VIVEKFEST, Pasadena, CA

October 21, 2024

Hidden Assumptions in Static Verification of Data-Race Free GPU Programs

Tiago Cogumbreiro / Julien Lange

University of Massachusetts Boston / Royal Holloway, University of London

About me

- 2012: Visiting PhD student
- 2015: Started my Postdoc at Rice
- 2017: Started my Postdoc at Georgia Tech
- Vivek has been instrumental in guiding me through the (US) academia

Goal: GPU static data-race analysis for everyone

- **Motivation:** Data-race freedom analysis of GPU kernels requires external assumptions (eg, thread config, parameter constraints)
- Challenge: how to configure these tools fully automatically?

Experiment

- **Data:** 191 data-race free kernels (pre-configured)
- **Test:** Disable/generalize analysis options to measure percentage of affected kernels

- RQ1: Which analysis features affect partial data-race freedom?
  - **Findings:** thread configuration most needed (98%), user-provided assumptions uncommonly needed (27%)
- RQ2: Can static data-race detection help with missing assumptions?
  - **Findings:** Yes, in 92% of the cases.

Background

GPU Programming

```
__global__ void saxpy(int n, float a, float *x, float *y) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    y[i] = a*x[i] + y[i];
}
```

GPU Programming

```
__global__ void saxpy(int n, float a, float *x, float *y) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    y[i] = a*x[i] + y[i];
}
```

Data-race free for `gridDim={4}, blockDim={8}`

	blockIdx.x=0	blockIdx.x=1	blockIdx.x=2	blockIdx.x=3
threadIdx.x=0	0	8	16	24
threadIdx.x=1	1	9	17	25
threadIdx.x=2	2	10	18	26
threadIdx.x=3	3	11	19	27
threadIdx.x=4	4	12	20	28
threadIdx.x=5	5	13	21	29
threadIdx.x=6	6	14	22	30
threadIdx.x=7	7	15	23	31

Faial: data-race freedom analysis

- Can verify data-race free CUDA kernels[FMSD23]
- Only uses information available ***in*** the kernel
  - Ignores kernel launch parameters

[FMSD23] **Memory Access Protocols: Certified Data-Race Freedom for GPU Kernels.** Tiago Cogumbreiro, Julien Lange, Dennis Liew, Hannah Zicarelli. FMSD, 2023.

Usage example

```
$ faial-drf saxpy1.cu
Kernel 'saxpy' is DRF!
```

Assumes thread configuration: blockDim=1024, gridDim=1

Ranging over all possible thread configurations

```
$ faial-drf --all-dims saxpy1.cu
```

```
Kernel 'saxpy' has 1 data-race.

~~~~ Data-race 1 (CIDI) ~~~~

6 |   if (i < n) y[i] = a*x[i] + y[i];

Globals
```

y[]	0
blockIdx	x = 0 | y = 0 | z = 0
n	1

```
Locals
```

threadIdx	x = 0 | y = 1 | z = 0	x = 0 | y = 0 | z = 0

```
True alarm detected!
```

	blockIdx.x=0	blockIdx.x=1	blockIdx.x=2	blockIdx.x=3
threadIdx.{x=0,y=0}	0	4	8	12
threadIdx.{x=1,y=0}	1	5	9	13
threadIdx.{x=2,y=0}	2	6	10	14
threadIdx.{x=3,y=0}	3	11	19	27
threadIdx.{x=0,y=1}	0	4	8	12
threadIdx.{x=1,y=1}	1	5	9	13
threadIdx.{x=2,y=1}	2	6	10	14
threadIdx.{x=3,y=1}	3	11	19	27

Observe a **true** data-race within a block (group of threads).

Static data-race detection for GPUs

- Faial[OOPSLA24] features ***static data-race detection***
- Distinguishes between **true** data-races and **potential** data-races.
- Aka **precise** data-race detection

[OOPSLA24] **Sound and partially-complete static analysis of data-races in GPU programs.** Dennis Liew, Tiago Cogumbreiro, Julien Lange. PACMPL, 8(OOPSLA2), 2024.

User-provided assumptions

- Constrain the thread configuration to rule out these data-races
- User-provided assumptions via `__assume` or early-return
- Faial supports both options

Option A: `__assume`

```
__assume(blockDim.y == 1 && blockDim.z == 1);
int i = blockIdx.x * blockDim.x + threadIdx.x;
y[i] = a*x[i] + y[i];
```

Option B: early return

```
if(blockDim.y ≠ 1 || blockDim.z ≠ 1) return;
int i = blockIdx.x * blockDim.x + threadIdx.x;
y[i] = a*x[i] + y[i];
```

Grid-level synchronization

- Grid-level analysis: data-races between groups of threads
- Grid-level analysis is disabled by default

```
faial-drf --all-dims --all-levels saxpy.cu
```

Fixed version

- Constrain the number of grids
- Data-race free in any possible usages

```
__assume(blockDim.y == 1 && blockDim.z == 1);
__assume(gridDim.y == 1 && gridDim.z == 1);
```

Kernel 'saxpy' has 1 data-race.

~~~~ Data-race 1 (CIDI) ~~~~

```
8 |   if (i < n) y[i] = a*x[i] + y[i];
```

Globals

y[]	0
blockDim	x = 4
gridDim	x = 1 | y = 2 | z = 2
n	1

Locals

blockIdx	x = 0 | y = 1 | z = 1	x = 0 | y = 0 | z = 0
threadIdx	x = 0 | y = 0 | z = 0	x = 0 | y = 0 | z = 0

True alarm detected!

Evaluation

Static data-race analysis for GPUs for everyone

UMASS BOSTON

Analysis features

- fixed vs ranging over all thread configurations
- user-provided constraints with `__assume`/early-return
- grid-level synchronization uncovers more data-races

Evaluation

- RQ1: Which analysis features affect partial data-race freedom?
- RQ2: Can static data-race detection help with missing assumptions?

Which analysis features affect partial data-race freedom?

Run		*DRF*	*Affected*
Baseline		191	0%
Grid-level	(-5)	186	3%
Ignore assume	(-52)	139	27%
Any thread conf	(-188)	3	98%
Every above	(-189)	2	99%

Data selection:

- Nvidia SDK
- Microsoft C++ samples
- gpgpu-sim benchmark

Baseline:

- fixed thread conf
- only block-level analysis
- may have `__assume`

Conclusions

- Almost every kernels assumes constraint thread configuration
- A relatively small (27%) number of kernels require user-assumptions
- Grid-level analysis the 5 missing are all timeouts
  Fix: set the SMT theory used to `AUFLIA`
- Every above: the 2 kernels had benign data-races and atomics

Can static data-race detection help with missing assumptions?

Consider every kernel that was not proved DRF (includes true-racy):

- Can static data-race detection find **true data races** in these kernels?

Run	*Racy*	*Non-DRF*	*Ratio*
Ignore assume	49	52	94%
Any thread conf	173	188	92%
Every above	173	189	92%

Yes, in at least 92% of kernels!

Additional findings

Our data-race detector helped fix the assumptions of **4 kernels**

- Grid-level analysis triggered incorrect thread configurations (3 kernels)
- Added a missing template-based constraint (1 kernel)

Enabling all-thread-dims and grid synchronization is a great sanity check!

> 6 kernels were ***incorrectly*** labelled DRF, due to limitations of Faial

- pointer to array used as loop variable
- &-references in function parameters

Conclusion

Hidden Assumptions in Static Verification of Data-Race Free GPU Programs

- Studied Faial a static verifier of GPU kernels: detects data-race freedom and data-races

`gitlab.com/umb-svl/faial/`

- **Data:** 191 data-race free kernels (pre-configured)
- **Test:** Disable/generalize analysis options to measure percentage affected

- RQ1: Which analysis features affect partial data-race freedom?
  - **Findings:** thread configurations crucial (affects 98%)
    user-provided assumptions uncommonly needed (affects 27%)
  - **Insight:** all-thread-configurations + grid sync + block sync = **good sanity check**
    (2 limitations found)
- RQ2: Can static data-race detection help with missing assumptions?
  - **Findings:** Yes, in 92% of the cases.