Aftermath:

Interactive Visualization of Cross-Layer Performance Anomalies in **Dynamic Task-Parallel Applications and Systems**

Andi Drebes

The University of Manchester School of Computer Science Advanced Processor Technologies andi.drebes@manchester.ac.uk

Joint work with: Antoniu Pop, Karine Heydemann, Albert Cohen

ISPASS 2016











Task-Parallel Programming



Principles of task-parallel programming

- Small units of work (tasks)
- Inter-task data dependences
- Expose large amounts of parallelism
- Efficient execution by a run-time system
- OpenMP 4, StarSs, OpenStream, ...



- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis



- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis



- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis



- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis

- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis

Performance analysis difficult

- Every component complex in itself
- Lots of complex interactions
- Performane anomalies? Which component(s)?
- Relation to execution model: Task-centric analysis

Aftermath: Make things easier with the right tool

Andi Drebes - Performance Analysis of Task-Parallel Applications

- Trace-based Analysis
- Overview of Aftermath
- 🔵 Demo
- Conclusion

Trace-Based Analysis

Trace data

- Captured during execution
- Run-time / hardware / application events
- Event types
 - Discrete events
 - State changes
 - Counter values
- Dump to trace file for post-mortem analysis

Support for Dependent Tasks

Support for Dependent Tasks

Data-flow Analysis

Data-flow Analysis

Cross-Layer Analysis

Data-flow Analysis

Cross-Layer Analysis

Support for NUMA

Support for Dependent Tasks

Data-flow Analysis

Cross-Layer Analysis

Support for NUMA

Powerful Filters

Support for Dependent Tasks

Data-flow Analysis

Cross-Layer Analysis

Support for NUMA

Powerful Filters

Reactive UI for large Files

Task execution (dark blue)

Task creation (white)

Searching for work (light blue)

-4	Aftermath - erents.stost	* • 0 X
Fie Data Edit View Help		
[🛃 163 🔍 164 🌆 🎆	2 🖼 🗃 💷 💷 💷 💷 🚾 📂 🗠 🖓 (1) 🛃	
Tasks Counters Frames	Graph Code	Statistics Heatmap
Task types		4879258384 - 22432062703
Address Symbol		
2 0x0 (no symbol found)		4.30 Mcycles / task (avg)
2 0x4019f0 (no symbol found)		Task length distribution
X 0x401b50 (no symbol found		×.
Octobread (no symbol found)		94
Ox4025e0 (no symbol found		
Ox402760 (no symbol found		5
2 0x4028c0 (no symbol found		123.50 K 125.46 M
R 0x402aa0 (no symbol found		Communication
R 0x408e40 (no symbol found		
2 0x408fb0 (no symbol found)		
2 0x409240 (no symbol found		Nodes O CPUs detach
- Select none		
		234.22 G local 89.06% 28.77 G remote local
Select all		RRWESEP
Writes to NUMA nodes		Ignore direction
Node		Exclude reflexive transfers
R Node 0		Rel. min:
😢 Node 1		Rel. max:
V Novia 2		
At least 0 8		Cyc Par
- Select core		Seeking 2.25% 1.44
- Succession		Tcreate: 0.17% 0.11
Select all		Resdep: 0.16% 0.10
Length		Tdec: 0.54% 0.35
Filter tasks by length		Init: 0.29% 0.19
Min	Selected avent Artive task In sever selected	Est. cost: 0.00% 0.00
Max		Reorder: 0.00% 0.00
apply		Select from graph
CBIL 21, ctate 1 (tackeyer) from 1323	UNISSING IN 13331756539. duration: 7-70 Microles. active task: 0x401x4cm (incl)	

Heatmap indicating task duration (white: short, red: long)

NUMA heatmap (blue: local accesses, pink: remote accesses)

Basic statistics for run-time states

Histogram for task duration + NUMA statistics

Filters: task type / duration, CPUs, event types,

14			Aftermath - e	vents.dt.ost			
Fie Data Edit View Help			1 53				
15 m of m m							
Tasks Counters Frames	Graph Code					Statistics Heatma	ID
Address Symbol * V Dx0 (no symbol found V Dx4019a0 (no symbol found	CRU A					4879258384 - 2 17.55 Gcycles sel 254913 tasks con 4.30 Mcycles / tas 254759 task crea	2432062703 — .: ected sidered ik (avg) tion events
Dx401910 Ino symbol found Dx401b50 (no symbol found Dx401b50 (no symbol found Dx401cd0 (no symbol found Dx4024d0 (no symbol found	005 007 007 007 007					Task length dist	ribution %15.97
Ox4025e0 ino symbol found Ox4025e0 ino symbol found Ox4028c0 ino symbol found Ox4028c0 ino symbol found						123.50 K	125.46 M
Ox402c80 (no symbol found Ox407b30 (no symbol found	CPU 13						
R 0x408e40 (no symbol found	00.0						
Ox408tb0 (no symbol found Ox409240 (no symbol found e	CPU 19 1941 30 CPU 31					Nodes CPU	s detach
- Select none	00011					234.22 G local 28.77 G remote	89.06% local
	09.8					RRWISC	P
Writes to NUMA nodes	00.8					Ignore direction	
Node	0.00						
R Node 0	00.16					Rel. min:	
✓ Node 1	00.00					Rel. max:	
V Nevto 2	09931						
At least 0 B	00.31						Cor Bar
	0.00					Seeking	2.25% 1.44
 Select none 	09.11					Texec:	96.56% 61.80
💠 Select all						Tcreate:	0.17% 0.11
Longth						Tdec:	0.54% 0.35
						Bcast:	0.01% 0.01
rinter casks cyliterigth	Selected en	trav		Active task		Init:	0.29% 0.19
Max	CPU: State	7 1/taskever1		Active task: 0x405fc0 (nul) Task duration: 3.42 M	Ê	Est. cost: Reorder:	0.00% 0.00
	From Duration:	4630388060 to 4633548496 3.16 Mcycles		Active frame: 0x7fbdf0d8b080		Select from	n graph
	469158 to 122	22450295 duration 2.99 Mourles, active task (0x40920.0x40				

Detailed task view: duration, dependences, accessed NUMA nodes

Current Iteration

Previous Iteration

Current Block

Current Iteration

Previous Iteration

Current Block

OpenStream implementation

Current Iteration

Previous Iteration

Current Block

OpenStream implementation

Current Iteration

Previous Iteration

Current Block

OpenStream implementation

Benchmark parameters

- Double precision floats
- ▶ Matrix: 2¹⁴ × 2¹⁴ (2 GiB)
- ▶ Block: 2⁸ × 2⁸ (512 KiB)
- 60 iterations
- 254,977 tasks

Current Iteration

Previous Iteration

😳 Current Block

OpenStream implementation

Benchmark parameters

- Double precision floats
- ▶ Matrix: 2¹⁴ × 2¹⁴ (2 GiB)
- ▶ Block: 2⁸ × 2⁸ (512 KiB)
- 60 iterations
- 254,977 tasks

Test system

- ▶ SGI UV 2000
- ► 24×Intel Xeon E5-4640
- Hyperthreading disabled
- ▶ 192 cores
- 24 NUMA nodes, 756 GiB RAM

DEMO

1. Default, random work-stealing 2. Run-time optimized for NUMA

Impact on OpenStream

Run-time development & performance debugging

- NUMA-aware task and data placement
- Topology-aware work-stealing
- Overhead analysis
 - Memory allocation
 - Task creation

Application development & performance debugging

- Task granularity / available parallelism
- Partitioning: main tasks / auxiliary tasks
- Task creation
- Dependence patterns
- Hardware performance counter analysis

Model-Centric Analysis of OpenMP Programs*

Analysis of OpenMP programs

- Parallel loops
 - Examine Partitioning
 - Execution time per worker
 - Filtering
- OpenMP 4 tasks
- OpenMP constructs: Barriers, single, ...
- Hardware performance counters

* Joint work with Jean-Baptiste Bréjon

Model-Centric Analysis of OpenMP Programs*

Analysis of OpenMP programs

- Parallel loops
 - Examine Partitioning
 - Execution time per worker
 - Filtering
- OpenMP 4 tasks
- OpenMP constructs: Barriers, single, ...
- Hardware performance counters

Easy-to-use scripts and library

- Instrumented Clang / LLVM OpenMP run-time (KMP / Intel)
 - \$ aftermath-openmp-trace -o events.ost -- <program> <args>
 - \$ aftermath events.ost

* Joint work with Jean-Baptiste Bréjon

Aftermath at a glance

- Fast and responsive UI, multiple connected views
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Joint analysis applications + run-time
- Successfully used for performance debugging of OpenStream

Aftermath at a glance

- Fast and responsive UI, multiple connected views
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Joint analysis applications + run-time
- Successfully used for performance debugging of OpenStream

Source code and tutorial available at

http://www.openstream.info/aftermath

- ► Free software license, small code base (< 30k lines of C code)
- ► Few dependences: GTK+, Cairo, Pango, Glibc, ...

Aftermath at a glance

- Fast and responsive UI, multiple connected views
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Joint analysis applications + run-time
- Successfully used for performance debugging of OpenStream

Source code and tutorial available at

http://www.openstream.info/aftermath

- ► Free software license, small code base (< 30k lines of C code)
- ▶ Few dependences: GTK+, Cairo, Pango, Glibc, ...

VM with Aftermath + OpenStream + sample traces available at http://www.openstream.info/vm

Aftermath at a glance

- Fast and responsive UI, multiple connected views
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Joint analysis applications + run-time
- Successfully used for performance debugging of OpenStream

Source code and tutorial available at

http://www.openstream.info/aftermath

- ► Free software license, small code base (< 30k lines of C code)
- ▶ Few dependences: GTK+, Cairo, Pango, Glibc, ...

VM with Aftermath + OpenStream + sample traces available at http://www.openstream.info/vm

OpenMP support very soon!