www.bsc.es

Brook GLES Pi: Democratising Accelerator Programming

Matina Maria Trompouki, Leonidas Kosmidis

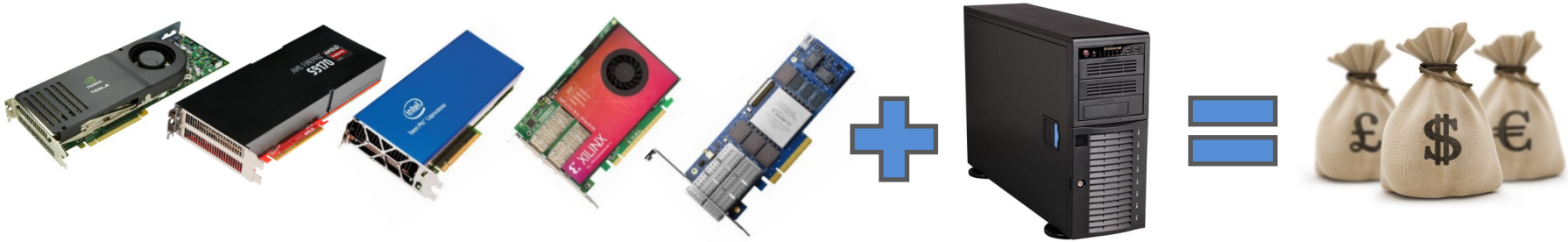
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HPG 2018

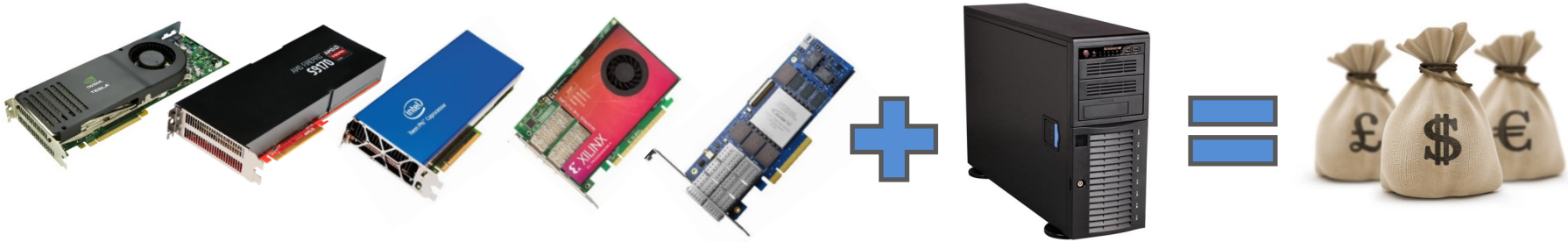
Introduction and Motivation

- Modern and future computing relies on general purpose accelerators
  - Increased public and scientific interest
  - Increased importance of learning and experimenting with their programming paradigm
- However their cost is high
  - Accelerator is expensive
  - Requires a high-end host computer to be used



Introduction and Motivation

- Traditional educational model shifted to self-education
  - Homework practice
  - Massively Open Online Courses (MOOC)
  - Experimentation with educational computers
- Unaffordable for these target groups
- Accelerator programming opportunities are limited!



Brook GLES Pi

Port of the open-source accelerator programming language Brook on the low-cost ($25) educational computer Raspberry Pi

- Enables the use of its embedded GPU VideoCore IV, capable of 24 Gflops
- Standalone development on the device
- Large and collaborative community
- Open-source implementation
- Portability across every embedded GPU supporting OpenGL ES 2 (99% of the embedded devices with a GPU in the market)
- Allows teaching, experimenting and learning GPGPU programming with affordable devices

Brook

We implemented our solution in the Brook programming language [1]

- open source language developed circa 2004
  - predecessor of CUDA and OpenCL
  - Commercially adopted by AMD before OpenCL, rebranded as Brook+
- Source-to-Source compiler and Runtime
  - Restricted subset of C (no recursion, no goto, no pointers)
  - transforms CUDA-like programs to graphics operations
  - supports multiple backends

[1] I. Buck et al, Brook for GPUs, SIGGRAPH 2004

Brook vs CUDA Example

```
#define MAX_ITERS 10000

kernel void foo(float a<>, float b[], out c<>){
 float acc=0.0;
 for(int i<0; i < a || i < MAX_ITERS; i++){
   acc += b[indexof(c).x];
 }

 c = a + acc;
}

int main(void){
 float a_h[100], b_d[100], c_h[100];
 float a_d<100>, b_d<100>, c_d<100>;

 streamRead (a_d, a_h);
 streamRead (b_d, b_h);
 foo (a_d, b_d, c_d);
 streamWrite (c_d, c_h);
}
```

```
__global__ void foo(float * a, float * b, float * c){
 unsigned int tid = blockIdx.x*blockDim.x + threadIdx.x;
 float acc=0.0;
 for(int i < 0; i < a[tid]; i++)
   acc += b[tid];

 c[tid] = a[tid] + acc;
}

int main(void){
 float a_h[100], b_d[100], c_h[100];
 float * a_d, * b_d, * c_d;

 cudaMalloc(&a_d, 100*sizeof(float));
 cudaMalloc(&b_d, 100*sizeof(float));
 cudaMalloc(&c_d, 100*sizeof(float));

 cudaMemcpy(a_d, a_h, 100*sizeof(float), cudaMemcpyHostToDevice);
 cudaMemcpy(b_d, b_h, 100*sizeof(float), cudaMemcpyHostToDevice);
 foo<<<1,100>>>(a_d, b_d, c_d);
 cudaMemcpy(c_h, c_d, 100*sizeof(float), cudaMemcpyDeviceToHost);
}
```

Additions

Ported Brook on the Raspberry Pi

- Introduced an OpenGL ES 2 compiler backend and runtime
- Optimised for the Raspberry Pi
- Based only on the core OpenGL ES 2 standard
  - No vendor specific extensions
  - Maximum portability across all embedded devices with GPUs supporting OpenGL ES 2 (>99% of the market)
- Shares common code base with Brook Auto [1]
  - 2K Lines of code in the compiler and runtime
  - 4K Lines of code in regression tests and benchmarks

[1] Trompouki et al, Brook Auto: High-Level Certification-Friendly Programming for GPU-powered Automotive Systems, DAC 2018

Additions

The compiler backend uses Nvidia's Cg compiler like the original Brook

- Provided only in binary form for x86
- Raspberry is ARM-based
  - Emulate compiler using the binary translator qemu-x86
  - Enables standalone development and compilation on the device
  - Compilation time is similar to the native compilers, e.g. gcc

Original Brook only supports floating point (and their vector) streams

- Added support for **char** and **int**
  - Signed and unsigned versions
  - Vector additions

Additions

Stream datatypes limitations due to OpenGL ES 2

- Input and output streams are limited to 32-bit
  - up to vectors of 4 **char**s, 1 **int**, or 1 **float**ing point
- Only a single output stream is permitted per kernel, up to 32-bit
- When a kernel violates these rules and the OpenGL ES 2 backend is enabled, the programmer is instructed to rewrite the code

Dropped Features

Iterators

- Unusual Brook feature, syntactic sugar for creating and initialising streams of indices
- No mapping with any CUDA/OpenCL concept
- AMD's Brook+ examples use **indexof** instead
- Would complicate unnecessarily the implementation
  - We want equivalence of CPU/GPU code but
  - OpenGL ES 2 only supports normalised coordinates

Dropped Features

GatherOp

- CPU emulation of gather operations
- Only needed in GPUs not supporting dependent texture lookups
  - All known OpenGL ES 2 GPUs support it

ScatterOp

- CPU emulation of read-modify-write
- Slow performance
- Scatter to gather transformation is encouraged for accelerators whenever is possible

Neither have an equivalent in CUDA/OpenCL

Currently Unsupported Features

Structs

- Supported in accelerators but their use is discouraged
  - Limits vectorisation
  - Under-utilises memory bandwidth
- Memory layout transformation is recommended
  - Array of Stuctures (AoS) to Structure of Arrays (SoA)
- Plans to be supported in the future to allow experiencing the performance difference

Results

Experimental Setup

- AMD Brook+ SDK applications, different input sizes up to 2048 elements
- Raspberry Pi
  - Brook GLES Pi backend and runtime, OpenGL ES 2
  - $25 cost

Comparison with:

- NVIDIA GeForce GTX 1050Ti (High/Medium-End desktop GPU)
  - $250 GPU cost, $2500 total cost including the host
- NVIDIA Jetson TX2 (High-End embedded GPU)
  - $600 platform cost
- Original Brook OpenGL backend and runtime

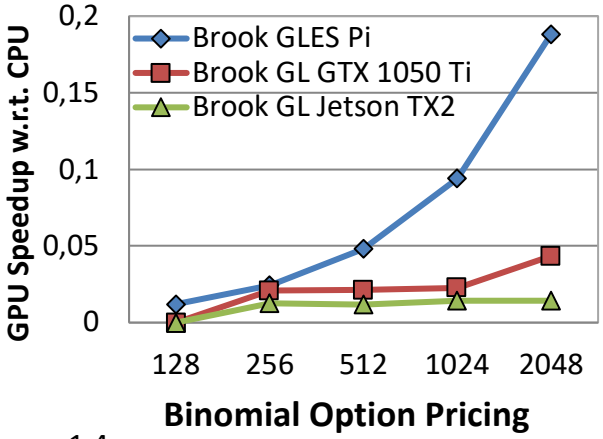
Relative Performance of the Systems used in the comparison

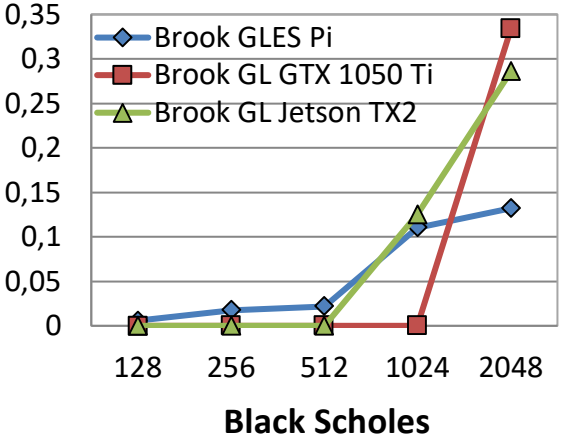
Relative performance CPU vs GPU of the systems used in comparison

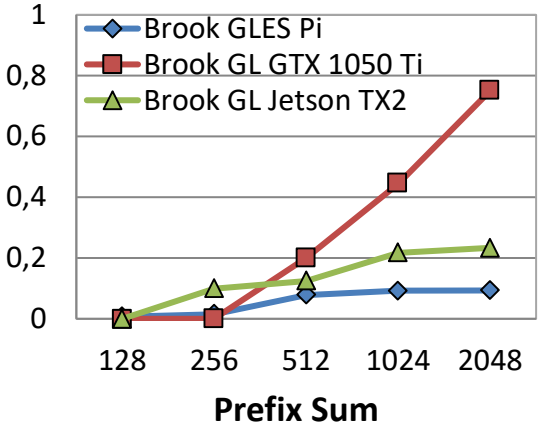
- Reported by the flops benchmark
- OpenMP CPU support is disabled in the multicore high-end systems
  - Their speedups are higher

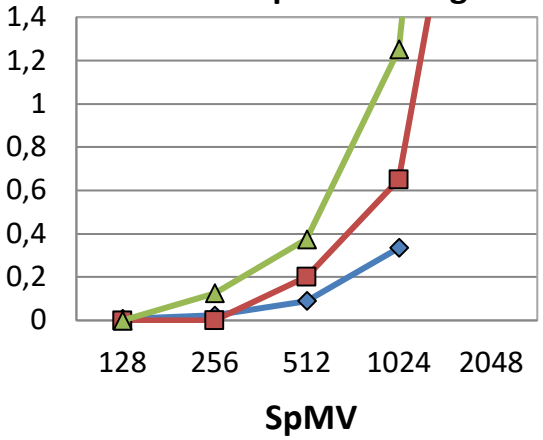
Platform (GPU/CPU)	Performance Ratio	Bandwidth Ratio
Raspberry Pi VideoCore IV vs ARM	23 x	1/33 x
NVIDIA GTX 1050 Ti vs AMD CPU	19 x	1/11 x
NVIDIA Jetson TX2 Pascal GPU vs ARM	11 x	1/4 x

Brook GLES Pi Evaluation: Weak scaling GPU Programs



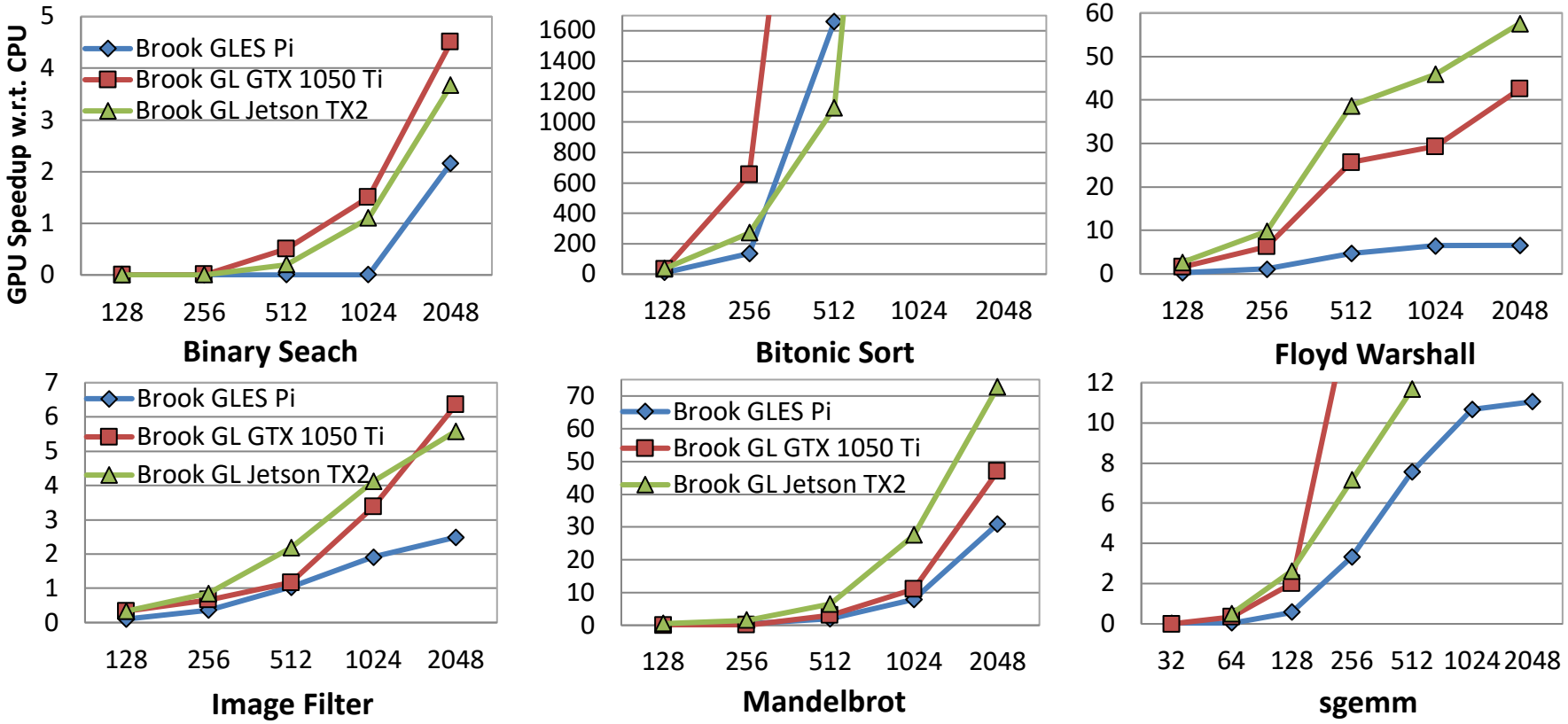






- Benchmarks that do not scale with input size in high-end GPUs do not scale either in Brook GLES Pi
- Relative performance in the same order of magnitude

Brook GLES Pi Evaluation: Strong scaling GPU Programs



Same performance trends, relative performance within 2 orders of magnitude

Conclusion

Brook GLES Pi

- Port of the Brook Programming Language over OpenGL ES 2
- Optimised for the low-cost educational computer Raspberry Pi
  - Portable OpenGL ES 2 allows the use of any embedded GPU
- Similar performance trends with original Brook on high-end GPU systems
- Democratises accelerator programming

- Code:

  http://github.com/lkosmid/brook

Thank you!

www.bsc.es

Brook GLES Pi: Democratising Accelerator Programming

Matina Maria Trompouki, Leonidas Kosmidis

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HPG 2018