PPoPP 2020

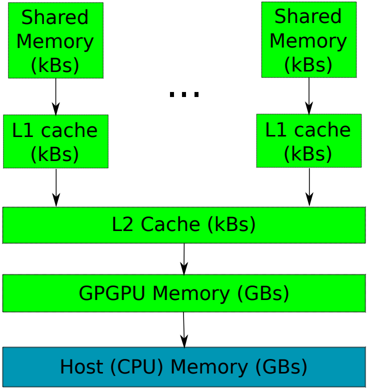
POSTER: Optimizing GPU Programs By Partial Evaluation

Aleksey Tyurin, Daniil Berezun, **Semyon Grigorev**

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

February 24, 2020

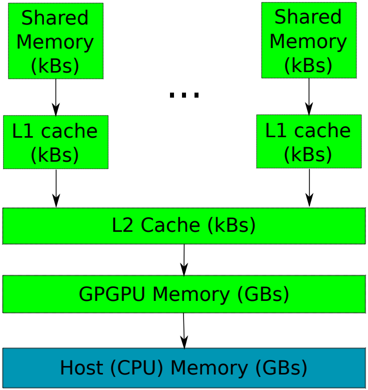
GPGPU Architecture



Navigation through a graph

- Global memory
  - ☺ Big

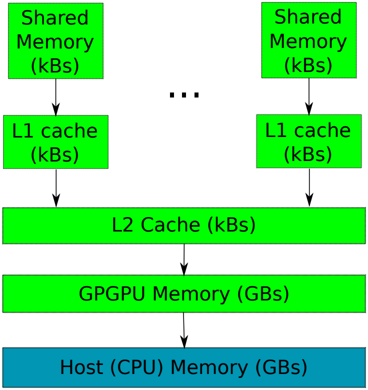
GPGPU Architecture



Navigation through a graph

- Global memory
  - ☺ Big
  - ☹ Slow
- Shared memory
  - ☺ Fast

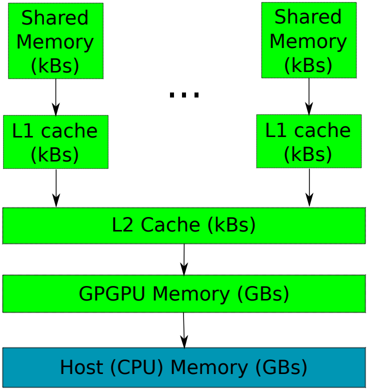
GPGPU Architecture



Navigation through a graph

- Global memory
  - ☺ Big
  - ☹ Slow
- Shared memory
  - ☺ Fast
  - ☹ Relatively small
  - ☹ Manual allocation mamagement
- Constant memory
  - ☺ Fast

GPGPU Architecture



Navigation through a graph

- Global memory
  - ☺ Big
  - ☹ Slow
- Shared memory
  - ☺ Fast
  - ☹ Relatively small
  - ☹ Manual allocation mamagement
- Constant memory
  - ☺ Fast
    - ☹ Only for appropriate access pattern
  - ☹ Small
  - ☹ Static allocation
- **Memory traffic is a bottleneck**

Data Processing

- Substring matching
- Filtering by using Hidden Markov Models (HMM)

Data Processing

- Substring matching
- Filtering by using Hidden Markov Models (HMM)

```
__global__ void estimateSimilarity
                     (int* filterParams, int* data, ...)
{
   __shared__ int cachedFilterParams[size];

   /*some code to load filterParams
     to cachedFilterParams*/
   ...
}
```

Data Processing

- Substring matching
- Filtering by using Hidden Markov Models (HMM)

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
   __shared__ int cachedFilterParams[size];

   /*some code to load filterParams
     to cachedFilterParams*/
   ...
}
```

Data Processing

- Substring matching
- Filtering by using Hidden Markov Models (HMM)

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
   __shared__ int cachedFilterParams[size];

   /*some code to load filterParams
     to cachedFilterParams*/
   ...
}
```

Data Processing

- Substring matching
- Filtering by using Hidden Markov Models (HMM)

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
   __shared__ int cachedFilterParams[size];

   /*some code to load filterParams
     to cachedFilterParams*/
   ...
}
```

Big Data Processing

- Substring matching $\Rightarrow$ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) $\Rightarrow$ Homology search (bioinformatics)

Big Data Processing

- Substring matching $\Rightarrow$ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) $\Rightarrow$ Homology search (bioinformatics)

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
   ...
}
```

Big Data Processing

- Substring matching ⇒ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

Many data chunks
⇒ many runs
of procedure

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
    ...
}
```

Big Data Processing

- Substring matching ⇒ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for many data chunks

Many data chunks ⇒ many runs of procedure

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
    ...
}
```

Big Data Processing

- Substring matching ⇒ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for many data chunks

Many data chunks ⇒ many runs of procedure

```
__global__ void estimateSimilarity
                    (int* filterParams, int* data, ...)
{
    ...
}
```

`filterParams` is a static during one data porcessing session.

Big Data Processing

- Substring matching ⇒ Data curving (digital forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for many data chunks

Many data chunks ⇒ many runs of procedure

```
__global__ void estimateSimilarity
                     (int* filterParams, int* data, ...)
{
    ...
}
```

`filterParams` is a static during one data porcessing session.
How can we use this fact to optimize our procedure?

Partial Evaluation

!!! Framework

[Scipy] Sparse matrices multiplication by using **Scipy** in **Python**

!!! Framework

[Scipy] Sparse matrices multiplication by using **Scipy** in **Python**

[M4RI] Dense matrices multiplication by using **m4ri** library which implements the Method of Four Russians in **C**

Evaluation: Data Curving

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

Evaluation: Data Curving

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

[GPU_N] Our own implementation of the naïve boolean matrix multiplication in **CUDA C**

Evaluation: Data Curving

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

[GPU_N] Our own implementation of the naïve boolean matrix multiplication in **CUDA C**

[GPU_Py] Our own implementation of naïve boolean matrix multiplication in **Python** by using **numba** compiler

Evaluation: !!!

[CuSprs]

- Rustam Azimov, 2018, "Context-free Path Querying by Matrix Multiplication"
- Implementation is based on NVIDIA cuSPARSE library (**CUDA C, GPGPU**)

Evaluation: !!!

[CuSprs]
- Rustam Azimov, 2018, "Context-free Path Querying by Matrix Multiplication"
- Implementation is based on NVIDIA cuSPARSE library (**CUDA C, GPGPU**)

[CYK]
- X. Zhang et al, 2016, "Context-free path queries on RDF graphs"
- CYK-based algorithm implemented in **Java** (CPU)

Limitations

[RDF]

- The set of the real-world RDF files (ontologies)
- Queries:
  $G_4 : s \to SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$
  $G_5 : s \to SCOR\ s\ SCO \mid SCO$

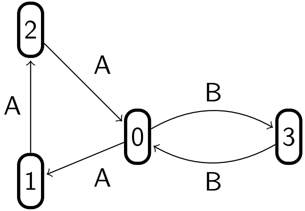
Limitations

[RDF]

- The set of the real-world RDF files (ontologies)
- Queries:
  $G_4 : s \to SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$
  $G_5 : s \to SCOR\ s\ SCO \mid SCO$

[Worst]

- The input graph is two cycles of coprime lengths with one shared vertex



- Query: $G_1 : s \to A\ s\ B \mid A\ B$

Dataset

[Full]

- The input graph is sparse, but the result is a full graph
- Queries:
  $G_2 : s \to s\ s \mid A$
  $G_3 : s \to s\ s\ s \mid A$

Dataset

[Full]

- The input graph is sparse, but the result is a full graph
- Queries:
  $G_2 : s \to s\ s \mid A$
  $G_3 : s \to s\ s\ s \mid A$

[Sparse]

- Sparse graphs are generated by GTgraph
- Query: $G_1 : s \to A\ s\ B \mid A\ B$

Conclusion

- OS: Ubuntu 18.04
- CPU: Intel core i7 8700k 3,7GHz
- RAM: DDR4 32 Gb
- GPGPU: NVIDIA GeForce 1080Ti (11Gb RAM)

Future Research

- Investigate implemented algorithms to explain nontrivial behaviors
- Create open extensible platform for CFPQ algorithms comparison
- Evaluate other CFPQ algorithms
  - Sparse matrices
  - Destributed matrix multiplication
  - LL- and LR-based algorithms
- Add new data and queries to the dataset
  - Bigger RDFs
  - Static code analysis

Contact Information

- Semyon Grigorev:
  - s.v.grigoriev@spbu.ru
  - Semen.Grigorev@jetbrains.com
- Nikita Mishin: mishinnikitam@gmail.com
- Iaroslav Sokolov: sokolov.yas@gmail.com
- Egor Spirin: egor@spirin.tech
- Vladimir Kutuev: vladimir.kutuev@gmail.com
- Egor Nemchinov: nemchegor@gmail.com
- Sergey Gorbatyuk: sergeygorbatyuk171@gmail.com

- Dataset and algorithm implementations: https://github.com/SokolovYaroslav/CFPQ-on-GPGPU

Thanks!