MANCHESTER 1824

The University of Manchester

Project Beehive:

A HW/SW co-designed stack for runtime and architecture research

Andy Nisbet
Research Fellow
Advanced Processors Technologies (APT) Group

MANCHESTER 1824

The University of Manchester

Team: EPSRC/EU Projects & Industry Funding

Compilers and Runtimes
Andrey Rodchenko
James Clarkson
Colin Barrett
Andy Nisbet
Christos Kotselidis

Binary Translation/Optimization
Amanieu d'Antras
Cosmin Gorgovan

FPGAs
Thanos Stratikopoulos
John Mawer

Hardware and Simulators
Yaman Cackmaci
Will Toms

Academic Staff
Jim Garside
Antoniu Pop
John Goodacre
Mikel Lujan

Open PhD Positions
LLVM-IR to Truffle (funded by Oracle Labs)
Several other Ph.D. positions

MANCHESTER 1824

The University of Manchester

Overview

- Challenges driving project Beehive's HW/SW co-design?
- What is project Beehive?
  - Application
  - Runtime
  - Hardware
  - Simulation
- Status of project Beehive?
- Next steps?

MANCHESTER 1824

The University of Manchester

Computing challenges

- Limited performance deltas: advances in micro-architecture and compilers.
- **Power**, Performance, **Resilience/Wearout**
- Heterogeneous manycore architectures
- Changing workloads (Computer Vision, Big Data)
- HW/SW co-design as a potential solution?
- Problematic, as requires vertical expertise across software stacks and micro-architecture

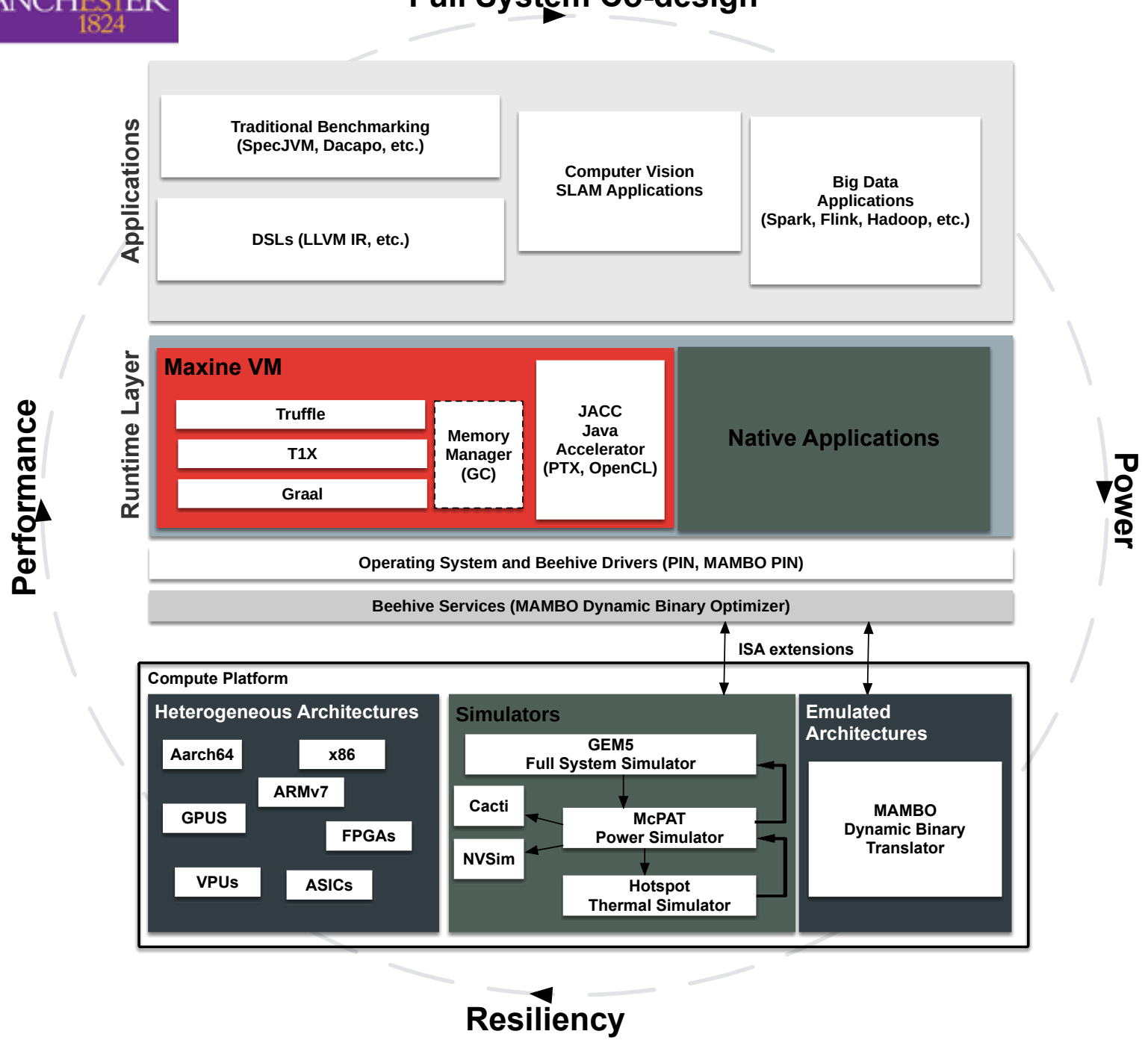
MANCHESTER 1824

The University of Manchester

Project Beehive

Goal: to provide an extensible state of art infrastructure for research/prototyping HW/SW co-designed systems.

- Traditional (native) and emergent application workloads.
- **Managed runtime environment with multiple guest languages**.
- Targets big.LITTLE, ARMv7, AArch64, GPGPU, FPGAs (accelerators/simulation), DSPs, x86.
- **Co-design: power, performance, and reliability/wearout modelling**
  - Dynamic binary instrumentation using *Pin* like tools
  - FPGA Accelerators and simulation/profiling tools that consume instrumentation information.
- **Design space exploration via fast simulation and investigation of application behaviour with minimal intrusion.**


Full System Co-design
Applications
Traditional Benchmarking (SpecJVM, Dacapo, etc.)
DSLs (LLVM IR, etc.)
Computer Vision SLAM Applications
Big Data Applications (Spark, Flink, Hadoop, etc.)
Runtime Layer
Maxine VM
Truffle
T1X
Graal
Memory Manager (GC)
JACC Java Accelerator (PTX, OpenCL)
Native Applications
Operating System and Beehive Drivers (PIN, MAMBO PIN)
Beehive Services (MAMBO Dynamic Binary Optimizer)
ISA extensions
Compute Platform
Heterogeneous Architectures
Aarch64
x86
ARMv7
GPUS
FPGAs
VPUs
ASICs
Simulators
GEM5 Full System Simulator
Cacti
McPAT Power Simulator
NVSim
Hotspot Thermal Simulator
Emulated Architectures
MAMBO Dynamic Binary Translator
Performance
Power
Resiliency

MANCHESTER 1824

The University of Manchester

Project Beehive: Application Layer

- Traditional use-cases: SPECint, SPECfp, SPECjvm, Dacapo …
- **Multiple guest languages**:
  - Native language support for C, C++, Java.
  - Guest language support via Truffle API to Graal. Ruby, R …
  - LLVM-IR to Truffle API – enable LLVM compiled languages.
- **Computer vision applications**
  - Simultaneous Localization & Mapping (SLAM) algorithms focusing on SLAMBench similar to *Google's Project Tango*
- **Big Data applications and frameworks**:
  - MapReduce, Spark, Flink, etc.

SLAMBench

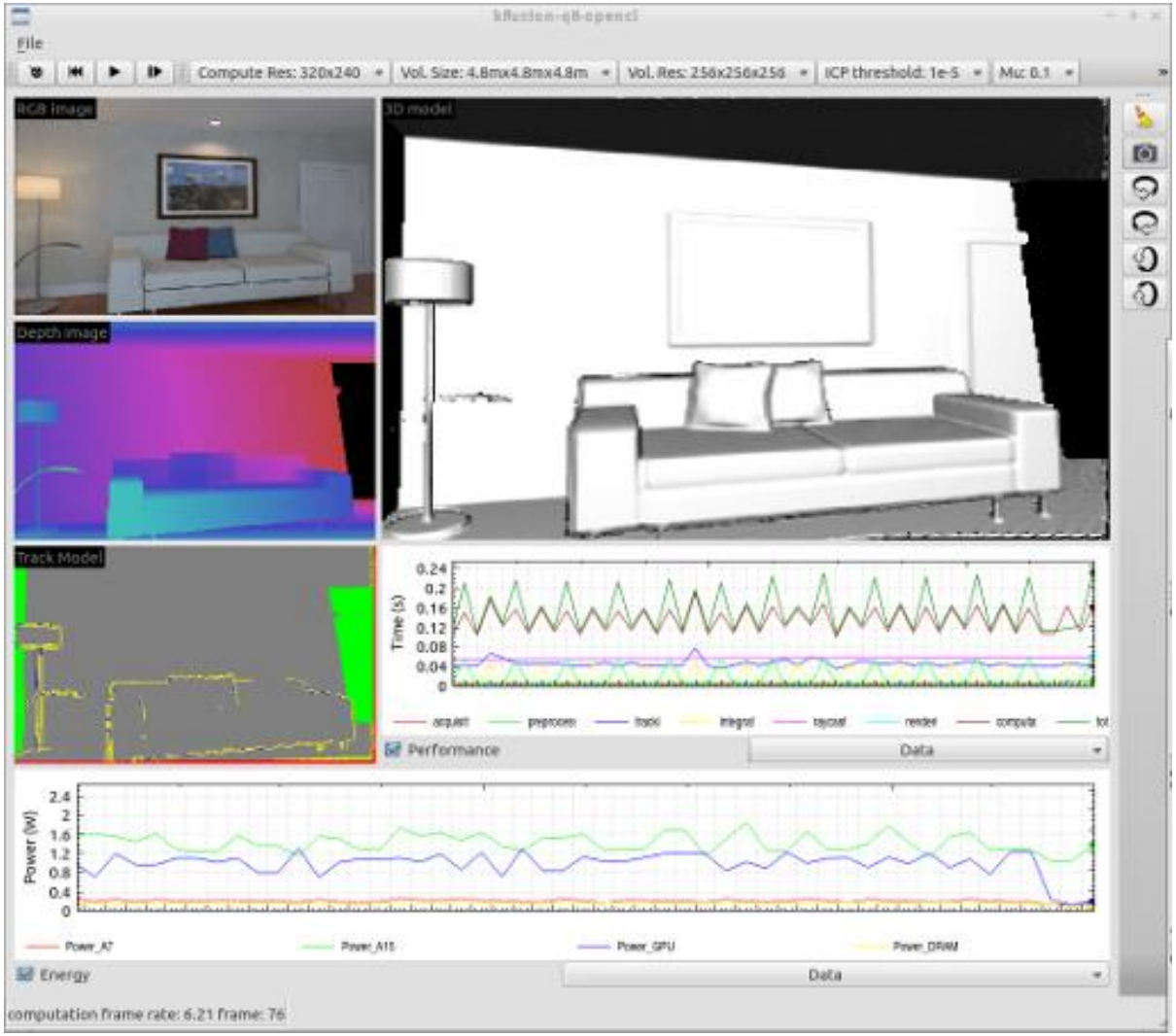
Project Web Page

http://apt.cs.manchester.ac.uk/projects/PAMELA/tools/SLAMBench/index.html

Android Application Download Link

https://play.google.com/store/apps/details?id=project.pamela.slambench

Joint with Edinburgh & Imperial



MANCHESTER 1824

The University of Manchester

Project Beehive: Runtime Layer

- **Maxine Research JVM**
  - Metacircular Java-in-Java VM with production quality optimizing compiler
  - T1X baseline and **Graal optimizing compiler**
  - Truffle AST Interpreter API for Graal **(multiple guest language support)**
- **Jacc: Java Accelerator**
  - Annotation based offloading onto heterogeneous hardware (GPGPUs and FPGAs), PTX, OpenCL code generation
- **Dynamic Binary Translators and Optimizers**
  - MAMBO: PIN-like tool for binary instrumentation
  - MAMBO64: DBT and DBO from ARMv7 to AArch64 – **enables emulation of new instructions/functionality in software as well as DBT**
- **System-level profiling & DVFS management**
  - Enable power management at runtime/application layers
- **In-house task-graph data-flow library**
  - Plan to co-design with the underlying compilers & Big-Data applications
  - **Exploit API semantics within MRE**

MANCHESTER 1824

The University of Manchester

Project Beehive: Hardware Layer

- **Targets**:
  - x86, ARMv7, AArch64, SIMD, GPGPUs, DSPs, FPGAs
- **Xilinx Zynq FPGA for rapid development**
  - Bluespec and other HDLs
  - Reconfigurable hardware
  - IP blocks development
  - Dual core Cortex-A9 ARMv7 + Xilinx FPGA.
- **SLAMBench**:
  - IP blocks enable cache simulation with MAMBO.
  - Acceleration of kernels using IP blocks
- **Maxine:**
  - Instrumentation for JIT-code **(soon!)**.
  - Exploitation of accelerator IP blocks.

MANCHESTER 1824

The University of Manchester

Project Beehive: Simulation Layer

- **Target platforms**: x86, ARMv7, AArch64, SIMD, GPGPUs, FPGAs
- **Simulators for fine grained power/performance/reliability modeling**:
  - GEM5 with McPAT and Hotspot for power and temperature measurements **(SLOW but ACCURATE)**
  - Non-volatile memory simulations via NVSim
  - **Fast customizable accuracy simulation of memory/full-systems using binary instrumentation & FPGAs**
- **Co-designed research on memory managers with Zsim**
  - Lightweight fast tagging of object pointers
  - Tags enable memory system performance/profiling to influence microarchitecture/JIT compilation (to improve cache/GC performance)

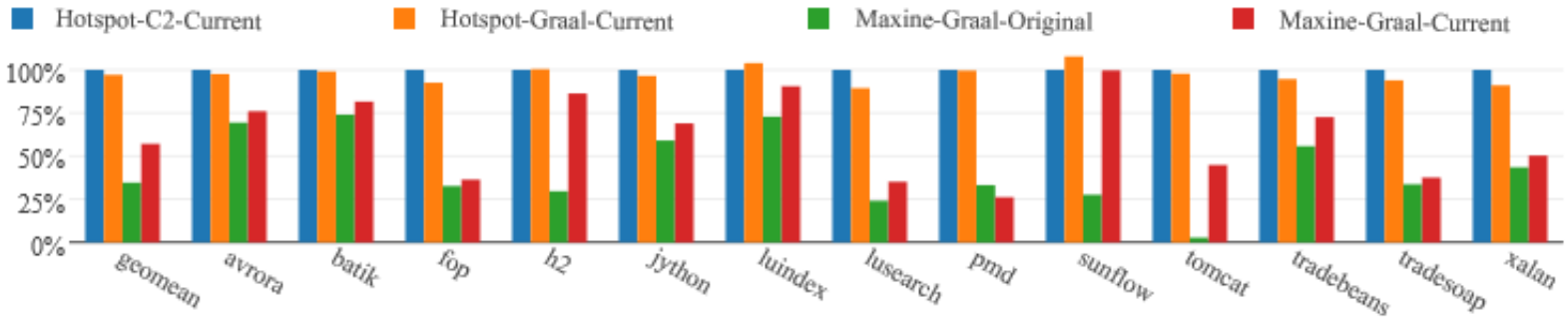
Evaluation

- **Beehive is WIP** and can not be evaluated holistically at present
- **Maxine VM**
  - Mature on x86 and available from Kenai
  - ARMv7 runs JIT-ed HelloWorld and other small benchmarks.
  - ARMv7 larger benchmarks can be ahead of time compiled into the Maxine “bootimage”
- **FPGA Simulation/acceleration**:
  - MAMBO: validation of cache hierarchy models against GEM5 is in progress
  - Maxine (**soon!**)
- **Mature components** that can be evaluated standalone:
  - MAMBO and MAMBO64 performance
  - Jacc GPGPU off-loading performance

MANCHESTER 1824

The University of Manchester

Graal-MaxineVM Improvement



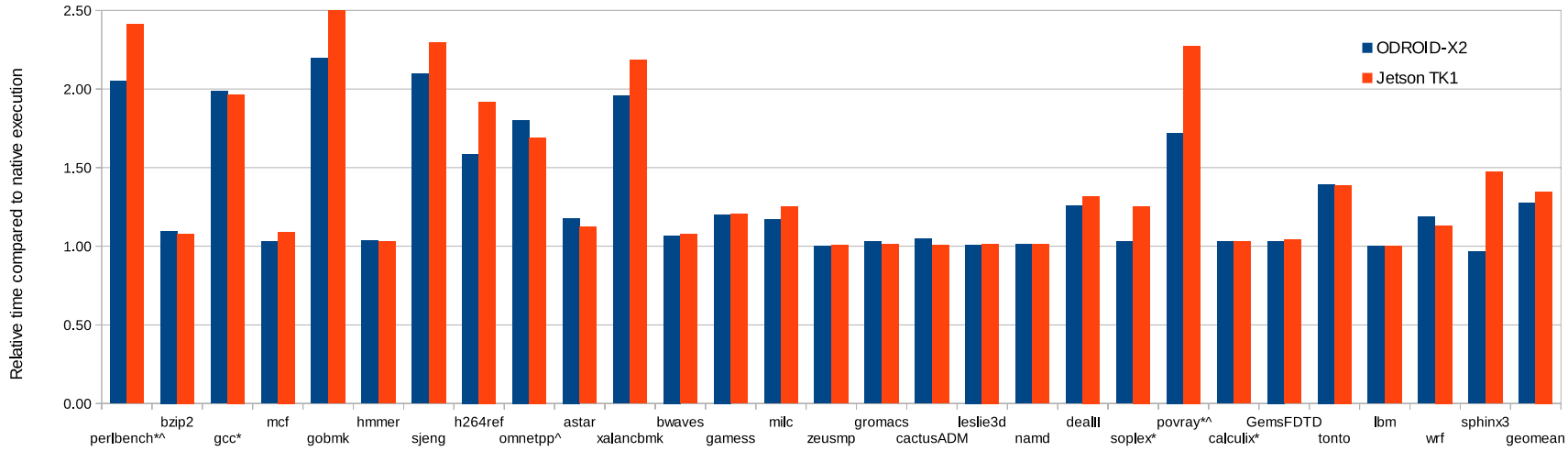
- **100% implies performance of Hotspot-C2-Current, higher is better**.
- Added profiling instrumentation in T1X baseline compiler (branches, type information, exceptions, etc.)
- Enabled aggressive Graal optimizations
- **Resulted in 1.64x speedup over Maxine-Graal-Original version**
- Changes upstreamed to Kenai repository for Maxine
- **Maxine VM provides a research MRE with an actively developed (Graal) optimizing compiler**
- **Andrey Rodchenko's work**

MANCHESTER 1824

The University of Manchester

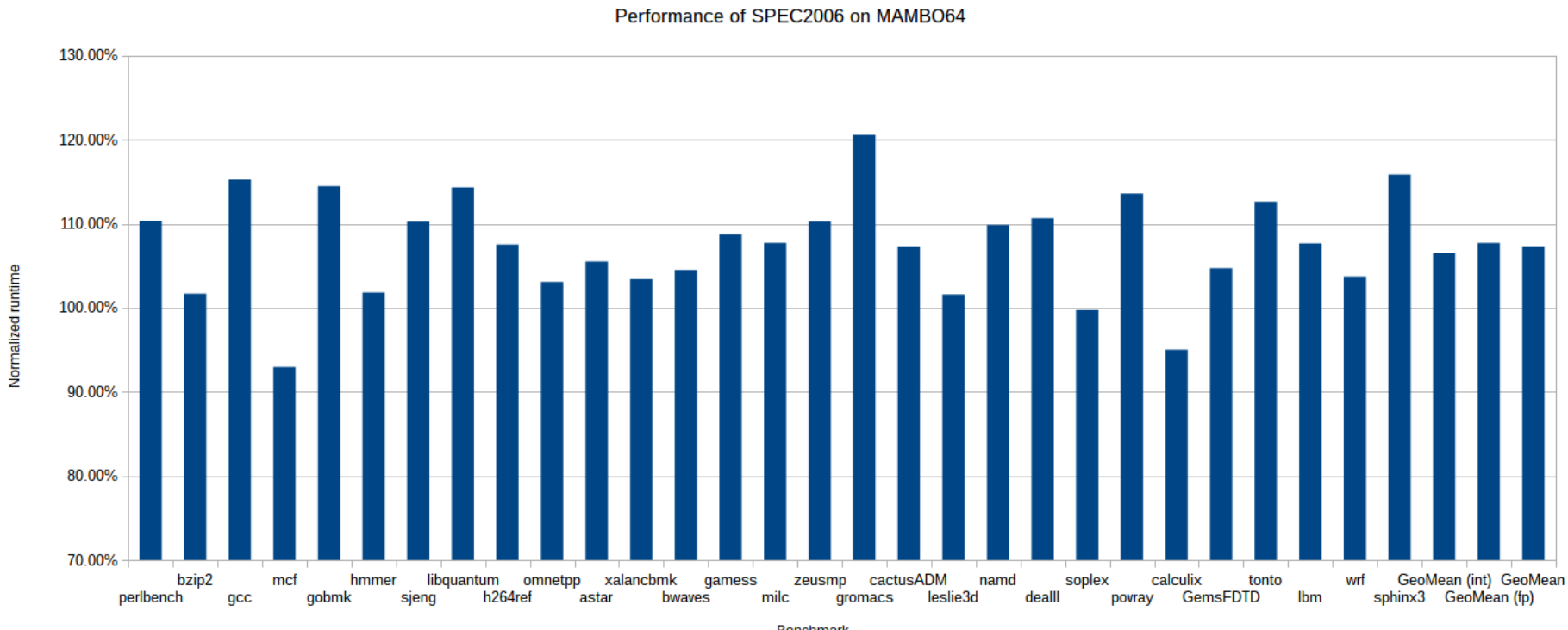
MAMBO (Cosmin Gorgovan)

- Dynamic Binary Modification Tool for ARMv7 (PIN-like)
- Evaluation of SPECcpu2006 on two different platforms
  - Odroid-X2: 4 Cortex-A9@1.7GHz ~ 28% overhead
  - Jetson TK1: 4 Cortex-A15@2.3GHz ~ 35% overhead



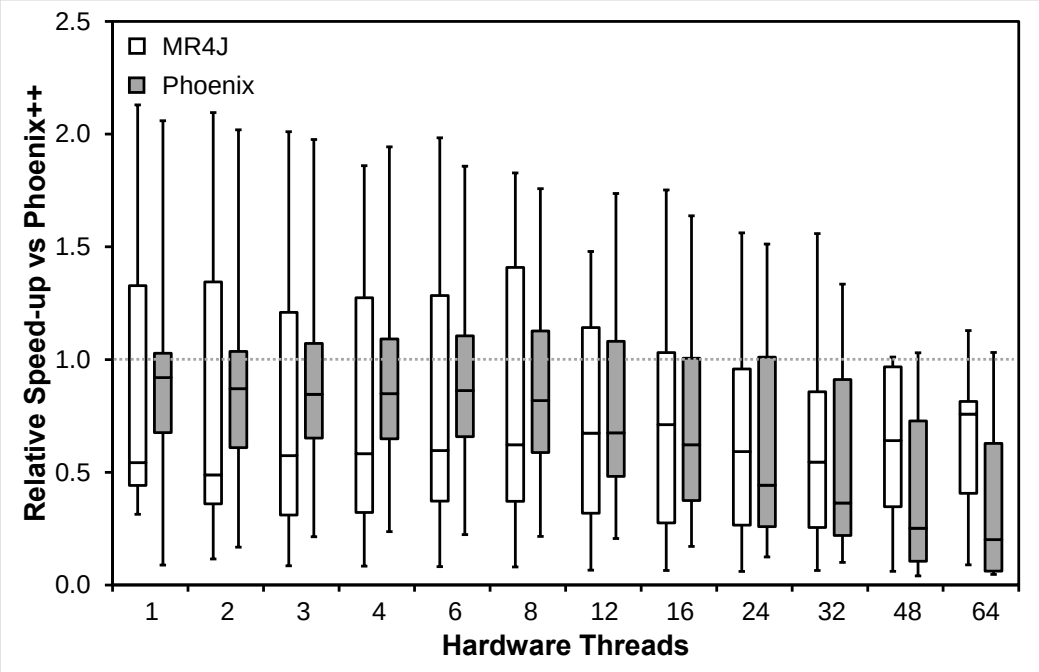
MAMBO64 (Amanieu d'Antras)

- Dynamic Binary Translator from ARMv7 to Aarch64
  - SPECcpu 2006 on Juno Dev. Platform < 10% overhead
- 100% represents the performance of the 32-bit binary running natively on the 64-bit processor (in 32-bit mode)
- Note that in some cases MAMBO64 can generate 64-bit code that runs faster than the 32-bit code running natively.



MR4J (Colin Barrett)

- Java MapReduce framework for CMPs
  - Co-designed optimization by detecting and inserting combining functions
  - Java agents are used to rewrite classes
  - Increases thread-local computation and reduces data communication
  - **Exploits semantics of Map-Reduce in order to produce more efficient code (inspired by C++ implementation)**
- AMD Opteron 6276 64HTs @2.3GHz, Phoenix++ vs MR4J (0.76) vs Phoenix (0.20)
- Benchmarks: Histogram, KMeans, Linear Regression, Matrix Multiply, Principal Component Analysis, String Match, Word Count



MANCHESTER 1824

The University of Manchester

Vectorization Abstractions (Colin Barrett)

- Co-designed vector class and optimization
  - Exploits the availability of SIMD instructions
  - Abstraction for points, translation and rotation on SLAM applications
- Extend Graal IR
  - Substitute specialized nodes for vector abstraction
  - Using existing optimization techniques
- Inspired by use of vector intrinsics in optimized C++
  - Frequent operations in SLAM applications with short vector length
  - Potential abstraction for SSE, AVX, NEON
- **Performance improvements upto 16x with SIMD vector length (4) due to removal of unnecessary deoptimisation guards**
- **Exploitation of semantics to remove Java overheads & utilize SIMD functional units.**

MANCHESTER 1824

The University of Manchester

Next Steps

- Complete the AArch64 port of Maxine
- Increase coverage and stability of Maxine-ARMv7
- **Improve the GC in Maxine** (semi-space stop the world for ARMv7) Generational GC may need additional modifications? ***Any helpers out there to implement other schemes?***
- Complete validation of the simulation platform, including timing (<5% difference GEM5, able to model big.LITTLE systems).
- Dissemination

MANCHESTER 1824

The University of Manchester

Questions?

JACC-GPGPU Offload (Tornado)

- Offloading *@Jacc* annotated code onto accelerators (GPGPUs, FPGAs, etc.)
  - Dedicated compiler for special compilation and optimizations
  - PTX and OpenCL code generators
  - Two Intel-Xeon E5-2620 (24 HTs@2.0GHz), NVIDIA Tesla K20m ~32x on average and 4.4x less code