Better Algorithms in Bioinformatics

FYP14020
Ng Tsz Ho
Supervised by Prof. Lam Tak Wah
Objective

To realize a server-client model of BALSA that runs on Amazon cloud platform (AWS)

- GPU separated from CPU, memory and hard disk
- GPU in server machine carries out computation, and transfers result back to client server
- Both machines launched in cloud
Remote CUDA (rCUDA)

In tradition, communication between nodes use message passing tools:

- Message Passing Interface (MPI)
- Parallel Virtual Machine (PVM)

Not for GPU programming, heavy overheads and bad memory management if bad programming.

Remote CUDA (rCUDA)

- As middleware that enables CUDA remoting over network
- Minimum changes to software structure
Advantages

● Reduce number of GPUs installed in a cluster -> increase GPU utilization to energy savings

● Flexible networking
  ○ In theory, every workstation can act as the server and the client
  ○ Several remote programs running in same remote GPU
  ○ 2 GPUs in same server side can be used at the same time

● Memory-friendly in Server side (with GPU)
  ○ only use <1GB memory in server side when running BALSA
# Architecture

<table>
<thead>
<tr>
<th>1. Local server (control)</th>
<th>64GB ram</th>
<th>24 cores</th>
<th>GTX680</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. rCUDA with 10GbE network</td>
<td>Client</td>
<td>64GB ram</td>
<td>24 cores</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>GTX680</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network</td>
<td>9.8Gbps</td>
<td></td>
</tr>
<tr>
<td>3. rCUDA with 10GbE network on AWS</td>
<td>Client</td>
<td>60GB ram</td>
<td>32 cores</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>M2050 x 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network</td>
<td>7.6Gbps</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing the architecture with local server, rCUDA with 10GbE network, and rCUDA with 10GbE network on AWS.]
Major Technical Issues

1. **GPU memories cannot be passed into thread functions**
   In BALSA, data structures loaded into GPU before thread, kernel functions in alignment thread cannot access them

**Solution**

- Free related GPU data structures before create alignment thread, and allocate them inside the thread
- This will cause only several seconds overhead per batch
Major Technical Issues

2. **CUDA library cannot be installed on a machine without a GPU card**
   - from CUDA5.0 onwards a GPU is needed for installation, current using CUDA6.0
   - cannot add a GPU to Amazon instances after it is launched

**Solution**

- Extract CUDA libraries from CUDA toolkit
- Match the links of CUDA libraries to rCUDA libraries
## Interim Progress

<table>
<thead>
<tr>
<th>Month</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov</td>
<td>SOAP3-dp and BALSA run successfully with rCUDA in BAL13 and BAL14</td>
</tr>
<tr>
<td>Dec</td>
<td>SOAP3-dp and BALSA run successfully with rCUDA on AWS</td>
</tr>
<tr>
<td></td>
<td>Accuracy tested with YH100 dataset on SOAP3-dp</td>
</tr>
<tr>
<td>Jan 2015</td>
<td>Testing and benchmarking</td>
</tr>
</tbody>
</table>
Benchmarking

3 Settings:
(i) GTX680 (control)
(ii) GTX680 + rCUDA
(iii) M2050(AWS) + rCUDA

5 Test cases:
100bp reads
6M, 12M, 18M, 24M, 30M
Benchmarking

3 Settings:
(i) GTX680 (control)
(ii) GTX680 + rCUDA
(iii) M2050(AWS) + rCUDA

5 Test cases:
100bp reads
6M, 12M, 18M, 24M, 30M

Alignment Time
(ii) ~18% slower
(iii) ~80% slower
Benchmarking

3 Settings:
(i) GTX680 (control)
(ii) GTX680 + rCUDA
(iii) M2050(AWS) + rCUDA

5 Test cases:
100bp reads
6M, 12M, 18M, 24M, 30M

Running time up to realignment
(ii) ~16.5% slower
(iii) ~48% slower
Some Suggestions

1. Incorporate **Non-uniform memory Access (NUMA) control** on AWS
   - AWS machines are of NUMA architecture

2. **Multiple GPU running in one server**
   - utilize 2 M2050 gpus in one workstation
Some Suggestions

3. **InfiniBand** instead of 10GbE

- adapts new Mellanox OFED GPUDirect RDMA feature, can bypass CPU in server side and goes directly into GPU.
- Not many instance types support InfiniBand on AWS
Thank you.