# Aftermath

A graphical tool for performance analysis and debugging of fine-grained task-parallel programs and run-time systems

#### Andi Drebes

Université Pierre et Marie Curie Laboratoire d'Informatique de Paris VI andi.drebes@lip6.fr

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

MULTIPROG'14, January 22nd, 2014





## Context

#### Hardware and software environment

- Multi-core / many-core NUMA systems
- ► Task-parallel languages based on fine-grained tasks

## Context

#### Hardware and software environment

- Multi-core / many-core NUMA systems
- Task-parallel languages based on fine-grained tasks

#### Efficient exploitation of the machine

- Enough parallelism to keep processors busy
- Data locality on NUMA systems
- Low management overhead
- Requires analysis of complex interactions: Application / Run-time / Machine

## Context

#### Hardware and software environment

- Multi-core / many-core NUMA systems
- Task-parallel languages based on fine-grained tasks

#### Efficient exploitation of the machine

- Enough parallelism to keep processors busy
- Data locality on NUMA systems
- Low management overhead
- Requires analysis of complex interactions: Application / Run-time / Machine

## Tools for performance debugging

Trace-based analysis

- 1. Trace-based analysis
- 2. Overview on Aftermath
- 3. Live demo
- 4. Use cases
- 5. Summary & Questions

# Capturing event data



#### Trace data

- Captured during execution
- Event types
  - Discrete events
  - State changes
  - Counter values
- High-level events
- Low-level events
- ► Dumped to trace file

# Off-line analysis

#### Off-line analysis scenarios

- Trace exploration
  - Optimization opportunities
  - Presence of a bottleneck?
  - No precise idea where to look
- Hypotheses testing
  - Precise idea of performance anomaly
  - Check if the hypothesis holds
- Cyclic pattern during application / run-time development and debugging
- Need for a tool supporting exploration *and* hypotheses testing



	-
	-

## Graphical representation

- ► Static and dynamic information
- Make strong correlations visible

_		
-	-	

## Graphical representation

- Static and dynamic information
- Make strong correlations visible



#### Filters

- Control over amount of detail
  - ► Task type, duration, processors, ...
- Ability to combine filters

		-
	_	
-		_

## Graphical representation

- Static and dynamic information
- Make strong correlations visible

5	2	
F	7	$\left( \right)$
L	ſ	

## Filters

- Control over amount of detail
  - ► Task type, duration, processors, ...
- Ability to combine filters



## Statistical counters

- Quantifying events
- Derived metrics

		-
	_	
-		_

## Graphical representation

- Static and dynamic information
- Make strong correlations visible

5	7	
2	5	1
	$\Lambda$	ſ
L	5	

## Filters

- ► Control over amount of detail
  - ► Task type, duration, processors, ...
- Ability to combine filters



## Statistical counters

- Quantifying events
- Derived metrics



## User interface

- Support for large trace files
- Reactive UI with immediate feedback

# Why a new tool?

#### **Existing approaches**

- ► Trace analysis: Well-known technique, widely used in HPC
- ► Existing tools: Paraver, Vampir, Paraprof / TAU, ...
- Optimized for clusters / message passing
- Many good ideas in these tools, but none of them fully satisfied our requirements

# Why a new tool?

## Existing approaches

- ► Trace analysis: Well-known technique, widely used in HPC
- ► Existing tools: Paraver, Vampir, Paraprof / TAU, ...
- Optimized for clusters / message passing
- Many good ideas in these tools, but none of them fully satisfied our requirements

## Additional requirements in the OpenStream project

- ► OpenStream: OpenMP extension, fine-grained data-flow tasks
- Native support for tasks
- Resource-model
  - NUMA analysis
  - Data-flow analysis
- Interactive exploration





Re Data 641 Mew Help	ekseldelistreare <sub>,</sub> df_seldelievents.md.est «@eloble»				• • •
( Total Counters Frames )	Stath Code				Statistics Heatman
Task types					3943095911 - 6922786085
Address Symbol					
2 Ee0 (no symbol found					5.43 Mcycles / task (avg)
2 8e401550 (no symbol found					
P 894210/0 (no symbol found 4					Task length distribution
2 8x401900 (no symbol found					200
C 8e401930 (no symbol found		and the second se			
2 EAGLERO (no symbol faunc	No.	CPU 0			6
2 8e431d30 (no symbol found		CDULT			57.42 M
2 8x401dd0 (no symbol faund		CPUI			
2 8e401f00 (no symbol found		CD11.2			
2 Beaution (no symbol fear)	2018 C	Cro L			
2 0x4021/0 (no symbol found		CPU 3			
20 8x4023c0 (no symbol found v					
	44 H ( )	CPU 4			
= Select none	2011				ve transfers
+ Select al	22.20 Contraction of the second s	CPU 5			
Writes to NUMA nodes		CDUL C			
Node	NA CONTRACTOR OF A CONTRACTOR OFTA	CPU 6			Parmax.)
V Note 0		CPU 7			
2 Node 2					Cris Par
	No. 1	CPU 8			0.61% 0.39
At least 0 B	ana				95.65% 61.21 1.23% 0.79
- Select none	27.11	CPU 9			1.17% 0.35
+ select all		CBU 10			2.31% 0.84 0.03% 0.02
Length		CFO AU			0.00 0.00
Filter tasks by length	Selected event	CPU 11			0.00% 0.00
Min	CR0. 2				m graph
Max	Trees 4433380000 to 4633540406		100 00 10 01: 0 X2 M		
di teshy	Duration: 3.16 Mcycles		Active frame: 0x75d50d8b660 dr none dr29v9rd8b000		
CPU 13: state 1 (taskexec) from 459	6460383 to 4605347184, duration: 6.69 Mcycles, active task:	0x405fc0 (null)			

## **Timeline component**

- Processor activity over time
  - ► State changes
  - Communication
  - Hardware counters

- ► One "lane" per CPU
- Multiple views:
  - States / Heatmap / NUMA



## Filters

- Task-related (type, length)
- Subset of CPUs
- Event types

- Communication (size, type)
- Data-flow
  - Specific data-flow buffers
  - Specific NUMA nodes



## Statistical graphs and counters

- ► Selectable area
- Number of tasks
- Average duration

- Time spent in different states
- Task duration histogram
- ► NUMA node comm. matrix



## **Detailed textual information**

- ► Selected task / state
- ► Task / state duration
- ► Task creation

- Data-flow information
  - Producers / Consumers
  - Reads / writes per NUMA node

	Aller Aller	idel\kream_dl_seide	il leventa.rnd.cot « (idable»							• •
tee Data Eat New	, Helb									
A Coll afterna the gale	die-		• •							
2000	(natia								Statistics Heatmap	
							Rel afterwath sigdst	14 P		
Course:	CHUJUJAEODES	IS_TO_MEMORI_OL	OCAL_CHO_HO_NEMOTE_HEM				7,94:	MUNA rode	centention	
Dividing counter:	CRUJO_REQUES	TS_TO_MEMORY_IOL	OCAL_CPU_10_REMOTE_NEM	<u></u>			NUMA node:	Node 0		0
Picce:	Plan division Le	L (A / B)					Access type:	<ul> <li>Write acce</li> </ul>	esses only	
Number of semples:	- United by same	J.C. AGATOJU	1000.0	Non alternath sign	keler			<ul> <li>Fead acce</li> </ul>	isses only	
Name:	Derived			7,94:	Task length	0		· Read and a	write accesses	
				Attach to CPU	CPU 0	0	Data direction	<ul> <li>Hereaters</li> </ul>	obes to ) from local nee	se only
			Carce Or	Number of samples		1008.0		<ul> <li>Local node</li> </ul>	to / from remote ced	es anky
Strassess in a stu	WATER .			CPUS:	CPU			🗴 Remote a	nd local nodes to / from	i local node
	ibol feand				CPU 0			<ul> <li>Everything</li> </ul>	g involving local node	
	ibol feund				0 070 1		Access model:	O Spikes		
	ibol found				0 0002		_	R Task Inear		
					0.0014		Exclude regs. from			
	and found				LT CPU 5		Attach to CPU:	CPU 8		
	that frame				CPU 6		Number of samples:			1000.0
				Date	11.0013		Name	Derved		
Not afternath + Sel	deter -				0.000	optimus on constant ferment			2 Cancel	40 OK
Jpc:	Appropate			0	0 0x402570 0	reate terrinal task, wst				
Counter:	CPU_IO_REQUES	TS_TO_MEMONY_IOL	OCAL_CPU_TO_REMOTE_MEM		0.0403530	reate terrinal task, wst			Ral mire -	
Attach to CPU:	CPU 0				0x402710 0	reate_terminal_task_wst			Ral mas:-	
Number of samples:			1000.0		0.0463820 0	reate_initial_task_watrea				max.1
Name:	Derived				0.0402950 0	reate_initial_task_wstrea				
			(Monut) after	- <b>-</b>	CONCLESS C	PEACE PADM TANK, WITPEN V				
			Carco O	Name	Derived		Kall aftermeth sigds			
At least 0	5					Cancel di OK	Type:	Parallelism		0
							State:	Task enecution		0
- Scherten							Attach to CPU:	CPU 0		
							Number of samples:			1000.0
Length	3					•	Name:	Derived		
recer calks by long	th Se	lected event			Active tasi				ancel 2	48 ox
						OXEDSTCD (MUR)				
MI C			000 to 4533545496							
		aration: 3.16 Mc;	rcles		Active frame					
				Tark AutoStro Indi						

## **Derived metrics**

- ► Aggregation of CPUs
- Ratio of counters
- Parallelism on average

- NUMA node contention
- Average task duration

# Example trace: Gauß-Seidel iterations



#### Benchmark from the OpenStream project

- Data-flow implementation of a 5-point stencil
- Main tasks + Auxiliary tasks (initialization, termination)
- ► Two traces: unoptimized / optimized task and data placement

#### Hardware environment

- ► 64 cores
- 8 NUMA domains

# LIVE DEMO

## Use cases

#### 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

# Summary

#### Aftermath at a glance

- ► Fast and responsive UI
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Multiple connected views
- Successfully used for performance debugging of the OpenStream run-time and applications

# Summary

#### Aftermath at a glance

- ► Fast and responsive UI
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Multiple connected views
- Successfully used for performance debugging of the OpenStream run-time and applications

#### Source code and tutorial available at

http://www.openstream.info/aftermath

- ▶ Free software license, small code base (< 15k lines of C code)
- ► Few dependences: GTK+, Cairo, Pango, Glibc, ...

# Summary

#### Aftermath at a glance

- ► Fast and responsive UI
- Various filters with immediate visual feedback
- Native support for dependent tasks
- Multiple connected views
- Successfully used for performance debugging of the OpenStream run-time and applications

## Source code and tutorial available at

http://www.openstream.info/aftermath

- ▶ Free software license, small code base (< 15k lines of C code)
- ► Few dependences: GTK+, Cairo, Pango, Glibc, ...

## Future Work

- More detailed resource model
- Trace compression
- Your idea here!