To Co-Run, or Not To Co-Run: A Performance Study on Integrated Architectures

Feng Zhang, Jidong Zhai, Wenguang Chen
Tsinghua University, Beijing, 100084, China

Bingsheng He and Shuhao Zhang
Nanyang Technological University, 639798, Singapore

http://pacman.cs.tsinghua.edu.cn

Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

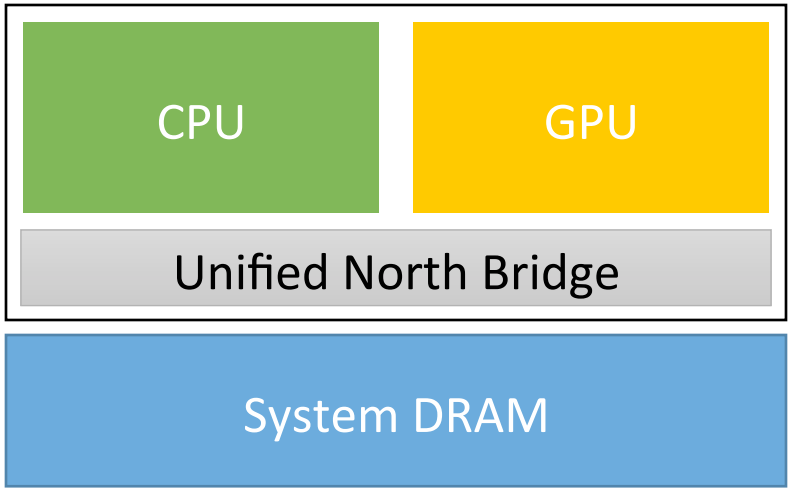
Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

Motivation

2011,
AMD APU

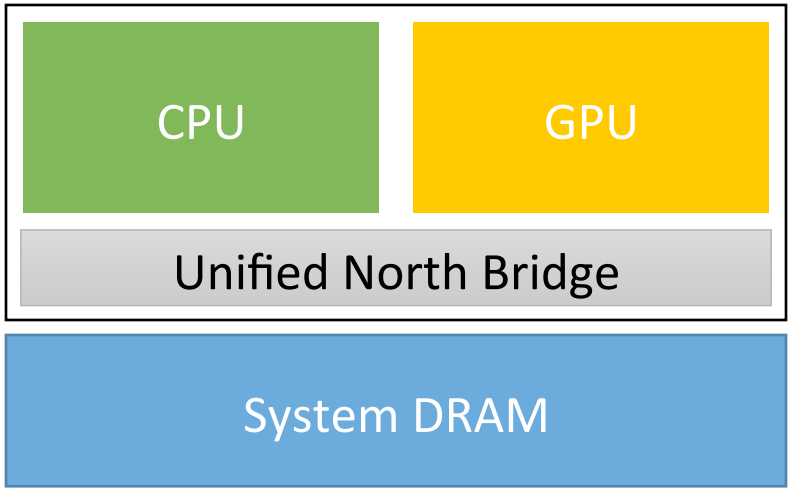


- [VLDB 13] Revisiting Co-processing for Hash Joins on the Coupled CPU-GPU Architecture.
- [ICPP 13] Parallel Radix Sort on the AMD Fusion Accelerated Processing Unit.
- [SC 12] Accelerating MapReduce on a coupled CPU-GPU architecture.

…

Motivation

2011,
AMD APU



- [VLDB 13] Revisiting Co-processing for Hash Joins on the Coupled CPU-GPU Architecture.
- [ICPP 13] Parallel Radix Sort on the AMD Fusion Accelerated Processing Unit.
- [SC 12] Accelerating MapReduce on a coupled CPU-GPU architecture.

…

Limitation: Few studies have been performed to analyze a wide range of applications on such architectures. It remains unclear which programs can benefit from co-running on integrated architectures.

Motivation

- Can all programs benefit from co-running on integrated architectures?

PACMAN

http://pacman.cs.tsinghua.edu.cn

Motivation

- Can all programs benefit from co-running on integrated architectures?

  - **To Co-Run, or Not To Co-Run: A Performance Study on Integrated Architectures**

PACMAN

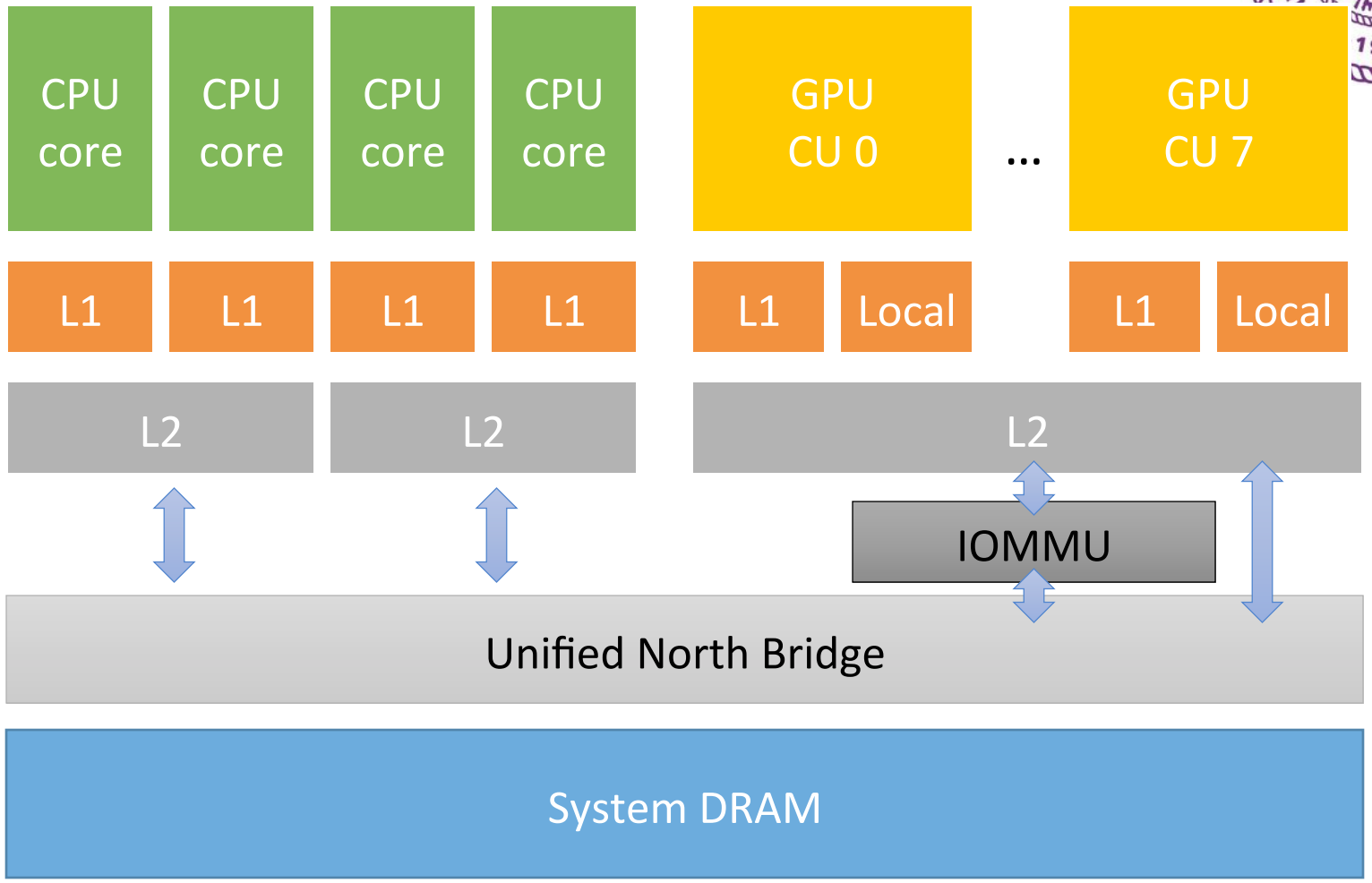
http://pacman.cs.tsinghua.edu.cn

Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

Background

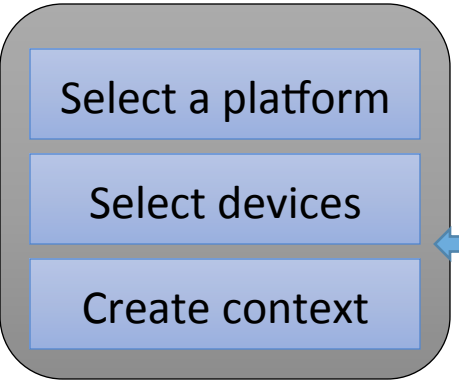


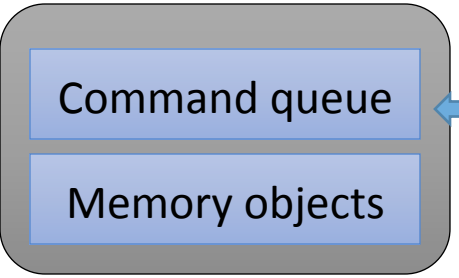
AMD integrated architecture (AMD A10-7850K)

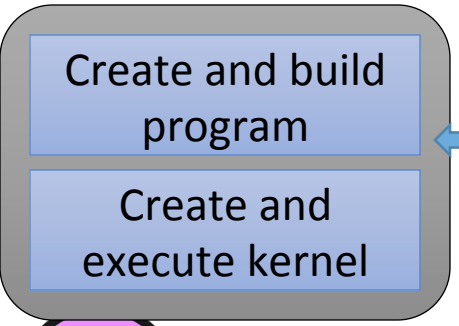
PACMAN

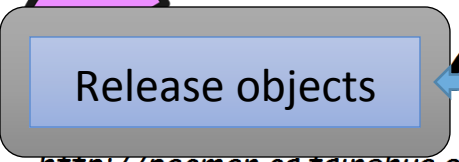
http://pacman.cs.tsinghua.edu.cn

Background









Start

```
int main(int argc, char *argv[])
{
...
```

Initialize devices

```
errcode = clGetPlatformIDs(1, &platform_id, &num_platforms);
errcode = clGetDeviceIDs(platform_id, CL_DEVICE_TYPE_GPU, 1,
&device_id, &num_devices);
clGPUContext = clCreateContext(NULL, 1, &device_id, NULL, NULL,
&errcode);
```

Copy data to devices

```
a_mem_obj = clCreateBuffer(clGPUContext, CL_MEM_READ_ONLY,
sizeof(DATA_TYPE) * NI * NJ, NULL, &errcode);
errcode = clEnqueueWriteBuffer(clCommandQue, a_mem_obj,
CL_TRUE, 0, sizeof(DATA_TYPE) * NI * NJ, A, 0, NULL, NULL);
```

Execute kernels on devices

```
clProgram = clCreateProgramWithSource(clGPUContext, 1, (const
char **)&source_str, (const size_t *)&source_size, &errcode);
errcode = clBuildProgram(clProgram, 1, &device_id, NULL, NULL,
NULL);
clKernel = clCreateKernel(clProgram, "Convolution2D_kernel",
&errcode);
errcode =  clSetKernelArg(clKernel, 0, sizeof(cl_mem), (void
*)&a_mem_obj);
errcode = clEnqueueNDRangeKernel(clCommandQue, clKernel, 2,
NULL, globalWorkSize, localWorkSize, 0, NULL, NULL);
```

Copy data back

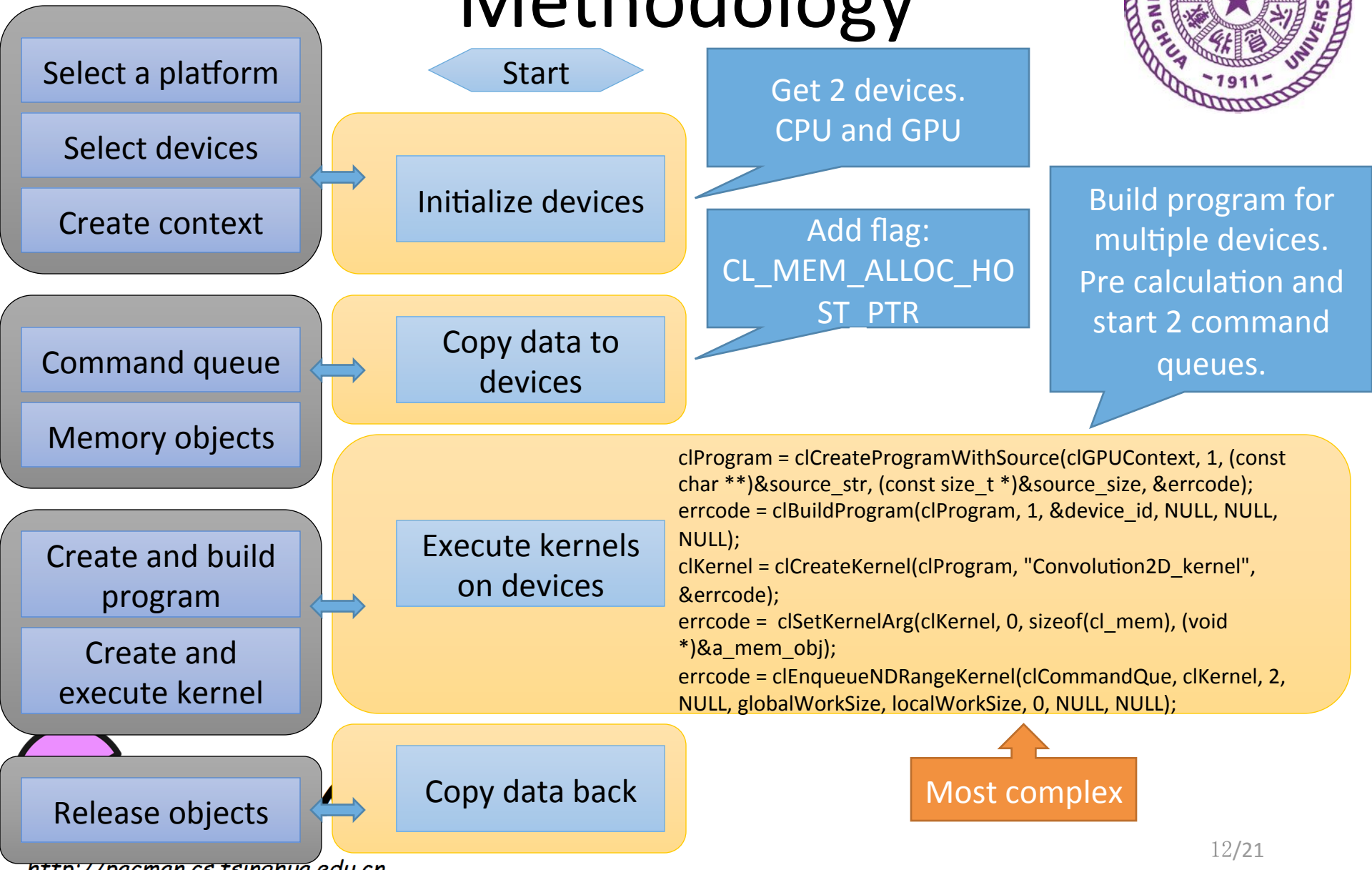
```
errcode = clEnqueueReadBuffer(clCommandQue, b_mem_obj,
CL_TRUE, 0, NI*NJ*sizeof(DATA_TYPE), B_outputFromGpu, 0, NULL,
NULL);
...
}
```

Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

Methodology



Methodology

Initialize devices

Copy data to devices

```
clProgram = clCreateProgramWithSource(clGPUContext, 1, (const char **)&source_str, (const size_t *)&source_size, &errcode);
errcode = clBuildProgram(clProgram, 0,0, NULL, NULL, NULL);
clKernel = clCreateKernel(clProgram, "Convolution2D_kernel", &errcode);
errcode =  clSetKernelArg(clKernel, 0, sizeof(cl_mem), (void *)&a_mem_obj);
Calculate workload partitions and set number of work-itmes;
errcode = clEnqueueNDRangeKernel_fusion(clCommandQue1, clKernel, …);
```

Copy data back

cl_int clEnqueueNDRangeKernel(…);

cl_int clEnqueueNDRangeKernel_fusion (…){

//Step 1: Calculate the number of work-items for CPUs and GPUs and get the offset.

```
gpu_global_work_size = compute_gpu_global_size(…);
cpu_global_work_size = compute_cpu_global_size(…);
global_work_offset = compute_global_offset(…);
```

//Step 2: If the related number of work-items is not zero, launch the kernels for CPUs and GPUs.

```
if(GPU_RUN) clEnqueueNDRangeKernel (…);
if(CPU_RUN) clEnqueueNDRangeKernel (…);
if(GPU_RUN) clFlush(gpu_command_queue);
if(CPU_RUN) clFlush(cpu_command_queue);
```

//Step 3: Synchronization.

```
if(CPU_RUN)   error|=clWaitForEvents (&cpu_event);
if(GPU_RUN)   error|=clWaitForEvents (&gpu_event);
```

}

Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

Experiment

• Platform

Platform	AMD A10-7850K
Operating System	Ubuntu 13.10
CPU Peak FLOPS	118.4 Gflops/s
Number of CPU cores	4
CPU frequency	3.7 GHz
GPU Peak FLOPS	737.28 Gflops/s
GPU Max frequency	0.72GHz
Memory Size	32 GB
Memory Frequency	1600 MHz
Max Bandwidth	25.6 GB/s

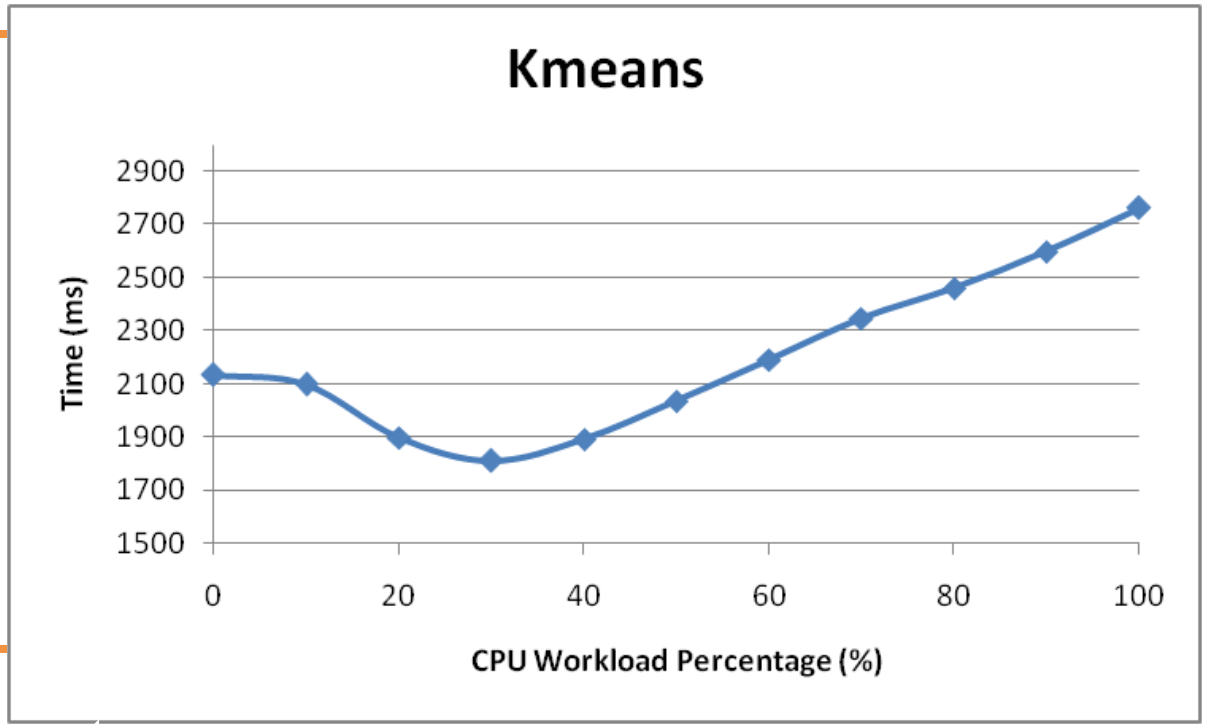
• Programs (Rodinia)

Applications	Dwarves	Domains
Leukocyte	Structured Grid	Medical Imaging
Heart Wall	Structured Grid	Medical Imaging
CFD Solver1	Unstructured Grid	Fluid Dynamics
LU Decomposition	Dense Linear Algebra	Linear Algebra
HotSpot	Structured Grid	Physics Simulation
Back Propagation	Unstructured Grid	Pattern Recognition
Needleman-Wunsch	Dynamic Programming	Bioinformatics
Kmeans	Dense Linear Algebra	Data Mining
Breadth-First Search1	Graph Traversal	Graph Algorithms
SRAD	Structured Grid	Image Processing
Streamcluster1	Dense Linear Algebra	Data Mining
Particle Filter	Structured Grid	Medical Imaging
PathFinder	Dynamic Programming	Grid Traversal
Gaussian Elimination	Dense Linear Algebra	Linear Algebra
k-Nearest Neighbors	Dense Linear Algebra	Data Mining
LavaMD2	N-Body	Molecular Dynamics
Myocyte	Structured Grid	Biological Simulation
B+ Tree	Graph Traversal	Search
GPUDWT	Spectral Method	Image/Video Compression
Hybrid Sort	Sorting	Sorting Algorithms

http://pacman.cs.tsinghua.edu.cn

Experiment

Program running time



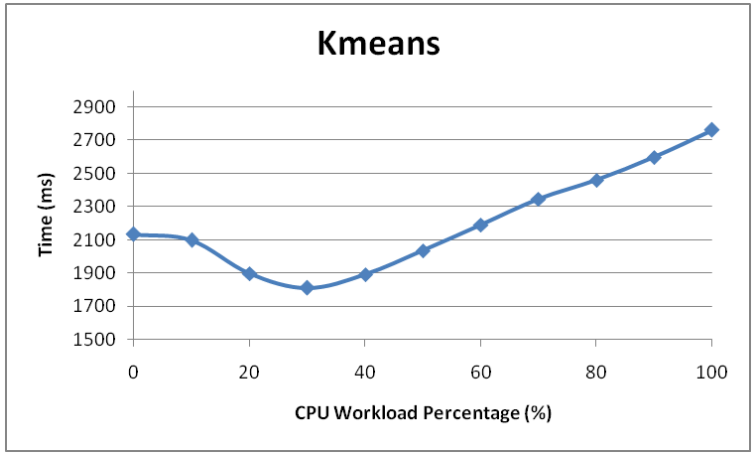
More GPU workload

More CPU workload

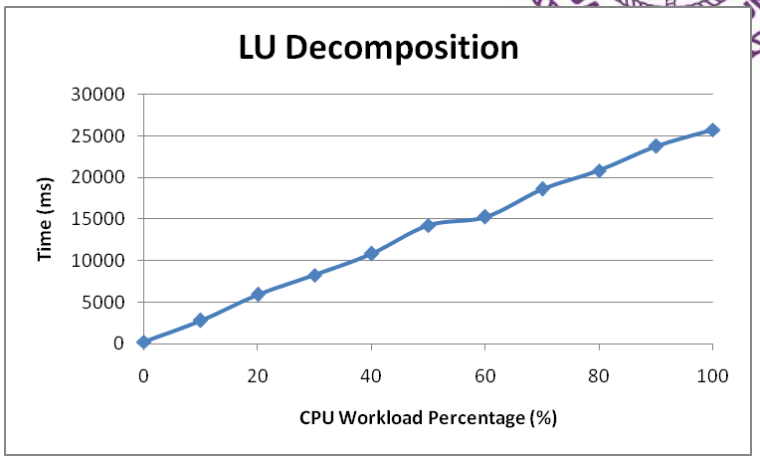
PACMAN

http://pacman.cs.tsinghua.edu.cn

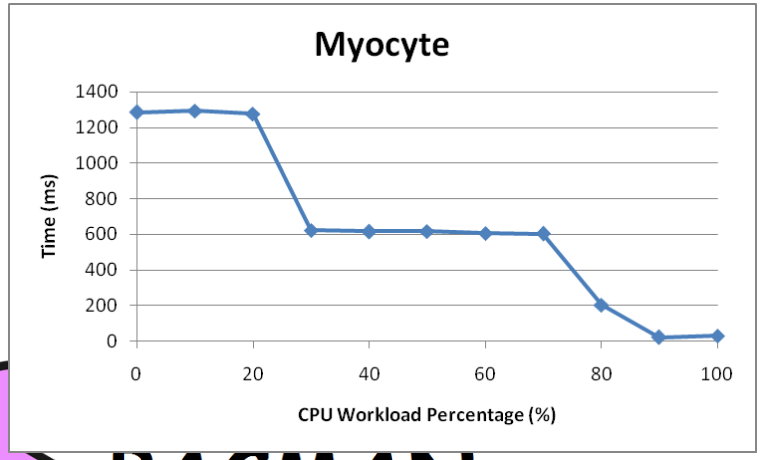
Experiment



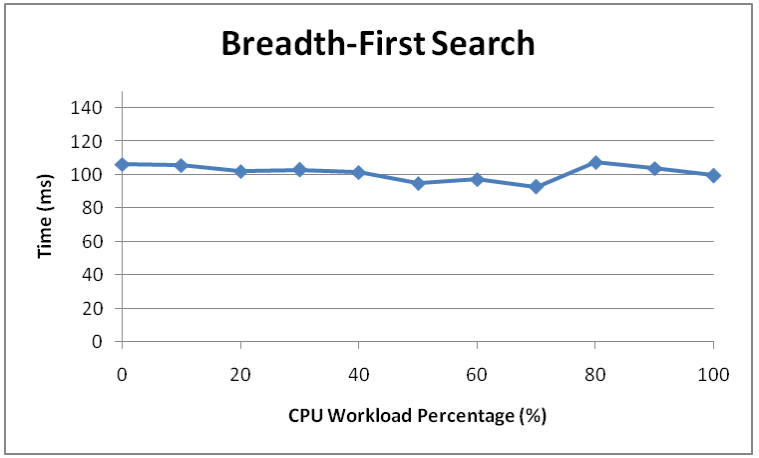
Co-run friendly program



GPU-dominant program



CPU-dominant program



Performance similar program

Experiment

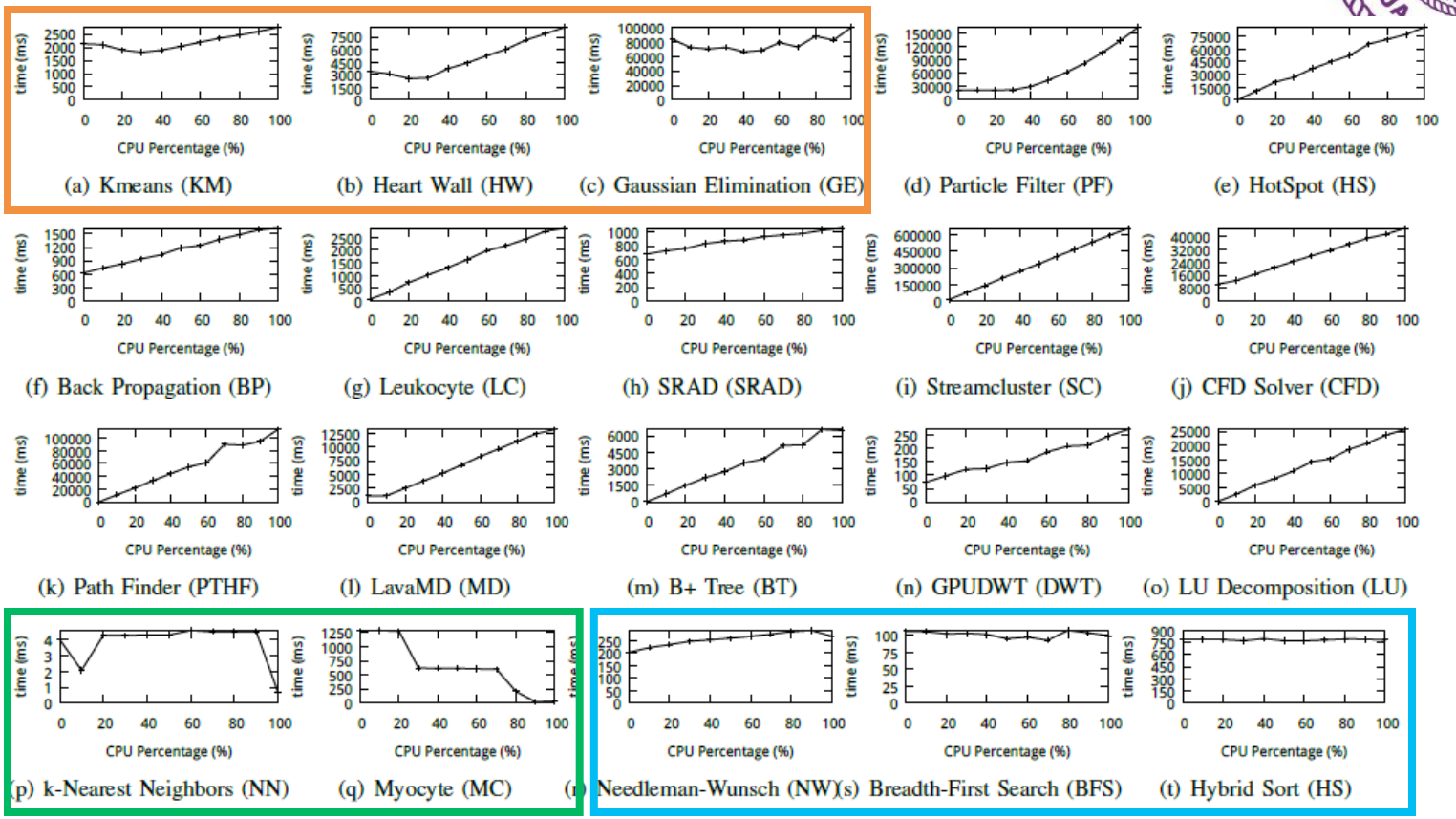


Fig. 3. Execution time for the Rodinia benchmark suite. Co-run-friendly programs are (a, b, c), GPU-dominant programs are (d, e, f, g, h, i, j, k, l, m, n, o), CPU-dominant programs are (p, q), and performance similar programs are (r, s, t).

PACMAN

http://pacman.cs.tsinghua.edu.cn

Outline

- Motivation
- Background
- Methodology
- Experiment
- Conclusion

http://pacman.cs.tsinghua.edu.cn

Conclusion

- Finding 1: Most programs using local memory are GPU-dominant.
- Finding 2: Programs with large number of threads are more likely to be GPU-dominant.
- Finding 3: Co-run friendly programs usually have long kernel execution time.
- Finding 4: Programs having low memory bandwidth tend to be co-run friendly.

http://pacman.cs.tsinghua.edu.cn

Not all programs can benefit from co-running, and more research is needed for this promising architecture.

Q&A

Thank you!

PACMAN

http://pacman.cs.tsinghua.edu.cn