Estimating the cost of locks in ROOT with VTune

E. Tejedor, D. Piparo, P. Canal, P. Mató

Concurrency Forum

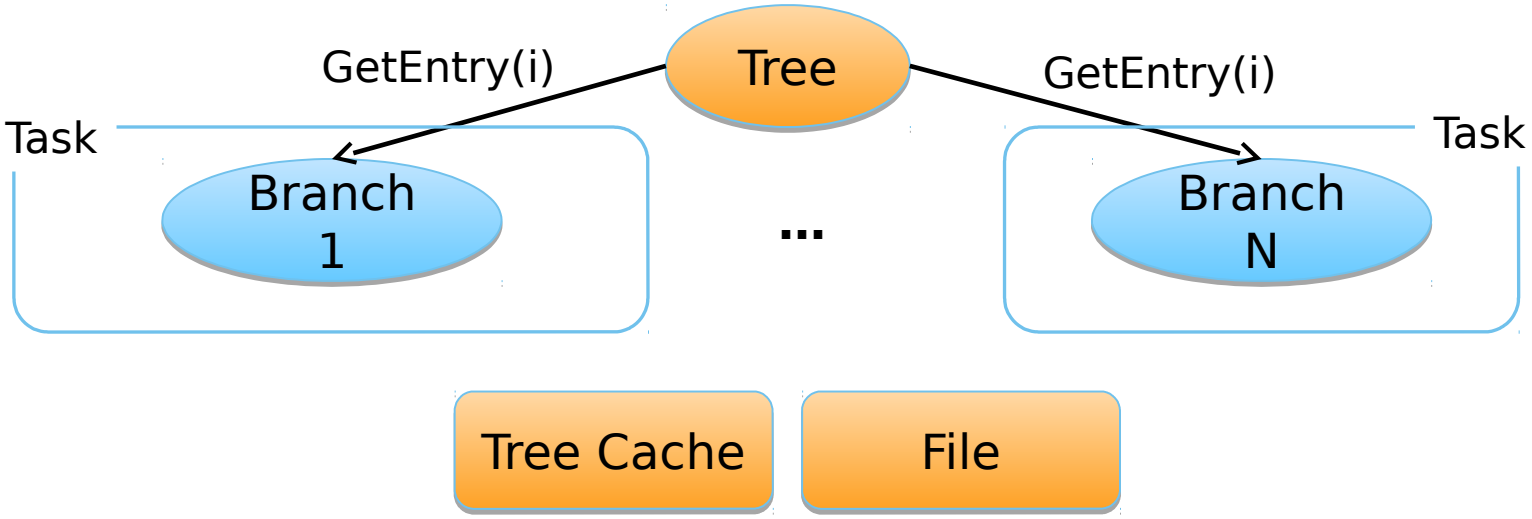
October 21st 2015

Outline

1. Locks in ROOT
   - TTree I/O parallelization
2. Estimation of lock cost
   - IgProf
   - Extrae + Paraver
   - VTune
3. Conclusions

Locks in ROOT

- When working in MT mode, ROOT protects the access to global/shared resources
- Example: TTree I/O parallelization
  - http://indico.cern.ch/event/395194/
  - Branches share the tree, cache and file



Motivation

- Performance analysis of TTree I/O parallelization for a set of trees
  - Custom tree with events of type $ROOTSRC/test/Event.h
  - CMS: GenSim data
  - ATLAS: xAOD data
- Intel machine, 4 cores (8 HT)
- Needed to find out if scaling issues were related to lock contention
- How to reliably estimate the cost of locks?

IgProf

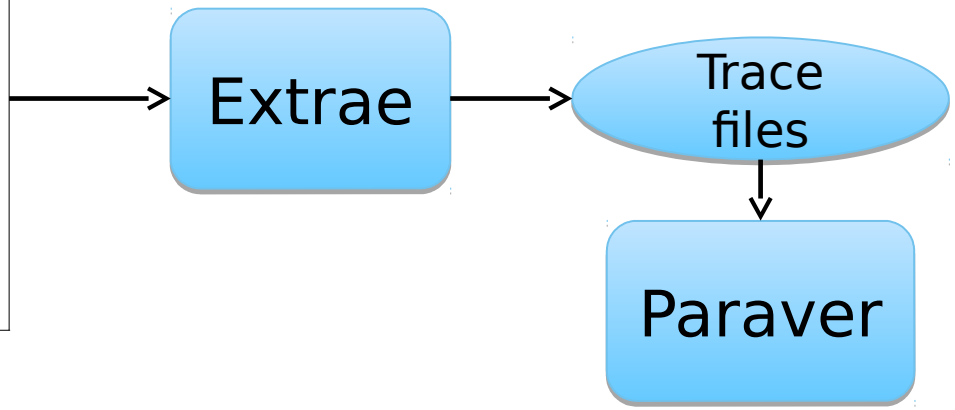
- Statistical sampling
- Low profiling overhead
- Issue with multi-threading
  - Merge of thread stacks not correct
  - Reported to Giulio Eulisse

Rank	Total %	Self	Symbol name
21	56.74	47.14	inflate_fast
44	16.51	13.71	frombuf(char*&, double*)
46	4.85	4.03	adler32
45	3.37	2.80	tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::receive_or_steal_task(long&)
52	2.44	2.03	__read_nocancel
54	2.29	1.90	inflate_table
40	2.26	1.88	TBufferFile::ReadFastArray(double*, int)
57	2.18	1.81	sched_yield
58	1.86	1.54	memcpy
20	1.01	0.84	inflate
63	0.95	0.79	__memset_sse2
90	0.44	0.37	__lll_unlock_wake
107	0.28	0.23	__lll_lock_wait

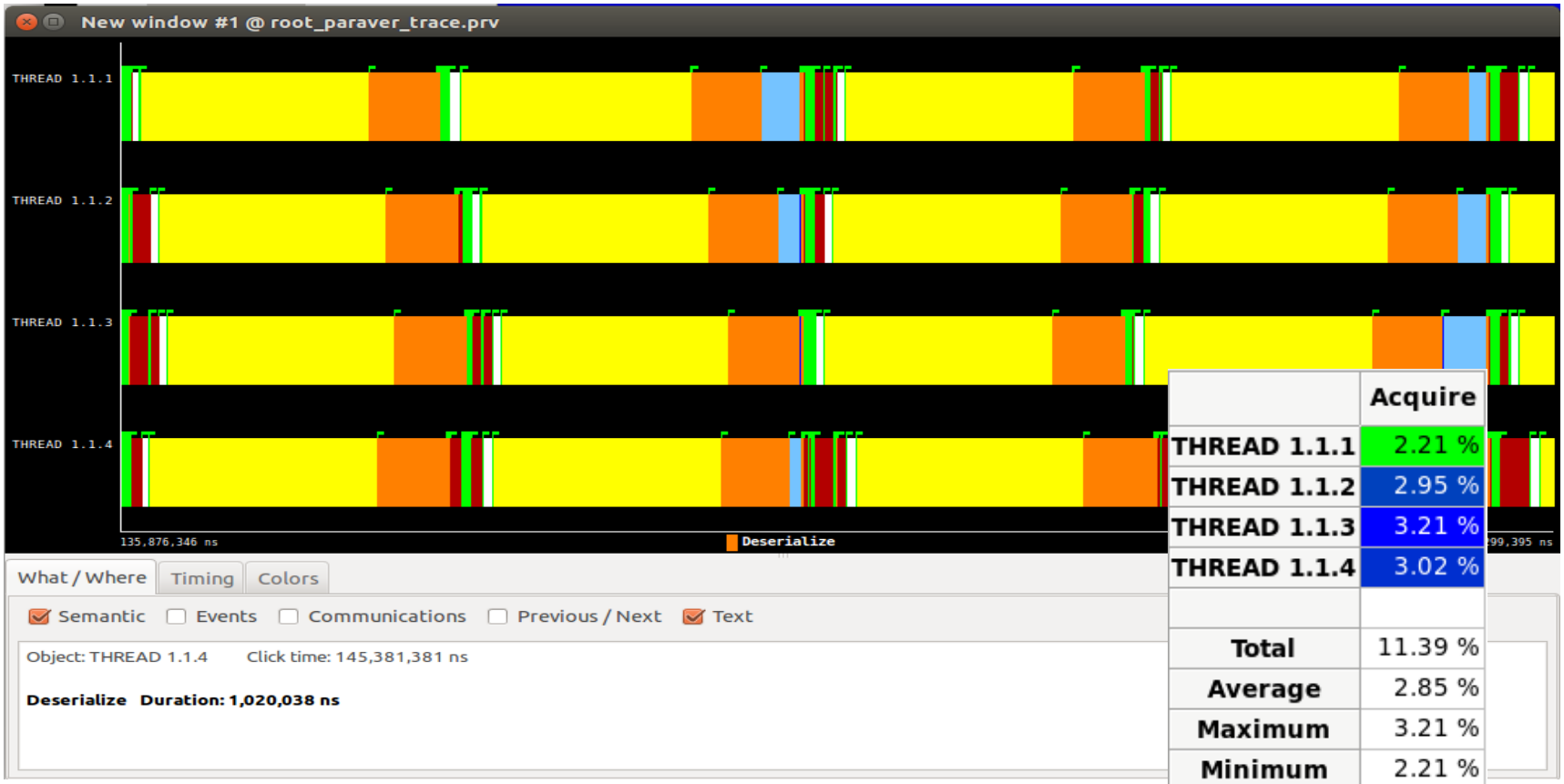
Extrae + Paraver

- Extrae
  - Instrumentation tool
  - Can be used to emit user events
  - Generates trace files
- Paraver
  - Graphical performance analysis
  - Displays trace files

```
User program {
    …
    Extrae_event(type, value);
    …
}
```



Paraver trace

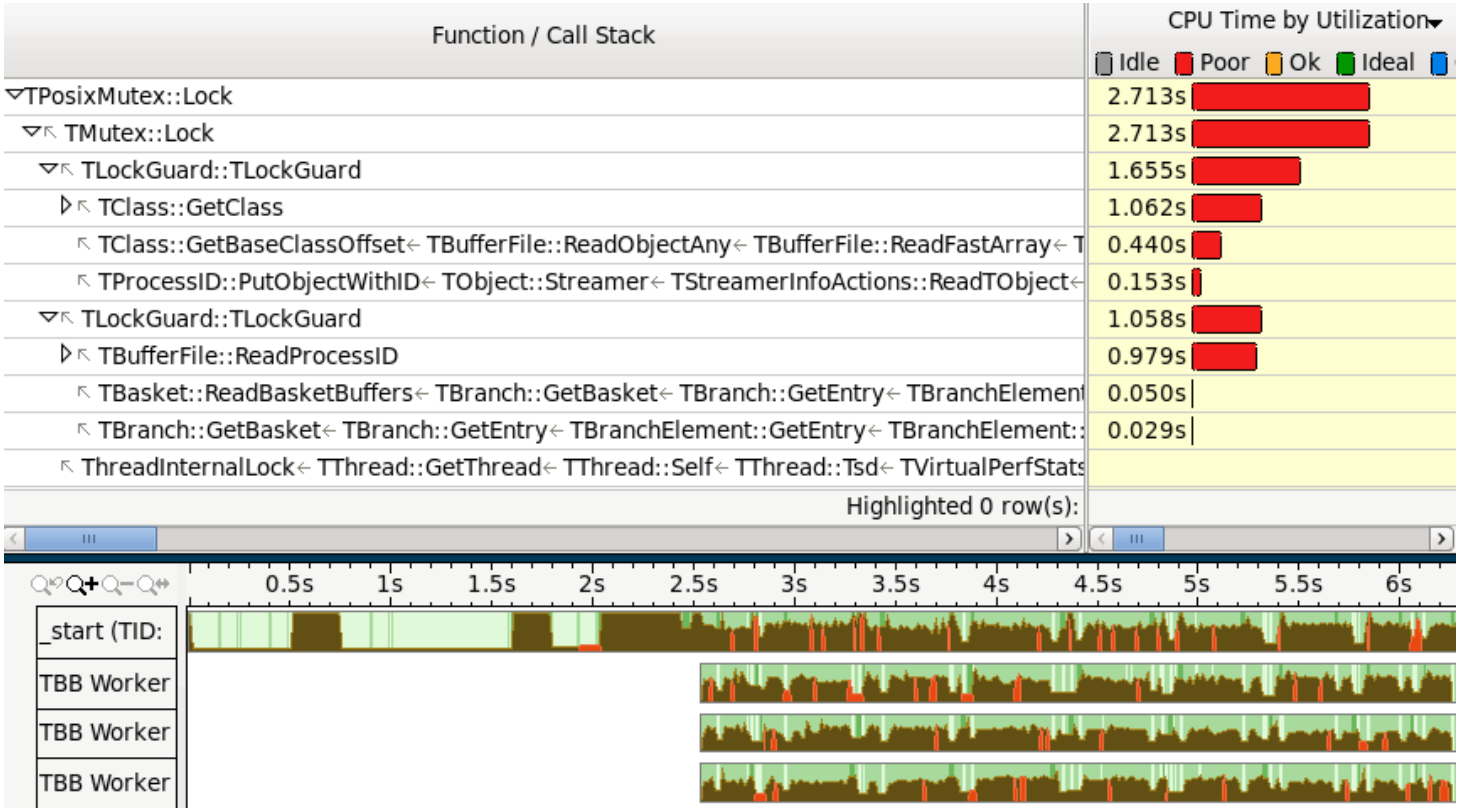


	Acquire
THREAD 1.1.1	2.21 %
THREAD 1.1.2	2.95 %
THREAD 1.1.3	3.21 %
THREAD 1.1.4	3.02 %
Total	11.39 %
Average	2.85 %
Maximum	3.21 %
Minimum	2.21 %
StDev	0.38 %
Avg/Max	0.89

- Count the elapsed time per event type
- Limitation: number of events

VTune: Concurrency Analysis

- Intel VTune Amplifier
  - Profiler
  - Concurrency analysis



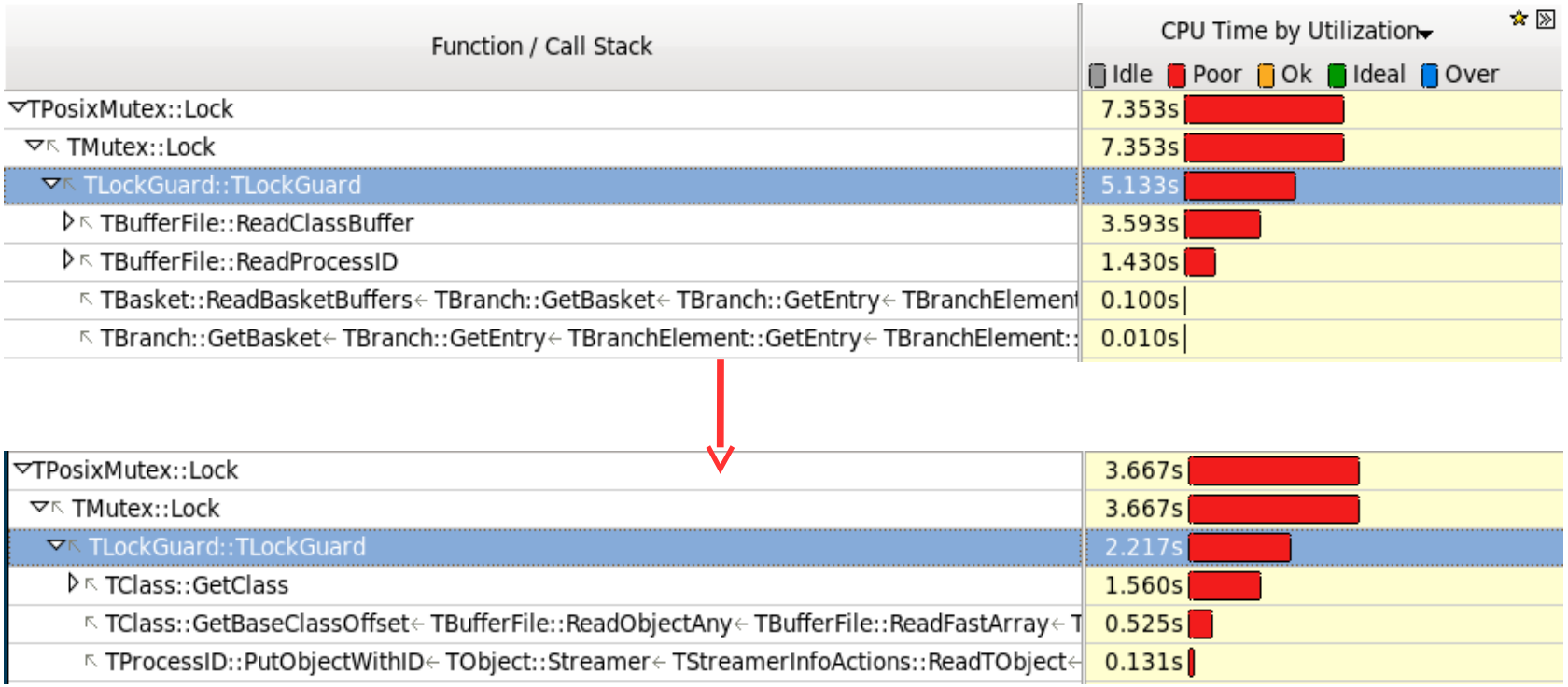
VTune: Waits analysis

- Study of waits
  - Use as input the report of the concurrency analysis
  - Generate a waits analysis with amplxe-cl

Index	% Wait Time:Total	Wait Time:Self	Wait Time:Children	Name	Index
[0]	100.0	0.0	63.986	<spontaneous>	[0]
		0.0	55.229	__libc_start_main	[1]
		0.0	7.003	clone	[25]
		0.0	1.162	main	[35]
		0.0	0.298	func@0x405f2c	[73]
		0.0	0.293	main	[77]
		0.0	0.001	_dl_start_user	[118]
		0.0	0.0	func@0x402260	[195]
		0.0	55.229	<spontaneous>	[0]
[1]	86.31	0.0	55.229	__libc_start_main	[1]
		0.0	55.023	ttree_iter_seq	[2]
		0.0	0.206	main	[90]
		0.0	0.0	toplev_main	[189]

VTune: Lock cost

- Study of locks in parallel TTree I/O
  - Custom tree
  - VTune (2013) reports a set of locks to be time consuming
  - Complete removal of most costly lock reduces execution time with VTune



VTune: Lock cost (II)

- However:
  - Execution time with VTune is significantly bigger (0.2 sec vs 5 sec)
  - Lock removal **has very little effect** in the execution time without VTune
- Same experience with ATLAS tree, VTune 2016 and using the waits analysis:
  - Execution with VTune: 65 sec -> 37 sec
  - No noticeable changes without VTune: 25 sec

Index	% Wait Time:Total	Wait Time:Self	Wait Time:Children	Name	Index
		38.744	0.0	TClass::GetBaseClassOffset	[14]
[15]	60.55	38.744	0.0	TLockGuard	[15]

Conclusions

- Still have not found an optimal tool to measure the lock contention in ROOT
- IgProf and Paraver can help, but they have limitations
- VTune results can be used as a hint, but sometimes they do not estimate correctly the real cost of locks
- Suggestions are welcome!