MANCHESTER 1824
The University of Manchester

Dynamic Application Reconfiguration on Heterogeneous Hardware

Juan Fumero, Michail Papadimitriou, Foivos S. Zakkak,
Maria Xekalaki, James Clarkson, Christos Kotselidis

The University of Manchester, UK

Virtual Execution Environments (VEE), 14th April 2019
Providence, RI, US

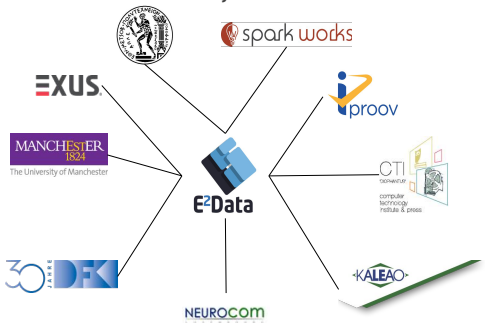
MANCHESTER 1824
The University of Manchester

Outline

- Motivation
- TornadoVM
  - JIT Compiler and Runtime
  - TornadoVM and Virtual Layer
- Evaluation
- Conclusions

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/`

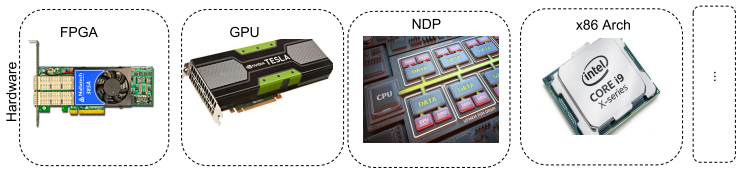
European Union's Horizon H2020 research and innovation programme under grant agreement No 780622

MANCHESTER 1824
The University of Manchester

1. Motivation

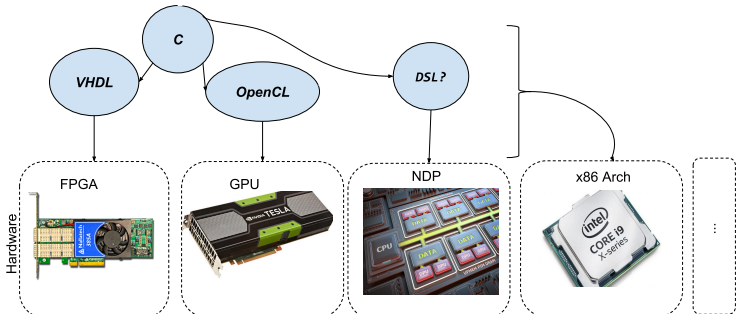
MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape



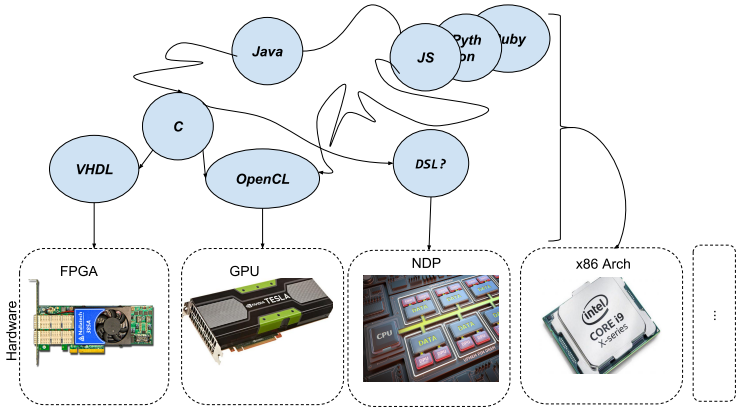
MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape

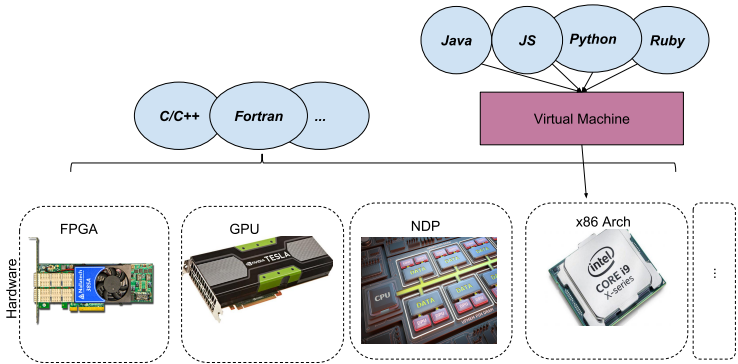


MANCHESTER 1824
The University of Manchester

Current Heterogeneous Computing Landscape

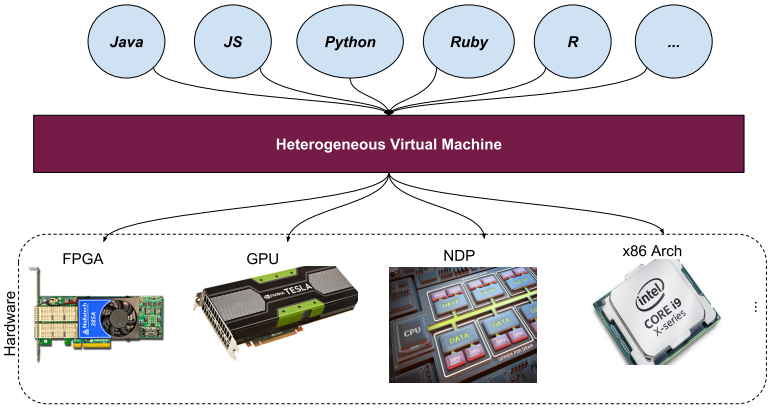


Current Heterogeneous Computing Landscape



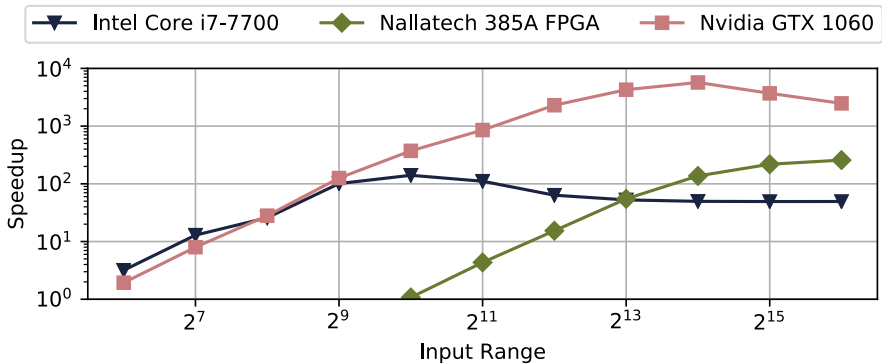
MANCHESTER 1824
The University of Manchester

Our Solution: VM + Heterogeneous Runtime



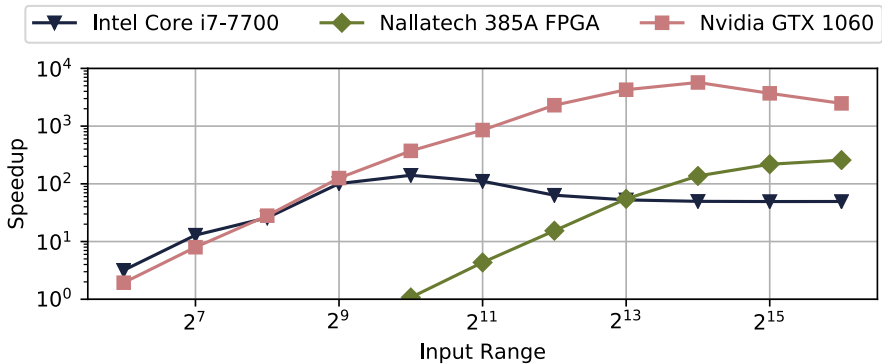
MANCHESTER 1824
The University of Manchester

... but, where to execute the code now?



MANCHESTER 1824
The University of Manchester

... but, where to execute the code now?



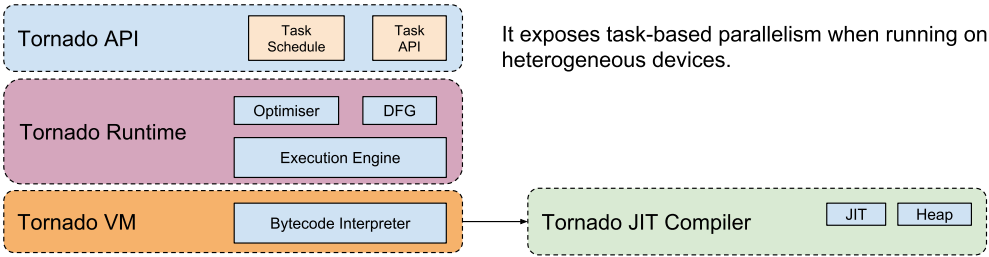
It depends, but can we make it automatically?

MANCHESTER 1824
The University of Manchester

2. TornadoVM: a heterogeneous Virtual Machine that automatically explores the best performing device at runtime

MANCHESTER 1824
The University of Manchester

TornadoVM Overview



It exposes task-based parallelism when running on heterogeneous devices.

Modular System currently using:

- OpenJDK/Graal
- OpenCL 1.2

It currently runs on multi-core CPUs, GPUs and FPGAs

MANCHESTER 1824
The University of Manc[illegible]

Tornado API: Example of enabling task-based parallelism

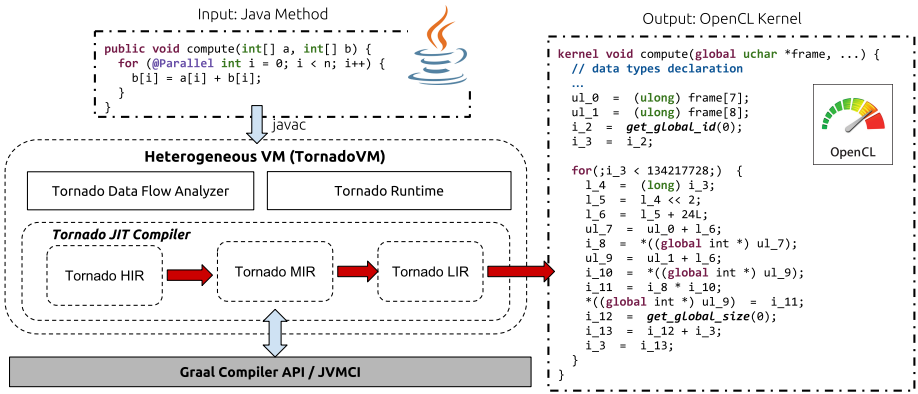
```
public static void add(int[] a, int[] b, int[] c) {
    for (@Parallel int i = 0; i < c.length; i++) {
        c[i] = a[i] + b[i];
    }
}

// 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

3. Tornado Runtime & JIT Compiler

MANCHESTER 1824
The University of Manchester

Tornado JIT Compiler



MANCHESTER 1824
The University of Manchester

More Info about the JIT Compiler: Manlang Paper 2018

Exploiting High-Performance Heterogeneous Hardware for Java Programs using Graal

James Clarkson
The University of Manchester
Manchester, UK
james.clarkson@manchester.ac.uk

Juan Fumero
The University of Manchester
Manchester, UK
juan.fumero@manchester.ac.uk

Michail Papadimitriou
The University of Manchester
Manchester, UK
michail.papadimitriou@manchester.ac.uk

Foivos S. Zakkak
The University of Manchester
Manchester, UK
foivos.zakkak@manchester.ac.uk

Maria Xekalaki
The University of Manchester
Manchester, UK
maria.xekalaki@manchester.ac.uk

Christos Kotselidis
The University of Manchester
Manchester, UK
christos.kotselidis@manchester.ac.uk

Mikel Luján
The University of Manchester
Manchester, UK
mikel.lujan@manchester.ac.uk

ABSTRACT

The proliferation of heterogeneous hardware in recent years means that every system we program is likely to include a mix of compute elements; each with different characteristics. By utilizing these available hardware resources, developers can improve the performance and energy efficiency of their applications. However, existing tools for heterogeneous programming neglect developers who do not have the time or inclination to switch programming languages or learn the intricacies of a specific piece of hardware.

This paper presents a framework that enables Java applications

KEYWORDS

Heterogeneous Hardware, Java, Virtual Machine, Graal, openCL

ACM Reference Format:

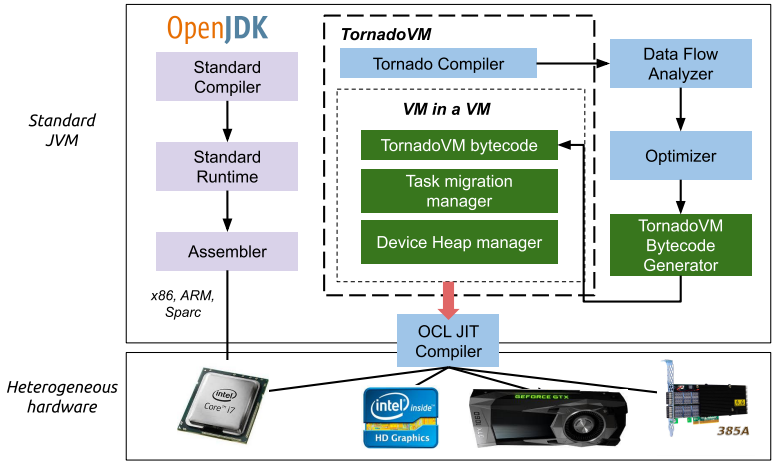
James Clarkson, Juan Fumero, Michail Papadimitriou, Foivos S. Zakkak, Maria Xekalaki, Christos Kotselidis, and Mikel Luján. 2018. Exploiting High-Performance Heterogeneous Hardware for Java Programs using Graal. In *15th International Conference on Managed Languages & Runtimes (ManLang'18), September 12–14, 2018, Linz, Austria.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3237009.3237016

MANCHESTER 1824
The University of Manchester

4. TornadoVM

MANCHESTER 1824
The University of Manchester

TornadoVM Overview



MANCHESTER 1824
The University of Manchester

TornadoVM

Input Java code

```
public class Compute {
  public void map(float[] in, float[] out) {
    for (@Parallel int i = 0; i < n; i++) {
      out[i] = in[i] * in[i];
  }}
  public void reduce(float[]in,@Reduce float[]out) {
    for (@Parallel int i = 0; i < n; i++) {
      out[0] += in[i];
  }}
  public static void compute(float[] in, float[] out,
                             float[] tmp, Compute obj){
    TaskSchedule t0 = new TaskSchedule("s0")
      .task("t0", obj::map, in, tmp)
      .task("t1", obj::reduce, tmp, out)
      .streamOut(out)
      .execute();
  }}
```

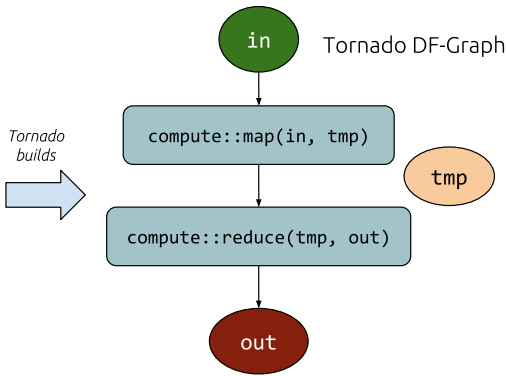
MANCHESTER 1824
The University of Manchester

TornadoVM

Input Java code

```
public class Compute {
  public void map(float[] in, float[] out) {
    for (@Parallel int i = 0; i < n; i++) {
      out[i] = in[i] * in[i];
  }}
  public void reduce(float[]in,@Reduce float[]out) {
    for (@Parallel int i = 0; i < n; i++) {
      out[0] += in[i];
  }}
  public static void compute(float[] in, float[] out,
                             float[] tmp, Compute obj){
    TaskSchedule t0 = new TaskSchedule("s0")
      .task("t0", obj::map, in, tmp)
      .task("t1", obj::reduce, tmp, out)
      .streamOut(out)
      .execute();
  }}
```

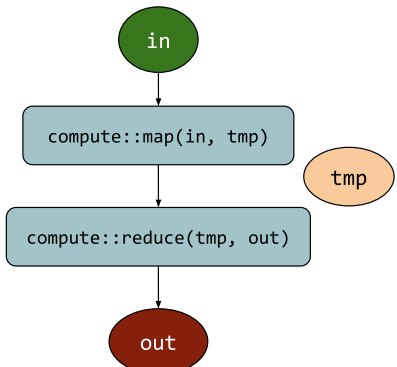
Tornado builds



MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph

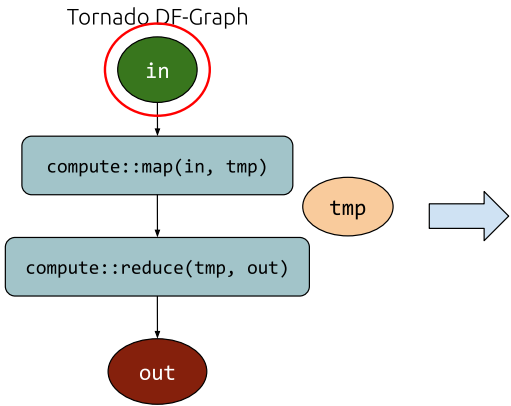


TornadoVM Bytecodes

```
BEGIN   <0>          // Starts a new context

END     <0>          // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM



TornadoVM Bytecodes

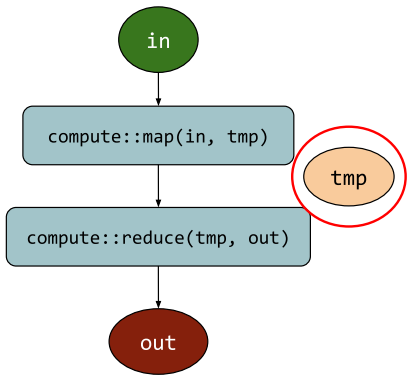
```
BEGIN    <0>            // Starts a new context
COPY_IN <0, bi1, in>    // Allocates and copies <in>
```

```
END      <0>            // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph



TornadoVM Bytecodes

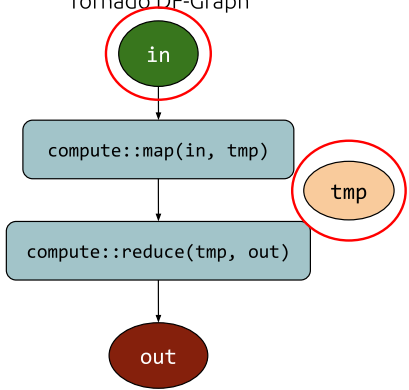
```
BEGIN    <0>             // Starts a new context
COPY_IN  <0, bi1, in>    // Allocates and copies <in>
ALLOC    <0, bi2, tmp>   // Allocates <tmp> on device
```

```
END      <0>             // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph



TornadoVM Bytecodes

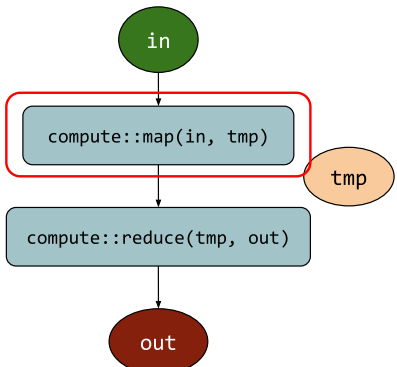
```
BEGIN    <0>             // Starts a new context
COPY_IN  <0, bi1, in>    // Allocates and copies <in>
ALLOC    <0, bi2, tmp>   // Allocates <tmp> on device
ADD_DEP  <0, bi1, bi2>   // Waits for copy and alloc
```

```
END      <0>             // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph



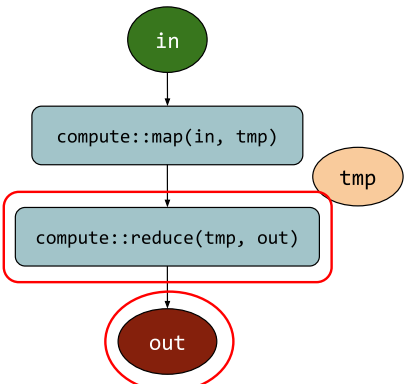
TornadoVM Bytecodes

```
BEGIN    <0>              // Starts a new context
COPY_IN  <0, bi1, in>     // Allocates and copies <in>
ALLOC    <0, bi2, tmp>    // Allocates <tmp> on device
ADD_DEP  <0, bi1, bi2>    // Waits for copy and alloc
LAUNCH   <0, bi3, @map, in, tmp> // Runs map


END      <0>              // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph



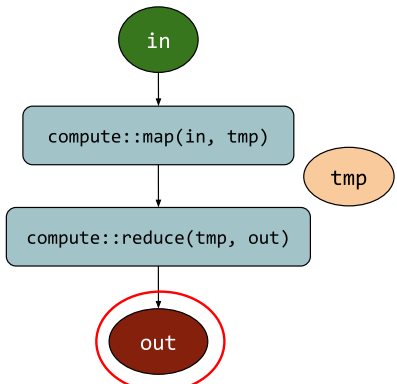
TornadoVM Bytecodes

```
BEGIN    <0>                  // Starts a new context
COPY_IN  <0, bi1, in>         // Allocates and copies <in>
ALLOC    <0, bi2, tmp>        // Allocates <tmp> on device
ADD_DEP  <0, bi1, bi2>        // Waits for copy and alloc
LAUNCH   <0, bi3, @map, in, tmp> // Runs map
ALLOC    <0, bi4, out>        // Allocates <out> on device
ADD_DEP  <0, bi3, bi4>        // Waits for alloc and map
LAUNCH   <0, bi5, @reduce, tmp, out> // Runs reduce


END      <0>                  // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM

Tornado DF-Graph



TornadoVM Bytecodes

```
BEGIN    <0>              // Starts a new context
COPY_IN <0, bi1, in>      // Allocates and copies <in>
ALLOC    <0, bi2, tmp>    // Allocates <tmp> on device
ADD_DEP <0, bi1, bi2>     // Waits for copy and alloc
LAUNCH   <0, bi3, @map, in, tmp> // Runs map
ALLOC    <0, bi4, out>    // Allocates <out> on device
ADD_DEP <0, bi3, bi4>     // Waits for alloc and map
LAUNCH   <0, bi5, @reduce, tmp, out> // Runs reduce
ADD_DEP <0, bi5>          // Wait for reduce
COPY_OUT_BLOCK <0, bi6, out> // Copies <out> back
END      <0>              // Ends context
```

MANCHESTER 1824
The University of Manchester

TornadoVM is a Composable VM

E.g., Processing Arrays of 16GB in a GPU of 1GB

Input Java user-code

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

```
// 16GB data
double[] a = new double[2000000000];
double[] b = new double[2000000000];
double[] c = new double[2000000000];
TaskSchedule ts = new TaskSchedule("s0");

ts.batch("300MB")
  .task(Compute::add, a, b, c)
  .streamOut(c)
  .execute();
```

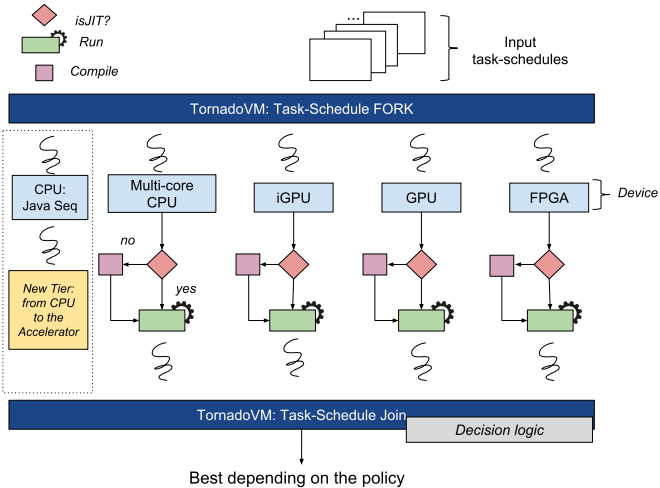
Tornado VM

```
vm: BEGIN
vm: COPY_IN bytes=300000000, offset=0
vm: COPY_IN bytes=300000000, offset=0
vm: ALLOCATE bytes=300000000
vm: LAUNCH s0.t0 threads=37500000, offset=0
vm: STREAM_OUT bytes=300000000, offset=0
vm: COPY_IN bytes=300000000, offset=300000000
vm: COPY_IN bytes=300000000, offset=300000000
vm: ALLOCATE bytes=300000000
vm: LAUNCH task s0.t0 threads=37500000, offset=300000000
vm: STREAM_OUT bytes=300000000, offset=300000000
vm: …
vm: …
vm: STREAM_OUT_BLOCKING bytes=100000000, offset=1500000000
vm: END
```

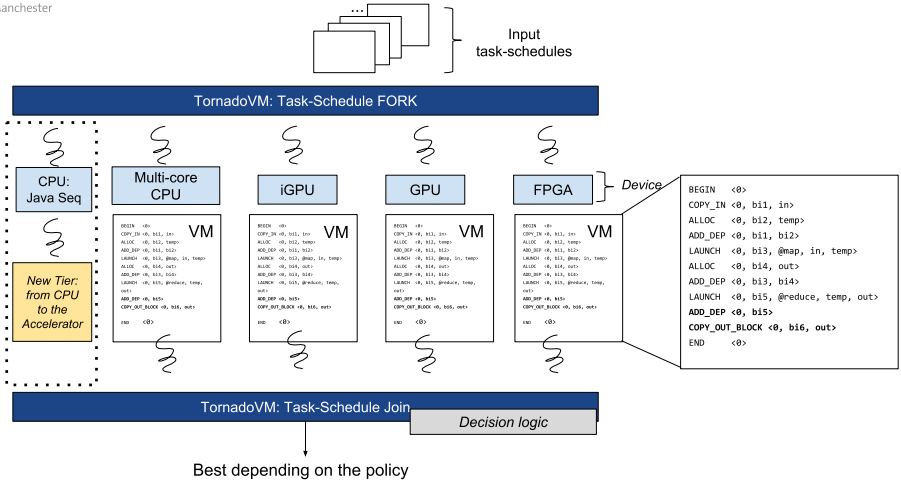
Easy to orchestrate heterogeneous execution

MANCHESTER 1824
The University of Manchester

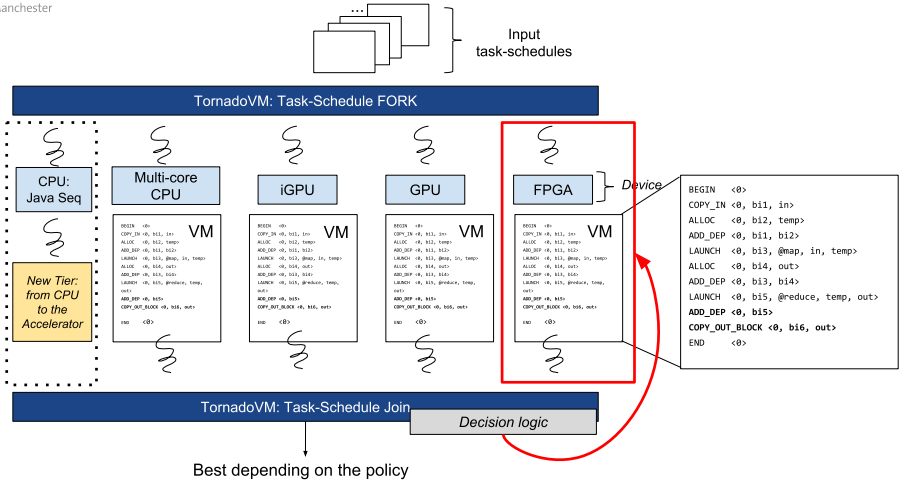
Dynamic Reconfiguration from Java to the "best" Accelerator



Dynamic Reconfiguration from Java to the "best" Accelerator



Dynamic Reconfiguration from Java to the "best" Accelerator



MANCHESTER 1824
The University of Manchester

How is the Decision Performed? → Three Policies

- End-to-end: including JIT compilation time
- Peak performance: without including JIT compilation
- Latency: does not wait for all threads

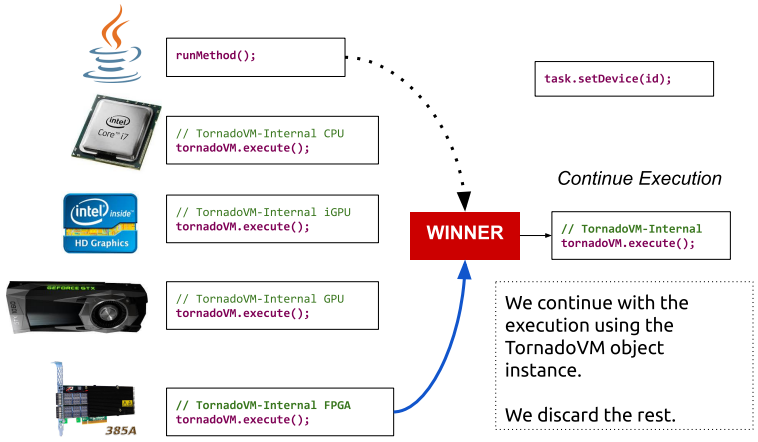
```
// END TO END PERFORMANCE
ts.task(Compute::add, a, b, c)
  .streamOut(c)
  .execute(Profiler.END2END);
```

```
// PEAK PERFORMANCE
ts.task(Compute::add, a, b, c)
  .streamOut(c)
  .execute(Profiler.PERFORMANCE);
```

```
// Latency - winner
ts.task(Compute::add, a, b, c)
  .streamOut(c)
  .execute(Profiler.LATENCY);
```

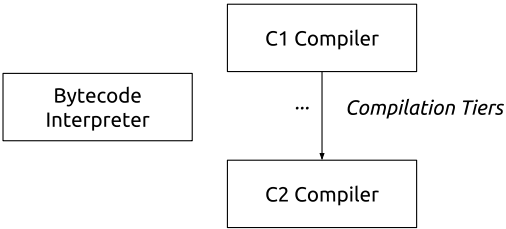
MANCHESTER 1824
The University of Manchester

How to Perform Runtime Task Migration?

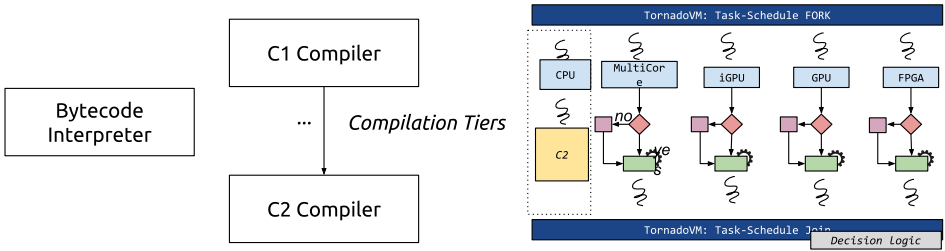


MANCHESTER 1824
The University of Manchester

New Compilation Tier for Heterogeneous Systems



New Compilation Tier for Heterogeneous Systems



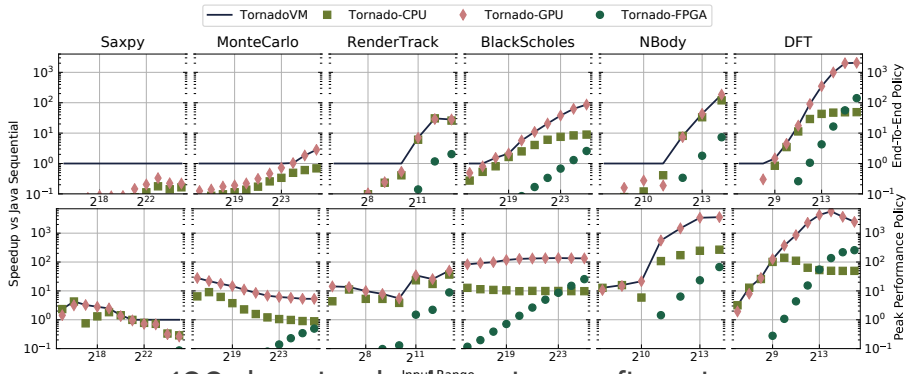
E.g., From C2 -> Multi-core -> GPU

MANCHESTER 1824
The University of Manchester

5. Evaluation

MANCHESTER 1824
The University of Manchester

TornadoVM - Reconfiguration in Action

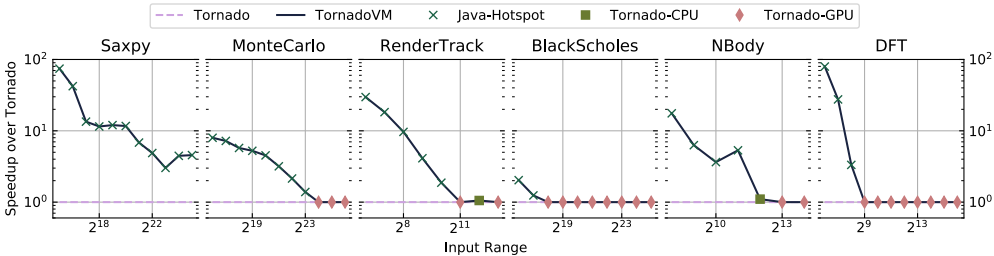


100x by using the dynamic reconfiguration

Up to 4000x on NVIDIA GTX 1060 GPU compared to Java sequential

MANCHESTER 1824
The University of Manchester

TornadoVM - Reconfiguration in Action II



Up to 100x compared to the best static device

Average of 7.7x compared to the best static device

MANCHESTER 1824
The University of Manchester

7. Conclusions

MANCHESTER 1824
The University of Manchester

Start Accelerating your Applications for Free!

https://github.com/beehive-lab/TornadoVM

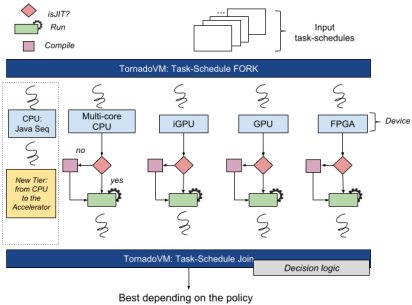
```
$ docker pull beehivelab/tornado-gpu

# And RUN !
$ ./run.sh javac.py
example/MatrixMultiplication.java
$ ./run.sh tornado example/MatrixMultiplication
```

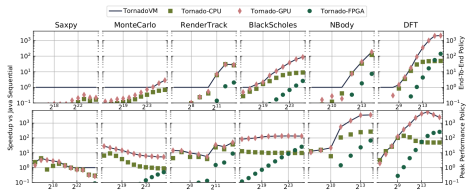
https://github.com/beehive-lab/docker-tornado

If you are interested, we can also show you demos on GPUs and FPGAs!

MANCHESTER 1824
The University of Manchester

Takeaways





- > 4000x compared to vanilla Java!
- 7.7x in average compared to the best parallel code

TornadoVM is open-source:

https://github.com/beehive-lab/TornadoVM

E²Data

https://e2data.eu

SCAN ME

MANCHESTER 1824
The University of Manchester

Thank you very much for your attention

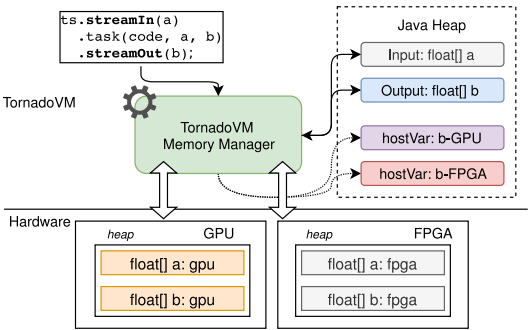
This work is partially supported by the EU Horizon 2020 E2Data 780245.

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

MANCHESTER 1824
The University of Manchester

Back-UP slides

Memory Management



- Host Variables: read-only in the JVM heap, R/W or W then we perform a new copy.
- Device Variables: a new copy unless OpenCL zero copy, e.g., iGPU

MANCHESTER 1824
The University of Manchester

Breakdown Analysis

Benchmark	Compilation Time			
	CPU	FPGA	Load	GPU
Saxpy	7.44	53 mins	1314	99.64
MonteCarlo	85.85	54 mins	1368	87.60
RenderTrack	111.10	51 mins	1380	105.03
BlackScholes	178.61	114 mins	1420	243.02
NBody	144.68	51 mins	1387	151.25
DFT	83.80	68 mins	1398	161.96

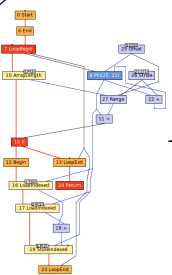
Host to Device			Execution			Device To Host			Rest		
CPU	FPGA	GPU	CPU	FPGA	GPU	CPU	FPGA	GPU	CPU	FPGA	GPU
19.78	19.85	59.01	57.04	248.64	2.72	10.06	13.54	20.57	1.15	20.14	1.50
1 ns	1 ns	1 ns	240.88	456.96	2.75	21.61	59.71	41.14	0.70	0.57	0.70
18.59	40.10	58.35	24.50	242.15	1.96	3.86	6.84	7.70	0.69	3.03	2.61
16.84	10.30	30.97	1036.12	400.31	4.43	21.07	20.45	41.14	1.09	2.36	0.93
0.05	0.04	0.08	101.81	441.48	7.47	0.04	0.10	0.08	1.31	1.21	1.04
0.09	0.10	0.16	31674.15	4424.13	460.68	0.05	0.10	0.08	1.03	1.85	1.24

JIT Specialisation - E.g., Granularity for Data Parallelism

Input Java code

```
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];
   }
}
```

Compiler IR



GPU Specialization

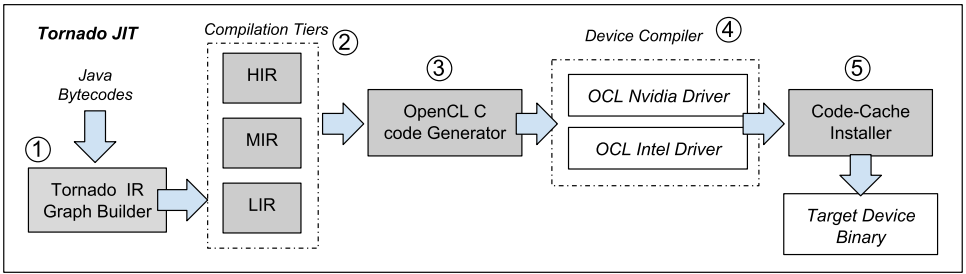

```
int idx = get_global_id(0);
int size = get_global_size(0);
for (int i = idx; i < c.length; i += size) {
  c[i] = a[i] + b[i];
}
```

CPU Specialization

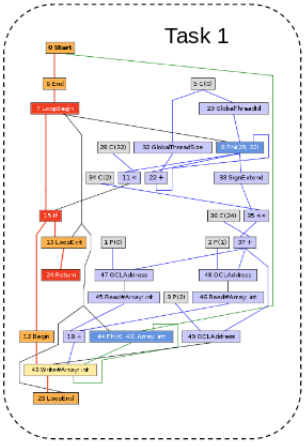

```
int id = get_global_id(0);
int size = get_global_size(0);
int block_size = (size + inputSize - 1) / size;
int start = id * block_size;
int end = min(start + block_size, inputSize);
for (int i = start; i < end; i++) {
  c[i] = a[i] + b[i];
}
```

MANCHESTER 1824
The University of Manchester

TornadoVM JIT Compiler

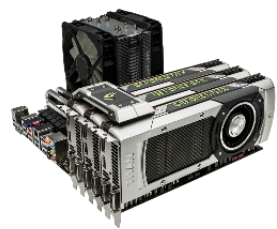


TornadoVM JIT Compiler

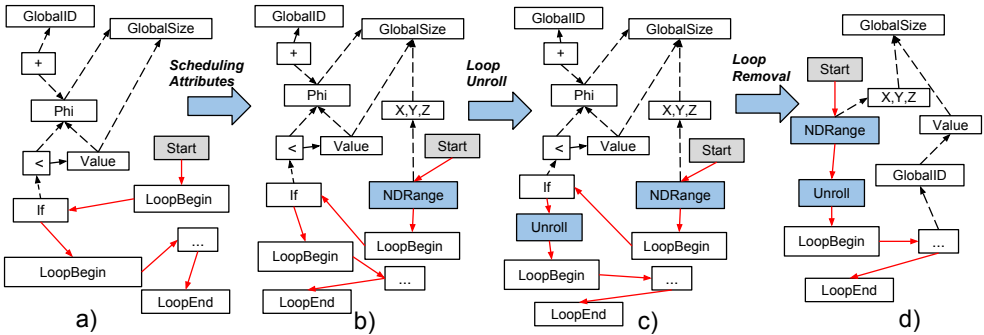



```
__kernel void add(...) {
  // BLOCK 0
  ul_0  =  (ulong) _frame[6];
  ul_1  =  (ulong) _frame[7];
  ul_2  =  (ulong) _frame[8];
  i_3  =  get_global_id(0);
  // BLOCK 1 MERGES [0 2 ]
  i_4  =  i_3;
  for(;i_4 < 32;)  {
    // BLOCK 2
    l_5  =  (long) i_4;
    l_6  =  l_5 << 2;
    l_7  =  l_6 + 24L;
    ul_8  =  ul_0 + l_7;
    i_9  =  *((__global int *) ul_8);
    ul_10  =  ul_1 + l_7;
    i_11  =  *((__global int *) ul_10);
    ul_12  =  ul_2 + l_7;
    i_13  =  i_9 + i_11;
    *((__global int *) ul_12)  =  i_13;
    i_14  =  get_global_size(0);
    i_15  =  i_14 + i_4;
    i_4  =  i_15;
  }
  // BLOCK 3
  return;
}
```

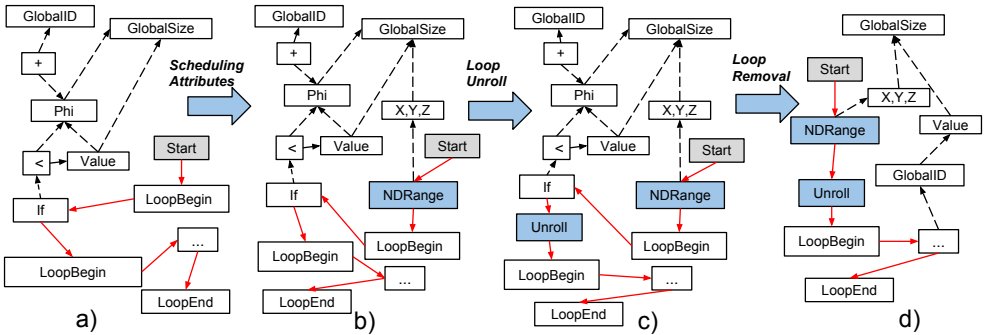
OpenCL



FPGA Specializations - IR level



FPGA Specializations - IR level



Java DFT on FPGA: from 5x to 270x compared to Java Hotspot!

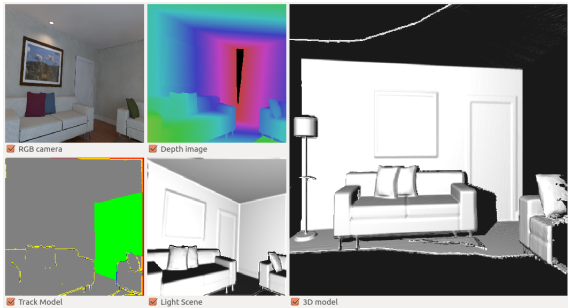
MANCHESTER 1824
The University of Manchester

Evaluation Setup

Running CentOS 7.4 with Linux Kernel 3.10.0-693.
JDK 1.8.0.131 with JVMCI Support.

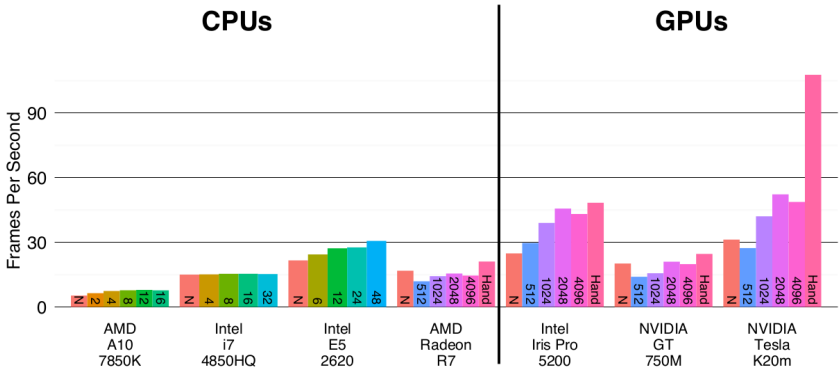
MANCHESTER 1824
The University of Manchester

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

What did we get with Tornado?



Running on NVIDIA Tesla, up to 150 fps