MANCHESTER 1824
The University of Manchester

Exploiting High-Performance Heterogeneous Hardware for Java Programs using Graal

James Clarkson[±], *Juan Fumero*[*], Michalis Papadimitriou[*], Foivos S. Zakkak[*], Christos Kotselidis[*] and Mikel Luján[*]

[±]Dyson, [*]The University of Manchester

ManLang'18, Linz (Austria), 12th September 2018

MANCHESTER 1824
The University of Manchester

Outline

- Background
- Tornado
  - Tornado-API
  - Tornado Runtime
  - Tornado JIT Compiler
- Performance Results
- Conclusions

MANCHESTER 1824
The University of Manchester

Context of this project

Started as the PhD thesis of **James Clarkson**: *Compiler and Runtime Support for Heterogeneous Programming*

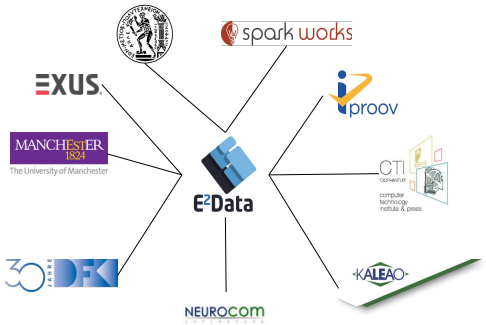
James Clarkson, Christos Kotselidis, Gavin Brown, and Mikel Luján.
Boosting Java Performance using GPGPUs.
In Proceedings of the 30th International Conference on Architecture of Computing Systems

Christos Kotselidis, James Clarkson, Andrey Rodchenko, Andy Nisbet, John Mawer, and Mikel Luján.
Heterogeneous Managed Runtime Systems: A Computer Vision Case Study
ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '17)

Partially funded by the EPSRC AnyScale grant EP/L000725/1

MANCHESTER 1824
The University of Manchester

Currently part of the EU H2020 E2Data Project



"End-to-end solution for heterogeneous Big Data deployments that fully exploits and advances the state-of-the-art in infrastructure" `https://e2data.eu/`

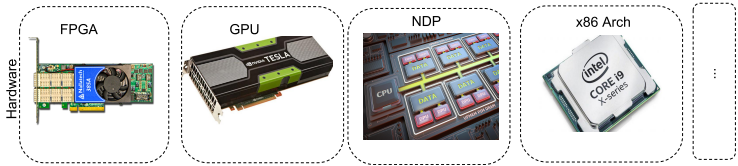
European Union's Horizon H2020 research and innovation programme under grant agreement No 780622

MANCHESTER 1824
The University of Manchester

1. Background

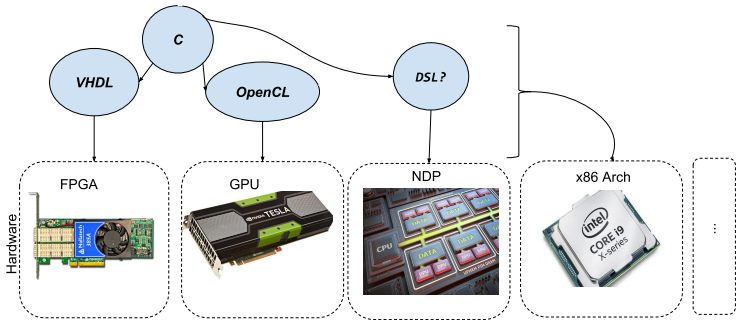
MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape



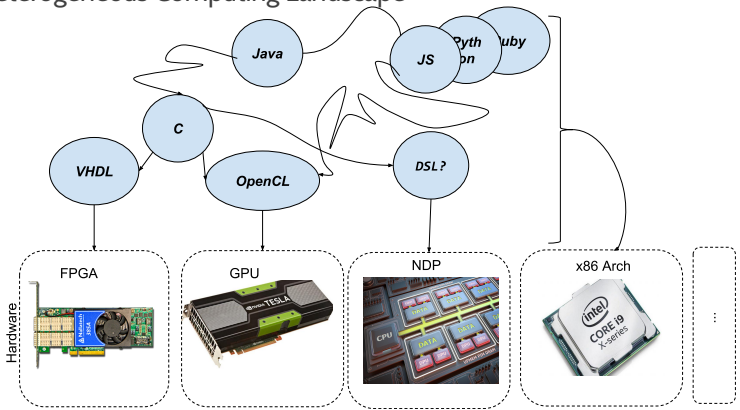
MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape



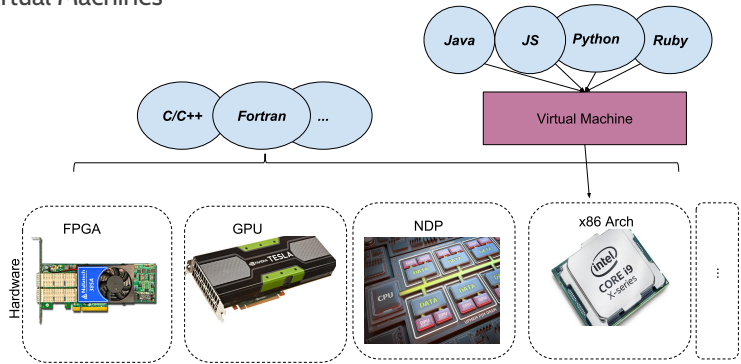
MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape



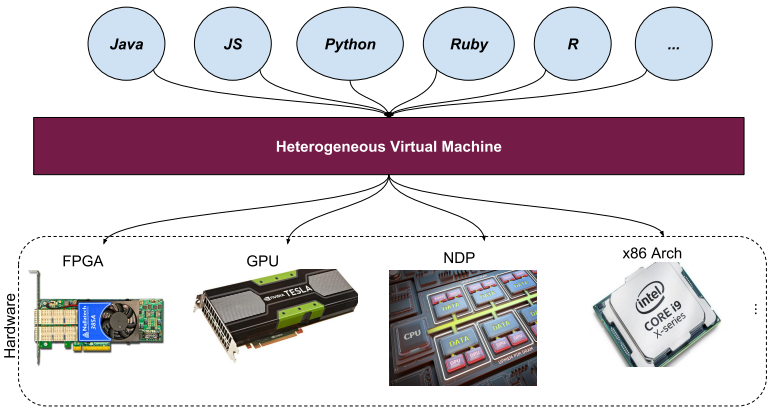
MANCHESTER 1824
The University of Manchester

Current Virtual Machines



MANCHESTER 1824
The University of Manchester

Our Solution: VM + Heterogeneous Runtime



MANCHESTER 1824
The University of Manchester

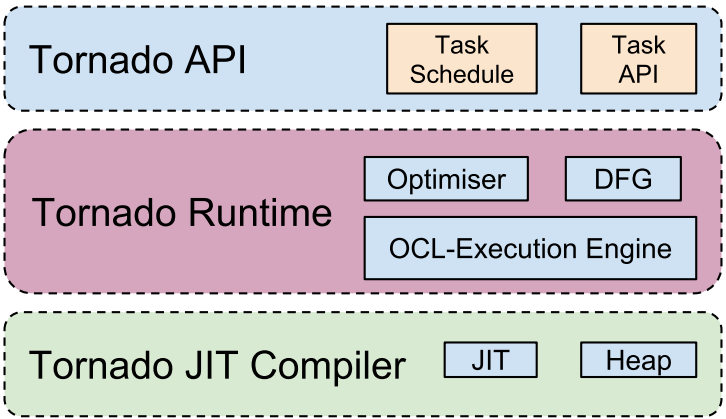
2. Tornado: A Practical Heterogeneous Programming Framework

Tornado

- A Java based Heterogeneous Programming Framework
- It exposes a task-based parallel programming API
- It contains an OpenCL JIT Compiler and a Runtime for running on heterogeneous devices
- Modular system currently using:
  - OpenJDK/Graal
  - OpenCL
- It currently runs on CPUs, GPUs and FPGAs*

MANCHESTER 1824
The University of Manchester

Tornado Overview



MANCHESTER 1824
The University of Manchester

Tornado API: @Parallel

"It's a developer provided annotation that instructs the JIT compiler that it is OK for each iteration to be executed independently."

It does not specify or imply:

- iterations should be executed in parallel;
- the parallelization scheme to be used

MANCHESTER 1824
The University of Manchester

Task Schedules

"A task schedule describes how to co-ordinate the execution of tasks across heterogeneous hardware.".

- Composability
- Sequential consistency
- Task-based parallelism
- Automatic and optimised data movement

MANCHESTER 1824
The University of Manchester

Tornado API: enabling task-based parallelism

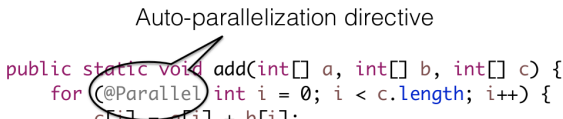
```
public static void add(int[] a, int[] b, int[] c) {
    for (@Parallel int i = 0; i < c.length; i++) {
        c[i] = a[i] + b[i];
    }
}

// execute array addition on an accelerator
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];


new TaskSchedule("s0")
    .task("t0", ArrayAddInt::add, a, b, c)
    .streamOut(c)
    .execute();
```

MANCHESTER 1824
The University of Manchester

Tornado API: enabling task-based parallelism




```
public static void add(int[] a, int[] b, int[] c) {
    for (@Parallel int i = 0; i < c.length; i++) {
        c[i] = a[i] + b[i];
    }
}

// execute array addition on an accelerator
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];


new TaskSchedule("s0")
    .task("t0", ArrayAddInt::add, a, b, c)
    .streamOut(c)
    .execute();
```

MANCHESTER 1824
The University of Manchester

Tornado API: enabling task-based parallelism

Auto-parallelization directive

```
public static void add(int[] a, int[] b, int[] c) {
    for (@Parallel int i = 0; i < c.length; i++) {
        c[i] = a[i] + b[i];
    }
}

// execute array addition on an accelerator
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];
```

```
new TaskSchedule("s0")
    .task("t0", ArrayAddInt::add, a, b, c)
    .streamOut(c)
    .execute();
```

Task-based execution model

MANCHESTER 1824
The University of Manchester

Task Schedules: example

```
1  class Ex {
2    public static void multiply
3        (Double4[] a, Double4[] b, Double4[] c) {
4      // code here
5    }
6
7    public static void add
8        (Double4 []a, Double4[] b, Double4[] c) {
9      // code here
10   }
11 }
```

MANCHESTER 1824
The University of Manchester

Task Schedules: example

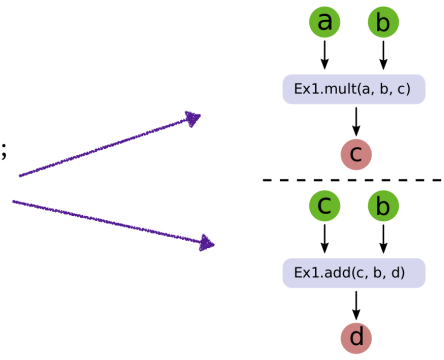
```
TaskSchedule schedule = new TaskSchedule("s0")
  .streamIn(a,b)                    // copy a & b to device
  .task("t0", Ex::multiply,a,b,c)   // multiply task
  .task("t1", Ex::add,c,b,d)        // add task
  .streamOut(d);                    // return d to host


while (true) {
   schedule.execute();
}
```

MANCHESTER 1824
The University of Manchester

Task Schedules: example

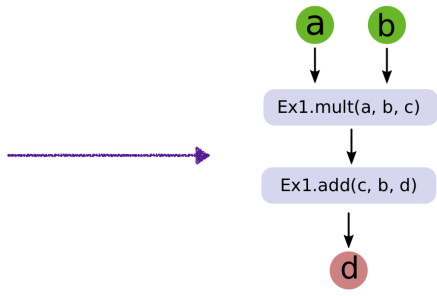
```
while (true) {
    task(multiply(a, b, c));
    task(add(c, b, d));
}
```



MANCHESTER 1824
The University of Manchester

Task Schedules: example

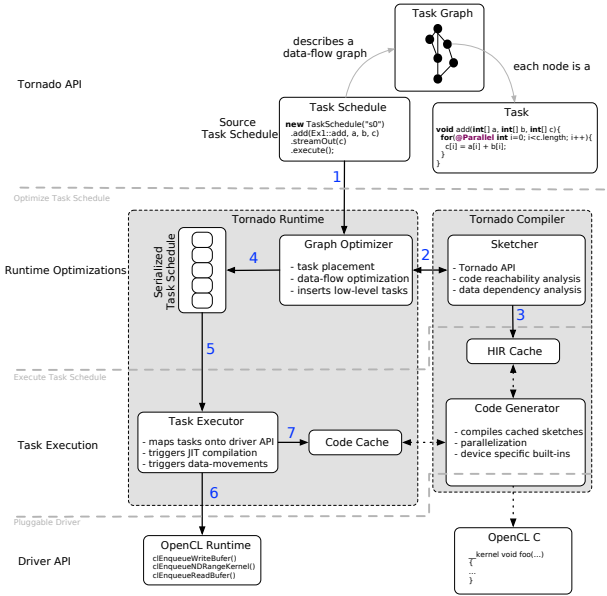
```
while (true) {
    schedule.execute();
}
```



MANCHESTER 1824
The University of Manchester

3. Tornado Runtime

Tornado: WorkFlow



MANCHESTER 1824
The University of Manchester

Data parallelism - Task specialisation

E.g., currently we have two parallel schemes: course-grain and fine-grain

```
1  // Loop for GPUs
2  int idx = get_global_id(0);
3  int size = get_global_size(0);
4  for (int i = idx; i < c.length;
5       i += size) {
6    // computation
7    c[i] = a[i] + b[i];
8  }
```

```
1  // Loop for CPUs
2  int id = get_global_id(0);
3  int size = get_global_size(0);
4  int block_size = (size +
5            inputSize - 1) / size;
6  int start = id * block_size;
7  int end = min(start + bs, c.length);
8  for (int i = start; i < end; i++) {
9    // computation
10   c[i] = a[i] + b[i];
11 }
```

MANCHESTER 1824
The University of Manchester

Memory Management

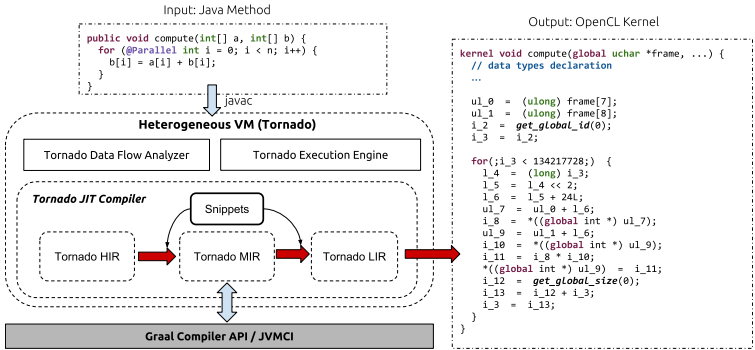
- Each heterogeneous device has a managed heap
- Enables objects to persist on devices
- Currently we duplicate objects which reside in the JVM heap
- No object creation on devices

MANCHESTER 1824
The University of Manchester

4. Tornado JIT Compiler

MANCHESTER 1824
The University of Manchester

Tornado JIT Compiler

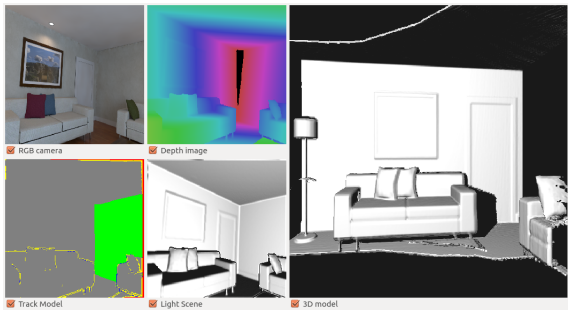


MANCHESTER 1824
The University of Manchester

5. Case study

Case study

Kinect Fusion: it is a complex computer vision application that is able to re-construct a 3D movel from RGB-D camera in real time.



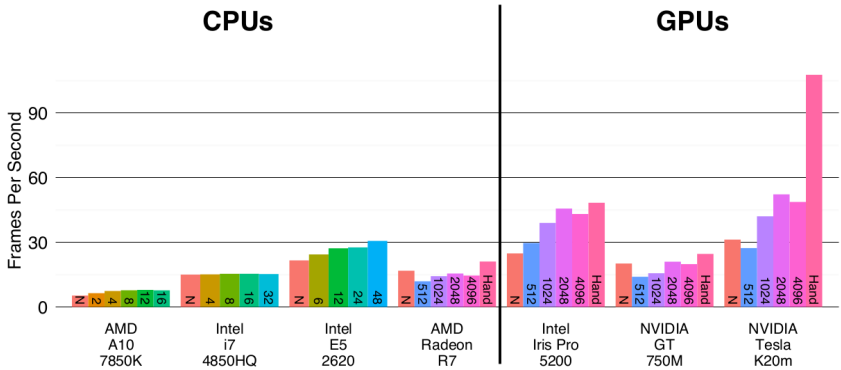
MANCHESTER 1824
The University of Manchester

Why KFusion?

- Not a normal Java application
- Complex multi-kernel pipeline
  - Sustained the execution of 540-1620 kernels per second.
  - SLA of 30 FPS
- Representative of cutting edge robotics/computer vision applications
- Want to deploy across many platform and accelerator combinations

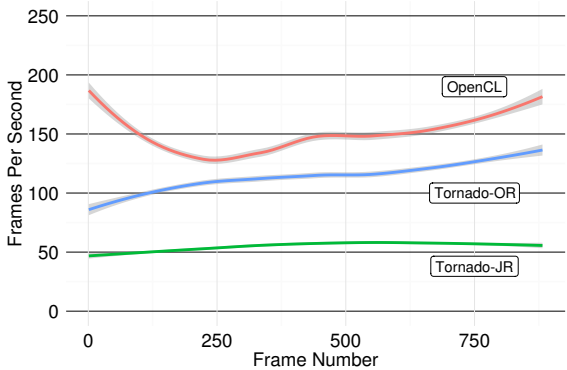
MANCHESTER 1824
The University of Manchester

What did we get with Tornado?



Running on NVIDIA Tesla, up to 150 fps

And compared to native code?



Tornado is 28% slower than the best OpenCL native code.

MANCHESTER 1824
The University of Manchester

6. Announcement & Conclusions

MANCHESTER 1824
The University of Manchester

Tornado is now Open Source!

```
$ git clone https://github.com/beehive-lab/tornado
$ make
$ tornado YourApplication
```

- We also have a poster tormorrow, come along!
- If you are interested, we can also show you demos on GPUs and FPGAs!

MANCHESTER 1824
The University of Manchester

Takeaway

- We have presented Tornado
- We have shown runtime code generation for OpenCL
- We have shown a case study for computer vision
- It is open-source, give a try!

We are looking forward for your feedback!

MANCHESTER 1824
The University of Manchester

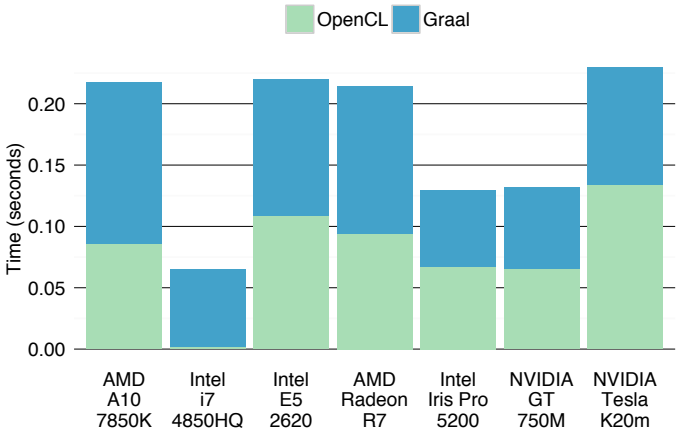
Thank you very much for your attention

This work is partially supported by the EPSRC grants PAMELA EP/K008730/1 and AnyScale Apps EP/L000725/1, and the EU Horizon 2020 E2Data 780245.

Juan Fumero <juan.fumero@manchester.ac.uk>

MANCHESTER 1824
The University of Manchester

Compilation times



MANCHESTER 1824
The University of Manchester

OpenCL Device Driver: Just In Time Compiler

OpenCL JIT Compiler and Runtime