Benchmarking and comparison of kernel and userspace tracers

Mohamad Gebai
Michel Dagenais

5 May, 2016
École Polytechnique de Montreal

Content

- The tracers
- The setup
- The benchmarks
- Results
  - Kernel space
  - System calls
  - Filtering
  - Userspace

The tracers

- **Kernel tracers**
  - LTTng
  - Ftrace
  - Perf
  - eBPF (kind of)
- **Userspace tracers**
  - LTTng
  - Printf()
  - LTTng using tracef()
  - Extrae
  - Lightweight homemade basic tracer

The setup

- Intel i7 @ 3.40 GHz
- 16 GB of RAM
- Linux kernel version 4.5.0
- LTTng suite built from tip of the branch stabe-2.7
- Deactivated:
  - Hyperthreading
  - C-states
  - CPU idle
  - Intel Turbo Boost
- LTTng
  - Snapshot mode
  - Subbuffer size 32 KB
- Ftrace
  - Global clock

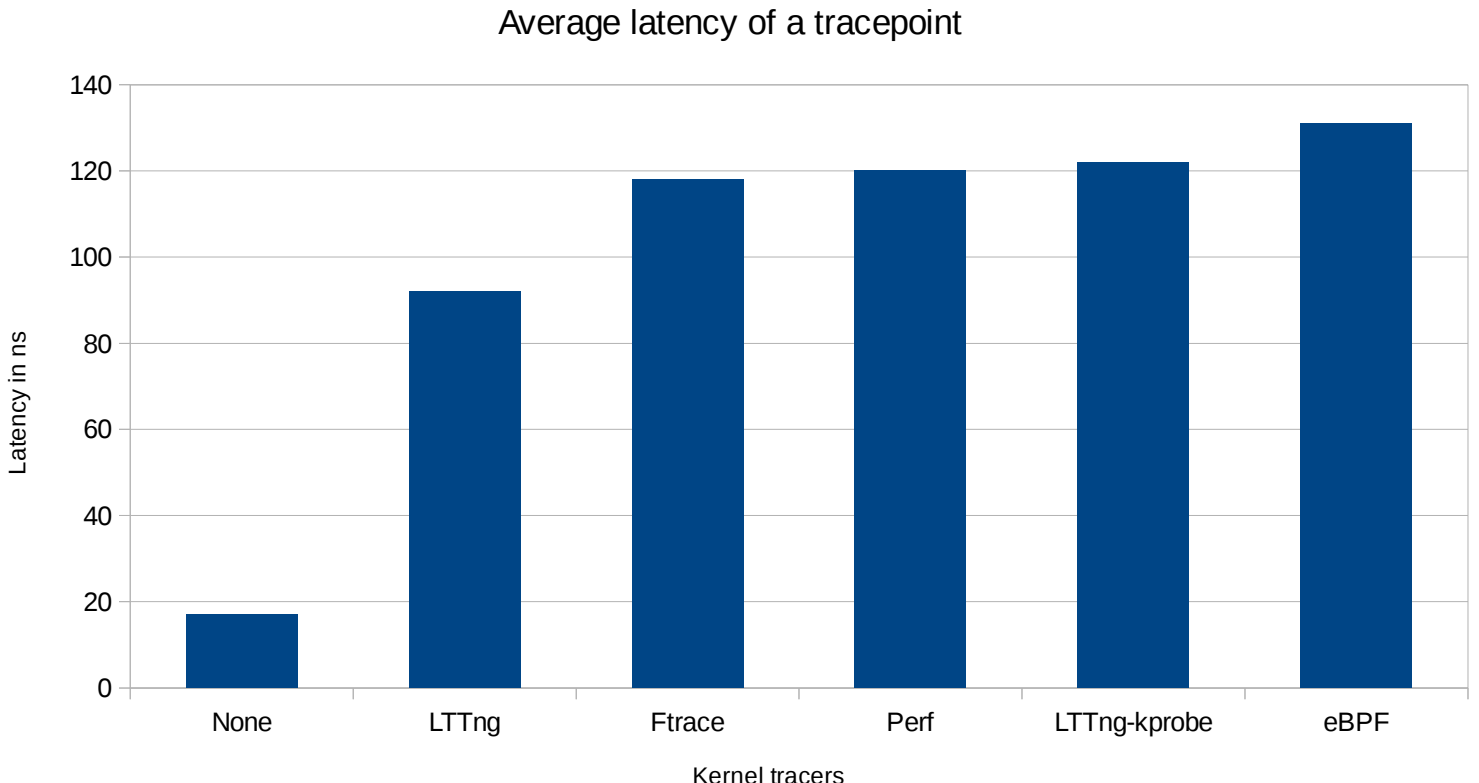
The benchmarks

- Objective: measure the cost of writing a tracepoint
- Kernel tracers:
  - Microbenchmark in kernel space
  - Payload: 32 bytes
- Userspace tracers:
  - Microbenchmark in userspace
  - Payload: 32 bytes
- Steady state (drop initial page faults, cache misses, initialization, etc.)
  - Do a full buffer write and start overwriting

Kernel tracing: the outline

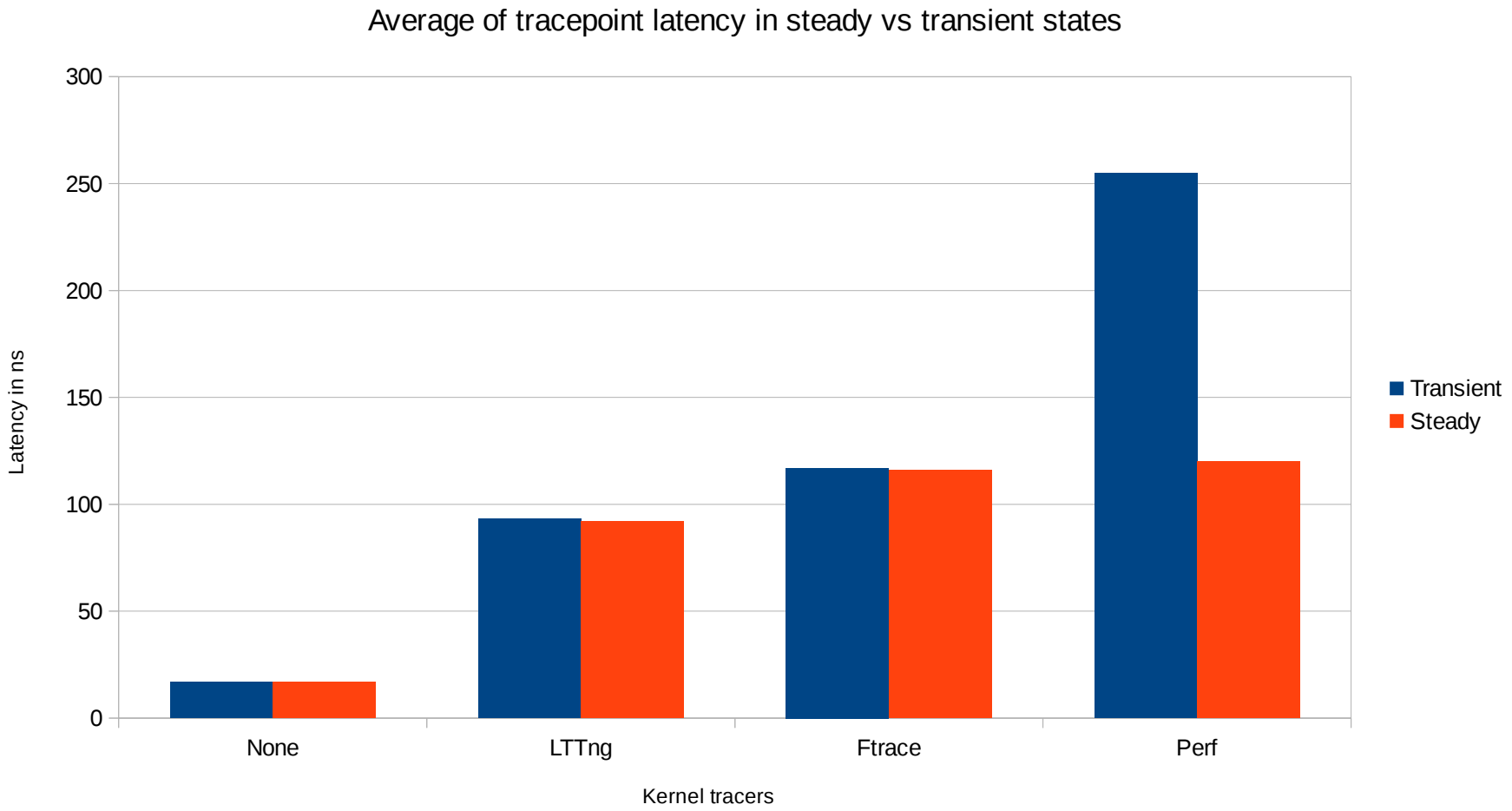
- Final numbers:

	Average		90th percentile	
	Value (ns)	Overhead	Value (ns)	Overhead
None	17	0%	17	0%
LTTng	92	441%	89	424%
Ftrace	118	594%	114	571%
Perf	120	606%	118	594%
LTTng-kprobe	122	618%	120	606%
eBPF	131	671%	126	641%

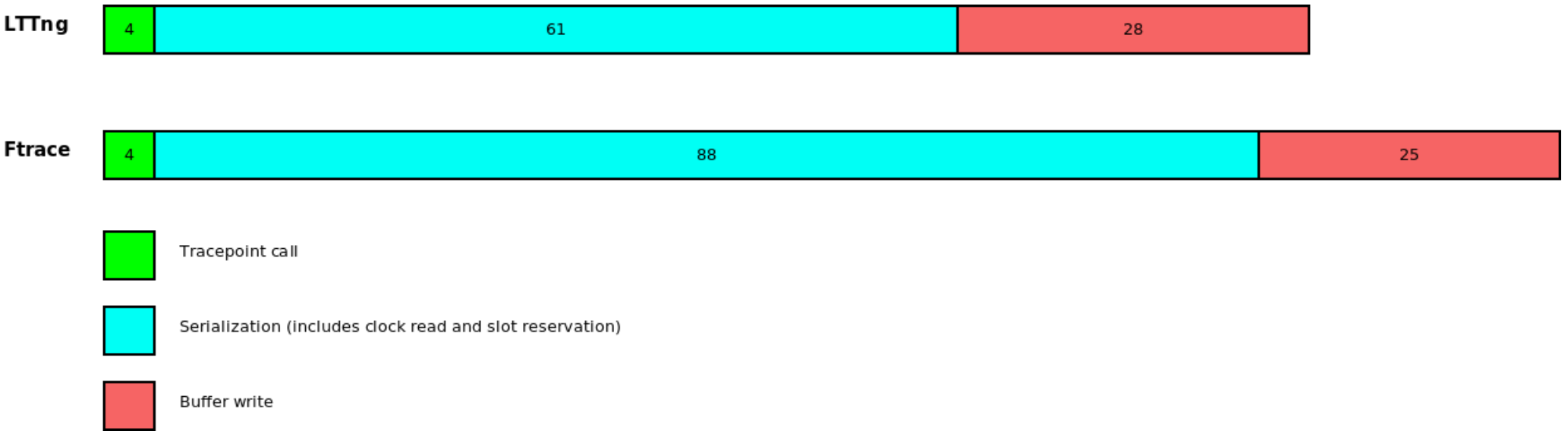


Kernel tracing: the outline

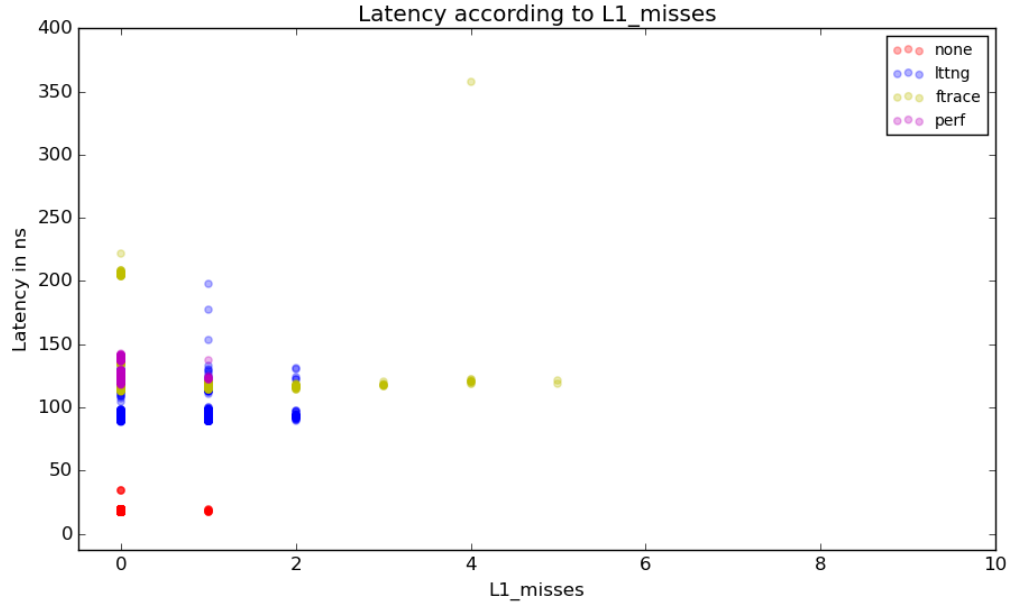
- Final numbers:

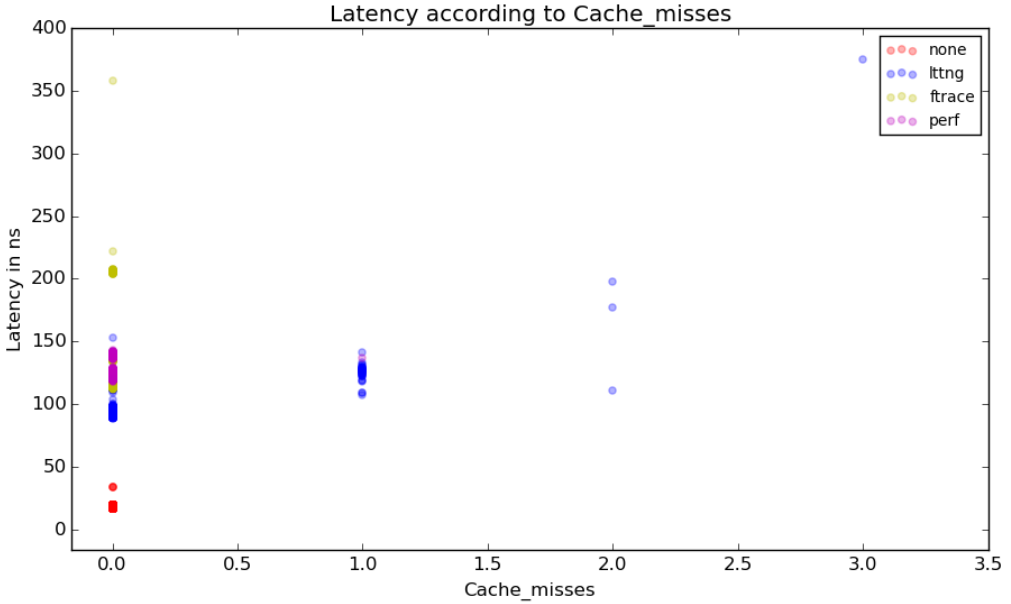


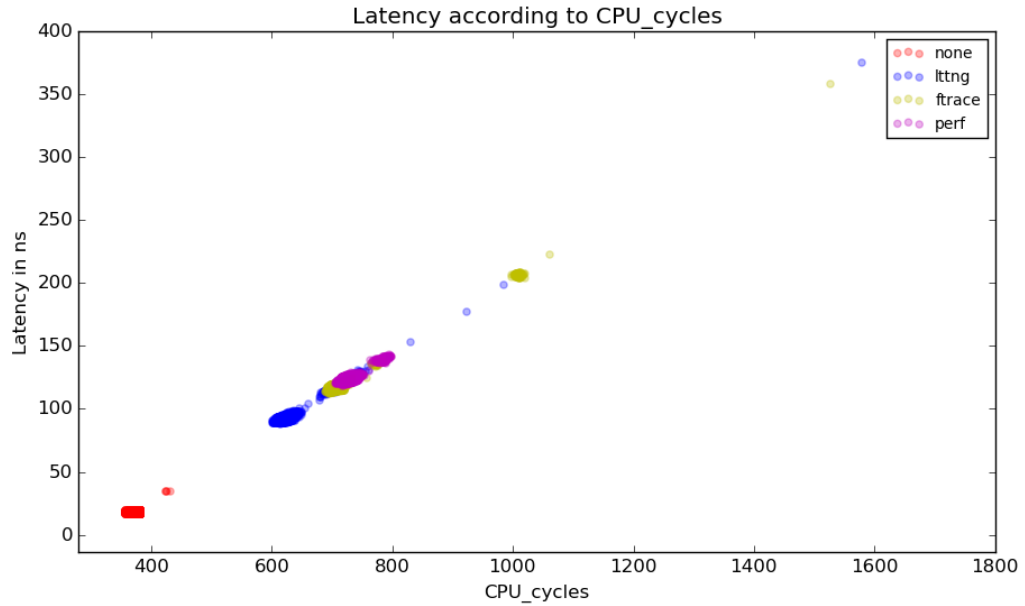
Kernel tracing: LTTng vs Ftrace

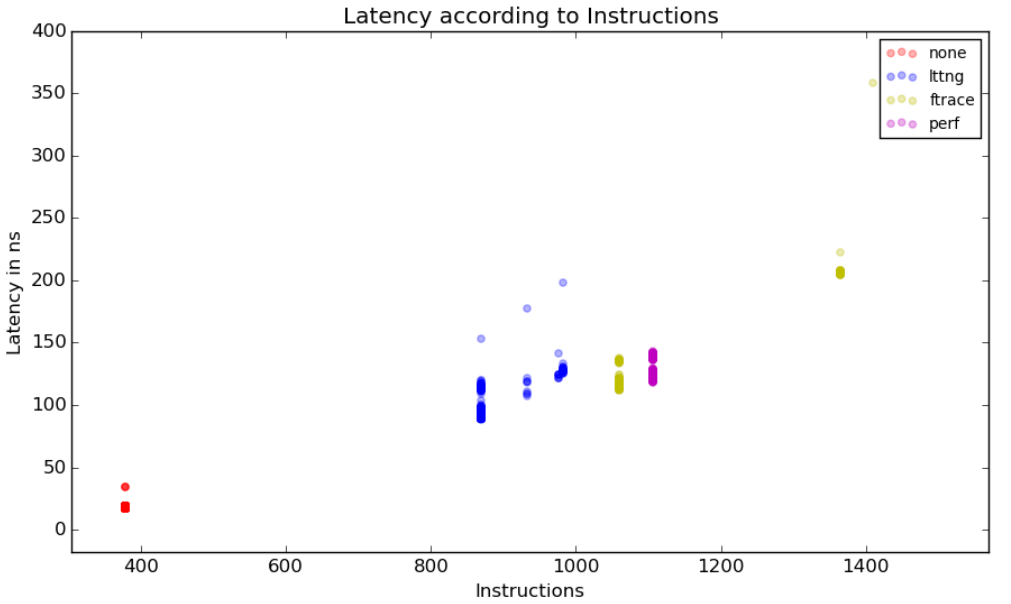


Kernel tracing: closer look

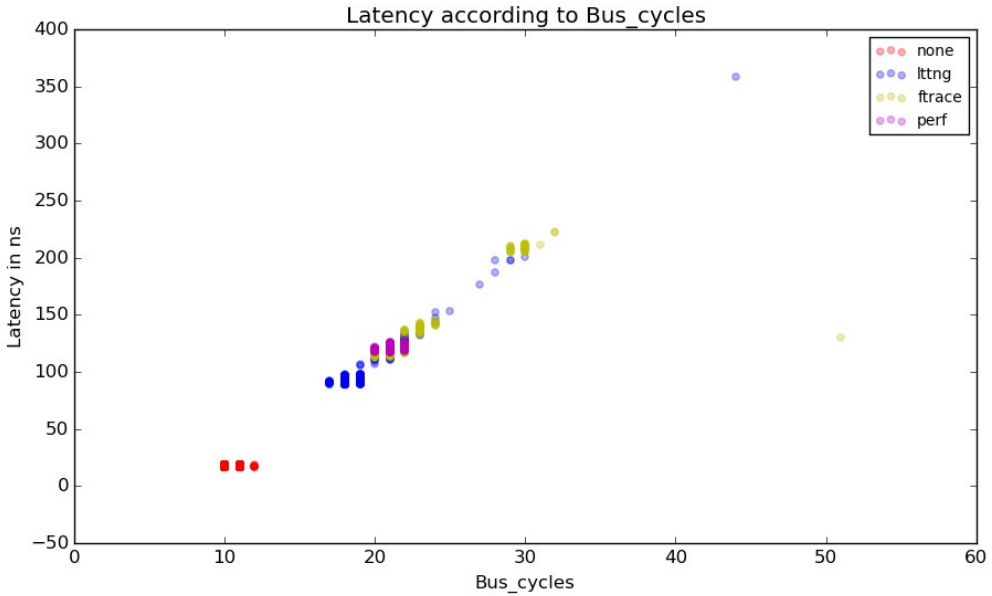


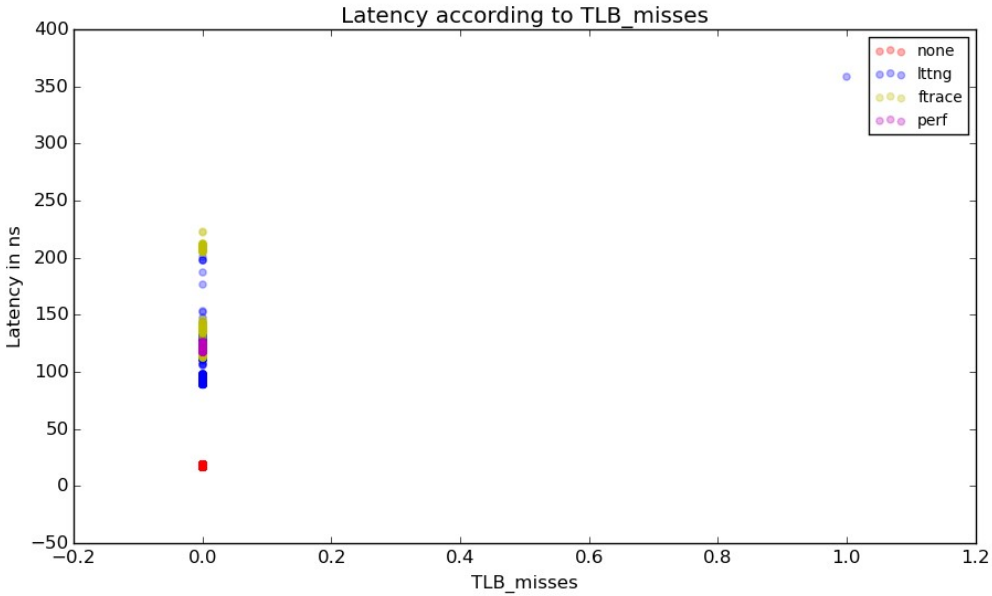


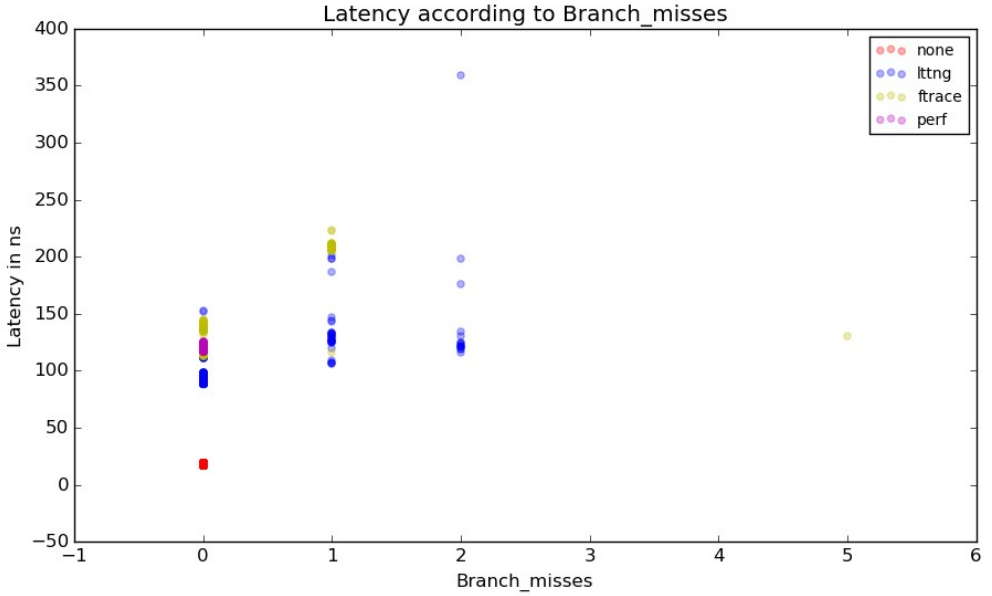


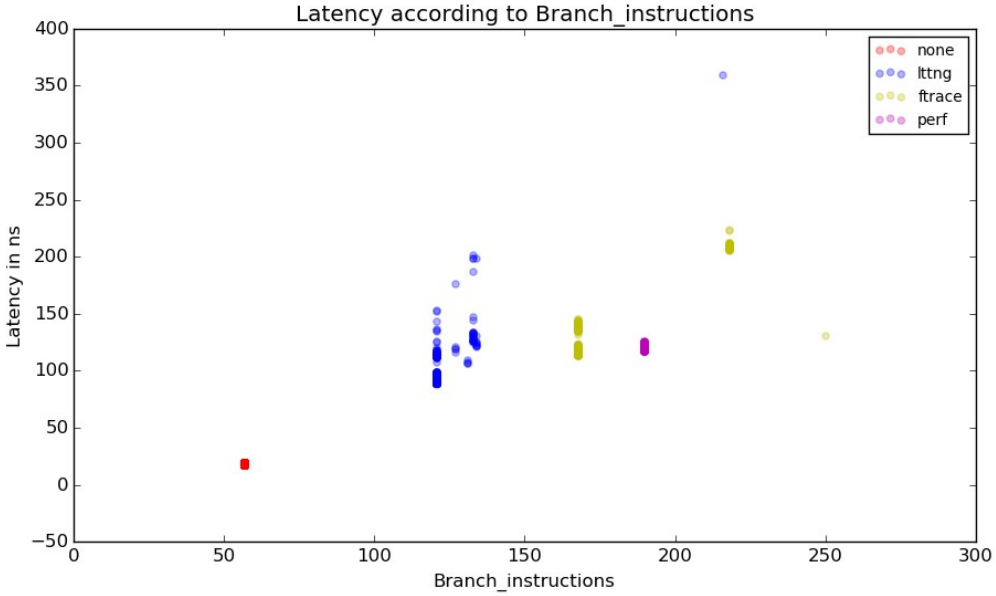


Kernel tracing: closer look





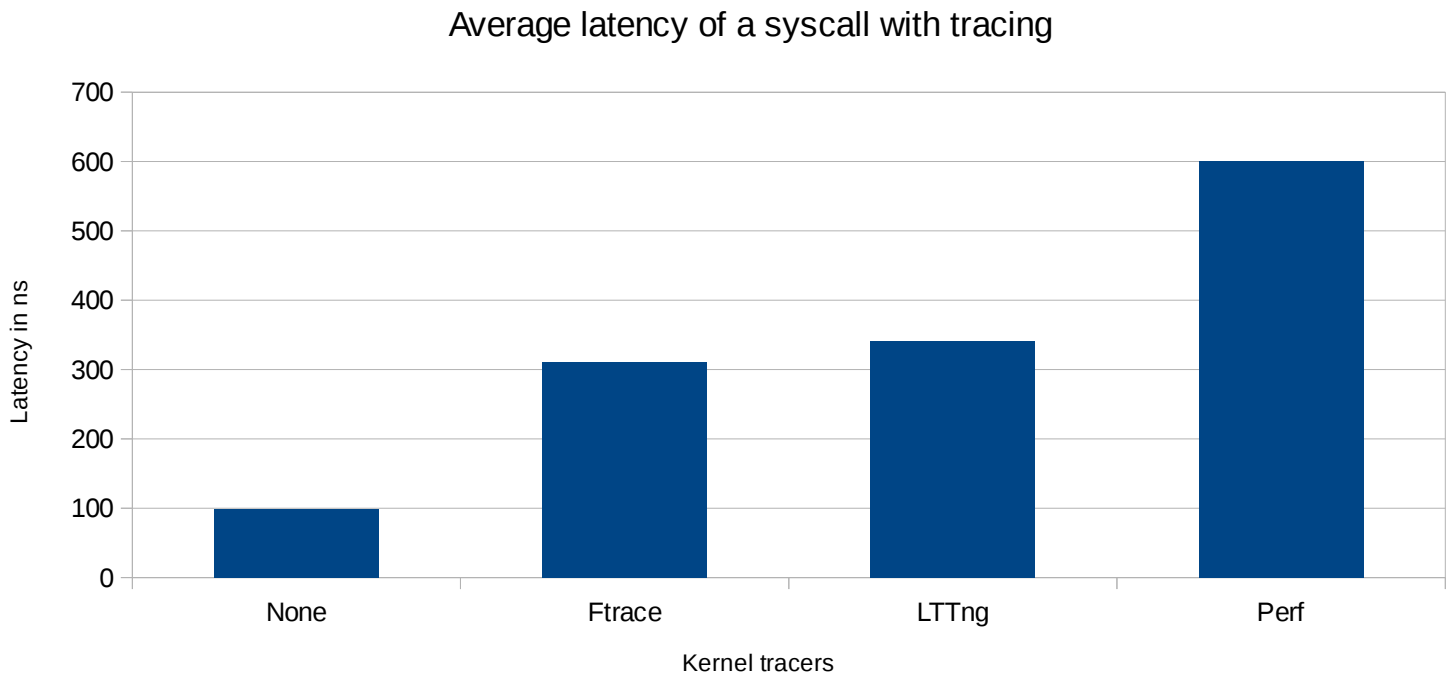




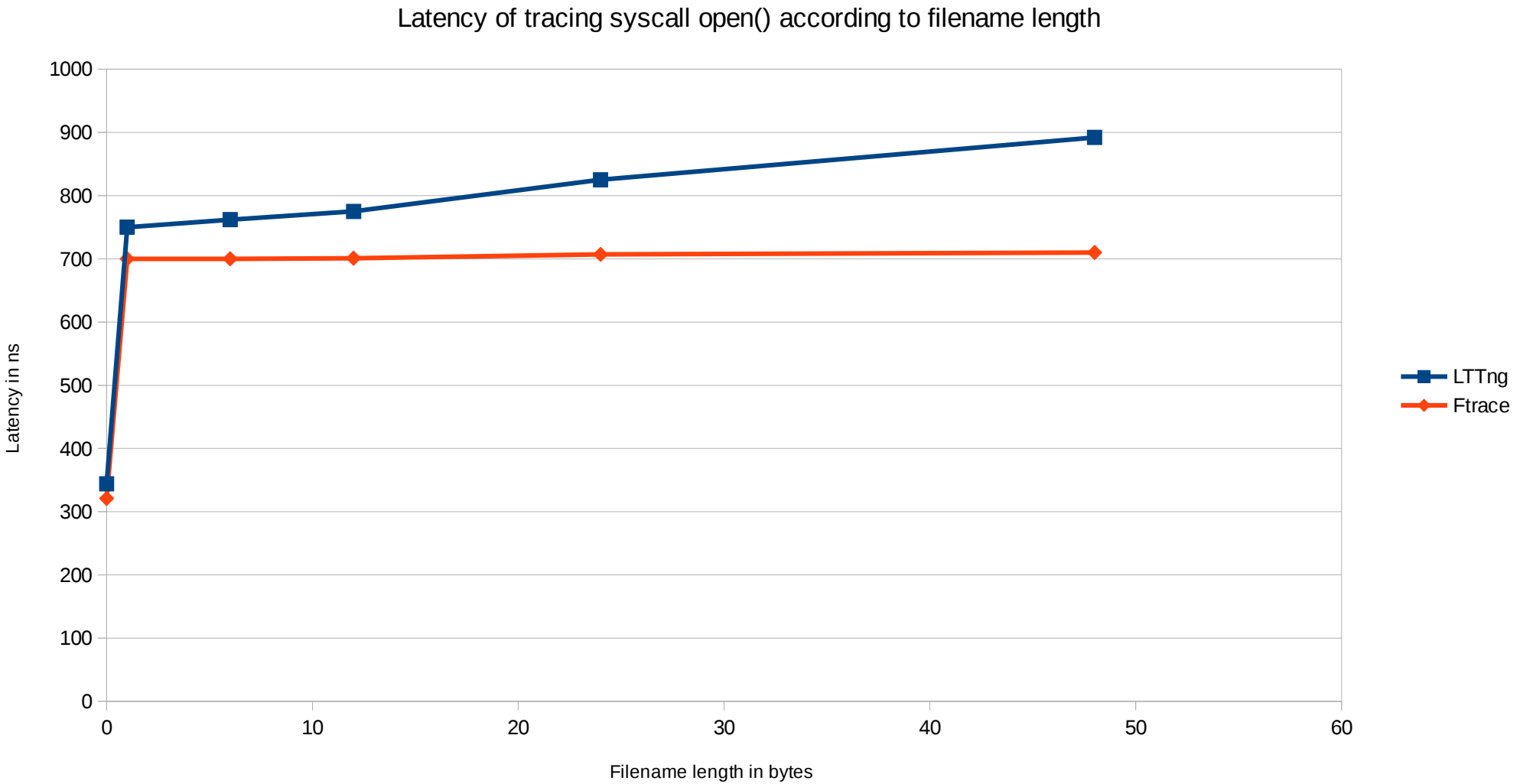
Kernel tracing: ioctl() syscall

- Final numbers:

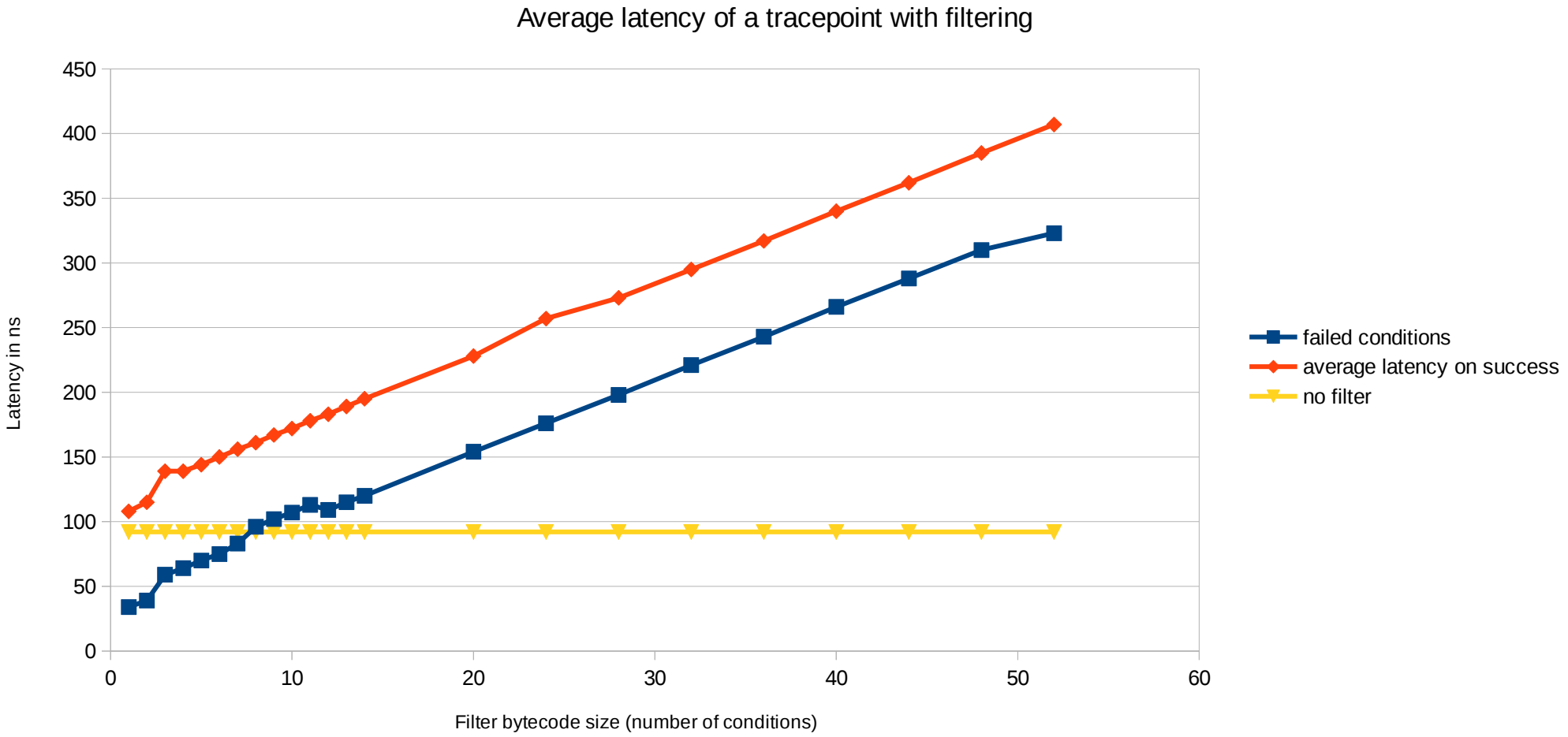
	Average		90th percentile	
	Value (ns)	Overhead	Value (ns)	Overhead
None	99	0%	98	0%
Ftrace	311	214%	302	208%
LTTng	341	244%	333	240%
Perf	600	506%	586	498%



Kernel tracing: open() syscall



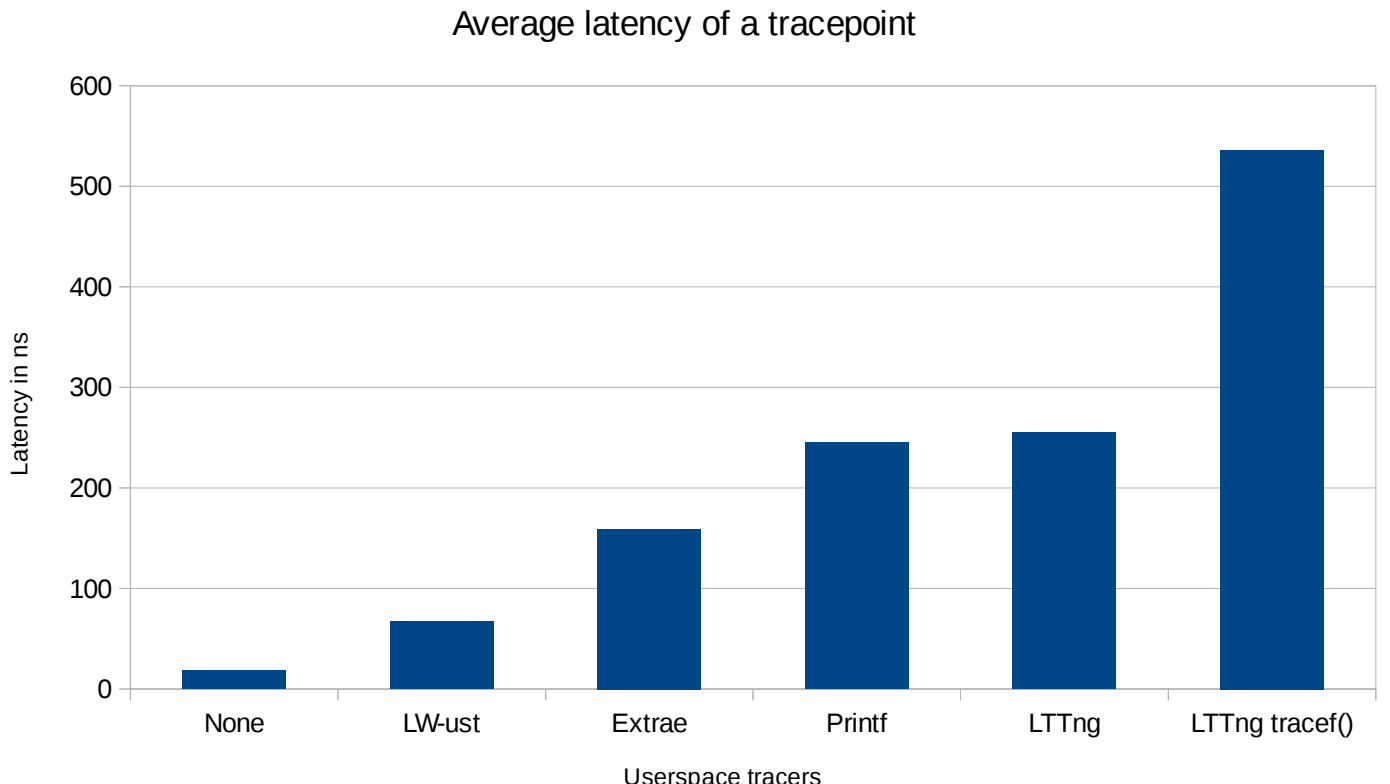
Kernel tracing: filtering with LTTng



Userspace tracing: the outline

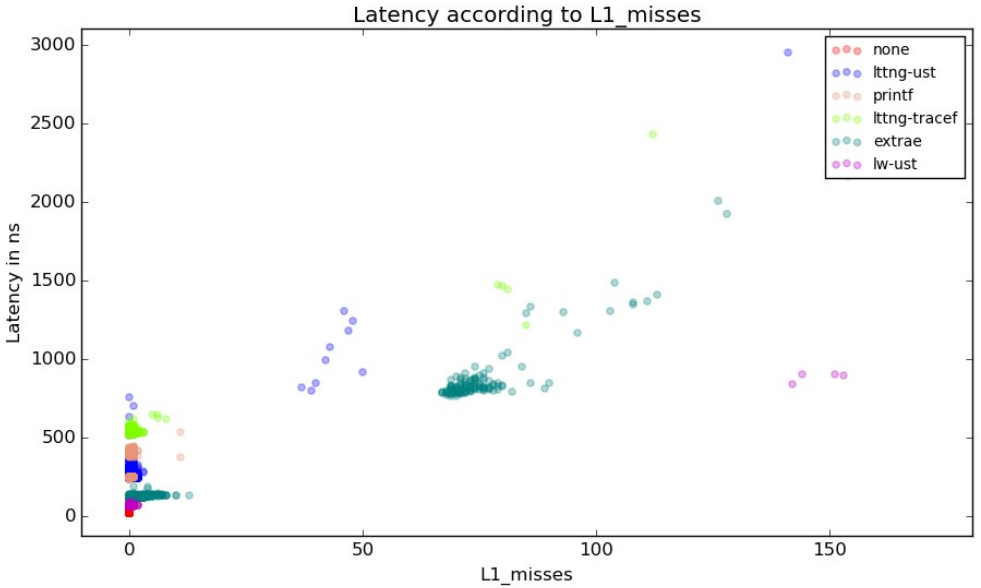
- Final numbers, **but**...
  - LTTng is signal-safe reentrant
  - LTTng is scalable

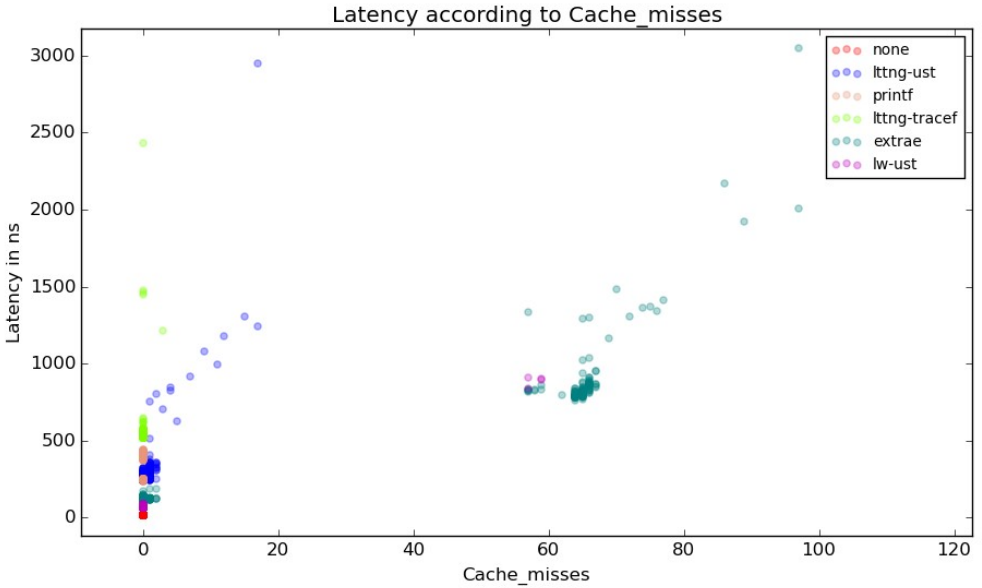
	Average		90th percentile	
	Value (ns)	Overhead	Value (ns)	Overhead
None	18	0%	17	0%
LW-ust	67	272%	64	276%
Extrae	159	783%	118	594%
Printf	245	1261%	243	1329%
LTTng	255	1317%	249	1365%
LTTng tracef()	536	2878%	523	2976%

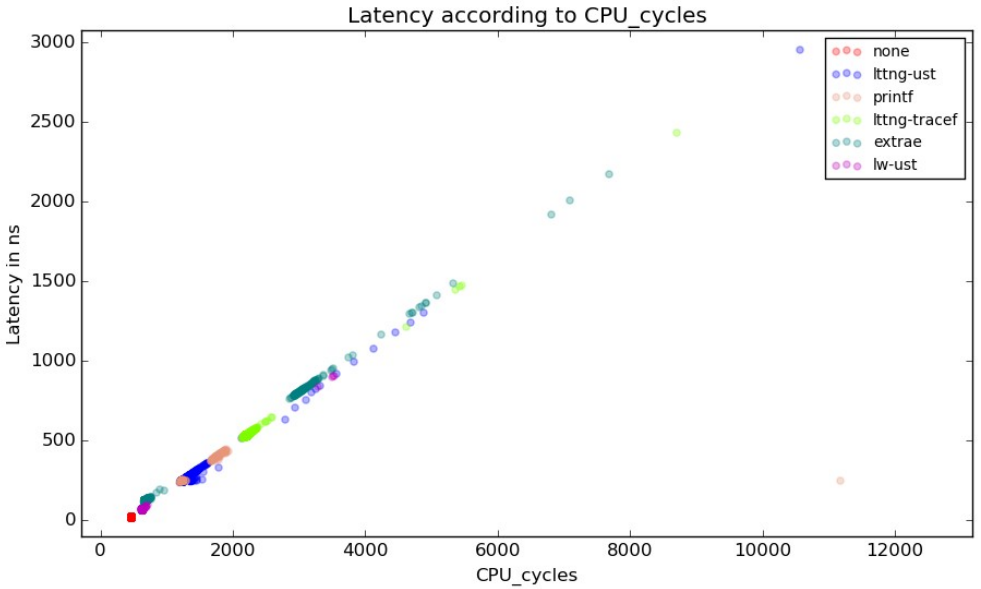


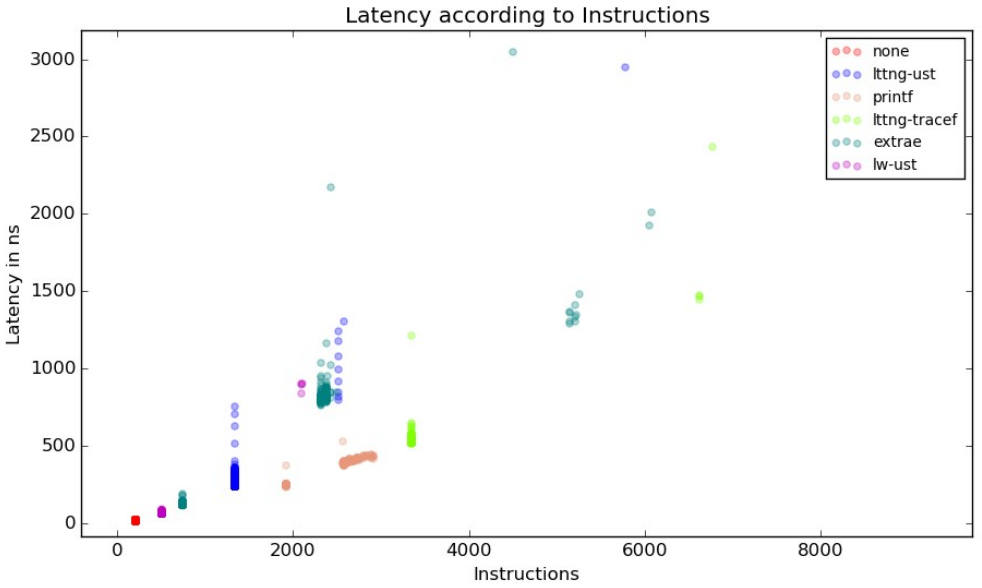
Benchmarking and compariso

Userspace tracing: closer look

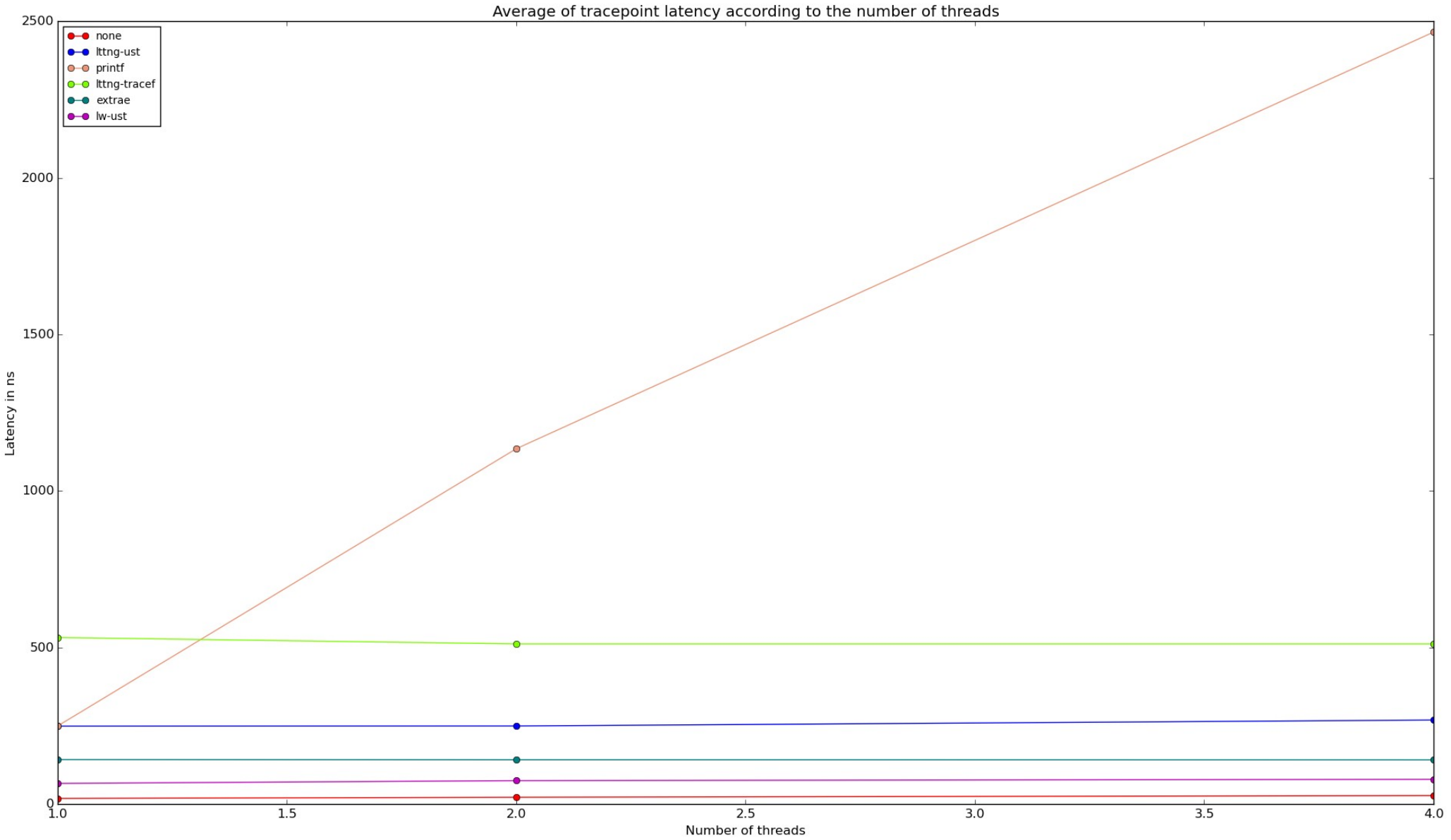








Userspace tracing: scalability



Related work

- More scalability runs
- Closer look at aggregators (eBPF, Systemtap)

Acknowledgements

- Professor Michel Dagenais
- Julien Desfossez
- Mathieu Desnoyers
- Ericsson