High-performance tracing of many-core systems with LTTng

Simon Marchi

Laboratoire DORSAL
Département de génie informatique

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Outline

- Work done
- Benchmarks
  - memcached
  - dragon
  - hello
  - packet processor
- Possible routes for improvement

Work, done and current

- Completed port of LTTng to Tilera
  - Added support for system call tracing
- Designed benchmarks to evaluate future improvements
- Performance analysis to find bottlenecks

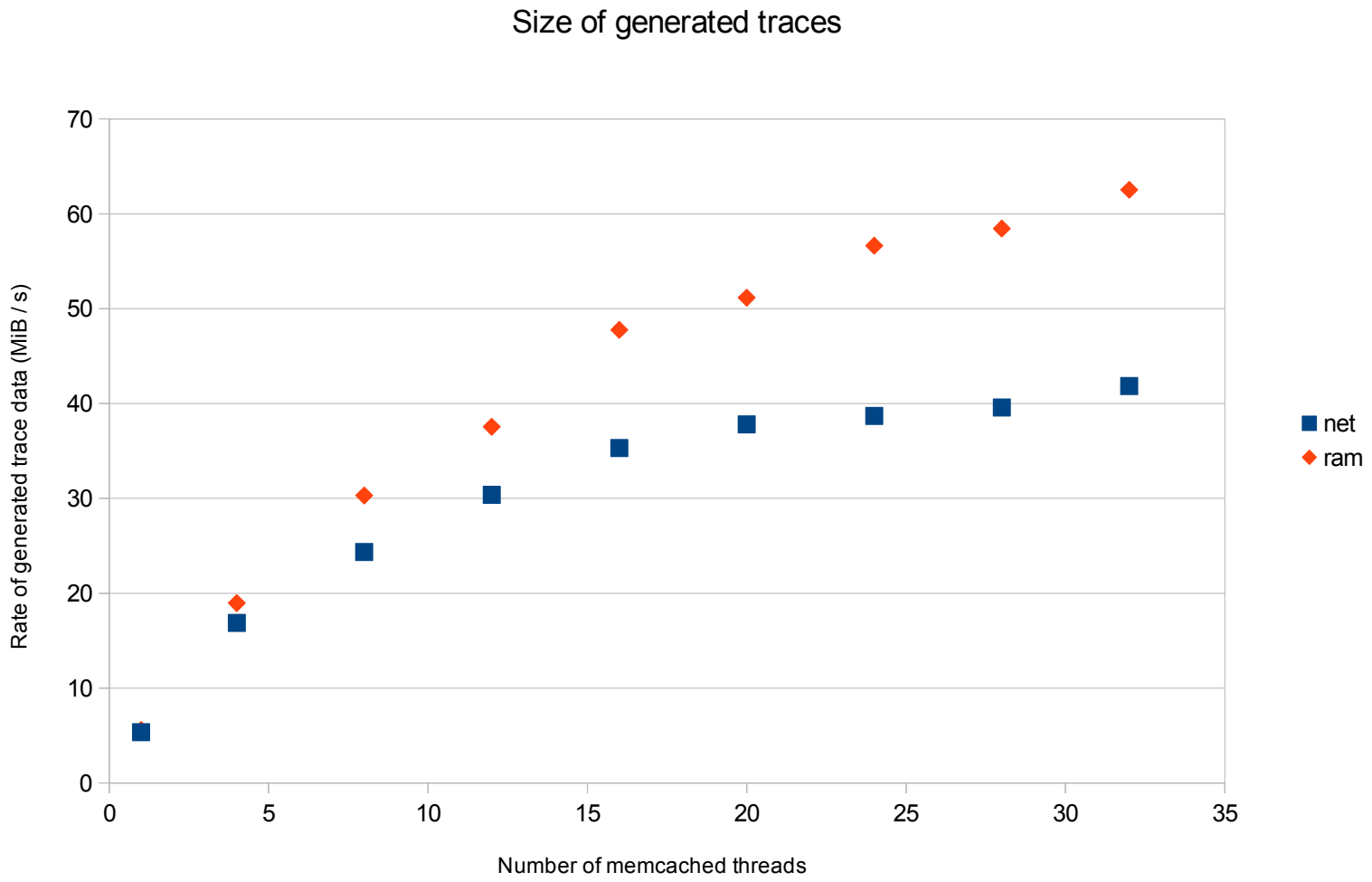
Benchmark: memcached

- Network application, a lot of interaction with the kernel: a lot of kernel events

- It is one of the featured applications on Tilera's website, so it must work pretty well...

Benchmark: memcached + tracing

Big traces !


Size of generated traces
Rate of generated trace data (MiB / s)
Number of memcached threads
0
10
20
30
40
50
60
70
0
5
10
15
20
25
30
35
net
ram

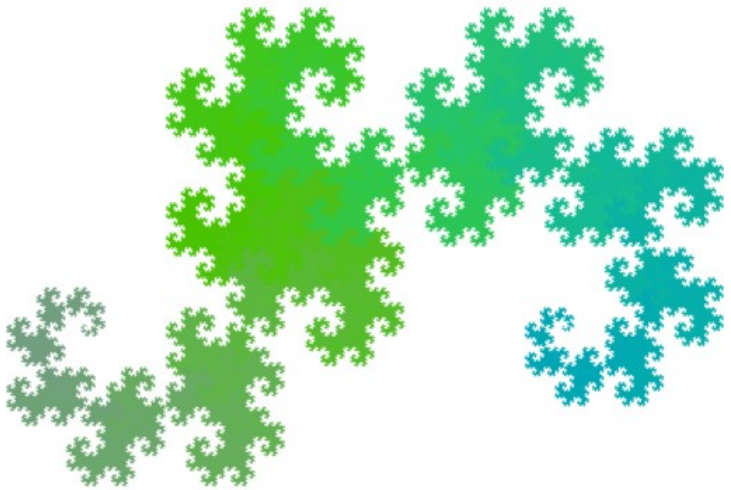
Network streaming hurts

- Saving to ram:

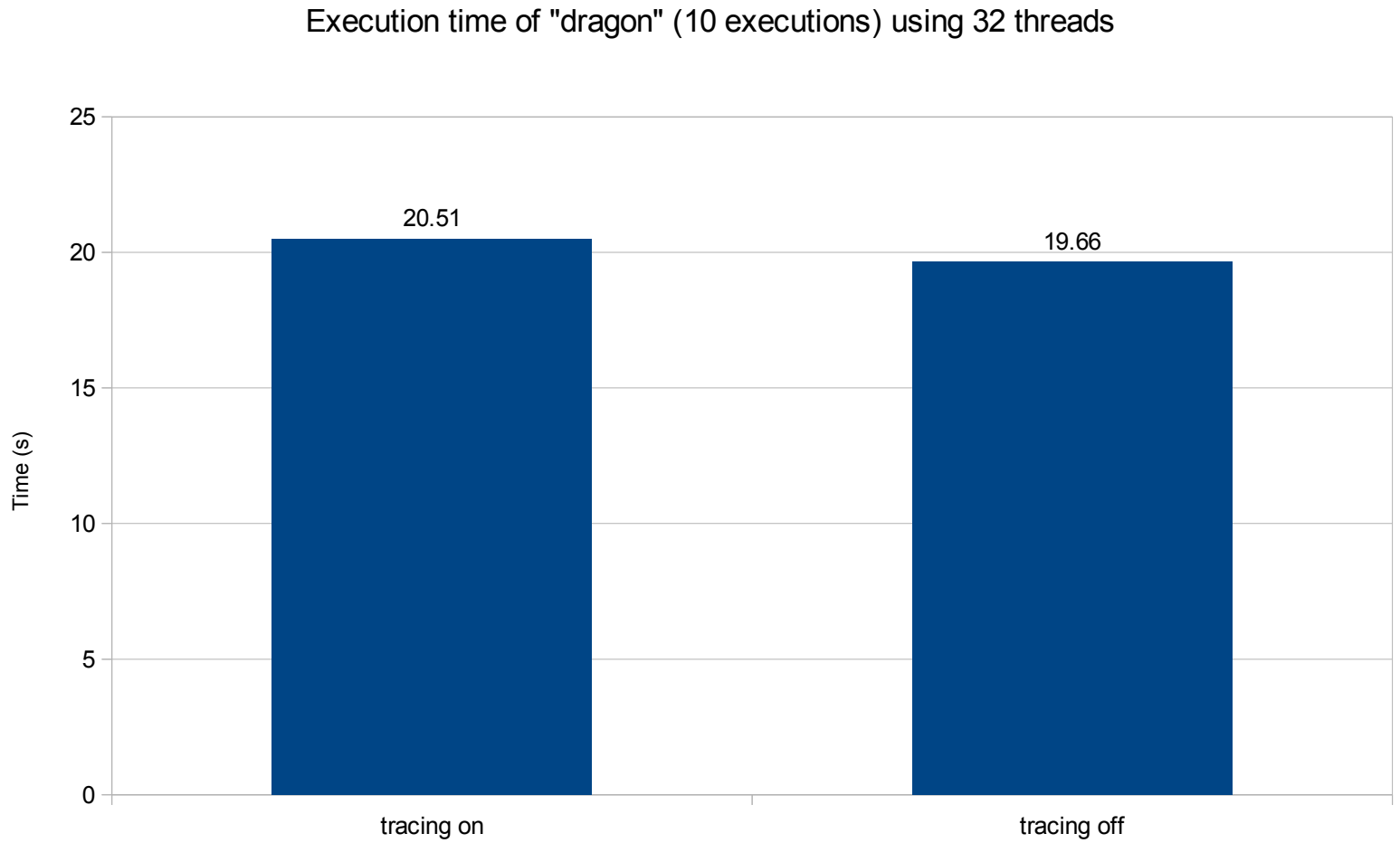
- Streaming on network:

Dragon

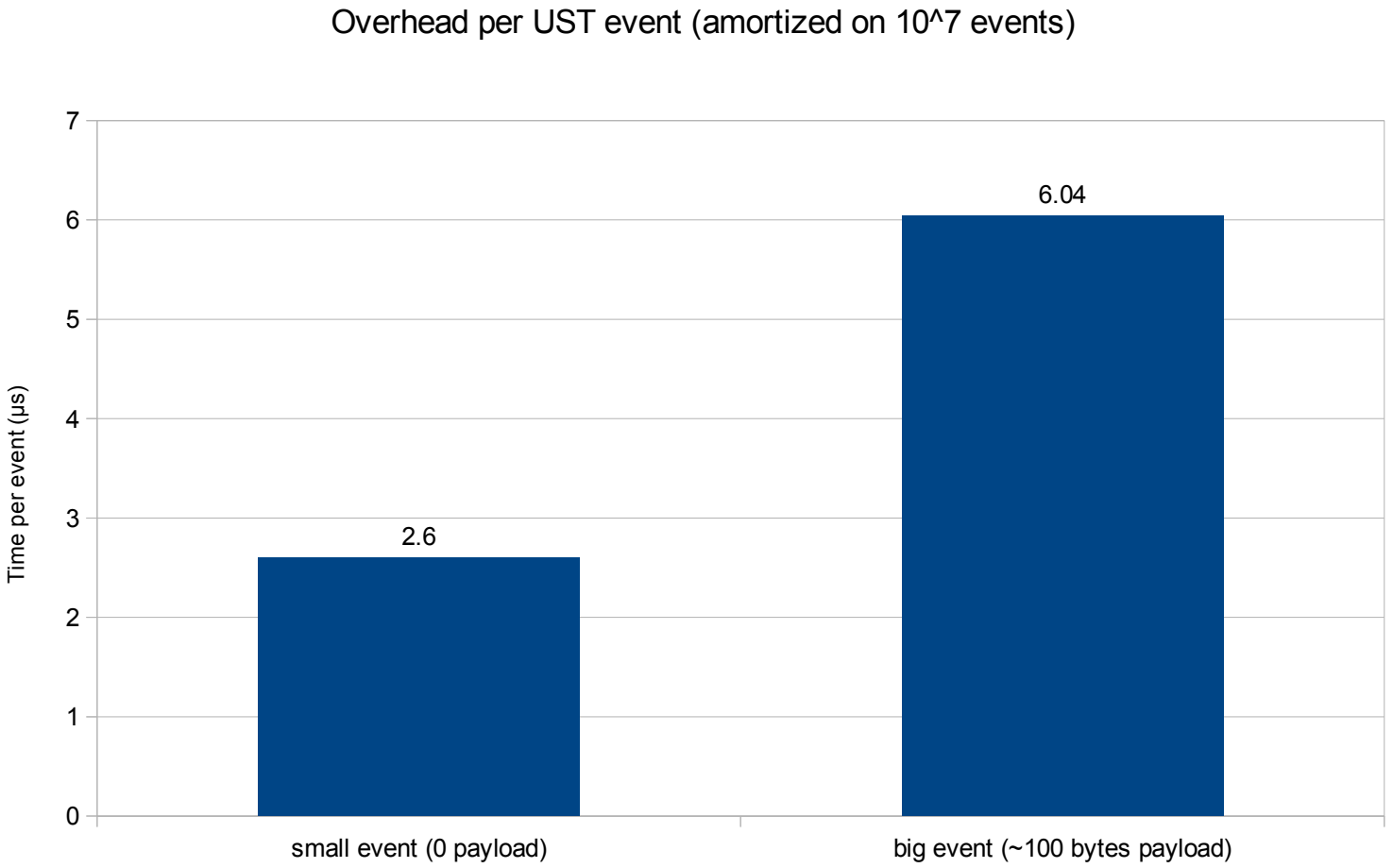
- Highly parallel, academic application that draws a fractal.
- Essentially userspace-only application, very little interaction with the kernel.



Dragon + kernel tracing

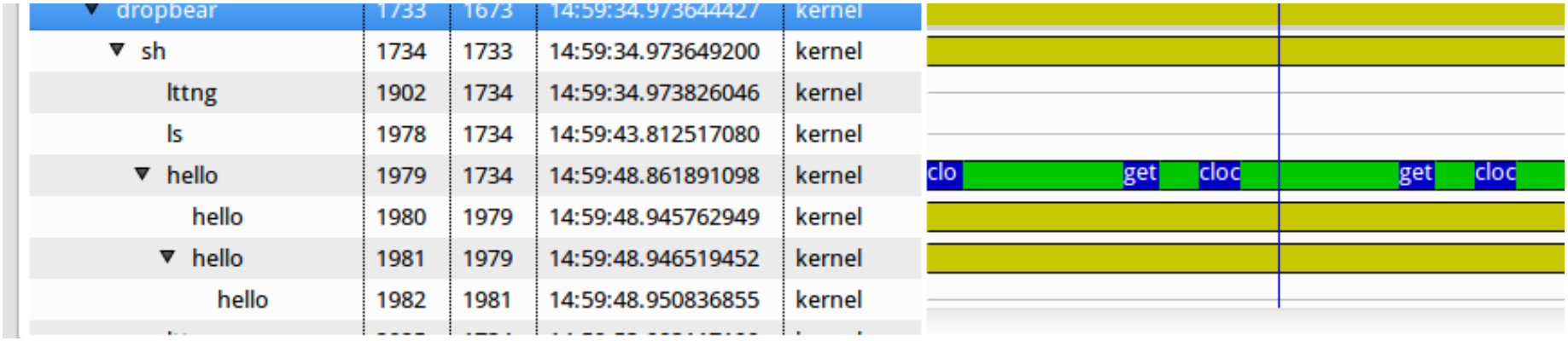


Hello UST test case



Hello UST test case

- Two system calls for each UST event
  - getcpu
  - clock_gettime



Planned: Packet processor

- Use the hardware network packet classifier available on the processor
- Write a sample application that analyzes incoming network packets
- Should be a good way to fully utilize all 36 cores

Possible routes for improvement

- Adapt caching strategy for trace data
  - Evaluate impact of tracing on the app's cache hit rate.
- Use huge pages
  - Reduce TLB stress
- Analyze memory controllers balance

Questions ?