

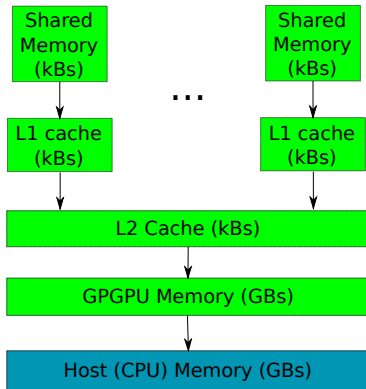
POSTER: Optimizing GPU Programs By Partial Evaluation

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

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

February 24, 2020

GPGPU memory hierarchy

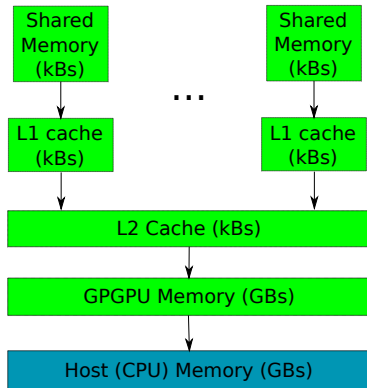


GPGPU Architecture

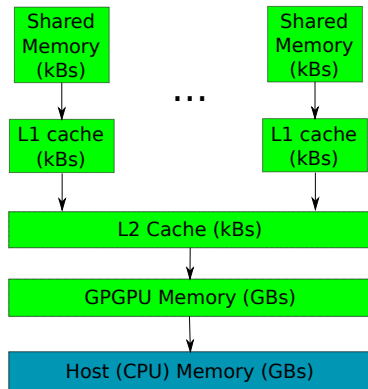
GPGPU memory hierarchy

- Global memory

- 😊 Big
- 😞 Slow



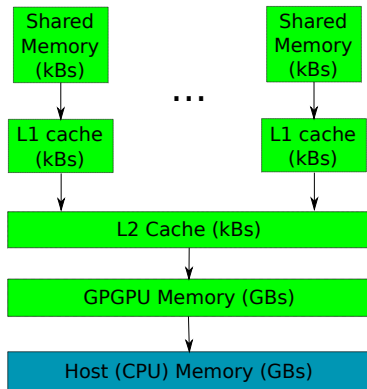
GPGPU Architecture



GPGPU memory hierarchy

- Global memory
 - 😊 Big
 - 😞 Slow
- Shared memory
 - 😊 Fast
 - 😞 Relatively small
 - 😞 Manual allocation management

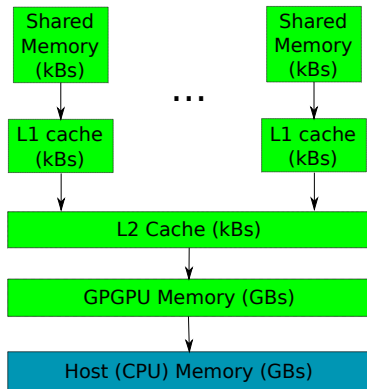
GPGPU Architecture



GPGPU memory hierarchy

- Global memory
 - 😊 Big
 - 😞 Slow
- Shared memory
 - 😊 Fast
 - 😞 Relatively small
 - 😞 Manual allocation management
- Constant memory
 - 😊 Fast
 - 😞 Only for appropriate access pattern
 - 😞 Small
 - 😞 Static allocation

GPGPU Architecture



GPGPU memory hierarchy

- Global memory
 - 😊 Big
 - ☹ Slow
- Shared memory
 - 😊 Fast
 - ☹ Relatively small
 - ☹ Manual allocation management
- 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 handleData
    (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 handleData
    (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 handleData
    (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 handleData
    (int* filterParams, int* data, ...)
{
    __shared__ int cachedFilterParams[size];

    /*some code to load filterParams
       to cachedFilterParams*/

    ...
}
```

Big Data Processing

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

Big Data Processing

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

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

Big Data Processing

- Substring matching \Rightarrow Data curving (cyber forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

Many data chunks
 \Rightarrow many runs
of procedure

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

Big Data Processing

- Substring matching \Rightarrow Data curving (cyber forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for
many data chunks

Many data chunks
 \Rightarrow many runs
of procedure

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

Big Data Processing

- Substring matching \Rightarrow Data curving (cyber forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for
many data chunks

Many data chunks
 \Rightarrow many runs
of procedure

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

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

Big Data Processing

- Substring matching \Rightarrow Data curving (cyber forensics)
- Filtering by using Hidden Markov Models (HMM) (bioinformatics)

One filter for
many data chunks

Many data chunks
 \Rightarrow many runs
of procedure

```
__global__ void handleData  
    (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 or Specialization

$$\underbrace{\llbracket \textit{handleData} \rrbracket}_{\textit{handleData}}[\textit{filterParams}, \textit{data}] = \underbrace{\llbracket \llbracket \textit{mix} \rrbracket[\textit{handleData}, \textit{filterParams}] \rrbracket}_{\textit{handleData}_{\textit{mix}}}[\textit{data}]$$

Partial Evaluation or Specialization

$$\underbrace{\llbracket \text{handleData} \rrbracket}_{\text{handleData}}[\text{filterParams}, \text{data}] = \underbrace{\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, \text{filterParams}] \rrbracket}_{\text{handleData}_{\text{mix}}}[\text{data}]$$

```
handleData (filterParams, data)
{
    res = new List()
    for d in data
        for e in filterParams
            if d % e == 0
                then res.Add(d)
    return res
}
```

Partial Evaluation or Specialization

$$\underbrace{\llbracket \text{handleData} \rrbracket}_{\text{handleData}}[\text{filterParams}, \text{data}] = \underbrace{\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, \text{filterParams}] \rrbracket}_{\text{handleData}_{\text{mix}}}[\text{data}]$$

$$\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, [2; 3]] \rrbracket$$

```
handleData (filterParams, data)
{
  res = new List()
  for d in data
    for e in filterParams
      if d % e == 0
        then res.Add(d)
  return res
}
```

Partial Evaluation or Specialization

$$\underbrace{\llbracket \text{handleData} \rrbracket}_{\text{handleData}}[\text{filterParams}, \text{data}] = \underbrace{\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, \text{filterParams}] \rrbracket}_{\text{handleData}_{\text{mix}}}[\text{data}]$$

$\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, [2; 3]] \rrbracket$

```
handleData (filterParams, data)
{
  res = new List()
  for d in data
    for e in filterParams
      if d % e == 0
        then res.Add(d)
  return res
}
```

```
handleData (data)
{
  res = new List()
  for d in data
    if d % 2 == 0 ||
       d % 3 == 0
      then res.Add(d)
  return res
}
```

Partial Evaluation or Specialization

$$\underbrace{\llbracket \text{handleData} \rrbracket}_{\text{handleData}}[\text{filterParams}, \text{data}] = \underbrace{\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, \text{filterParams}] \rrbracket}_{\text{handleData}_{\text{mix}}}[\text{data}]$$

$\llbracket \llbracket \text{mix} \rrbracket[\text{handleData}, [2; 3]] \rrbracket$

```
handleData (filterParams, data)
{
    res = new List()
    for d in data
        for e in filterParams
            if d % e == 0
                then res.Add(d)
    return res
}
```

```
handleData (data)
{
    res = new List()
    for d in data
        if d % 2 == 0 ||
           d % 3 == 0
            then res.Add(d)
    return res
}
```

- AnyDSL framework for specialization
 - ▶ Special DSL which can be specialized and compiled
 - ▶ Ahead-of-time specialization

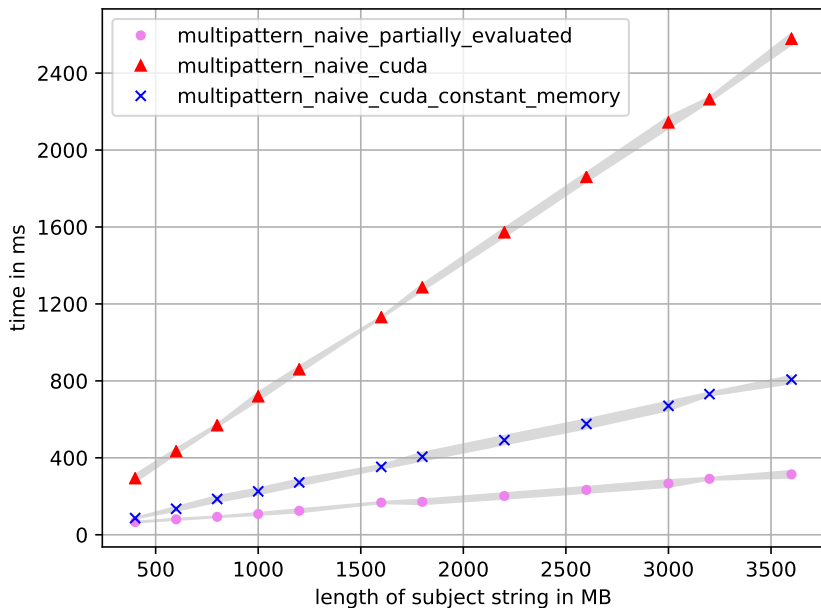
Evaluation Setup

- AnyDSL framework for specialization
 - ▶ Special DSL which can be specialized and compiled
 - ▶ Ahead-of-time specialization
- Algorithms
 - ▶ Naïve multiple substring matching
 - ▶

Evaluation Setup

- AnyDSL framework for specialization
 - ▶ Special DSL which can be specialized and compiled
 - ▶ Ahead-of-time specialization
- Algorithms
 - ▶ Naïve multiple substring matching
 - ▶
- Environment
 - ▶ Environment
 - ▶

Evaluation: Data Curving



Limitations

[RDF]

- ▶ The set of the real-world RDF files (ontologies)

- ▶ Queries:

$$G_4 : s \rightarrow SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$$

$$G_5 : s \rightarrow SCOR\ s\ SCO \mid SCO$$

Limitations

[RDF]

- ▶ The set of the real-world RDF files (ontologies)

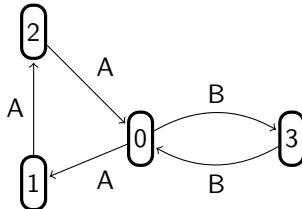
- ▶ Queries:

$G_4 : s \rightarrow SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$

$G_5 : s \rightarrow SCOR\ s\ SCO \mid SCO$

[Worst]

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



- ▶ Query: $G_1 : s \rightarrow A\ s\ B \mid A\ B$

- [Full]
- ▶ The input graph is sparse, but the result is a full graph
 - ▶ Queries:
 $G_2 : s \rightarrow s \ s \mid A$
 $G_3 : s \rightarrow s \ s \ s \mid A$

[Full]

- ▶ The input graph is sparse, but the result is a full graph

- ▶ Queries:

$$G_2 : s \rightarrow s \ s \mid A$$

$$G_3 : s \rightarrow s \ s \ s \mid A$$

[Sparse]

- ▶ Sparse graphs are generated by GTgraph

- ▶ Query: $G_1 : s \rightarrow A \ s \ B \mid A \ B$

Conclusion

- Just In Time speciaization
-
-
-

- Switch to CUDA C partial evaluator
- Reduce specialization overhead
- Integrete with shared memory register spilling
- Evaluate on real-world examples

Contact Information

- Semyon Grigorev:
 - ▶ s.v.grigoriev@spbu.ru
 - ▶ Semen.Grigorev@jetbrains.com
- Aleksey Tyurin: alekseytyurinspb@gmail.com
- Daniil Berezun: daniil.berezun@jetbrains.com
- Dataset and algorithm implementations:
<https://github.com/SokolovYaroslav/CFPQ-on-GPGPU>

Thanks!