

**Barcelona Supercomputing Center** Centro Nacional de Supercomputación

# Correlating Performance, Code Location and Memory Access

Harald Servat, Jesus Labarta, Judit Gimenez Scalable Tools Workshop - Lake Tahoe, Aug 2<sup>nd</sup> 2016

# Folding: instantaneous metric with minimum overhead

(Combine instrumentation and sampling

Centro Nacional de Supercomputación

- Instrumentation delimits regions (routines, loops, ...)
- Sampling exposes progression within a region

(Capture performance counters and call-stack references



### ( Memory related data in the trace

#### PEBS events

- · Loads: address, cost in cycles, level providing the data
- Stores: only address
- Sampling frequency:
  - Possibly different rate for both loads and stores
  - One entry PEBS buffer. Signal Extrae on individual event.
- Multiplexing: alternate periods sampling loads and stores





# Memory object references

### ( Memory related data in the trace

- Interception of mallocs and frees
  - Emit object id/call stack
  - With threshold on allocated size (potential unresolved objects)
- Identification of memory object on sampled references
  - Static object from symbol table  $\rightarrow$  Identify variable name
  - Dynamic objects from instantaneous memory map → Identify malloc where object was allocated

## ( Observation

- Same source code → different per process address space
  - Randomization Linux security

### ( Insight

Folding should be applied on a per process basis





# Analytics

# ( Identification of coarse grain repetitive structure (prerequisite)

- Computation bursts
  - Between calls to the runtime (MPI, OpenMP)
  - Clustering
- Iteration (longer intervals with runtime calls)
  - Manually:
    - Extrae\_event API call
    - Paraver analysis
  - Automatic: Using spectral analysis (WIP)
  - Clustering
    - Isolate different modes, eliminate outliers

# ( Folding generates:

- Gnuplot
- Paraver trace
  - All PEBS related events are projected and **ordered** into a representative instance of the repetitive region
  - The same Paraver configuration files can be applied





## Looking at Lulesh: 1. Performance

#### 27 MPI ranks in 2 nodes (2 sockets x 12 cores each node)



#### MPI calls

Useful Duration @ lulesh27\_pebs\_jureca.prv.gz THREAD 1.1.1 THREAD 1.5.1 THREAD 1.9.1 THREAD 1.17.1 THREAD 1.21.1 THREAD 1.21.1 THREAD 1.25.1 1.248,765 us 2,536,953 us





#### Useful instructions



# Looking at Lulesh: 1. Performance

#### Histogram useful duration



#### Histogram useful instructions



#### Process mapping



#### Histogram clock frequency



# Looking at Lulesh: 1. Performance

#### One iteration

Useful Duration @ lules	h27_pebs_jureca.chop1.prv		
THREAD 1.1.1			
THREAD 1.2.1			
THREAD 1.3.1			
THREAD 1.4.1			
THREAD 1.5.1			
THREAD 1.6.1			
THREAD 1.7.1			
THREAD 1.8.1			
THREAD 1.9.1			
THREAD 1.10.1			
THREAD 1.11.1			
THREAD 1.12.1		_	
THREAD 1.13.1			
THREAD 1.14.1			
THREAD 1.15.1			
THREAD 1.16.1			
THREAD 1.17.1			
THREAD 1.18.1			
THREAD 1.19.1			
THREAD 1.20.1			
THREAD 1.21.1			
THREAD 1.22.1			
THREAD 1.23.1			
THREAD 1.24.1			
THREAD 1.25.1			
THREAD 1.26.1			
THREAD 1.27.1			
98,891 us			533,380 us

MPI call @ lul	esh27_pebs_jureca.chop1.prv					
THREAD 1.1.1						
THREAD 1.2.1						
THREAD 1.3.1						
THREAD 1.4.1						
THREAD 1.5.1						
THREAD 1.6.1		1	1			
THREAD 1.7.1						
THREAD 1.8.1						
THREAD 1.9.1						
THREAD 1.10.1						
THREAD 1.11.1						
THREAD 1.12.1						
THREAD 1.13.1						
THREAD 1.14.1				· · · · ·		
THREAD 1.15.1						
THREAD 1.16.1			11			
THREAD 1.17.1						
THREAD 1.18.1			÷.			
THREAD 1.19.1			÷.			
THREAD 1.20.1			11			
THREAD 1.21.1						
THREAD 1.22.1					- 71	
THREAD 1.23.1					÷.	
THREAD 1.24.1						
THREAD 1.25.1						
THREAD 1.26.1						
THREAD 1.27.1						
	98,891 us					533,380 us



Barcelona Supercomputing Center Centro Nacional de Supercomputación

#### 4 tasks selected



## Looking at Lulesh: 2. Code location

#### Approximation based on call stack @ MPI calls





#### Approximation based on folded call stack











9 (Unresolved)
2074 (kmp\_runtime.c, libiomp5.so)
493 (lulesh-comm.cc, lulesh2.0)
511 (lulesh-comm.cc, lulesh2.0)
1158 (lulesh.cc, lulesh2.0)
1161 (lulesh.cc, lulesh2.0)
1998 (lulesh.cc, lulesh2.0)
2010 (lulesh.cc, lulesh2.0)
2414 (lulesh.cc, lulesh2.0)
2461 (lulesh.cc, lulesh2.0)
2464 (lulesh.cc, lulesh2.0)









455.000

#### PEBS level providing the data





Sampled location 2020om range [2,2] @ lulesh_s4tasks_	rolded.prv						
THREAD 1.13.1							
Thread 1.14.1							
THREAD 1.21.1			Nev	v Histogram #1 @ lulesh	_s4tasks_fol	lded.prv <2>	
THREAD 1.23.1			o 🙉 i 🖬 ī			I I I I I I I I I I I I I I I I I I I	ed.brv
e us		455,000 us	× × I				
			L1 cache	Line Fill Buffer (LFB)	L2 cache	L3 cache	DRAM (loca
Sampled location 2DZoom range [3,3] @ lulesh_s4tasks_	folded.prv	THREAD 1.13.1	1 6,487	160	42	11	
THREAD 1.13.1		THREAD 1.14.1	1 7,580	172	58	12	
THREAD 1.14.1		THREAD 1.21.	1 6,860	131	27	12	
THREAD 1.21.1		THREAD 1.23.4	1 9,523	359	60	23	
THREAD 1.23.1							
e us		455,000 us Total	30,450	822	187	58	
		Average	7,612.500000	205.500000	46.750000	14.500000	10.2500
Sampled location 2DZoom range [4,4] @ lulesh_s4tasks_	folded.prv	Maximum	9,523	359	60	23	
THREAD 1.13.1		Minimum	6,487	131	27	11	
THREAD 1.14.1		StDev	1,170.904885	89.867959	13.367404	4.924429	2.5860
THREAD 1.21.1		Avg/Max	0.799380	0.572423	0.779167	0.630435	0.7321
THREAD 1.23.1							
e us		455,000 us				_	
		L2 cache					
Sampled location 2DZoom range [7,7] @ lulesh_s4tasks_	folded.prv	_					
THREAD 1.13.1							
THREAD 1.14.1							
THREAD 1.21.1							
THREAD 1.23.1							
e us		455,000 us					

#### PEBS cost in cycles (avg.)

Ѥ 10 30   🔍 💐   🔳 н Η 11 ⅔ Σ							
	L1 cache	Line Fill Buffer (LFB)	L2 cache	L3 cache	DRAM (local)		
HREAD 1.13.1	7.951	40.089	18.649	62.559	129.148		
HREAD 1.14.1	7.667	42.436	18.486	72.550	80.468		
HREAD 1.21.1	7.769	84.112	16.750	567.189	487.700		
HREAD 1.23.1	8.604	109.264	20.556	78.679	371.227		
Total	31.992	275.901	74.441	780.977	1,068.542		
Average	7.998	68.975	18.610	195.244	267.136		
Maximum	8.604	109.264	20.556	567.189	487.700		
Minimum	7.667	40.089	16.750	62.559	80.468		
StDev	0.365	29.116	1.348	214.819	168.352		
Avg/Max	0.930	0.631	0.905	0.344	0.548		

Line Fill Buff New Histogram #1 @ lulesh\_s4tasks\_folded.prv <2>

С 🛛 🕸   🔍 🕰   🚺 Н М 👖 🌾
-------------------------

 $\sim \otimes$ 

	THREAD 1.13.1	THREAD 1.14.1	THREAD 1.21.1	THREAD 1.23.1
Unresolved	8.421	7.894	8.563	8.329
(new_op.ccc:2741) [new_op.cc:50 > lulesh.cc:2741]	7.286	7.550	9.791	9.199
(new_op.cit.cc:68) [new_op.cc:50 > new_opv.cc:33 > lulesh-init.cc:68]	-	-	-	-
(new_op.cr.h:1073) [new_op.cc:50 > new_allocator.h:104 > stl_vector.h:1073]	8.319	8.564	49.044	19.812
(new_op.c.,it,cc:71) [new_op.cc:50 > new_allocator,h:104 > lulesh-init,cc:71]	7	8.998	256.564	7
(new_op.cit.cc:73) [new_op.cc:50 > new_allocator.h:104 > lulesh-init.cc:73]	21.294	20.568	14.157	19.357
(new_op.ct.cc:359) [new_op.cc:50 > new_opv.cc:33 > lulesh-init.cc:359]	7.113	7.285	71.140	13.900
(new_op.ct.cc:361) [new_op.cc:50 > new_opv.cc:33 > lulesh-init.cc:361]	7.243	7.008	11.449	184.014
(new_op.csh.h:396) [new_op.cc:50 > new_allocator.h:104 > lulesh.h:396]	7.625	9.455	63.011	8.497
(new_op.ct.cc:474) [new_op.cc:50 > new_opv.cc:33 > lulesh-init.cc:474]	7.343	7.179	7.001	9.081
(lulesh.csh.h:396) [lulesh.cc:176 > lulesh.h:396]	8.758	7.247	7.185	7.252
(lulesh.csh.h:396) [lulesh.cc:1098 > lulesh.h:396]	8.828	7.312	7.197	7.214
(lulesh.csh.h:396) [lulesh.cc:1034 > lulesh.h:396]	8.366	8.979	7.484	7.165
(lulesh.cc:176)	13.970	13.416	23.714	72.325
(lulesh.ccc:2410) [lulesh.cc:176 > lulesh.cc:2410]	11.432	12.959	19.289	25.314
(lulesh.ccc:2410) [lulesh.cc:2089 > lulesh.cc:2410]	9.542	11.208	21.497	19.731
Total	142.541	145.623	577.086	418.192
Average	9.503	9.708	38.472	27.879
Maximum	21.294	20.568	256.564	184.014
Minimum	7	7.008	7.001	7
StDev	3.631	3.523	61.734	44.675
Avg/Max	0.446	0.472	0.150	0.152



Unresolved

 $\sim$   $\sim$   $\otimes$ 

## Looking at Lulesh: Comparing gnuplots



BSC

14

Code line

5deac6faec90

4ce1550c5898

000009298648 000007fa6bf9 000005fbd0fb 000003fd35fd 000001fe9aff

7000

6000

5000

3000

2000

1000

Code line

b9##974e53f70 2 6f1aa6e017906

5e3638daefca 4d51cad5c7f7

3c6d5cd0a02

0b86ee25285

0000081db92

000006164ae

0000040edc9 000002076e4C 000000000000001 8000

7000

6000

5000

3000

2000

1000

0

433.42

4000 €

0

433.33

[2424] angeElen

ForEle

4000 €

3bd7e31dc4a

[2424]

ForElei

346.66

MIPS -

346.74

MIPS -

### Conclusions

- ( Folding can provide low overhead detailed analysis on accesses to memory
  - Wide range of new metrics: access pattern, memory objects, memory level, cost in cycles,...
- ( Paraver provides huge flexibility combining and correlating the new data :)
  - Only required to implement new "paint as" punctual information

( How much far/close to reverse engineering?

