MANCHESTER 1824
The University of Manchester

Tornado: Accelerating Java Applications Using GPGPUs

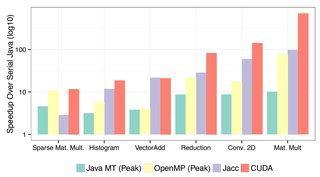
James Clarkson Christos Kotselidis Gavin Brown Mikel Luján
University of Manchester, UK
{first.last}@manchester.ac.uk

Introduction

- Tornado is a programming framework which allows developers to write complex heterogeneous applications in Java.
- It can be used to target OpenCL compatible devices.
- The dynamic compiler generates OpenCL C code directly from Java bytecode.
- Applications are programmed using a task-based programming model.
- Tasks can be assigned to execute on specific devices.
- It enables applications to schedule multiple-tasks across multiple-devices.

History - Java for Accelerators (JACC)

- Using Jacc [1] we demonstrated that it was possible to increase the performance of some Java benchmarks by an order of magnitude.



Programming Model

- Tornado uses a task-based programming model.
- Tasks can be executed serially or by using parallel threads.
- Parallelism is expressed using the `@Parallel` annotation within tasks.
- Tasks are assigned to execute on specific devices - either statically or dynamically.
- The dynamic compiler parallelises the code depending on the target device.
- Multiple tasks can be composed into a task-graph.
- Tasks are free to execute on different devices.

Creating A Task

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

Task addTask = createTask(VectorAddFloat.class, "add");
```

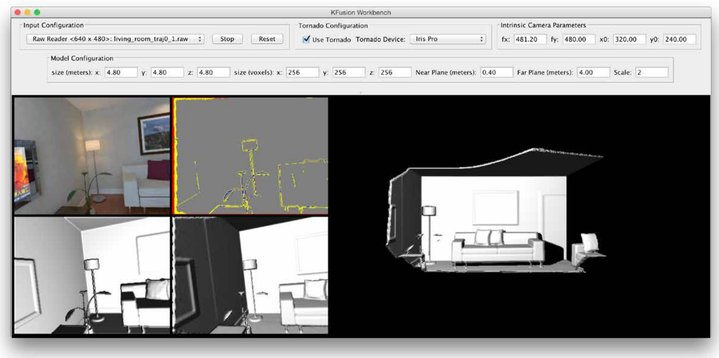
Executing A Task

```
// Invoke add with parameter a, b, c.
TaskInvocation add = addTask.invoke(a, b, c);

// Assign task to run on a GPGPU
add.mapTo(EXTERNAL_GPU);

// block until complete
add.execute();
```

Case Study: Kinect Fusion

- Our case study is an implementation of the Kinect Fusion application [4].
- KFusion is a computer vision application which reconstructs a 3D scene from a stream of RGB-D data.
- Ported KFusion into Java from the CUDA version of SLAMBENCH [3].
- The processing pipeline uses 12 unique tasks.
- To work in real-time the application needs to execute 20 tasks in 1/30th of a second.

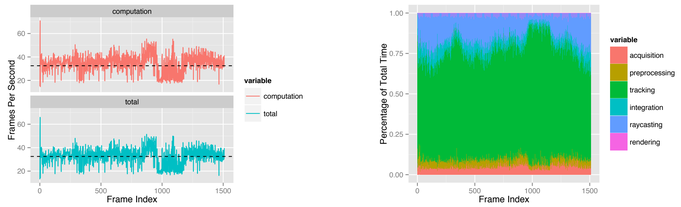
Kinect Fusion Java



Runtime and Language Support

- The Tornado runtime is able to automatically schedule data transfers between devices.
- By using task-graphs the runtime system can eliminate redundant data transfers.
- The runtime system is responsible for satisfying all inter-task data
- A library of common numerical functions has been developed.
- Support for short-vector types, e.g. float3, double4.

Kfusion Performance

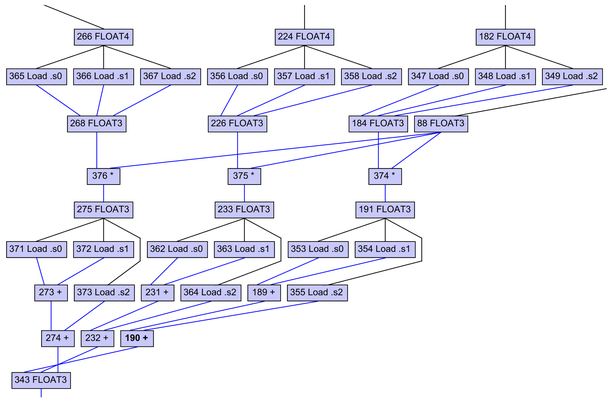


- Our serial Java implementation achieves between 0.2 and 1.2 FPS.
- Using Tornado we have achieved over 30 FPS.
- Frame rate is heavily on dependent performance of the tracking algorithm.
- Our Java implementation has been developed to produce bit-exact results when compared to SLAM BENCH implementations.

Kfusion Code

```
public void vertex2normal(@Write Image3 normals, @Read Image3 verticies) {
  for (@Parallel int y = 0; y < normals.Y(); y++)
    for (@Parallel int x = 0; x < normals.X(); x++) {

      final Float3 left = verticies.get(max(x - 1, 0), y);
      final Float3 right = verticies.get(min(x + 1, verticies.X() - 1), y);
      final Float3 up = verticies.get(x, max(y - 1, 0));
      final Float3 down = verticies.get(x, min(y + 1, verticies.Y() - 1));


      final Float3 dxv = sub(right, left);
      final Float3 dyv = sub(down, up);

      boolean invalidNormal = left.getZ() == 0 || right.getZ() == 0
            || up.getZ() == 0 || down.getZ() == 0;

      final Float3 normal = (invalidNormal) ?
        new Float3(INVALID,0f,0f)
        : normalise(cross(dyv,dxv));

      normals.set(x, y, normal);
    }
}
```

Dynamic Compiler



- The compiler is developed using GRAAL [2].
- It supports developer-driven parallelisation of tasks.
- It Support for vectorisation of memory accesses and arithmetic optimisations.
- Uses partial evaluation to eliminate all possible exception checks.

References

James Clarkson, Christos Kotselidis, Gavin Brown, and Mikel Luján.
Boosting java performance using gpgpus.
CoRR, abs/1508.06791, 2015.

G. Duboscq, L. Stadler, T. Würthinger, D. Simon, C. Wimmer, and H. Mössenböck.
Graal ir: An extensible declarative intermediate representation.
In *Asia-Pacific Programming Languages and Compilers*, 2013.

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 D. Riley, Nigel Topham, and Steve Furber.
Introducing slambench, a performance and accuracy benchmarking methodology for SLAM.
CoRR, abs/1410.2167, 2014.

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon.
Kinectfusion: Real-time dense surface mapping and tracking.
In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

Acknowledgements This work is supported by the AnyScale Apps and PAMELA projects funded by EPSRC EP/L000725/1 and EP/K008730/1. Dr Luján is supported by a Royal Society University Research Fellowship.