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

Big Data Processing

- Substring matching ⇒ Data curving (cyber forensics)
- 2D convolution ⇒ Image processing
- Filtering by using HMMs ⇒ Homology search (bioinformatics)

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

Big Data Processing

- Substring matching ⇒ Data curving (cyber forensics)
- 2D convolution ⇒ Image processing
- Filtering by using HMM

One filter for many data chunks

Many data chunks ⇒ many procedure runs

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

Big Data Processing

- Substring matching ⇒ Data curving (cyber forensics)
- 2D convolution ⇒ Image processing
- Filtering by using HMM

One filter for many data chunks

Many data chunks ⇒ many procedure runs

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

filterParams is static during one data processing session

How can we use this fact to optimize our procedure?

Partial Evaluation or Specialization

$$[\![\underbrace{handleData}_{handleData}]\!][filterParams, data] = \underbrace{[\![[\![\overbrace{mix}^{\text{partial evaluator}}]\!][handleData, filterParams]]\!]}_{handleData_{mix}}[data]$$

Partial Evaluation or Specialization

$$[\![\underbrace{handleData}_{handleData}]\!][filterParams, data] = \underbrace{[\![[\![\overbrace{mix}^{\text{partial evaluator}}]\!][handleData, filterParams]]\!]}_{handleData_{mix}}[data]$$

```
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

$$[\![\underbrace{handleData}_{handleData}]\!][filterParams, data] = \underbrace{[\![[\![\overbrace{mix}^{\text{partial evaluator}}]\!][handleData, filterParams]]\!]}_{handleData_{mix}}[data]$$

$$[\![[\![mix]\!][handleData, [2; 3]]]\!]$$

```
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

$$[\![\underbrace{handleData}_{handleData}]\!][filterParams, data] = [\![\underbrace{[\![\overbrace{mix}^{\text{partial evaluator}}]\!][handleData, filterParams]}_{handleData_{mix}}]\!][data]$$

$$[\![[\![mix]\!][handleData, [2;3]]]\!]$$

```
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

$$[\![\underbrace{handleData}_{handleData}]\!][filterParams, data] = \underbrace{[\![\,[\![\overbrace{mix}^{\text{partial evaluator}}]\!][handleData, filterParams]\,]\!]}_{handleData_{mix}}[data]$$

$$[\![\,[\![mix]\!][handleData, [2;3]]\,]\!]$$

```
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
}
```

Evaluation Setup

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

Evaluation Setup

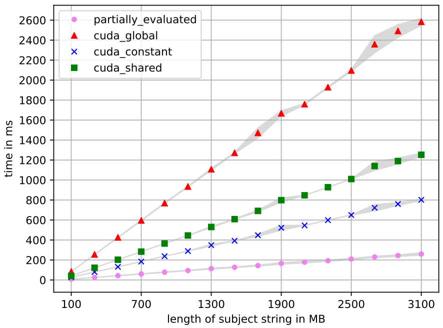
- We use AnyDSL framework for specialization
  - Special DSL which can be specialized and compiled
  - Ahead-of-time specialization
- Algorithms
  - Naïve multiple substring matching
  - 2D convolution

Evaluation Setup

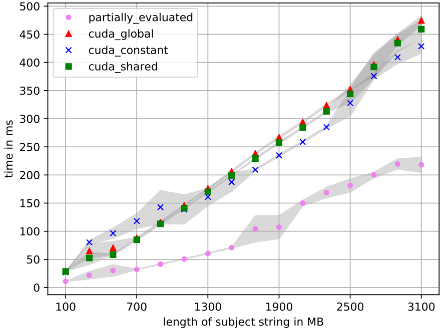
- We use AnyDSL framework for specialization
  - Special DSL which can be specialized and compiled
  - Ahead-of-time specialization
- Algorithms
  - Naïve multiple substring matching
  - 2D convolution
- Hardware
  - **GTX-1070**: Pascal architecture, 8GB GDDR5, 1920 CUDA cores
  - **Tesla T4**: Turing architecture, 16GB GDDR6, 2560 CUDA cores

Evaluation: Substring Matching

- Application: data curving
- Subject string: byte sequence from real hard drive
- Patterns: 16 file signatures from GCK's file signatures table[1]



Results for GTX-1070

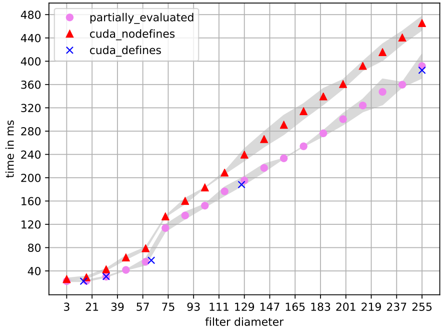


Results for Tesla T4

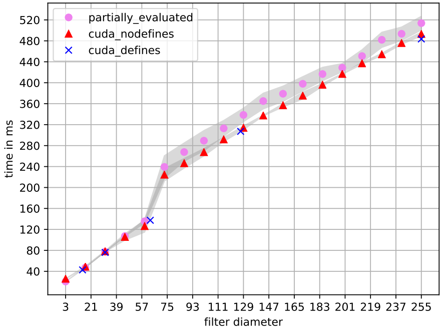
[1] `https://www.garykessler.net/library/file_sigs.html`

Evaluation: 2D Convolution

- Application: image processing
- Subject image: random image of size 1GB
- Filters: random square filters with diameter 3 to 255



Results for GTX-1070



Results for Tesla T4

Future Research

- Migration to CUDA C partial evaluator
  - LLVM.mix: partial evaluator for LLVM IR
- Reduction of specialization overhead
  - To be applicable in run-time
- Integration with shared memory register spilling
  - "RegDem: Increasing GPU Performance via Shared Memory Register Spilling" (Putt Sakdhnagool et.al. 2019)
- Evaluation on real-world examples
  - Homology search in bioinformatics
  - Regular expression matching for traffic analysis, log processing
  - Graph database querying
  - Ray tracing, path tracing

Contact Information

- Semyon Grigorev:
  - s.v.grigoriev@spbu.ru
  - Semen.Grigorev@jetbrains.com
- Aleksey Tyurin: alekseytyurinspb@gmail.com
- Daniil Berezun: daniil.berezun@jetbrains.com

Thanks!