Facilitating Irregular Applications on Many-core Processors

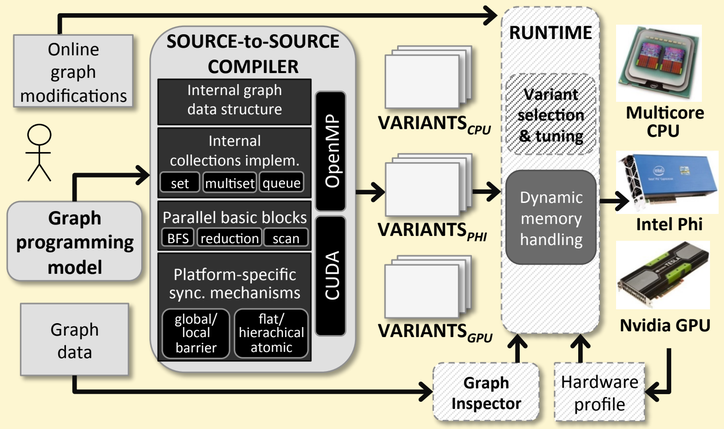
Da Li, Michela Becchi
University of Missouri

Contributions

Many-core processors are considered relatively difficult to program, in that they require the programmer to be familiar with both parallel programming and the hardware features of these devices. Irregular applications are characterized by irregular and unpredictable memory access patterns, frequent control flow divergence, and runtime (rather than compile time) parallelism. My research focuses on addressing important issues related to the deployment of irregular computations on many-core processors. My contributions are in three directions:

- Unifying programming interfaces for many-core processors
- Runtime support for efficient execution of applications on irregular datasets
- Compiler support for efficient mapping of applications onto hardware

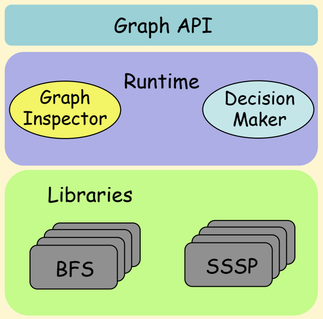
Unifying Programming Interfaces

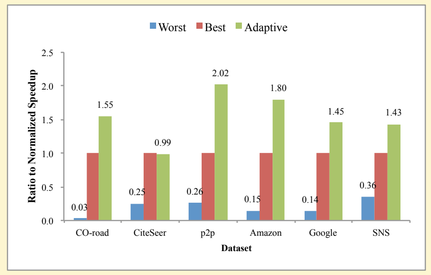


Application	Application & Graph Type	Attributes	Comp. pattern & arith. intensity	Best Speedup		
				m-CPU	GPU	Phi
BFS	read-only static	Level (int)	Set level (simple& low)	2.5x	50x	3.5x
PageRank	read-only static/dynamic	Rank (double)	Calculate Rank (intermediate & intermediate)	6.3x	6.5x	9x
A-DFA	read-only static	Default Trans (int)	Compare trans. (complex, low)	8x	6x	45x
DFA construction	read-write dynamic	Trans Table (int)	Compare Subset (complex, low)	6.5x	5.5x	4.3x

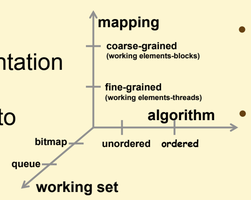
- Platform selection depends on application and graph type
- Unified programming interfaces facilitates fast-prototyping

Adaptive Computing





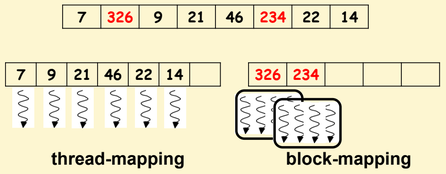
3-d exploration space:
- working set representation
- algorithm
- mapping method onto hardware



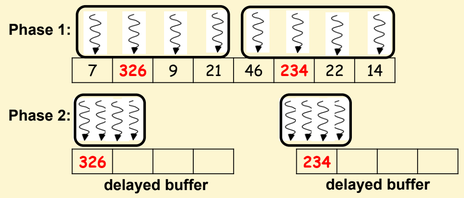
- Performance of static solution varies from dataset to dataset
- Adaptive solution outperforms static ones

Parallelization Templates

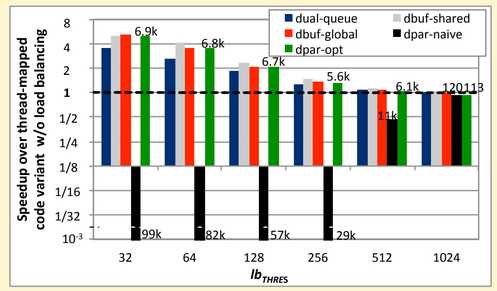
dual-queue



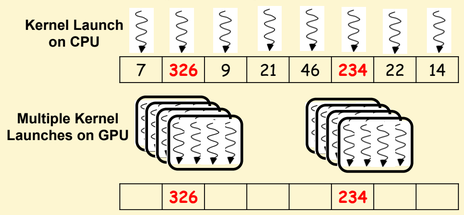
delayed-buffer (global/shared)



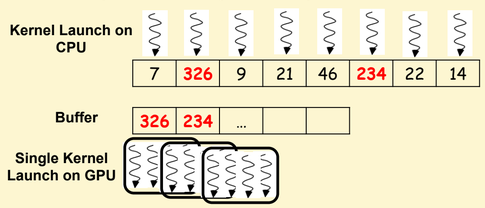
SSSP



naïve dynamic parallelism



optimized dynamic parallelism



- Naïve dynamic parallelism incurs significant overhead, thus degrading the performance
- lb_{THRES} (load balancing threshold) also affects the performance

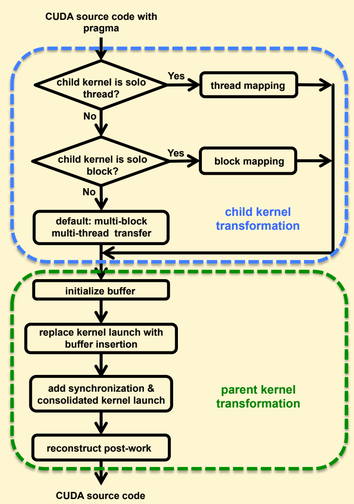
Workload Consolidation

Annotated CUDA source code

```
__global__ void parent_kernel() {
  work_item = get_work_item(…)
  prework(work_item)
  if (condition) {
  #pragma dp consldt(block) buffer(default, 256) work(work_item)
    child_kernel<<<block_dim, thread_dim>>>(…, work_item, …)
  } else  work(work_item)
  postwork(work_item)
}
```

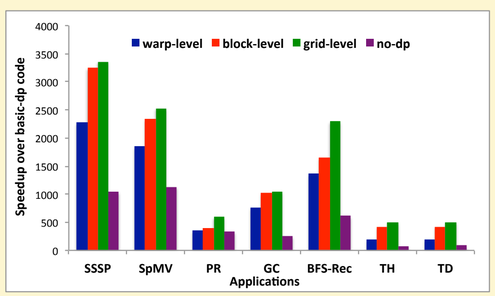
Generated CUDA kernel

```
__global__ void parent_kernel() {
  work_item = get_work_item(…);
  prework(work_item)
  if (condition)  insert_buffer(curr)
  else  work(work_item)
  synchronize
  if (thread_id==selected)
    child_kernel_consolidate<<<b_dim_con, t_dim_con>>>()
  synchronize
  postwork(work_item)
}
```



#pragma dp consldt(type) buffer([type, perBufferSize, totalSize]) work(varlist)

Performance



- Compiler-assisted workload consolidation can improve the performance significantly
- Grid-level consolidation achieves better performance than warp- and block-level consolidation

Publications

[1] Da Li, Michela Becchi, “Software Support for Regular and Irregular Applications in Parallel Computing”, SC 2012.
[2] Da Li, Michela Becchi, “Deploying Graph Algorithms on GPUs: an Adaptive Solution”, IPDPS 2013.
[3] Da Li, Srimat Chakradhar, Michela Becchi, “Grapid: a Compilation and Runtime Framework for Rapid Prototyping of Graph Applications on Many-core Processors”, ICPADS 2014.
[4] Da Li, Michela Becchi, “Designing Code Variants for Applications with Nested Parallelism on GPUs”, GPU Technology Conference 2015.
[5] Da Li, Hancheng Wu, Michela Becchi, “Nested Parallelism on GPU: Exploring Parallelization Templates for Irregular Loops and Recursive Computations”, ICPP 2015.
[6] Da Li, Hancheng Wu, Michela Becchi, “Exploiting Dynamic Parallelism to Efficiently Support Irregular Nested Loops on GPUs”, COSMIC 2015.

This work has been supported by NSF awards CNS-1216756 and CCF-1452454 and by equipment donations from Nvidia Corporation.

Contact us:
da.li@mail.missouri.edu