BDSC-Based Automatic Task Parallelization: Experiments

Dounia KHALDI

CRI, Mathématiques et systèmes
MINES ParisTech

Septièmes rencontres de la communauté française de compilation,
Dammarie-les-Lys, France
December 04, 2013

Context and Motivation

- *Anyone can build a fast CPU. The trick is to build a fast system.* Attributed to Seymour Cray
- Parallelism handling:
  - Parallel software developed by converting sequential programs by hand
  - Automatic task parallelism extraction: **Scheduling** problem
  - Resource constraints: memory requirements, processor features...
  - Scientific, signal and image processing benchmarks

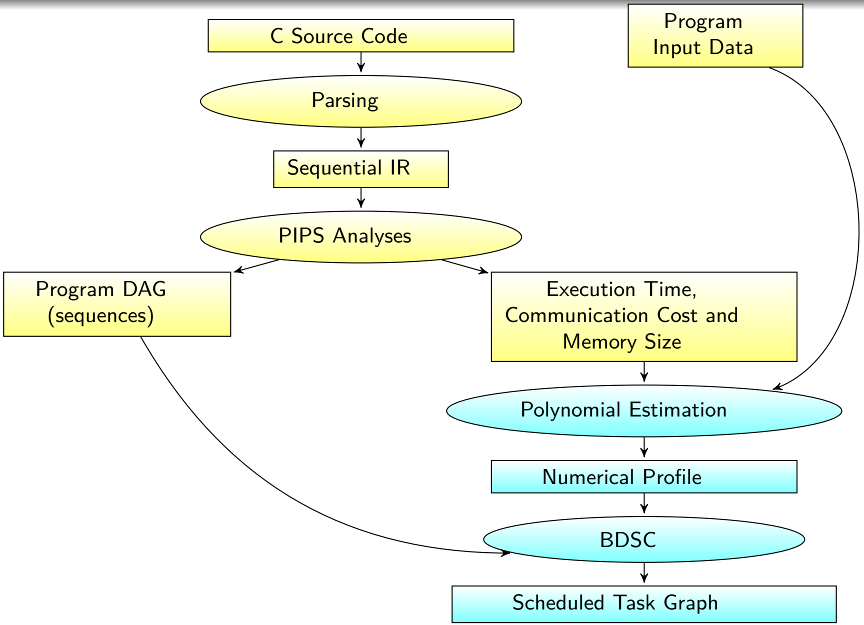
Automatic Resource-Constrained Static Task Parallelization

- ***BDSC****: a memory-constrained, number of processor-bounded extension of DSC*
- *Experimentation on shared and distributed memory systems*

Contents

Parallelization Process

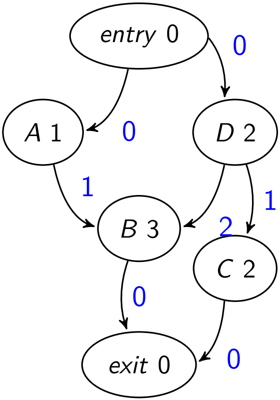
blue indicates contributions; an ellipse, a process; and a rectangle, results



Contents

List-Scheduling Heuristics

- Priorities are computed for all unscheduled vertices using:
  - Top level (tlevel(τ)): length of the longest path from entry to τ
  - Bottom level (blevel(τ)): length of the longest path from τ to exit

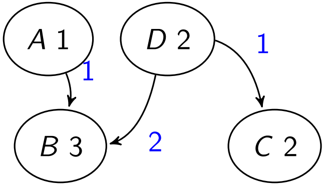


task	tlevel	blevel
D	0	7
C	3	2
A	0	5
B	4	3

- Vertex τ with the highest priority is selected for scheduling
- τ is added to the cluster (logical process) with the earliest start-time

A List-Scheduling Heuristic: Dominant Sequence Clustering (DSC)

- DSC (Dominant Sequence Clustering) [Yang and Gerasoulis 1994]
- Task list-scheduling heuristic for an unbounded number of clusters
- priority(τ) = tlevel(τ) + blevel(τ)
- zeroing(τ_p, τ) puts τ in the cluster of a predecessor τ_p $\Rightarrow$ reduces tlevel(τ) by setting to zero the cost of the edge (τ_p, τ)



step	task	tlevel	blevel	prio	scheduled tlevel		
					κ_0	κ_1	κ_2
1	*D*	0	7	7	0*		
2	*C*	3	2	5	2	3*	
3	*A*	0	5	5			0*
4	*B*	4	3	7	2*		4

- Complexity: $\mathcal{O}(n^2 log(n))$

A List-Scheduling Heuristic: Dominant Sequence Clustering (DSC)

- DSC algorithm weaknesses for our purpose:
  - Unbounded number of clusters
  - Number of clusters is not predefined $\rightarrow$ blind clustering
  - Memory size is not predefined $\rightarrow$ blind clustering
  - Creates long idle slots in already existing clusters

Proposal

BDSC: A Memory-Constrained, Number of Processor-Bounded Extension of DSC

Bounded DSC: Resource Constraint Warranty

1. Memory Constraint Warranty (MCW):
   - Do not exceed a memory threshold M
   - Overapproximation of the amount of memory used in tasks
   - data_size(cluster_data (κ) $\cup$ task_data (τ)) $\leq$ M
2. Bounded number of clusters P:
   - Number of cluster allocations do not exceed Threshold P
   - Maintain the constraint MCW
   - $\text{argmin}_{k \in clusters}$ cluster_time(κ)
3. Efficient cluster allocation by exploiting idle slots
4. Complexity: $\mathcal{O}(n^3)$

Related Work: Static Task Parallelization Tools

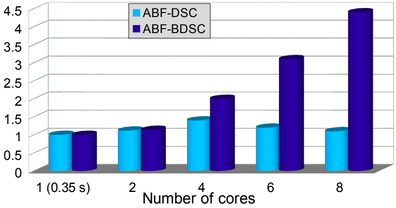
	blevel	tlevel	Resource constraints	Dependence control	Dependence data	Execution time estimation	Communication time estimation	Memory model
BDSC Parallelization	√	√	√		√	√	√	Shared, distributed
Sarkar's work [Sarkar, 1989]	√				√	√	√	Shared, distributed
OSCAR [Kasahara et al., 1992]		√		√	√	√		Shared
Pedigree [Newburn and Shen, 1996]	√	√		√	√	√		Shared
SPIR [Choi et al., 2009]		√	√	√	√	√	√	Shared

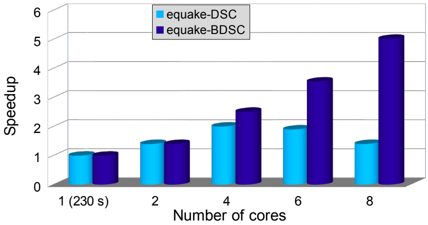
Contents

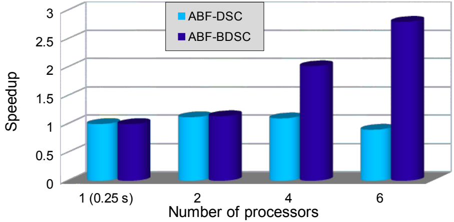
Experimental Setting

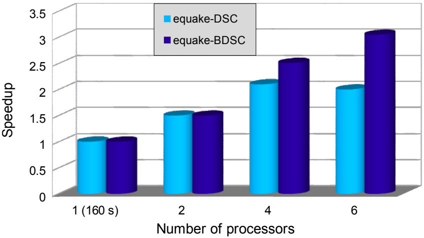
1. **Benchmarks**
   - *Thales ABF* (Adaptive Beam Forming), with 1,065 lines
   - SPEC benchmark *equake*, with 1,432 lines
   - *Harris* corner detector, with 105 lines
   - NAS Parallel Benchmark *IS* (Integer Sort), with 1,076 lines
2. **Machines**
   - **Shared Memory**: host Linux (Ubuntu)
     2-socket AMD quadcore Opteron, 2.4 GHz
     $M = 16$GB of RAM
     gcc 4.6.3 -O3
     OpenMP 3.0
     Cluster $\sim$ Thread
   - **Distributed Memory**: host Linux (RedHat)
     6 dual-core processors Intel® Xeon®, 2.5 GHz
     $M = 32$GB of RAM per processor
     gcc 4.4.6 -O3
     Open MPI 1.6.2
     Cluster $\sim$ Process

ABF and equake
Speedups with OpenMP and MPI

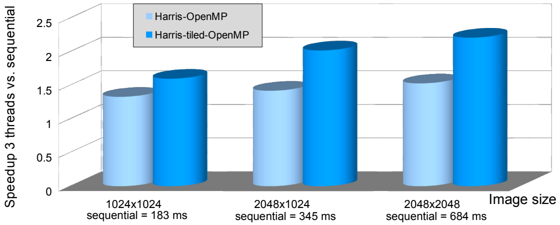


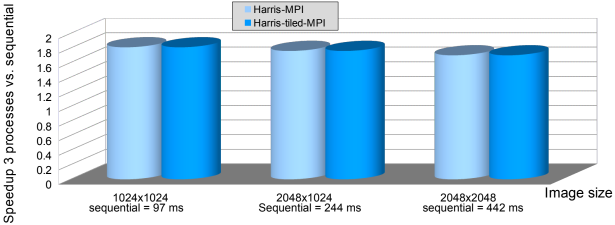




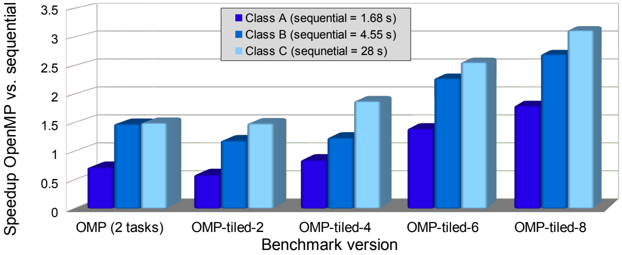


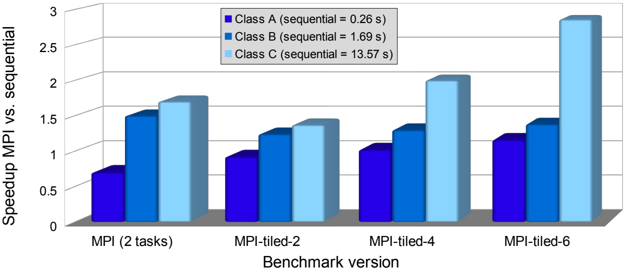
Harris
Speedups with OpenMP and MPI: Impact of Tiling (P=3)





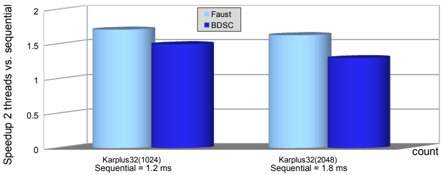
NAS Parallel Benchmark IS
Speedups with OpenMP and MPI: Different Class Sizes

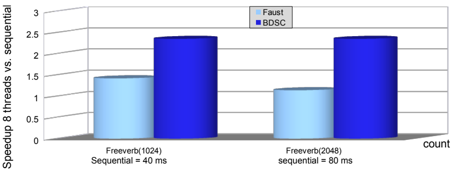




Faust Parallel Scheduling vs. BDSC

- Faust (Functional AUdio STream) [Orlarey et al., 2009]
- DSL for real-time audio signal processing and synthesis
- Generation of C or C++ with or without OpenMP directives
- `omp task` (BDSC) vs. `omp section` (Faust Parallelizing Compiler)
- Scheduling: BDSC vs. Faust topological ordering
- Speedups for two programs: Karplus32 and Freeverb





Contents

Conclusion

1. **BDSC-based hierarchical scheduling algorithm**
   - Memory constraint, bounded number of clusters, efficient cluster allocation
   - BDSC-based task parallelization algorithm
   - Communication, data and time cost models

2. **Experiments:**
   - BDSC-based automatic parallelization in PIPS
   - Code generation in OpenMP and MPI
   - Good speedups for coarse-grained parallelism

Future Work

1. **BDSC Scheduling**
   - Handling of heterogeneous devices
   - More precise cost models

2. **Parallel Code Generation**
   - More experimentation needed
   - Solving communication generation problems (MPI)
   - Hybrid task + data parallelism

BDSC-Based Automatic Task Parallelization: Experiments

Dounia KHALDI

CRI, Mathématiques et systèmes
MINES ParisTech

Septièmes rencontres de la communauté française de compilation,
Dammarie-les-Lys, France
December 04, 2013

References I

Choi, Y., Lin, Y., Chong, N., Mahlke, S., and Mudge, T. (2009).
Stream Compilation for Real-Time Embedded Multicore Systems.
In *Proceedings of the 7th annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '09, pages 210–220, Washington, DC, USA.

Kasahara, H., Honda, H., Mogi, A., Ogura, A., Fujiwara, K., and Narita, S. (1992).
A Multi-Grain Parallelizing Compilation Scheme for OSCAR (Optimally Scheduled Advanced Multiprocessor).
In *Proceedings of the Fourth International Workshop on Languages and Compilers for Parallel Computing*, pages 283–297, London, UK. Springer-Verlag.

Nandivada, V. K., Shirako, J., Zhao, J., and Sarkar, V. (2013).
A Transformation Framework for Optimizing Task-Parallel Programs.
ACM Trans. Program. Lang. Syst., 35(1):3:1–3:48.

Newburn, C. J. and Shen, J. P. (1996).
Automatic Partitioning of Signal Processing Programs for Symmetric Multiprocessors.
In *IEEE PACT*, pages 269–280. IEEE Computer Society.

Sarkar, V. (1989).
Partitioning and Scheduling Parallel Programs for Multiprocessors.
MIT Press, Cambridge, MA, USA.