The University of Manchester

TORNADO VM

A Brief Retrospective

James Clarkson

Neo4j (was University of Manchester)

james.clarkson@neo4j.com

Aims

- How is Tornado different to *x, y,* and *z?*
- Is the motivation behind Tornado still applicable today?
- What is the performance story?

The University of Manchester

How is Tornado different to *x*, *y*, and *z*?

MANCHESTER 1824
The University of Manchester

TORNADOVM

What is TornadoVM?

A virtual-machine in a virtual-machine that enables seamless execution across a range of heterogeneous architectures.

Now an open-source project at UoM.

tornadovm.org

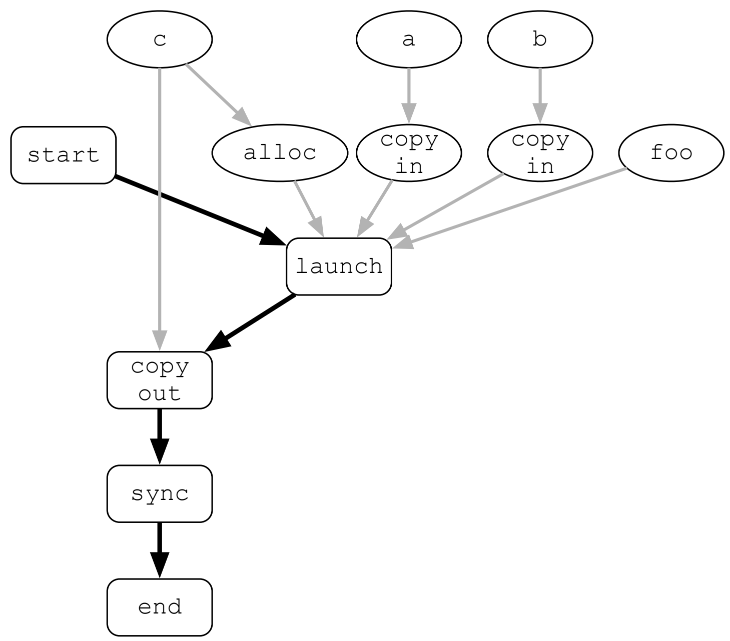
github.com/beehive-lab/tornadovm

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

// execute add on an accelerator
 new TaskSchedule("s0")
    .task("t0", SimpleMath::vectorAdd, a, b, c)
    .execute();
```

```
new TaskSchedule("s0")
    .task("t0", Example::foo, a, b, c)
    .execute();
```

MANCHESTER 1824
The University of Manchester

```
new TaskSchedule("s0")
    .task("t0", Example::foo, a, b, c)
    .execute();
```

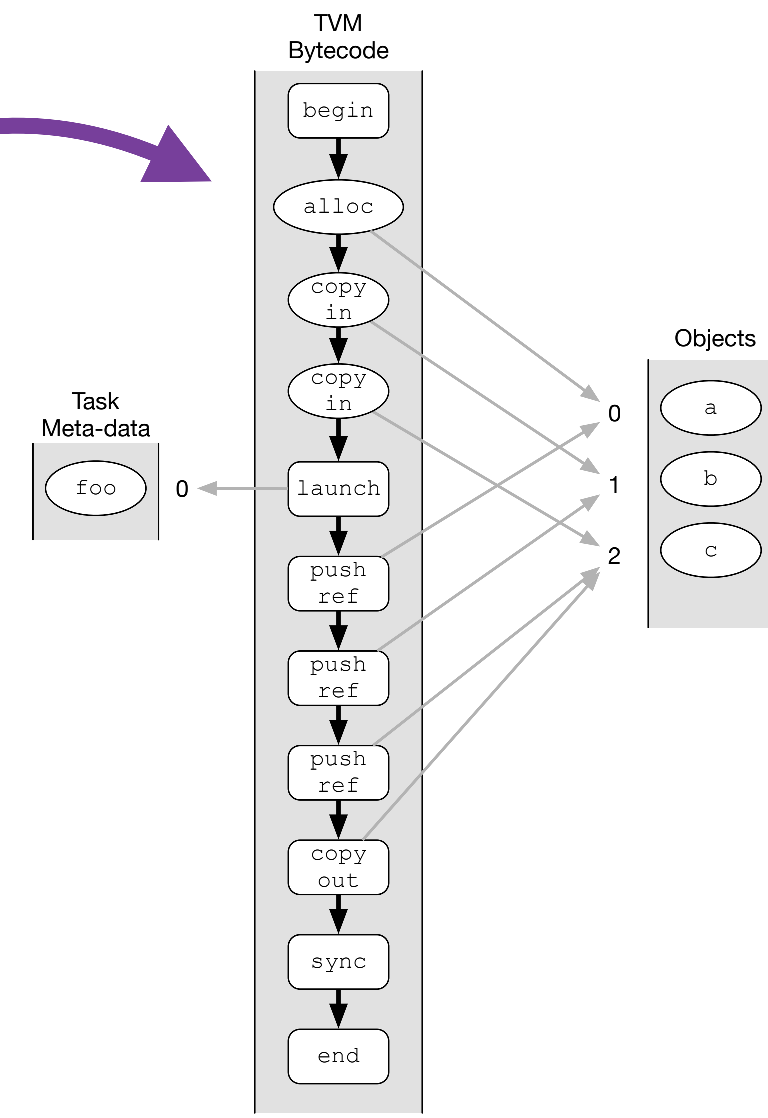


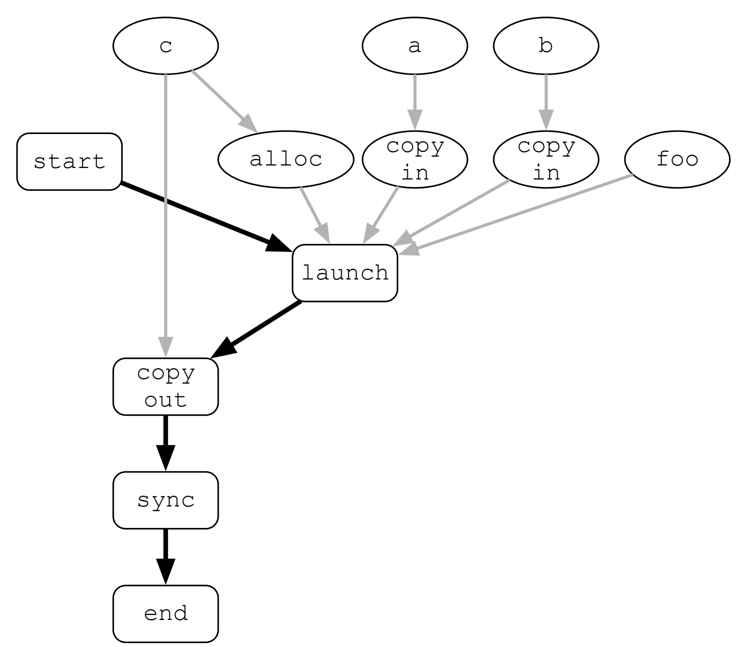
Generates a graph-IR that allows us to optimise both the data and the code that is to be executed.

MANCHESTER 1824
The University of Manchester

```
new TaskSchedule("s0")
    .task("t0", Example::foo, a, b, c)
    .execute();
```

Graph-IR is compiled into bytecode and run on A Virtual Machine.

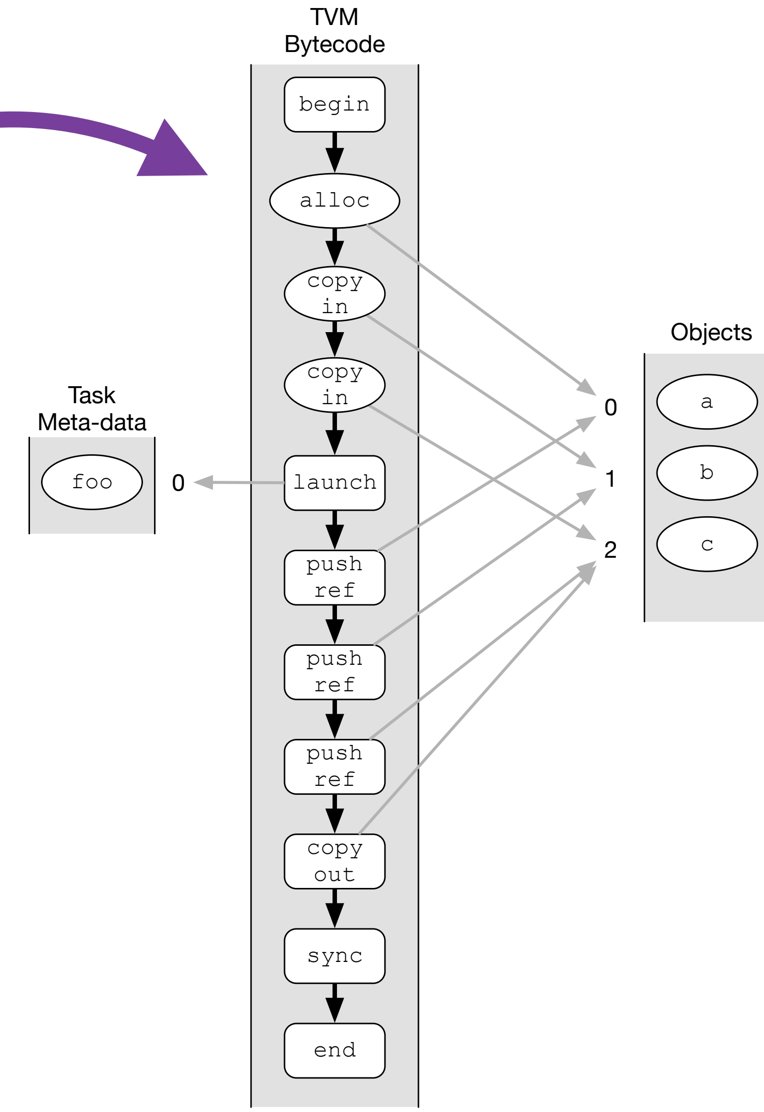


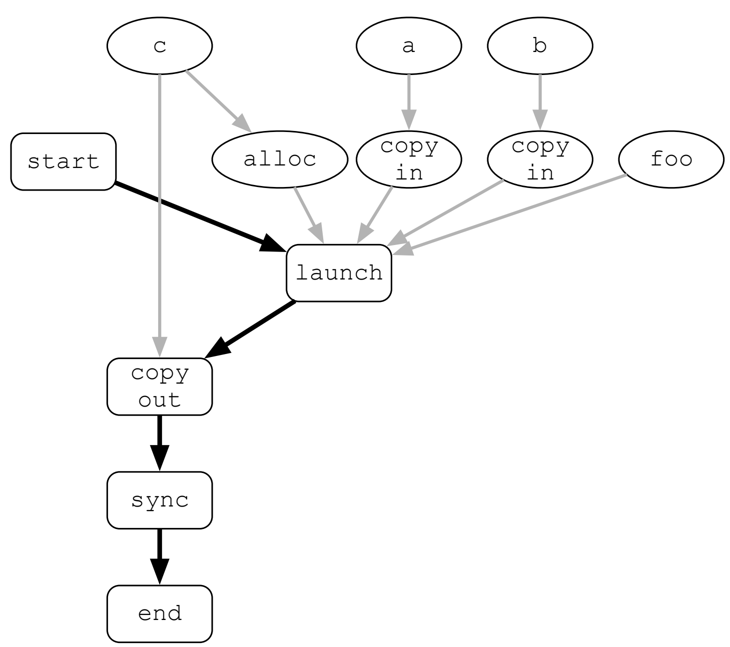


Generates a graph-IR that allows us to optimise both the data and the code that is to be executed.

```
new TaskSchedule("s0")
    .task("t0", Example::foo, a, b, c)
    .execute();
```

Graph-IR is compiled into
bytecode and run on
A Virtual Machine.





Generates a graph-IR that
allows us to optimise both the
data and the code that
is to be executed.

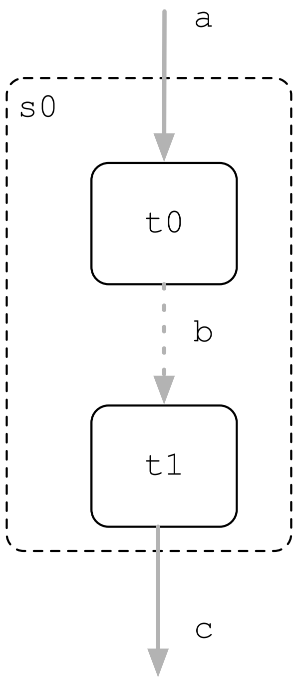
Each byte code performs an operation
on a disparate device.
e.g. copy memory to a GPGPU

MANCHESTER 1824
The University of Manchester

```java
// define a two stage pipeline
TaskSchedule schedule = new TaskSchedule("s0")
  .volatile(a)
  .task("t0", SimpleMath::vectorMultiply, a, b, c)
  .task("t1", SimpleMath::vectorAdd, c, b, d)
  .sync(d);


// query the number of devices attached to the system
TornadoDriver driver = getTornadoRuntime().getDriver(0);
int maxDevice = driver.getDeviceCount();
final Random rand = new Random(7);
final int[] devices = new int[2];

// invoke the pipeline multiple times
for (int i = 0; i < num_iterations; i++) {

  // randomly select a device for each task
  devices[0] = rand.nextInt(maxDevice);
  devices[1] = rand.nextInt(maxDevice);

  // update the task meta-data
  schedule.getTask("t0").mapTo(driver.getDevice(devices[0]));
  schedule.getTask("t1").mapTo(driver.getDevice(devices[1]));

  // execute the pipeline
  schedule.execute();
}
```

The University of Manchester

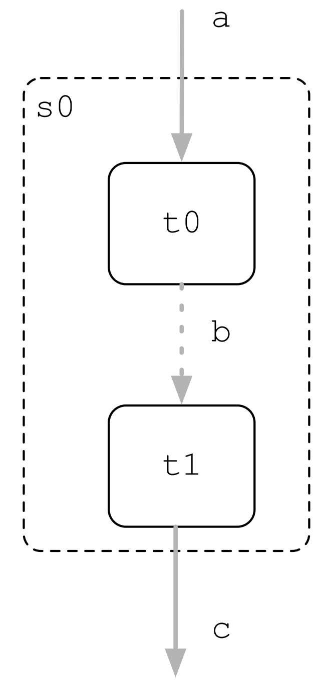
```
// define a two stage pipeline
TaskSchedule schedule = new TaskSchedule("s0")
  .volatile(a)
  .task("t0", SimpleMath::vectorMultiply, a, b, c)
  .task("t1", SimpleMath::vectorAdd, c, b, d)
  .sync(d);


// query the number of devices attached to the system
TornadoDriver driver = getTornadoRuntime().getDriver(0);
int maxDevice = driver.getDeviceCount();
final Random rand = new Random(7);
final int[] devices = new int[2];

// invoke the pipeline multiple times
for (int i = 0; i < num_iterations; i++) {

  // randomly select a device for each task
  devices[0] = rand.nextInt(maxDevice);
  devices[1] = rand.nextInt(maxDevice);

  // update the task meta-data
  schedule.getTask("t0").mapTo(driver.getDevice(devices[0]));
  schedule.getTask("t1").mapTo(driver.getDevice(devices[1]));

  // execute the pipeline
  schedule.execute();
}
```



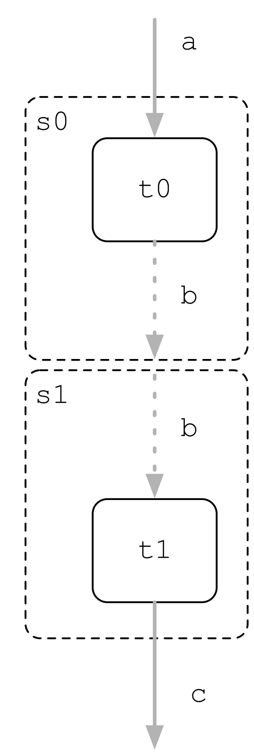
Two tasks running back-to-back on the same device.

MANCHESTER 1824
The University of Manchester

```java
// define a two stage pipeline
TaskSchedule schedule = new TaskSchedule("s0")
  .volatile(a)
  .task("t0", SimpleMath::vectorMultiply, a, b, c)
  .task("t1", SimpleMath::vectorAdd, c, b, d)
  .sync(d);


// query the number of devices attached to the system
TornadoDriver driver = getTornadoRuntime().getDriver(0);
int maxDevice = driver.getDeviceCount();
final Random rand = new Random(7);
final int[] devices = new int[2];

// invoke the pipeline multiple times
for (int i = 0; i < num_iterations; i++) {

  // randomly select a device for each task
  devices[0] = rand.nextInt(maxDevice);
  devices[1] = rand.nextInt(maxDevice);

  // update the task meta-data
  schedule.getTask("t0").mapTo(driver.getDevice(devices[0]));
  schedule.getTask("t1").mapTo(driver.getDevice(devices[1]));

  // execute the pipeline
  schedule.execute();
}
```



Two tasks running back-to-back on the same device.



Two tasks running back-to-back on the different devices.

The University of Manchester

Is the motivation behind Tornado still applicable today?

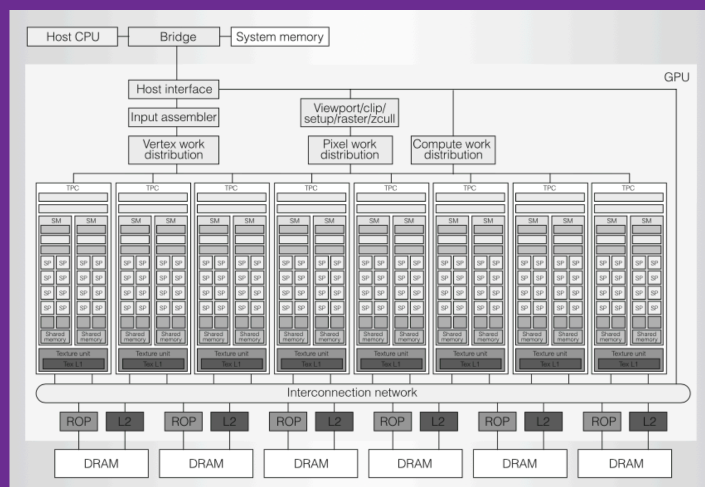
Rewind to 2010

Credit: Argonne National Laboratory [1]

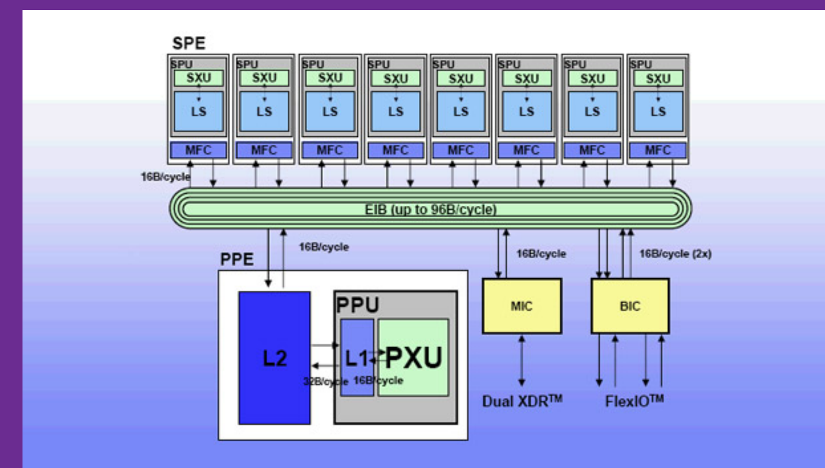
MANCHESTER 1824
The University of Manchester

Innovation circa 2010

NVIDIA Tesla [3]



IBM Cell [5]



IBM PowerEN [2]

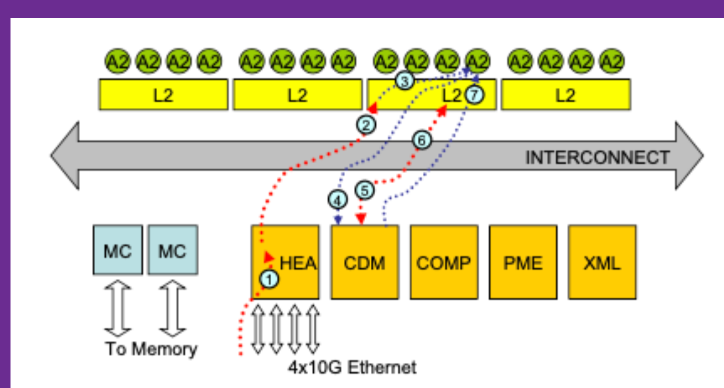
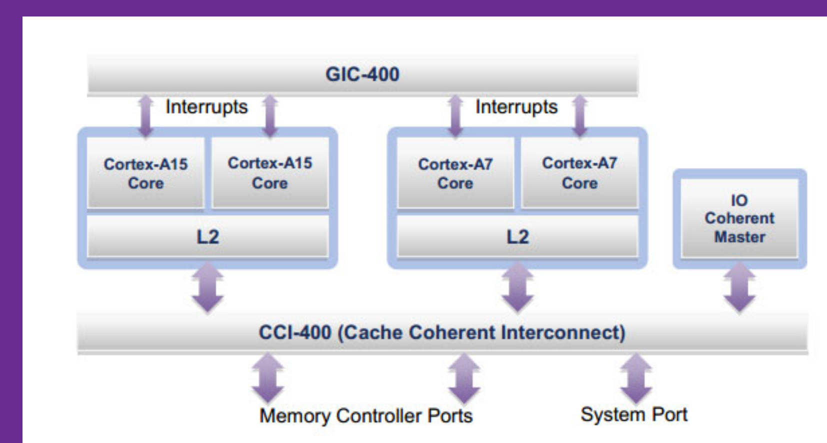


Figure 1: PowerEN Chip Overview

ARM big.LITTLE [6]

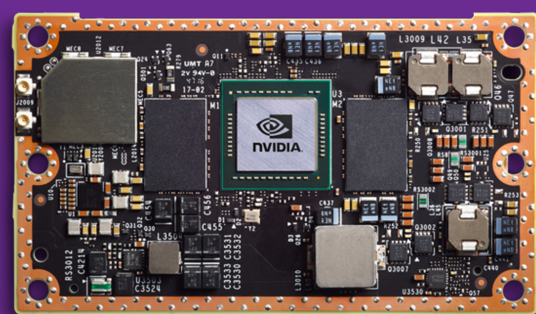


Alongside FPGAs, VLIW, and a range of multi-threaded architectures.

MANCHESTER 1824
The University of Manchester

Innovation circa 2021

NVIDIA Jetson [8]



HiSilicon Kirin [12]

Xilinx Ultra96 [10]

Intel Xe GPUs [9]

Heterogeneous technology has moved down into the mobile space.

What is the performance story?

Real-world Performance

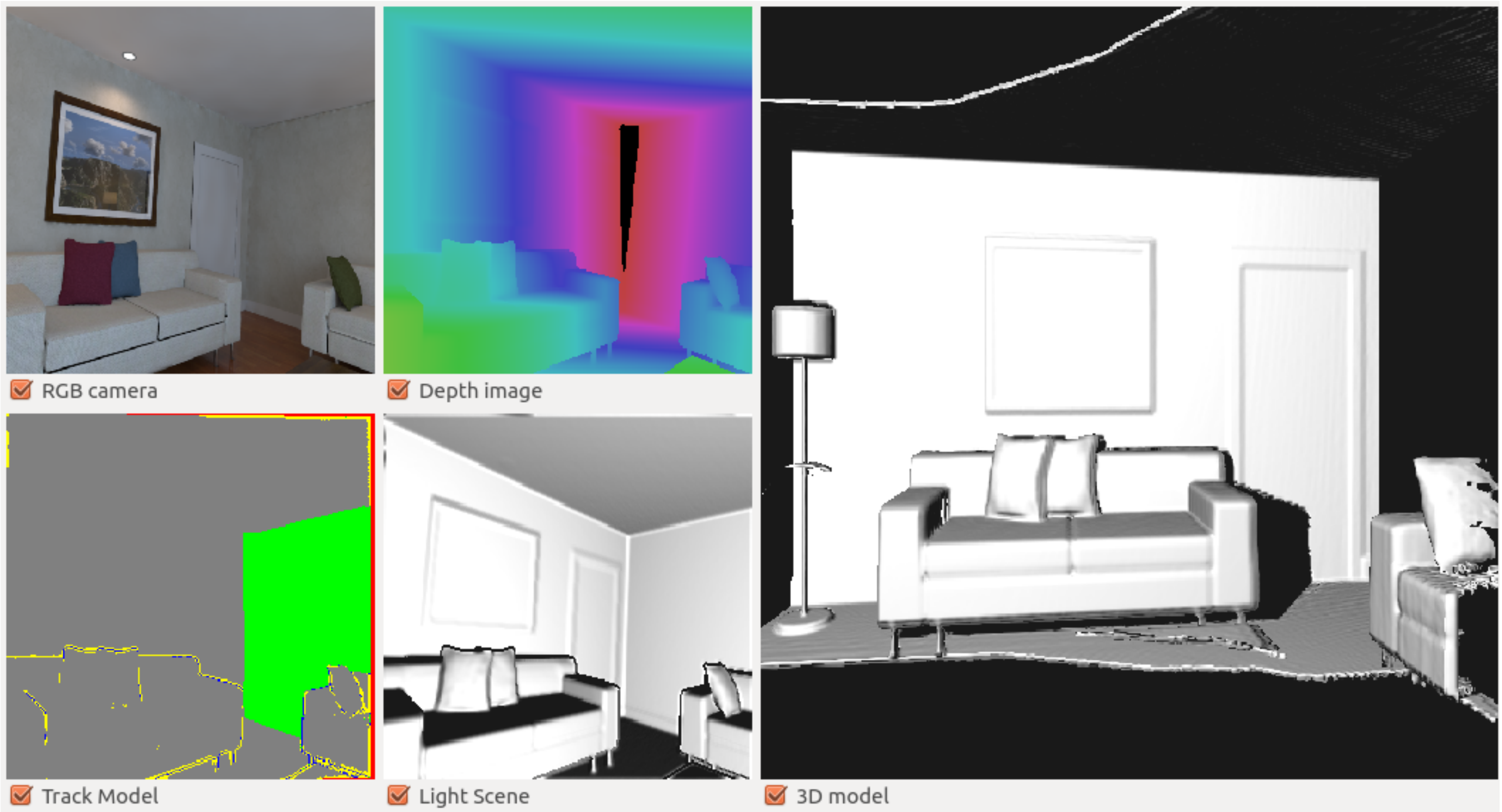
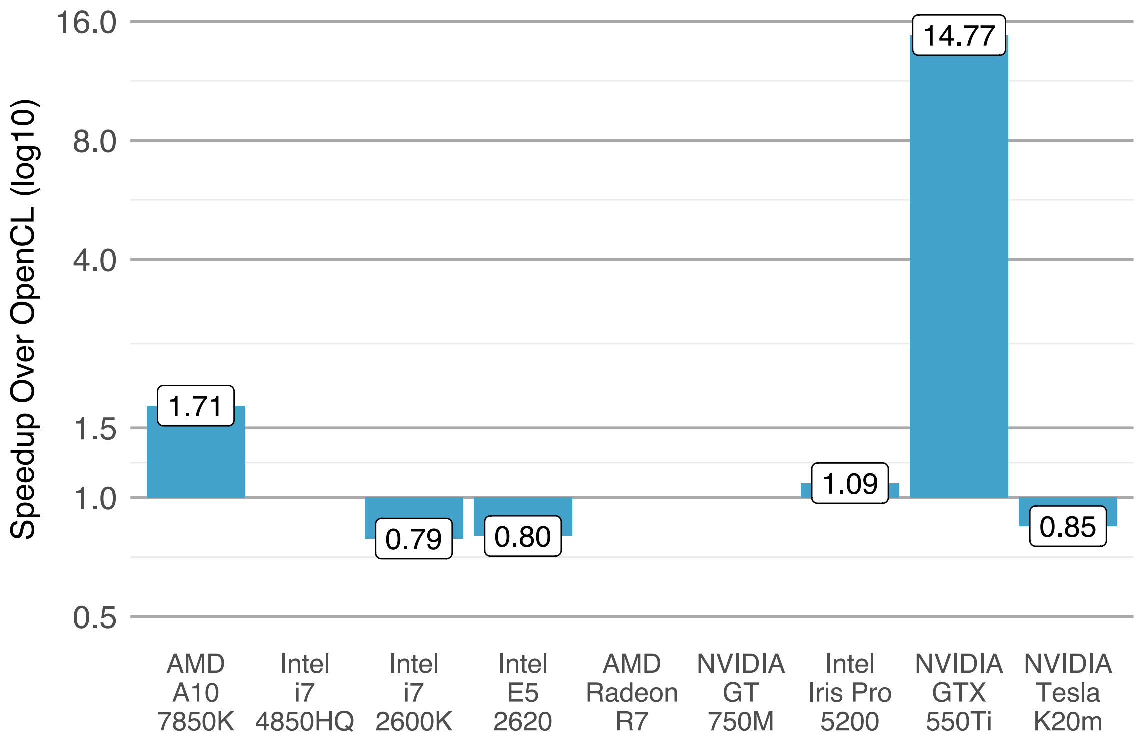
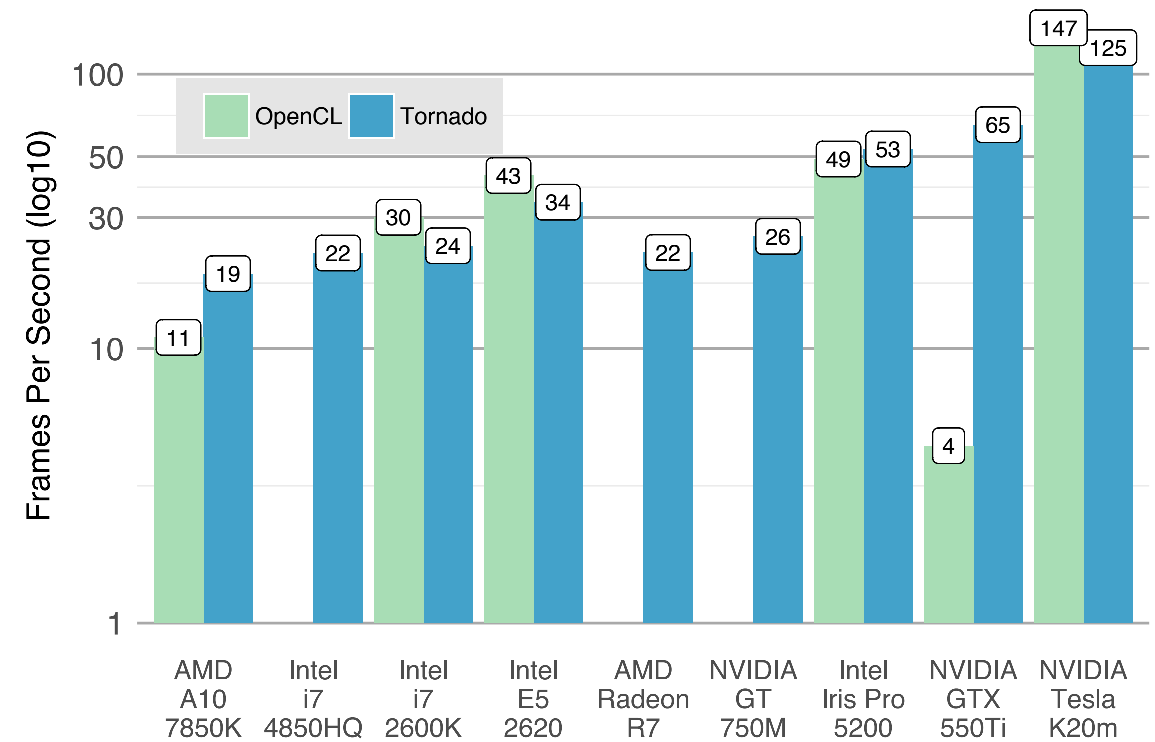
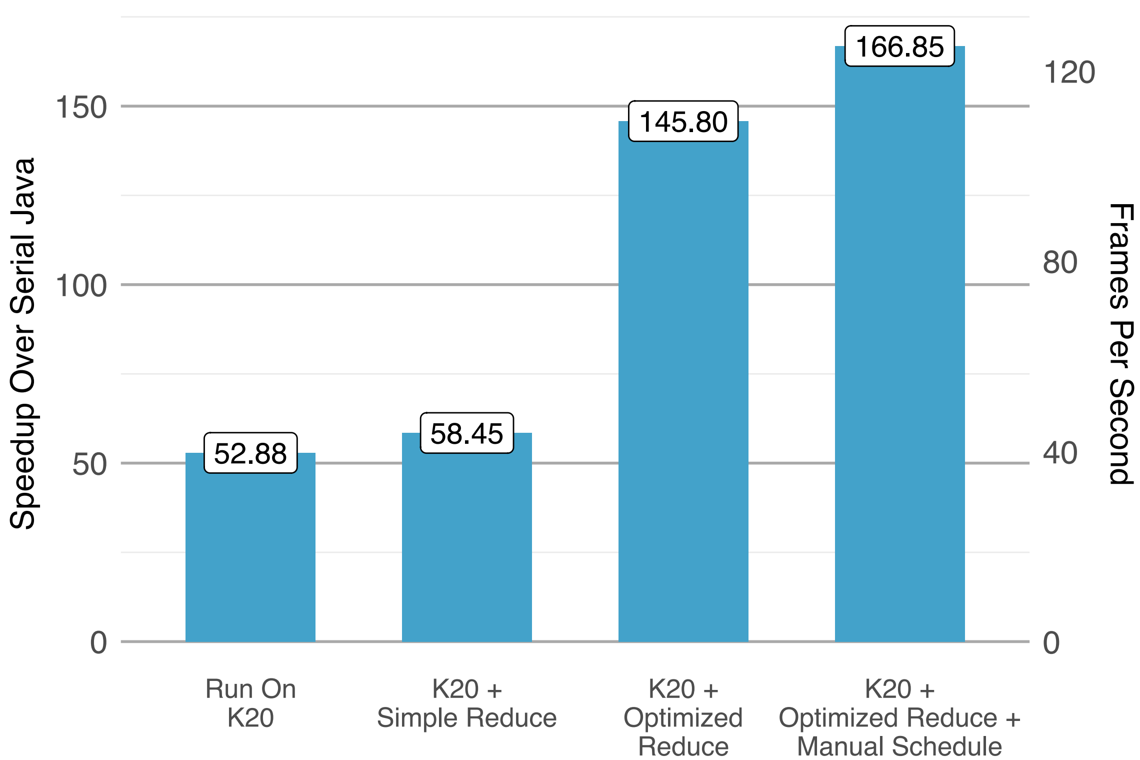


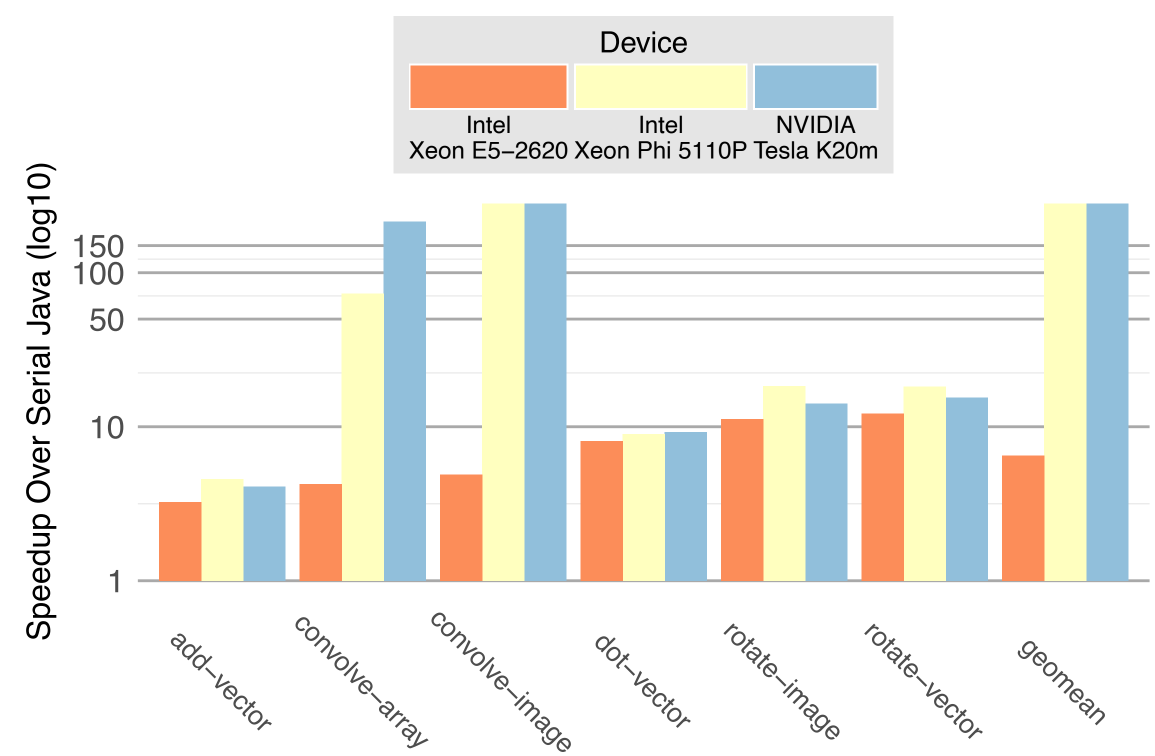
Image from SLAMbench [4]

Performance









Results taken from [1]

The University of Manchester

Summary

- There is lots of cool kit out there that is inaccessible!
- With Tornado we tried to look at the problem differently — from an ergonomic viewpoint; not performance.
- Able to show that there is a route to programming this hardware in more virtual-machine based languages that:
    - Retains their productivity features…
    - …without degrading performance (too much).
    - (Tried to) open hardware accelerators up to a new demographic.

The University of Manchester

Acknowledgements

PhD Supervisors

Mikel Lujan

Gavin Brown

Christos Kotselidis

Tornado VM Team

Juan Fumero

Thanos Stratikopoulos

Maria Xekalaki

Florin Balance

Michalis Papadimitriou

Funding

This work is supported by the AnyScale Apps and PAMELA projects funded by EPSRC EP/L000725/1 and EP/K008730/1.

MANCHESTER 1824
The University of Manchester

References

[1] Clarkson, James. 2019. Compiler and Runtime Support for Heterogeneous Programming. University of Manchester. Retrieved December 3, 2021 from https://www.research.manchester.ac.uk/portal/en/theses/compiler-and-runtime-support-for-heterogeneous-programming(3a83a155-390c-41d8-ab44-20cf963b4f94).html

[2] A. Krishna, T. Heil, N. Lindberg, F. Toussi, and S. VanderWiel. 2012. Hardware acceleration in the IBM PowerEN processor: architecture and performance. In *2012 21st international conference on parallel architectures and compilation techniques (PACT)*, 389–399.

[3] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. 2008. NVIDIA tesla: A unified graphics and computing architecture. *IEEE Micro* 28, 2 (March 2008), 39–55. DOI:https://doi.org/10/fcfgb5

[4] Luigi Nardi, Bruno Bodin, M. Zeeshan Zia, John Mawer, Andy Nisbet, Paul H. J. Kelly, Andrew J. Davison, Mikel Luján, Michael F. P. O'Boyle, Graham Riley, Nigel Topham, and Steve Furber. 2015. Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. In *IEEE intl. Conf. on robotics and automation (ICRA)*.

[5] 2012. IBM100 - The Cell Broadband Engine. Retrieved December 3, 2021 from http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/cellengine/

[6] 2013. *big.LITTLE Technology: The Future of Mobile*. ARM Ltd. Retrieved from https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/big-little-technology-the-future-of-mobile.pdf

[7] 2014. ARM: big.LITTLE processing. (2014). Retrieved from http://www.arm.com/products/processors/techn%20ologies/biglittleprocessing.php

[8] 2017. Jetson TX2 Module. *NVIDIA Developer*. Retrieved December 3, 2021 from https://developer.nvidia.com/embedded/jetson-tx2

[9] Xe-HPG Microarchitecture. *Intel*. Retrieved December 3, 2021 from https://www.intel.com/content/www/us/en/architecture-and-technology/visual-technology/arc-discrete-graphics/xe-hpg-microarchitecture.html

[10] Ultra96. *Linaro*. Retrieved December 3, 2021 from https://www.96boards.org/product/ultra96/

[11] Argonne's Leadership Computing Facility working to get more science per watt | Argonne National Laboratory. Retrieved December 3, 2021 from https://www.anl.gov/article/argonnes-leadership-computing-facility-working-to-get-more-science-per-watt

[12] Kirin 970 Chipset | HiSilicon Official Site. Retrieved from https://www.hisilicon.com/en/products/Kirin/Kirin-flagship-chips/Kirin-970

The University of Manchester

Questions?

james.clarkson@neo4j.com