scaling Storage
- Maintaining Large Databases
- Hadoop/MapReduce
- Cassandra, Mongdob, Elasticsearch
- Cost associated with Scaling
Amazon Web Services

AWS’s Products

- Deployment & Administration
- Application Services
- Compute (EC2)
- Storage (S3)
- Database (RDS)
- Networking
- AWS Global Infrastructure
Amazon EC2

- Amazon Elastic Compute Cloud provides compute capacity in the cloud.
- EC2 allows users to rent virtual computers on which to run their own computer applications.
- EC2 allows scalable deployment of applications by providing a web service and API.
- EC2 provides users control over geographic location for better optimization.

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>CentOS</th>
<th>Debian</th>
<th>SUSE Linux Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red Hat Enterprise Linux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle Enterprise Linux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ubuntu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Name</td>
<td>Memory (GiB)</td>
<td>Cores/Compute Units</td>
<td>Instance Storage (GB)</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>m1.small</td>
<td>1.7</td>
<td>1/1</td>
<td>160</td>
</tr>
<tr>
<td>m1.medium</td>
<td>3.75</td>
<td>1/2</td>
<td>410</td>
</tr>
<tr>
<td>m1.large</td>
<td>7.5</td>
<td>2/4</td>
<td>850</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>15</td>
<td>4/8</td>
<td>1600</td>
</tr>
<tr>
<td>m3.xlarge</td>
<td>15</td>
<td>4/13</td>
<td>0 (EBS only)</td>
</tr>
<tr>
<td>m3.2xlarge</td>
<td>30</td>
<td>8/26</td>
<td>0 (EBS only)</td>
</tr>
<tr>
<td>t1.micro</td>
<td>0.6</td>
<td>1/(up to 2)</td>
<td>0 (EBS only)</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>17.1</td>
<td>2/6.5</td>
<td>420</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>34.2</td>
<td>4/13</td>
<td>850</td>
</tr>
<tr>
<td>m2.4xlarge</td>
<td>68.4</td>
<td>8/26</td>
<td>1690</td>
</tr>
<tr>
<td>c1.medium</td>
<td>1.7</td>
<td>2/5</td>
<td>350</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>7</td>
<td>8/20</td>
<td>1690</td>
</tr>
<tr>
<td>cc1.4xlarge</td>
<td>23</td>
<td>2/33.5 (2 Intel Xeon X5570)</td>
<td>1690</td>
</tr>
<tr>
<td>cc2.8xlarge</td>
<td>60.5</td>
<td>2/88 (2 Intel Xeon E5-2670)</td>
<td>3370</td>
</tr>
<tr>
<td>cr1.8xlarge</td>
<td>244</td>
<td>2/88 (2 Intel Xeon E5-2670)</td>
<td>240 (SSD)</td>
</tr>
<tr>
<td>cg1.4xlarge</td>
<td>22</td>
<td>2/33.5 (2 Intel Xeon X5570) + 2 NVIDIA Tesla &quot;Fermi&quot; M2050 GPU</td>
<td>1960</td>
</tr>
<tr>
<td>hi1.4xlarge</td>
<td>60.5</td>
<td>16/35 (8 cores + 8 hyperthreads)</td>
<td>2*1024 (SSD)</td>
</tr>
<tr>
<td>hs1.8xlarge</td>
<td>117</td>
<td>16/35 (8 cores + 8 hyperthreads)</td>
<td>48000 (24 * 2TB drives)</td>
</tr>
</tbody>
</table>
Launching EC2 instance

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

- 0 Running Instances
- 0 Volumes
- 1 Key Pair
- 0 Placement Groups
- 0 Elastic IPs
- 0 Snapshots
- 0 Load Balancers
- 7 Security Groups

Focus on application development and offload database management to AWS - Try Amazon RDS Now!

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US East (N. Virginia) region

Service Health

Service Status:

- US East (N. Virginia):
  - This service is operating normally

Availability Zone Status:

- us-east-1a:
  - Availability zone is operating normally

Scheduled Events

US East (N. Virginia):

No events
### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of capacity management role to the instance, and more.

<table>
<thead>
<tr>
<th><strong>Number of instances</strong></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purchasing option</strong></td>
<td>Request Spot Instances</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Launch into EC2-Classic</td>
</tr>
<tr>
<td><strong>Availability Zone</strong></td>
<td>us-east-1a</td>
</tr>
<tr>
<td><strong>IAM role</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Shutdown behavior</strong></td>
<td>Stop</td>
</tr>
<tr>
<td><strong>Enable termination protection</strong></td>
<td>Protect against accidental termination</td>
</tr>
<tr>
<td><strong>Monitoring</strong></td>
<td>Enable CloudWatch detailed monitoring</td>
</tr>
</tbody>
</table>

Additional charges apply.
Monitoring activity
Amazon EC2 API

Key Pairs
- CreateKeyPair (p. 77)
- DeleteKeyPair (p. 141)
- DescribeKeyPairs (p. 238)
- ImportKeyPair (p. 379)

Elastic IP Addresses
- AllocateAddress (p. 13)
- AssociateAddress (p. 19)
- DescribeAddresses (p. 180)
- DisassociateAddress (p. 363)
- ReleaseAddress (p. 418)

Spot Instances
- CancelSpotInstanceRequests (p. 54)
- CreateSpotDatafeedSubscription (p. 112)
- DeleteSpotDatafeedSubscription (p. 158)
- DescribeSpotDatafeedSubscription (p. 299)
- DescribeSpotInstanceRequests (p. 301)
- DescribeSpotPriceHistory (p. 309)
- RequestSpotInstances (p. 433)

Elastic Block Store
- AttachVolume (p. 30)
- CopySnapshot (p. 61)
- CreateSnapshot (p. 109)
- CreateVolume (p. 119)
- DeleteSnapshot (p. 156)
- DeleteVolume (p. 164)
- DescribeSnapshotAttribute (p. 291)
- DescribeSnapshots (p. 294)
Connecting to Amazon EC2 instance
Auto-Scaling

- Automatically adapt computing capacity to site traffic
- Schedule based (e.g. time of the day), rule-based (e.g. CPU utilization thresholds) automated scaling

```
home$ as-create-or-update-trigger app-trigger -auto-scaling-group webapp
   --namespace "AWS/EC2" --measure CPUUtilization --statistic Average
   --lower-threshold 40 --upper-threshold 70 --lower-breach-increment=-1
   --upper-breach-increment=1 --breach-duration 120
```

Use command line tools to automate the process. For e.g. if CPU utilization goes above 70% for 120 secs, launch 1 machine. If goes below 40% for 120 secs, remove one machine
Understanding Scaling

- Infrastructure cost $\$
- Too much capacity
- Huge Cap-ex
- You lost your customers here
- Predicted Demand
- Actual Demand
- Scale up Approach
- Traditional scale out
- Automated scaling

Source: Amazon AWS
Overview of dynamic Scaling
Amazon S3: Scaling Storage

- S3 = Simple Storage Service
- Storage in EC2 is destroyed once the instance is terminated. Amazon uses this infrastructure for its own websites
- Data organized in the form of buckets. Accessed as bucketname.s3.amazonaws.com
- Allows unlimited storage: in the increments of 1GB to 5GB
- Objects are stored and retrieved using a developer-assigned key. Can be used along with Amazon EC2 compute instances
- Objects can be made available to public by http or bittorrent protocol
S3 Web Console

https://console.aws.amazon.com/s3/home?
If you wanted to back up data from your computer, at 0.01$/GB it will cost you around 5$/month for 500GB.
Data Transfer Pricing

The pricing below is based on data transferred "in" to and "out" of Amazon S3.

| Region: US Standard |

<table>
<thead>
<tr>
<th>Data Transfer IN To Amazon S3</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>All data transfer in</td>
<td>$0.000 / GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Transfer OUT From Amazon S3 To</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2 in the Northern Virginia Region</td>
<td>$0.000 / GB</td>
</tr>
<tr>
<td>Another AWS Region or Amazon CloudFront</td>
<td>$0.020 / GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Transfer OUT From Amazon S3 To Internet</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>First 1 GB / month</td>
<td>$0.000 / GB</td>
</tr>
<tr>
<td>Up to 10 TB / month</td>
<td>$0.120 / GB</td>
</tr>
<tr>
<td>Next 40 TB / month</td>
<td>$0.090 / GB</td>
</tr>
</tbody>
</table>
Advantages of Using S3

- Scalability: The amount of storage and bandwidth you need can scale as you like.
- Availability, speed, throughput, capacity and robustness is not affected even if you gain 10k users overnight.
- Leaves out lot of system administration overhead.
- Seamlessly integrates with other Amazon AWS tools. Could also use it for backing up your data.

Many well known websites use this for their storage requirements.
Amazon RDS: Scaling Databases

- Distributed relational database service

- Complex admin processes like patching software, backing up databases are managed automatically

- Can get started in minutes, instead of in days

- Scaling storage and compute resources can be performed by API calls

- Support for MySQL, Microsoft SQL server, Oracle Database and PostgreSQL
Amazon RDS Web Console
Other AWS Products

List of products  [edit]

Compute  [edit]

- Amazon Elastic Compute Cloud (EC2) provides scalable virtual private servers using Xen.
- Amazon Elastic MapReduce (EMR) allows businesses, researchers, data analysts, and developers to easily and cheaply process vast amounts of data. It uses a hosted Hadoop framework running on the web-scale infrastructure of EC2 and Amazon S3.

Networking  [edit]

- Amazon Route 53 provides a highly available and scalable Domain Name System (DNS) web service.
- Amazon Virtual Private Cloud (VPC) creates a logically isolated set of Amazon EC2 instances which can be connected to an existing network using a VPN connection.
- AWS Direct Connect provides dedicated network connections into AWS data centers, providing faster and cheaper data throughput.

Content delivery  [edit]

- Amazon CloudFront, a content delivery network (CDN) for distributing objects to so-called "edge locations" near the requester.

Storage and content delivery  [edit]

- Amazon Simple Storage Service (S3) provides Web Service based storage.
- Amazon Glacier provides a low-cost, long-term storage option (compared to S3). High redundancy and availability, but low-frequent access times. Ideal for archiving data.
- AWS Storage Gateway, an iSCSI block storage virtual appliance with cloud-based backup.
- Amazon Elastic Block Store (EBS) provides persistent block-level storage volumes for EC2.
- AWS Import/Export, accelerates moving large amounts of data into and out of AWS using portable storage devices for transport.

Database  [edit]

- Amazon DynamoDB provides a scalable, low-latency NoSQL online Database Service backed by SSDs.
- Amazon ElastiCache provides in-memory caching for web applications. This is Amazon’s implementation of Memcached and Redis.
- Amazon Relational Database Service (RDS) provides a scalable database server with MySQL, Informix, Oracle, SQL Server, and PostgreSQL support.
- Amazon Redshift provides petabyte-scale data warehousing with column-based storage and multi-node compute.
- Amazon SimpleDB allows developers to run queries on structured data. It operates in concert with EC2 and S3 to provide “the core functionality of a database”.

Infrastructure as a Service

- Google Compute Engine
- Amazon Web Services
- Windows Azure
- Rackspace
- IBM SmartCloud
- OpenShift
- Red Hat
- Google Compute Engine
Hadoop, MapReduce ....

Hadoop distributes data and computation across a large number of computers.
DISCLAIMER

• Don’t use Hadoop if your data and computation fit on one machine

• Getting easier to use, but still complicated

http://www.wired.com/gadgetlab/2008/07/patent-crazines/
What exactly is **Hadoop**?

- Actually a growing collection of subprojects
An overview of Hadoop Map-Reduce

Traditional Computing

(one computer)

Hadoop

(many computers)
An overview of Hadoop Map-Reduce

(Actually more like this)

(many computers, little communication, stragglers and failures)
What is Hadoop?

- A large scale distributed batch processing infrastructure
- True power lies in its ability to scale to hundreds or thousands of machines
- Hadoop includes distributed file system which breaks in input data and sends fractions of the original data to several machines in your cluster
- It includes a distributed file system which breaks up input data and sends fractions of the original data to several machines in your cluster
- Similar to NFS but lot more efficient
Challenges at Large Scale

- Data distributed over multiple machines
  - Increases probability of failure
  - Network failure, machine failure, router failure etc.
  - Drive failures, desynchronized clocks etc.

- Synchronization between multiple machines remains biggest challenge for distributed systems

- For e.g. in a system with 100 machines, if 1 fails it should be equivalent to loss of 1% of the work and not 100% of work
Hadoop Approach

- Connect multiple computers together and efficiently process large volumes of information by using commodity machines.

- A theoretical 1000-core CPU costs more than 1000 single CPU machines of 250 quad-core machines.

- Hadoop will tie smaller and more reasonably priced machines together into a single cost-effective compute cluster.

- Data is distributed to different nodes in the cluster instead of a single NFS drive.
**Network File System (NFS) Approach**

Large amount of input data

Store it on a huge disk that can be accessed by all machines

Distributed System

Node 1

Node 2

.. 

Node N

Data loading step
Hadoop Data Approach

Large amount of input data...

Data loading step

Node 1
Slice of input

Node 2
Slice of input

Node 3
Slice of input
Data Management in Hadoop

- Data is conceptually record-oriented in Hadoop.
- Individual input files are broken into lines or into other formats specific to application logic
- Each node in the cluster processes subset of the record
- Data is stored on local disks: thus reduces overhead on network bandwidth and transfers
- **Moving computation to the data** instead of moving data to computation
MapReduce: Isolated Processes

- Hadoop limits the amount of communication which can be performed by the processes.
- Though this looks like limitation, it increases reliability of the system.
- Hadoop will not run just any program and distribute it across cluster (just like GPUs).
- Programs must be written to conform “MapReduce” programming model.
- Records are processed in isolation by tasks called Mappers.
- The output of mappers is then brought together into second set of tasks calledReducers, where results from different mappers can be merged.
MapReduce Model

- Pre-loaded local input data
- Intermediate data from mappers
- Values exchanged by shuffle process
- Reducing process generates outputs
- Outputs stored locally
MapReduce: Isolated Processes

- Communication between nodes is implicit.

- Pieces of data are tagged with key names, which inform Hadoop how to send related bits of information to common destination node.

- Hadoop internally manages all of the data transfer and cluster topology issues.

- Less message passing between nodes compared to MPI (Message Passing Interface).

- Offers flat scalability compared to MPI. In MPI there is significant overhead for large scale systems. Have to manually engineer how message passing between all the machines work.

- Same code works for MBs of data and TBs of data. No overhead for refactoring, I/O, node failure etc. Hadoop takes care of it.