Building an AI that Codes

http://**chris**cummins.**cc**

2013

intel

01.ORG INTEL OPEN SOURCE TECHNOLOGY CENTER

2014

Aston University

pip-db

Protein Isoslectric Point Database

2015

THE UNIVERSITY of EDINBURGH

informatics

EPSRC Centre for Doctoral Training in Pervasive Parallelism

EPSRC

Engineering and Physical Sciences Research Council

2016

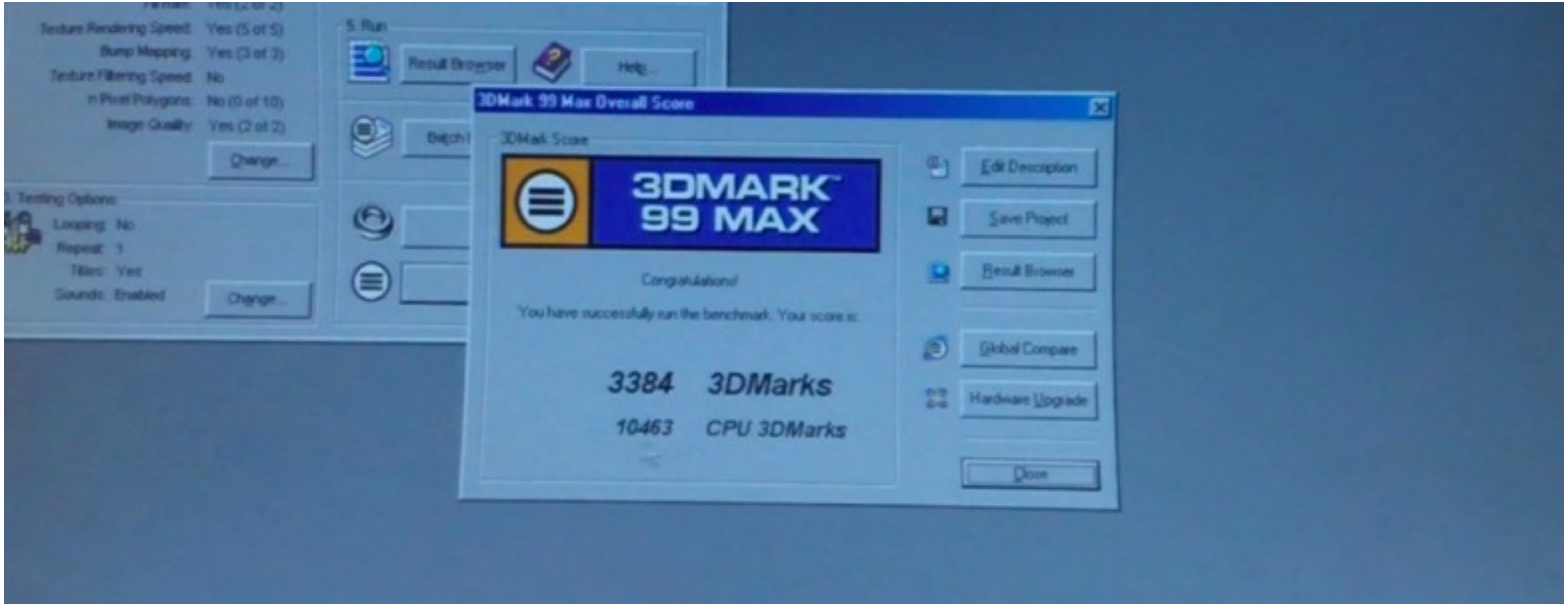
codeplay

TensorFlow + SYCL


52x/32x/52x
Low Radiation


GIFTS
POLICE

What makes a good computer?



big numbers != smooth games

fast forward …

Consider the “implementation space”

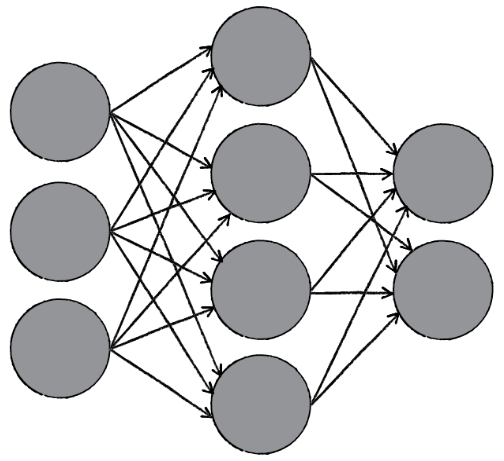
Hypothesis: real source codes form clusters

Weight space to match clustering

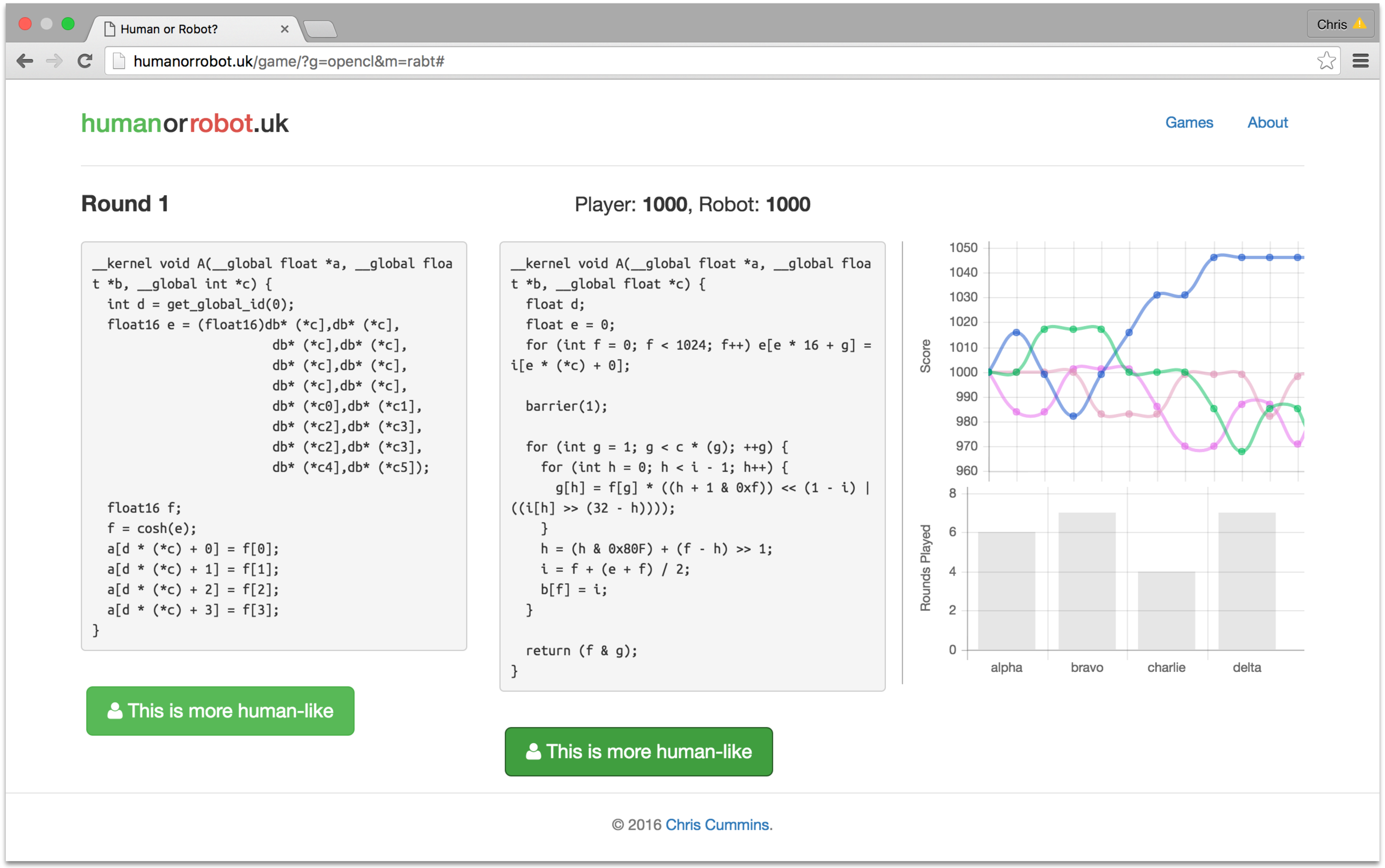
Sample from weighted space to generate new, representative benchmarks on-demand.

The idea

Mine programs from

Apply  over implementation space

Generate representative benchmarks on-demand



http://**human**or**robot**.uk


Curing the Benchmark Deficit: On-Demand Compute Kernel Synthesis using Deep Learning

Chris Cummins
University of Edinburgh
c.cummins@ed.ac.uk

Pavlos Petoumenos
University of Edinburgh
ppetoume@inf.ed.ac.uk

Zheng Wang
Lancaster University
z.wang@lancaster.ac.uk

Hugh Leather
University of Edinburgh
hleather@inf.ed.ac.uk

Abstract

The quality of performance tuning is bound by the quantity and quality of benchmarks used. Too few benchmarks leads to overfitting; non-representative benchmarks lead to invalid predictions.

We present a novel methodology for generating OpenCL compute kernels. Given a corpus of example programs, we apply deep learning across the implementation space, learning a language model from which we obtain new kernels through a process of rejection sampling. We demonstrate our approach for a state-of-the-art machine learning OpenCL autotuner. With the addition of synthesised compute kernels, we improve the accuracy of machine learning predictions from XXX% to XXX%, demonstrating up to X× speedup over the hand-selected benchmarks.

Keywords Synthetic program generation, OpenCL, Deep Learning, GPUs

1. Introduction

Benchmarking parallel applications is hard. State of the practice is lacking [1].

OpenCL benchmark suites: Rodinia [2], Parboil [3], Polybench [4], SHOC [5], AMD SDK [1] and NVIDIA SDK [2]. TODO: Tease the small number of benchmarks used.

Benchmarking OpenCL [6].

Are benchmarks suites representative? Exploring the full performance spectrum [7]. Characterising workloads of Rodinia and Parsec [8].

In previous works we used stochastic template substitution to generate stencil benchmarks for autotuning [9, 10]. This template based approach is not general purpose, requires laborious human effort, and does not guarantee to generate representative benchmarks.

RQ1: Can the quality of machine learning predictions be improved with the addition of representative benchmarks?

For this, we need a methodology for generating novel source codes. This leads to the subsequent research question:

RQ2: Given a program checker and a corpus of example programs, can a language model learn to generate new programs?

The key contributions of this work are as follows:

- a novel methodology for generating synthetic compute kernels for machine learning-based performance tuning;
- a large scale source code evaluation and re-writing for character-level language modelling using deep learning;
- improved tuning performance of [CGO'13] on hand written OpenCL kernels.

2. Motivation

We surveyed the benchmarking methodologies of GPU research papers from top tier conferences between 2013–2016: CGO, HiPC, PACT, and PPoPP. By aggregating the sources and quantities of benchmark kernels from 27 papers, we find that 79% of benchmark kernels come from four benchmark suites [3]. Figure 1 and Figure 2.

3. Generating Compute Kernels

Overview of methodology. Figure 3.

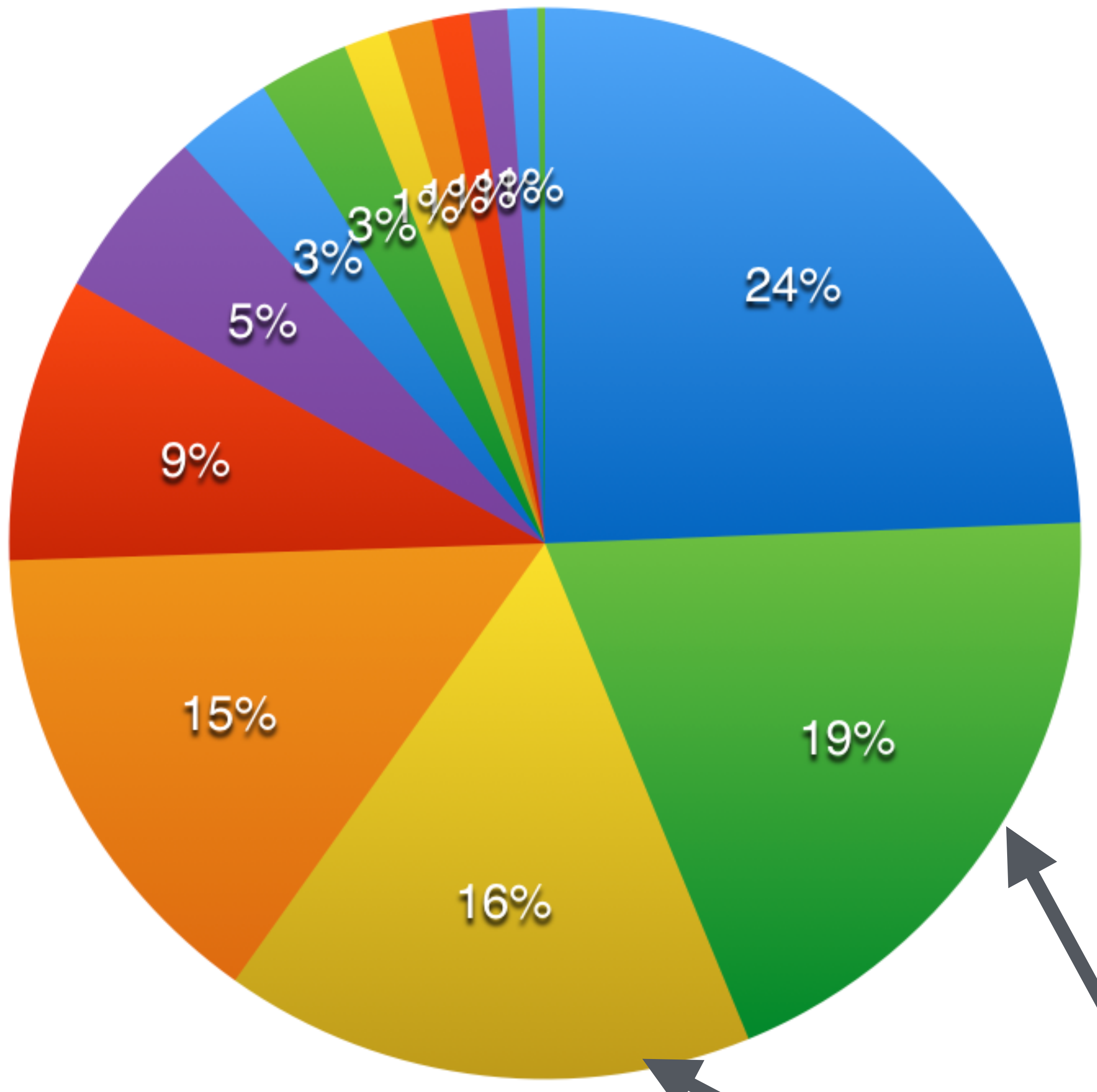
[1] http://developer.amd.com/
[2] https://developer.nvidia.com/opencl/
[3] Raw data available at: http://bit.ly/foo

1

2016/7/18

2013-2016 state of practise
27 top-tier GPU papers

79% of results from **4**
benchmark suites



(SDK sample codes)