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 Java [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(float[][] a, float[][] b) {
  final float[][] c = new float[a.length][b[0].length];
  for (@Parallel int i = 0; i < a.length; i++)
    for (@Parallel int j = 0; j < b[0].length; j++)
      c[i][j] = a[i][j] + b[i][j];
}

Task Invocation add = addTask.invoke(a, b, c);
// Assign task to run on a GPGPU
add.mapTo(EXTERNAL_GPU);
// block until complete
add.execute();

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.

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 dependent on the performance of the tracking algorithm.
- Our Java implementation has been developed to produce bit-exact results when compared to SLAM BENCH implementations.

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

References
  Boosting java performance using gpgpus.

  Graal ir: An extensible declarative intermediate representation.
  2015.

  Tornado: Accelerating Java Applications Using GPGPUs.

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.