Support Tools for Porting Legacy Applications to Multicore

Natsuki Kawai, Yuri Ardila,
Takashi Nakamura, Yosuke Tamura

Agenda

- Introduction
- PEMAP: Performance Estimator for MAny core Processors
  - **The overview of PEMAP**
  - **Estimation Methods**
  - **Demonstration of PEMAP**
- BEMAP: BEnchMarks for Automatic Parallelization
  - **The overview of BEMAP**
  - **Optimization Methods**
  - **BEMAP as a benchmark for PEMAP**
  - **BEMAP as a standalone benchmark**

Introduction

- In the first step of software parallelization, we need to estimate performance benefit of parallelization
- Also, this performance estimator needs a benchmark to examine its performance liability
- We propose two development tools for them
  - ✓ **PEMAP: Performance Estimator for MAny core Processors**
  - ✓ **BEMAP: BEnchMark for Automatic Parallelization**

PEMAP:

Performance Estimator for MAny-core Processors

Overview

- PEMAP estimates performance benefits of parallelization of existing sequential programs
  - ✓ **Without any parallel programming**
- Users have to insert two annotations to target sequential programs

```
void abs(float *d)
{
    int i;
    PEMAP_LOOP_START;
    for (i = 0; i < 1000000; i++) {
        d[i] = d[i] >= 0.0 ? d[i] : -d[i];
    }
    PEMAP_LOOP_END;
}
```

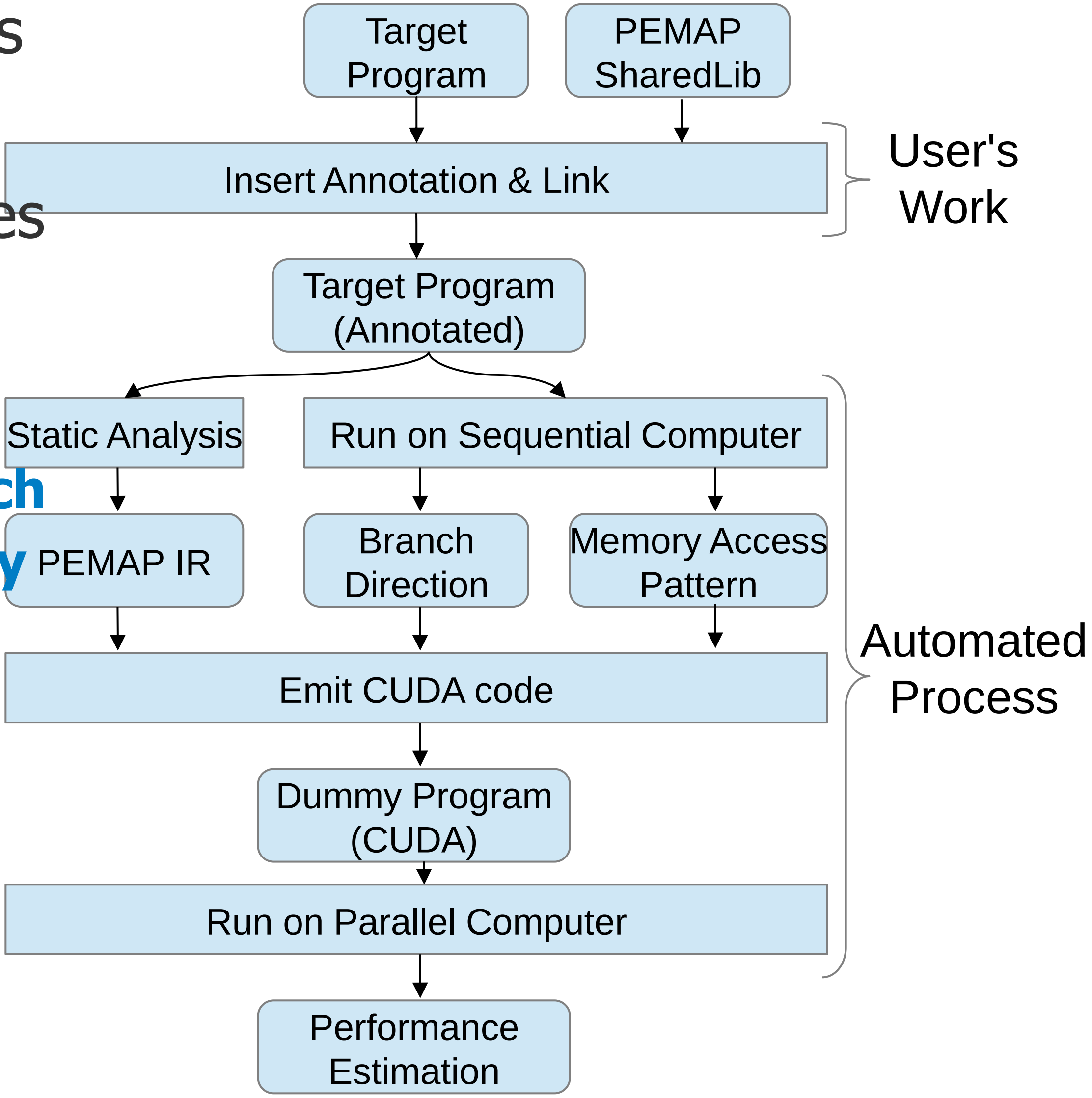
Idea

- Performance of GPU programs is strongly depends on their calculation costs and memory accesses.
- We propose to reconstruct them on GPU by existing methods of loop analysis.
  - ✓ **PEMAP does NOT analyze inter-loop data dependency because it is not important for performance estimation**

Reference: www.nvidia.com

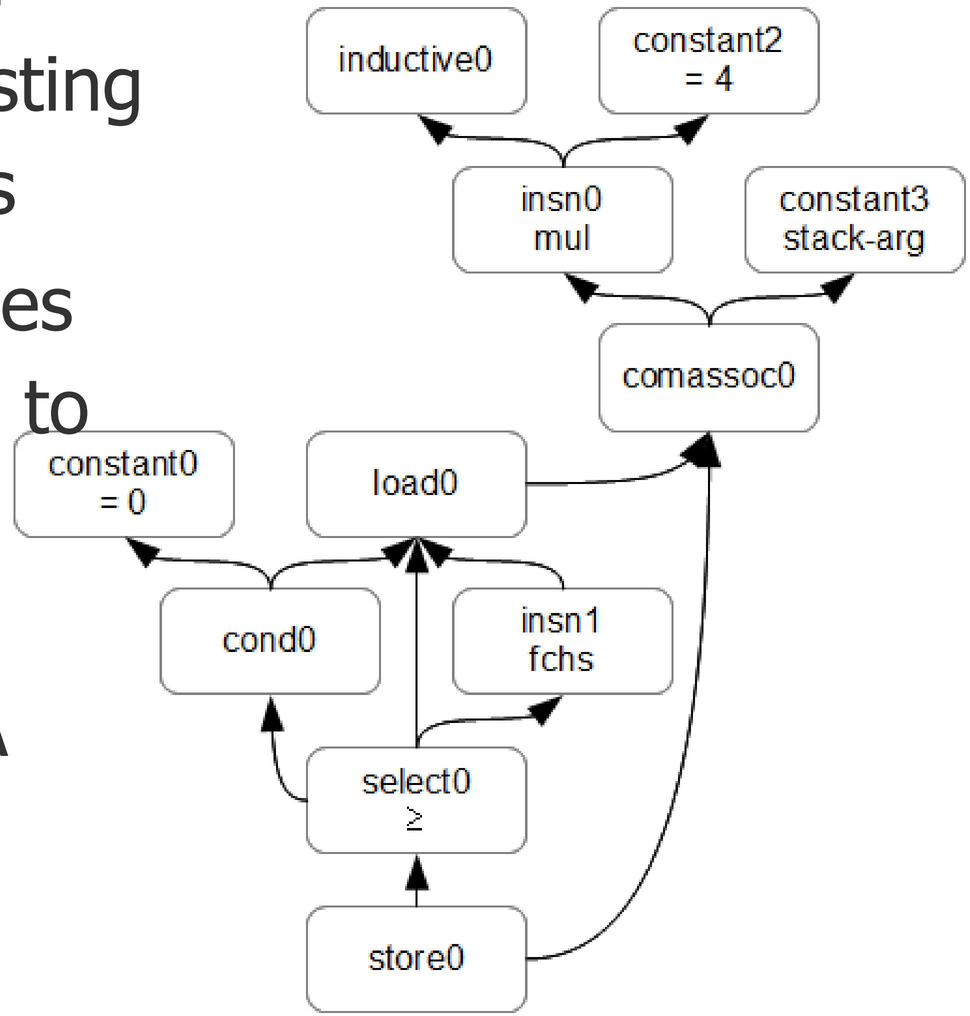
Estimating Processes

- The first annotation calls PEMAP
- PEMAP statically analyzes the target
- Runs the target
  - ✓ **To collect actual branch directions and memory access patterns**
- Generates a dummy program
  - ✓ **It has virtually same performance as a hand-parallelized program**



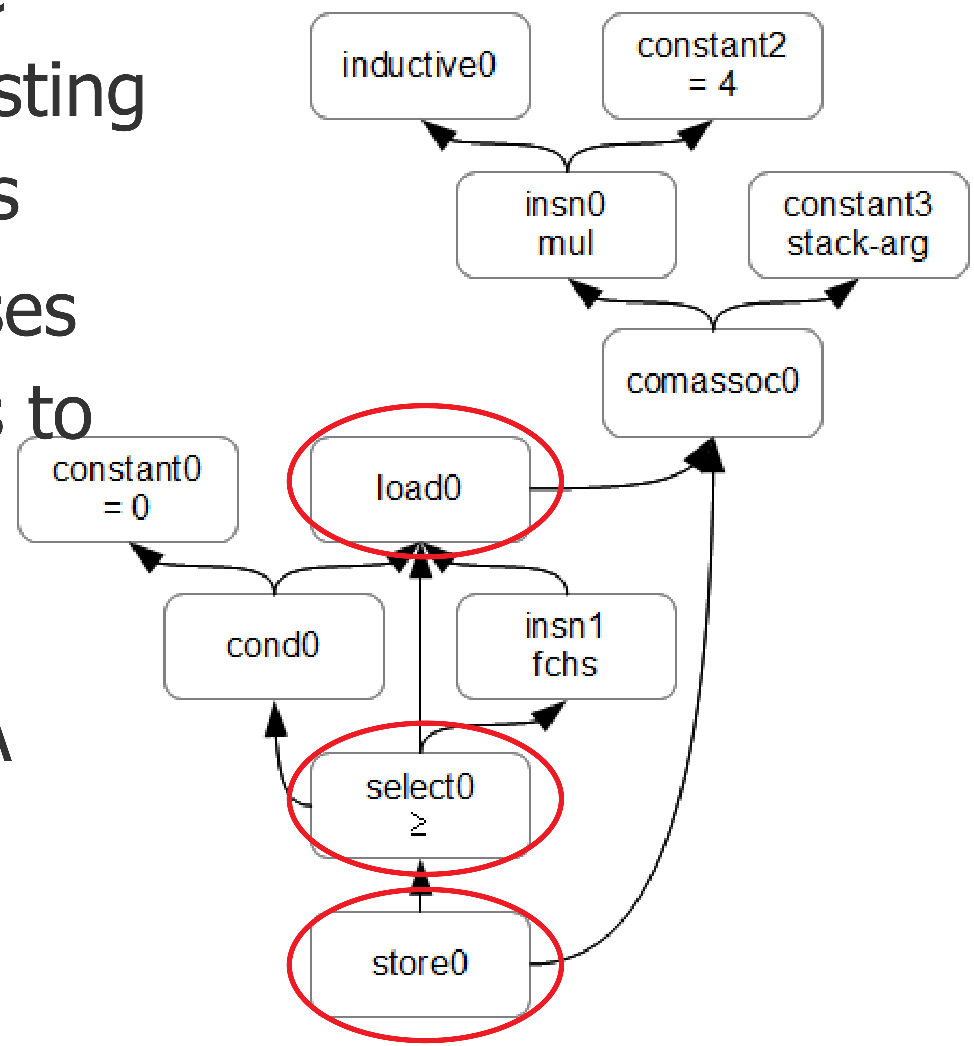
The analyzing method

1. Create graphs from the target programs by existing loop-analyzing methods
2. Pick up memory accesses and condition branches to collect
3. Traverse the graphs to generate dummy CUDA programs



The analyzing method

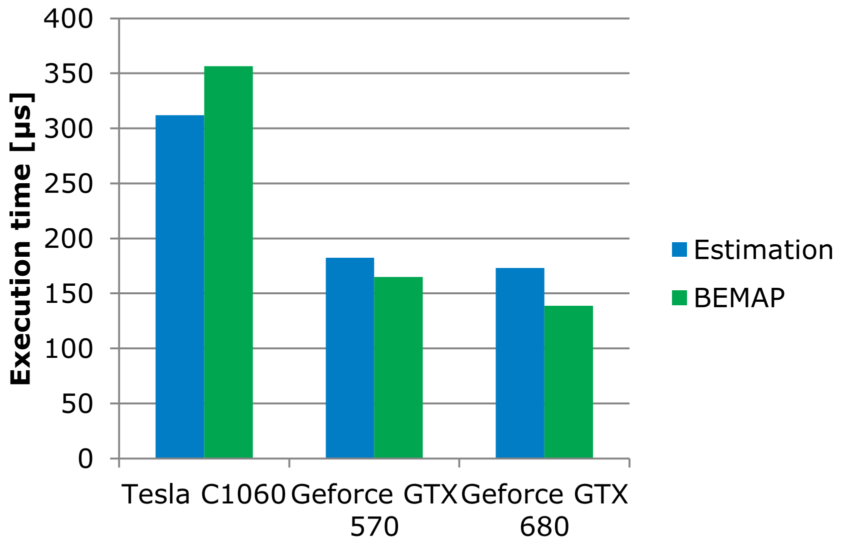
1. Create graphs from the target programs by existing loop-analyzing methods
2. Pick up memory accesses and condition branches to collect
3. Traverse the graphs to generate dummy CUDA programs



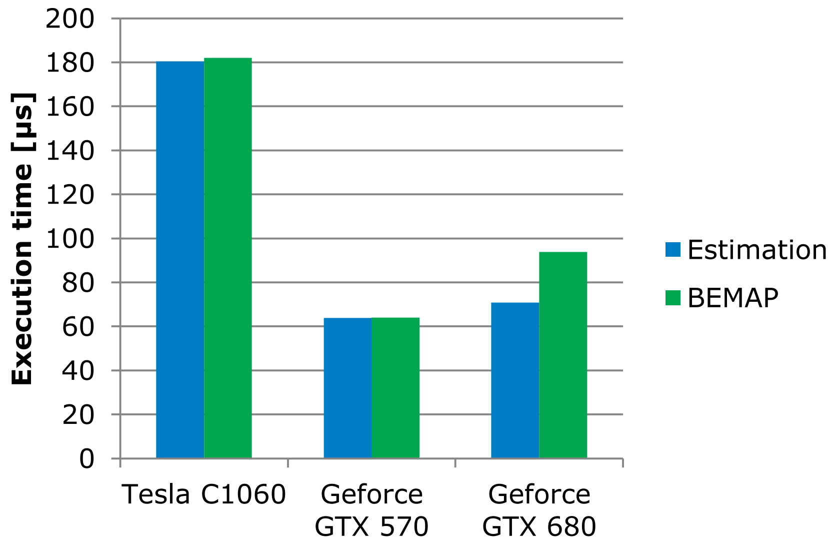
Evaluation

- We selected Black-Scholes and Grayscale benchmarks from BEMAP
  - ✓ **Estimated the performance of sequential samples in BEMAP**
  - ✓ **Ported OpenCL samples in BEMAP to CUDA**

Black-Scholes Benchmark



Grayscale Benchmark



User Interface

PEMAP: Performance Estimator for MAny-core Processors

Cloud PEMAP Front Page

Cloud PEMAP is a web service version of a performance estimation tool PEMAP.

Language:
- ( ) C
- (•) C++

Device:
- [x] Quadro FX 3700

```
void grayscale_gold(ImgStream & t_out, const ImgStream & inp, const int h,
                    const int w)
{
    float rr, gg, bb;
    float v;
    int i, j;
    int idx;

    PEMAP_LOOP_START;
    for (i = 0; i < h / 2; ++i) {
        for (j = 0; j < w / 2; ++j) {
            idx = i * w + j;
            rr = (float) inp.data_r[idx];
            gg = (float) inp.data_g[idx];
            bb = (float) inp.data_b[idx];
            v = R_RATE * rr + G_RATE * gg + B_RATE * bb;
            t_out.data_r[idx] = (pixel_uc) v;
        }
    }
    PEMAP_LOOP_END;
}
```

[Estimate!!]

User Interface

PEMAP: Performance Estimator for MAny-core Processors

Cloud PEMAP Result

PEMAP succeeded to estimate the performance of your program. Please confirm the result table and PEMAP-result below.

You always can access this result from http://localhost:4567/68/.

Return to the top-page of Cloud PEMAP to estimate your next program.

The result of estimation

GPU	# threads	exec time
CPU	1	4.958791318000001ms
Quadro FX 3700	32	2.024175ms

The output of PEMAP (including output of your program before PEMAP was called)

```
num_all_reg_arg: 15
load0: 0
load1: 1
load2: 2
store0: 0
load0 is NOT a random access.
load1 is NOT a random access.
load2 is NOT a random access.
store0 is NOT a random access.
tmpxft_00001dc0_00000000-14_dummy.ii
Target GPU: Quadro FX 3700
nitr=1 nthread=32 : 2.024175[msec]
```

BEMAP:

BEnchMark for Auto Parallelizer

Background

aims to :

- ✓ **measure an auto-parallelizer tool's performance and liability**
- ✓ **provide a simple interface to conduct a comparison between reference code and parallelized code**
- ✓ **provide a multi-platform benchmark using the OpenCL framework**

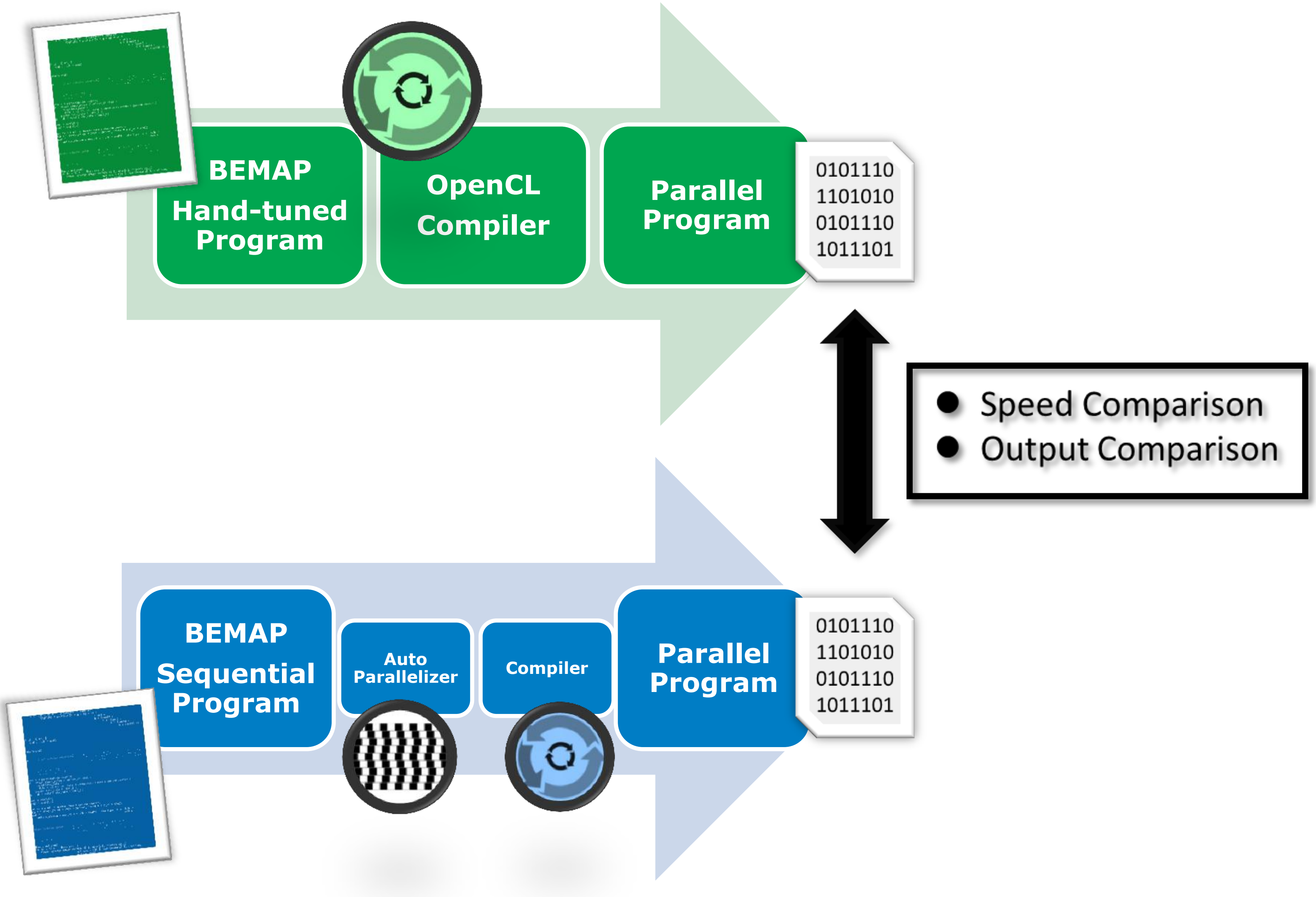
Overview

- A benchmark for parallelizer consisting of reference (single thread) codes and hand-tuned (parallel) OpenCL codes
- An open-source project, can be downloaded from:
  - ✓ **http://sourceforge.net/projects/bemap/**
- Currently consists of 8 algorithms:
  - ✓ **Black-Scholes for European Option**
  - ✓ **Gaussian Blur**
  - ✓ **Grayscale**
  - ✓ **Linear Search**
  - ✓ **Monte-Carlo for European Option**
  - ✓ **Runlength Encoding**
  - ✓ **Backprojection**
  - ✓ **Scale Invariant Feature Transform (SIFT)**

Overview (cont.)

- Has well-optimized OpenCL kernel codes
  - ✓ **Both platform independent and platform dependent OpenCL kernels are provided**
- Optimization methods and benchmark results are well-documented
  - ✓ **http://sourceforge.net/projects/bemap/files/Documentations/**

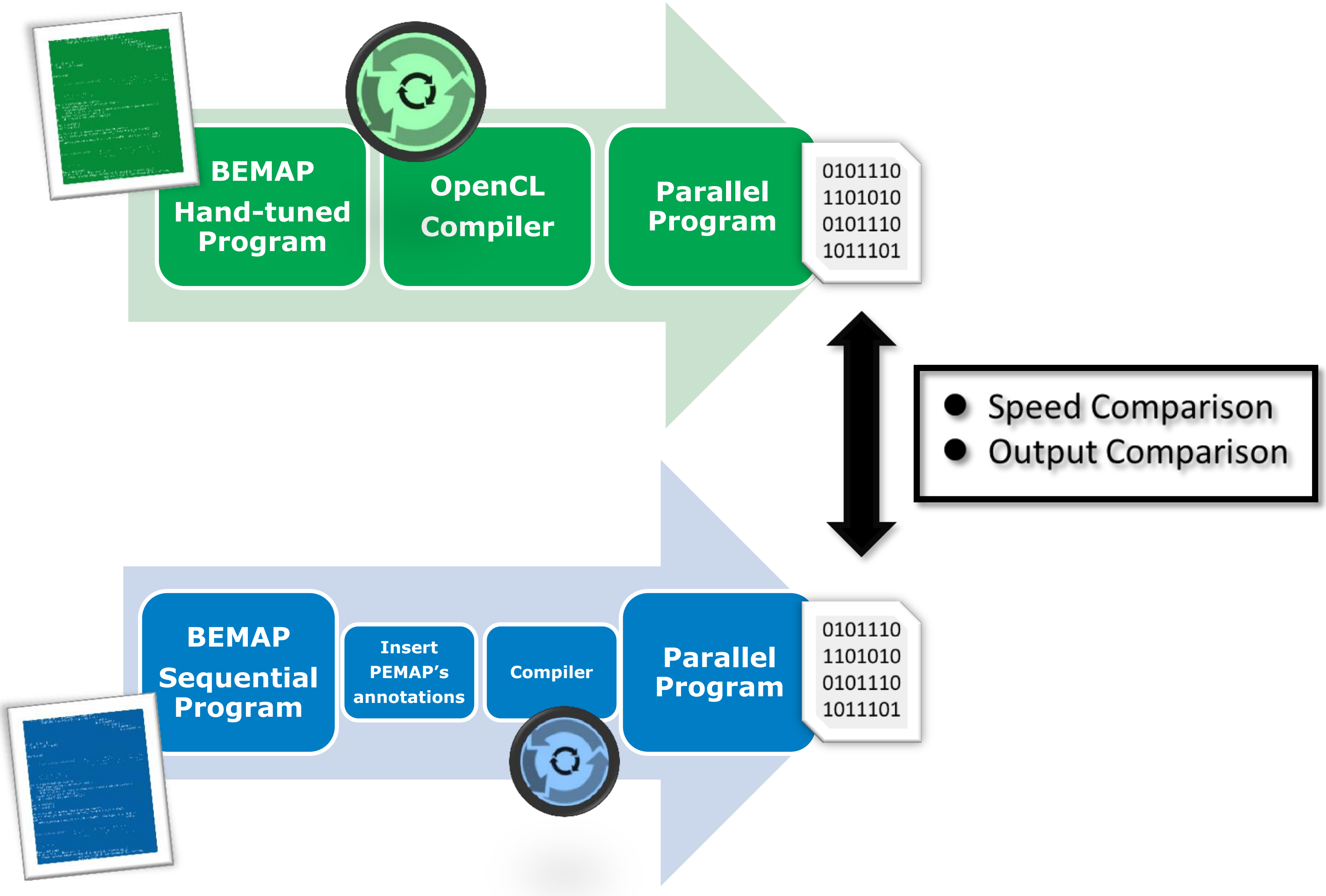
Overview (cont.)



Optimization Methods

- SIMD (explicit vectorization)
  - **Force the compiler to use wide registers to do a single operation for multiple data (e.g. Intel's [X|Y]MM registers)**
- Simple loop-unrolling
- Memory Access
  - **Coalesced, Back-conflicted, Random**
- Caching with shared memory
  - **Programmable cache may boost up the memory transfer performance (DMA architecture)**
- Memory Mapping
  - **Mapped (Pinned) memory gives asynchronous memory access for the host-device communication**
- Native Math Functions
  - **Built-in math functions are provided in some platforms**

BEMAP as a benchmark for PEMAP



BEMAP as a standalone benchmark

Time unit is **milliseconds (ms)**, **using BEMAP's default parameters**

Workload	Ref Plat1	CPUOCL Plat1	GPUOCL Plat1	Ref Plat2	CPUOCL Plat2	GPUOCL Plat2
BlackScholes	6109.07	32.05	1.70	1587.80	21.41	1.40
Gaussian	227.42	1.24	0.39	179.30	1.40	0.27
Grayscale	4.32	0.53	0.06	5.47	0.52	0.07
LinearSearch	26.25	8.42	47.14	33.33	14.24	24.08
MonteCarlo	193817.4	696.53	49.68	27423.70	264.49	41.50
Runlength	518.28	17.07	90.29	370.50	16.77	43.36
Backprojection	11297.43	195.03	75.83	11872.04	116.08	69.70
SIFT	1860.00	194.11	55.54	1452.00	170.51	48.02

RefPlat1 & CPUOCLPlat1:
Intel i7-X990 @ 3.47GHz (Nehalem)
8GB memory, 6 cores, 12 threads (HT)
GPUOCLPlat1:
NVIDIA GTX 570 @ 1.46GHz (Fermi GF-110)
480 CUDA cores, 1.2GB GDDR5 memory

RefPlat2 & CPUOCLPlat2:
Intel Core i7-3770K @ 3.50GHz (Ivy Bridge)
8GB memory, 4 cores, 8 threads (HT)
GPUOCLPlat2:
NVIDIA GTX 680 @ 1.06 GHz (Kepler GK-104)
1536 CUDA cores, 2GB GDDR5 memory

Conclusion

- We proposed two development tools for parallelization.
- PEMAP is an estimation tool of performance increase through parallelization.
  - ✓ **Users of PEMAP need to do nothing other than inserting two annotations.**
- BEMAP is a benchmark suite to assist the development of an auto-parallelizer