Integrating Acceleration Devices using CometCloud

Thomas Beach

School of Computer Science & Informatics
Cardiff University

June 2013
A heterogeneous cloud system often contains many different types of computing resources.

From standard multi core systems, to many core systems, GPUs, physics co-processing devices and FPGAs.

GPU Accelerated compute nodes are already available in mainstream cloud providers i.e. Amazon EC2.

Often making best use of these devices is difficult within a large system.

This presentation will give an overview of our work in utilising CometCloud to integrate these devices into a Cloud Computing system.
1 CometCloud

2 Integrating Acceleration Devices

3 Initial Results

4 Future Work
CometCloud
Comet Cloud is an autonomic computing engine for Cloud environments.

Self Managing.

Utilises several programming models such as Master/Worker, Workflow and MapReduce/Hadoop.
Structured into a set of masters and secure workers, linked by a distributed coordination space.

Unsecured workers are connected to the coordinating space via a Proxy (Request Handler).

Task Based approach is used.

Each task is an XML tuple, which is inserted into the coordination space.
Comet Cloud

- **out (ts, t):** a non-blocking operation that inserts tuple t into space ts

- **in (ts, t):** a blocking operation that removes a tuple t matching template t from the space ts and returns it

- **rd (ts, t):** a blocking operation that returns a tuple t matching template t from the space ts. The tuple is not removed from the space
CloudBursting and Cloud-Bridging

- Supports CloudBursting - The ability to dynamically scale out to utilise additional resources.
- This can be governed by user defined policies.
- CloudBridging - create a virtual cloud consisting of multiple smaller clouds.
- Controlled via a scheduler.
System Architecture

Adopted from Kim et al

Robust/Secure Masters

Management agent | Computing agent
Scheduling | Monitoring
Comet

Secure Workers

Computing agent
Comet

Request Handler

Proxy

Space sharing zone

Generate tasks

Get tasks

Directly send results

3. Get a task

4. Send a task

2. Forward requests in Round-robin

Unsecured workers

Computing agent

Clouds/Grids

1. Request a task

5. Send results directly to master

June 2013
System Architecture

Adopted from Kim et al
Integrating Acceleration Devices
Currently CometCloud will attempt to balance the task load across multiple workers.

However, many workers have different characteristics and more intelligent selection is necessary.

Selection by worker capability is the first step and works well.
Integration of Acceleration Devices

Currently CometCloud will attempt to balance the task load across multiple workers.

However, many workers have different characteristics and more intelligent selection is necessary.

Selection by worker capability is the first step and works well.

There scope to gain performance improvement from finer grained decision making.
Steps 1,2 - Code Analysis

- Analyse code and extract kernels (possible parallelisable loops)

- Extract metrics from each kernel within the application.

- Generate XML tuples for each kernel containing metrics and any device restrictions.

*Parallelisable loops could either be detected automatically or specified by the user*
Analysing the input code (or specifying via annotation) keeps the input device independent:

- This gives far more flexibility in terms of matching to a device.
- Utilising a specific API will restrict what devices can be used, i.e. CUDA.
- Frees the programmer from having to learn multiple APIs.
System Architecture

Integrating Acceleration Devices using CometCloud

June 2013
XML tuples representing kernels are only examined by workers which have the capability to execute them.

This means only workers that have the capability to actually execute the task will reply.

Decisions made on simple rules - could include: software versions, presence of hardware or more detailed hardware specification i.e. CUDA5 compatible device.

Device must then reply with *Estimated Time to execute* and *Estimated time until device is available*. 
In order to undertake performance prediction a series of metrics are extracted by the code analyser:

- **Intensity** - A count of mathematical operations per iteration.

- **I/O** - A count of the number of memory accesses per iteration (read and write).

- The number of times branching that occurs per iteration.

- The number of iterations of the kernel that are performed.

- Size of data that must be loaded to/from the device.
Performance prediction made utilising WEKA library and the K-Nearest Neighbour algorithm.

This enables each device to estimate how long a particular application may take and how long their current job may take.

Local decisions on whether application should be accelerated processed with a decision tree.
Performance Prediction

```
Performance Prediction

Intuitively, the performance of a program can be predicted by considering the following factors:

1. **Intensity**
   - If the intensity is less than or equal to 72, the performance is determined by the CPU.
   - If the intensity is greater than 72, the performance is determined by the Data.

2. **Data**
   - If the data size is less than or equal to 192000000, the performance is determined by the CPU.
   - If the data size is greater than 192000000, the performance is determined by the GPU.

3. **Iterations**
   - If the iterations are less than or equal to 9600000, the performance is determined by the CPU.
   - If the iterations are greater than 9600000, the performance is determined by the GPU.

4. **Intensities**
   - If the intensity is less than or equal to 512, the performance is determined by the CPU.
   - If the intensity is greater than 512, the performance is determined by the GPU.

5. **Intensities**
   - If the intensity is less than or equal to 25000000, the performance is determined by the CPU.
   - If the intensity is greater than 25000000, the performance is determined by the GPU.

6. **Intensities**
   - If the intensity is less than or equal to 4000, the performance is determined by the CPU.
   - If the intensity is greater than 4000, the performance is determined by the GPU.
```
Currently, one device is selected that gives best overall performance for the application.

- *Estimated runtime* + *Waiting time for device to be free*

- Other methods of selection could be: cost, power consumption.
Once a device has been selected:

- CUDA bootstrapping code can be generated.

- Identified Kernels can be wrapped in CUDA method stubs.

- This allows the kernels extracted to remain device independent.
The availability of accurate performance data therefore is critical:

- The system can self update.

- New devices entering the system or existing devices that may be under trained can update their training dataset.

- Inter worker communication using the CometSpace is used to achieve this.

- Takes advantage of idle runtime.
Self-Updating

Accelerator Device

Device
Performance Data
Device Agent

Accelerator Device

Device
Performance Data
Device Agent

CometSpace
Request Tuple

1
2
3
Initial Results
System Architecture

Integrating Acceleration Devices using CometCloud

June 2013 28 / 38
Step 1: Code Analysis - of un-annotated code

Step 2 - Metric Extraction

- Intensity-111
- I/O- 27
- Branching-1
Knowledge base of performance data was trained with results from similar applications.

<table>
<thead>
<tr>
<th>Total Number of Performance Measures</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurements within 0.5s</td>
<td>13</td>
</tr>
<tr>
<td>Measurements within 1.0s</td>
<td>28</td>
</tr>
<tr>
<td>Measurements within 1.5s</td>
<td>7</td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.70s</td>
</tr>
</tbody>
</table>
Step 6-7: Decision

- Decision made on device to utilise for application. Currently decision is made based on lowest execution time.

- In the case of our example - device offering best performance is generally NVidia FERMI GPU.
Initial Results - Canny Edge Detector

![Graph showing execution time against size of N in N x N dataset for different devices (CPU, Quad Core CPU, TESLA GPU, FERMI GPU).]
Utilising the system provides a performance improvement of a single core implementation of the algorithm.

Performance results are comparable with quad-core implementation.
Future Work
Is the method of performance prediction used here the most appropriate?
Key Questions

- Is the method of performance prediction used here the most appropriate?
- Is it feasible to execute kernels from the same application across multiple devices?
When operating in a federated environment, multiple cloud sites would only want to take tasks that they are capable of executing.

It is in the interest of each site within the federation to make best use of its resources internally.

CometClouds support of *CloudBursting* also allows for the possibility of utilising devices from external public clouds.
Multiple cloud systems each with workers with varying capabilities.
Any Questions?