Accelerating irregular bioinformatics algorithms on GPUs.

Muaaz Gul Awan
Application Performance Group
● Exabiome Project:
  ○ Common algorithmic motifs.
  ○ Irregular memory access patterns among bio-algorithms.

● ADEPT, a GPU accelerated sequence aligner:
  ○ Implementation on GPUs.
  ○ Results and Integration in exabiome software stack.

● GPU Local Assembly:
  ○ Local Assembly in MetaHipMer.
  ○ Challenges and implementation on GPUs.
  ○ Integration and results.
## ExaBiome: Project Overview

**Exascale algorithms & systems for previously intractable problems**

<table>
<thead>
<tr>
<th>Problem Domain</th>
<th>Metagenome Assembly</th>
<th>Protein Clustering and Annotation</th>
<th>Comparative Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Science need</strong></td>
<td>Find species, genes and relative abundance in microbial communities</td>
<td>Improve understanding of tree of life for microbes; aid in identifying gene function</td>
<td>Track microbiome over time or space, changes in environment, climate, etc.</td>
</tr>
<tr>
<td><strong>Computing techniques</strong></td>
<td>hash tables, alignment, k-mer counts, graph walks</td>
<td>Hash tables tables, alignment, k-mer counts, sparse matrices, ML (clustering, GNNs)</td>
<td>hash tables, alignment, k-mer counts, ML (dimensionality reduction)</td>
</tr>
</tbody>
</table>

[http://exabiome.org](http://exabiome.org)
Motifs of Bio Data Analysis

Short Read Assembly:
- Dynamic Programming Algorithms.
- Distributed and local Graph Traversals.
- Distributed and local hash tables.

Long Read Assembly:
- Dynamic Programming Algorithms.
- Sparse matrix multiplication.
- Distributed hash tables.

Protein Similarity and Clustering:
- Dynamic Programming Algorithms.
- Sparse Matrix Multiplication.

<table>
<thead>
<tr>
<th>kmers</th>
<th>Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGCA</td>
<td>T</td>
</tr>
<tr>
<td>GCAT</td>
<td>G</td>
</tr>
<tr>
<td>CATG</td>
<td>C</td>
</tr>
</tbody>
</table>
**Ideal for GPUs:**
- Localized and predictable memory access pattern.
- A lot of computations per each memory access.
- Equal amount of work can be distributed across threads.

**What we have:**
- Random or along diagonals memory access pattern.
- Integer only computations bound by memory bandwidth.
- Non-deterministic amount of work.
- Varying or limited parallelism (DP and graph algorithms).
Sequence Alignment
Diagonal Major Indexing allows for coalesced memory access in the global memory.
ADEPT’s performance on DNA

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Query Set</th>
<th>Reference Set</th>
<th>Total Alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min. Size</td>
<td>Max. Size</td>
<td>Min. Size</td>
</tr>
<tr>
<td>DNA-1</td>
<td>150</td>
<td>200</td>
<td>99</td>
</tr>
<tr>
<td>DNA-2</td>
<td>201</td>
<td>250</td>
<td>99</td>
</tr>
<tr>
<td>DNA-3</td>
<td>251</td>
<td>300</td>
<td>99</td>
</tr>
</tbody>
</table>

GCUPS for Multiple GPUs vs 32 Haswell Cores.

ADEPT’s performance on Protein

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Query Set</th>
<th>Reference Set</th>
<th>Total Alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min. Size</td>
<td>Max. Size</td>
<td>Min. Size</td>
</tr>
<tr>
<td>Protein-1</td>
<td>20</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Protein-2</td>
<td>20</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Protein-3</td>
<td>20</td>
<td>600</td>
<td>600</td>
</tr>
</tbody>
</table>

**GCUPS for Multiple GPU vs 32 Haswell Cores**

ADEPT in MetaHipMer

Diagram showing the flow of data through the ADEPT system in MetaHipMer, with steps including Seed Extraction, Distributed Index, Candidate Lookup, and Batch processing, leading to GPU outputs.
ADEPT in PASTIS

Dataset size: 5 million protein sequences
9.9 billion candidate alignments
~853 Million alignments

Protein similarity graph
▷ 5 Million nodes
▷ 64.6 million edges

100 nodes of NERSC Cori and Summit
▷ ADEPT
▷ SeqAn

Alignment: 5.2x speedup

Slide from Oguz Selvitopi
Local Assembly
MetaHipMer Time Breakdown

- 64 nodes of Summit system using the marine communities dataset*
- Total time 2128 seconds.
- Local assembly module takes the most amount of time.

*https://gold.jgi.doe.gov/biosamples?id=Gb0192059
Local Assembly

Reads aligning to ends of contigs are obtained from the alignment stage.

Reads are broken down into kmers to construct De Bruijn graphs.

De Bruijn graphs are traversed to extend contigs.
Building kmer hashtable

contig
ATGCATGCATGCA

reads
ATGCATG
ATGCATG
ATGCATG

kmers | Ext
---|---
TGCA | T
ATGC | A
GCAT | G
CATG | C
DNA Walks

<table>
<thead>
<tr>
<th>kmers</th>
<th>Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGCA</td>
<td>T</td>
</tr>
<tr>
<td>GCAT</td>
<td>G</td>
</tr>
<tr>
<td>CATG</td>
<td>C</td>
</tr>
</tbody>
</table>

ATGCATGCA
ATGCATGCA
ATGCATGCAT
ATGCATGCATG
ATGCATGCATGC
Implementation Challenges on GPU

- Not a typical GPU problem
- No dynamic memory allocation on GPUs and no support for STL containers.
- Length of walks is non-deterministic.
- Static memory allocation can be used.
- Accurate memory usage needs to be known before kernel launch time.
Local Assembly Kernel

CUDA warps

One extension per warp

Hash table construction, all threads participate

DNA walks, one thread participates

Broadcast the state of walk, all threads participate

Repeat with different k-mer size if walk not accepted.
CUDA’s atomic CAS paired with `match_any_sync` was used for mimicking an atomic region.
• Driver function packs the data and makes calls to GPU kernels.
• Driver is called from a separate CPU thread which runs in background.
• This allows for CPU stealing.
• NVIDIA’s MPS handles multiple kernel launches from different ranks.
GPU Accelerated Local Assembly

MetaHipMer Comparison

<table>
<thead>
<tr>
<th></th>
<th>arctic, N=1, 0.9G</th>
<th>arcticsynth, N=8, 11G</th>
<th>WA0, N=32, 71G</th>
<th>WA, N=256, 813G</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-mer analysis</td>
<td>3.15</td>
<td>3.67</td>
<td>4.35</td>
<td>4.79</td>
</tr>
<tr>
<td></td>
<td>3.35</td>
<td>4.79</td>
<td>5.80</td>
<td>9.42</td>
</tr>
<tr>
<td>alignment</td>
<td>2.32</td>
<td>2.13</td>
<td>3.22</td>
<td>4.27</td>
</tr>
<tr>
<td></td>
<td>2.13</td>
<td>5.70</td>
<td>4.69</td>
<td>4.69</td>
</tr>
<tr>
<td>local assembly</td>
<td>3.11</td>
<td>4.53</td>
<td>2.26</td>
<td>2.26</td>
</tr>
<tr>
<td>overall</td>
<td>3.11</td>
<td>4.53</td>
<td>2.26</td>
<td>2.26</td>
</tr>
</tbody>
</table>

speedup GPU/CPU
Porting to non-NVIDIA devices

- Planned and current exascale supercomputers do not have NVIDIA devices.
- HIP and SYCL are possible choices for AMD and Intel GPUs respectively.
- Some efforts within DOE are focused on supporting multiple backends and runtimes.
Acknowledgements

Rob Egan  Steven Hofmeyr  Jack Deslippe  Oguz Selvitopi  Aydin Buluc  Leonid Oliker  Katherine Yelick
Thank you!