Distributed system-level critical path analysis

Progress Report Meeting
December 11th 2014

Francis Giraldeau
francis.giraldeau@polymtl.ca

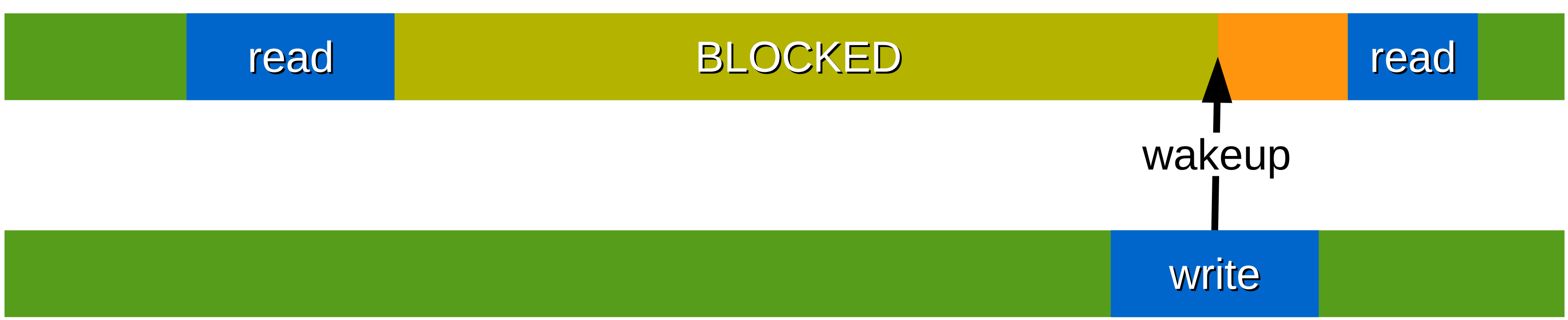
Under the direction of Michel Dagenais
DORSAL Lab, École Polytechnique de Montréal

UNIVERSAL* DISTRIBUTED PROFILER

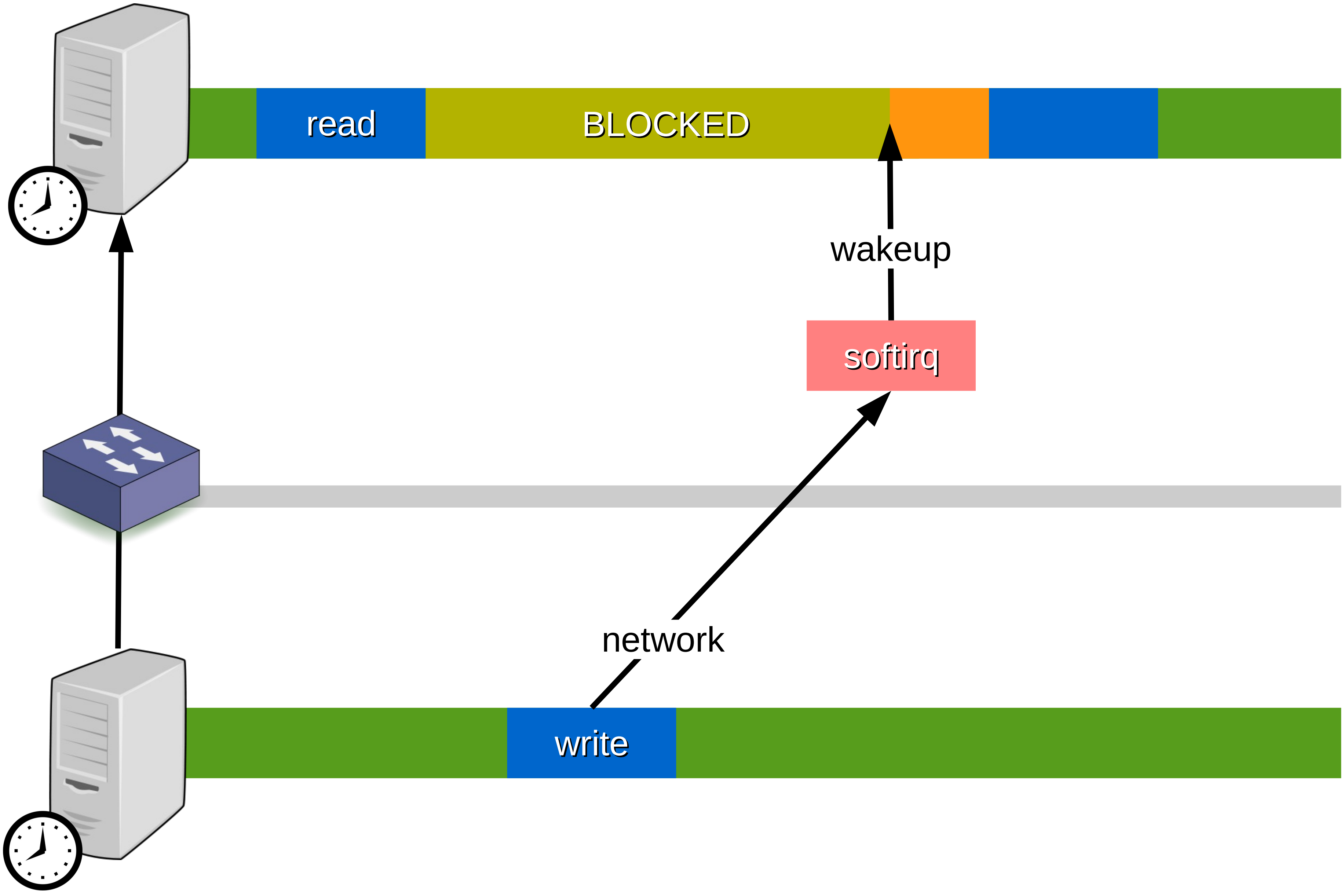
*Requires Linux

Device wake-up

Task wake-up



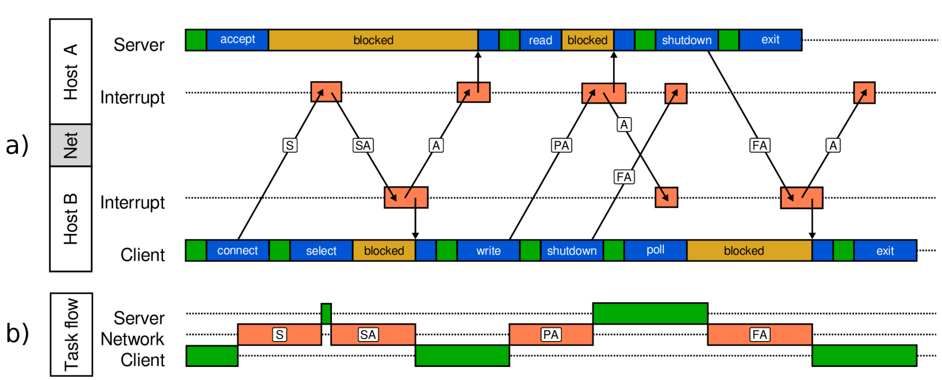
Remote wake-up



Parameters exploration

Effect of host type

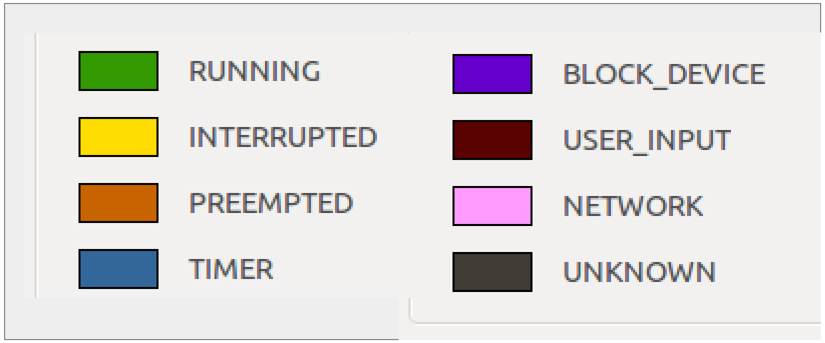
- Django web app experiment
- One host configuration
- Virtual machines
- Bare-metal



- loopback and eth0 same mechanism w.r.t. events.
- Some programs detects remote process is local and fall back to shared memory instead of socket.
- Some send/receive can be seen as preempted instead of blocked.

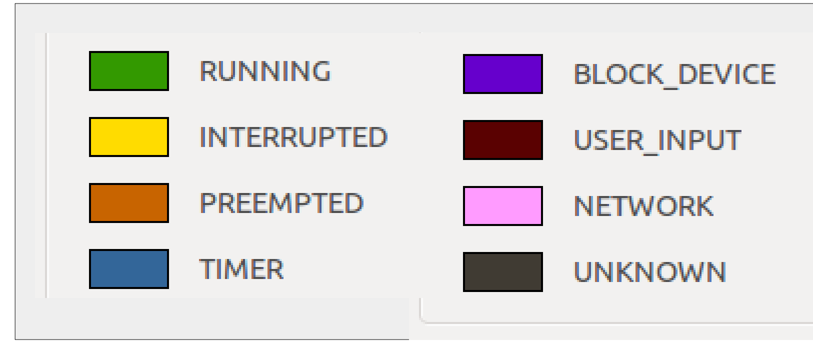
Effect of latency

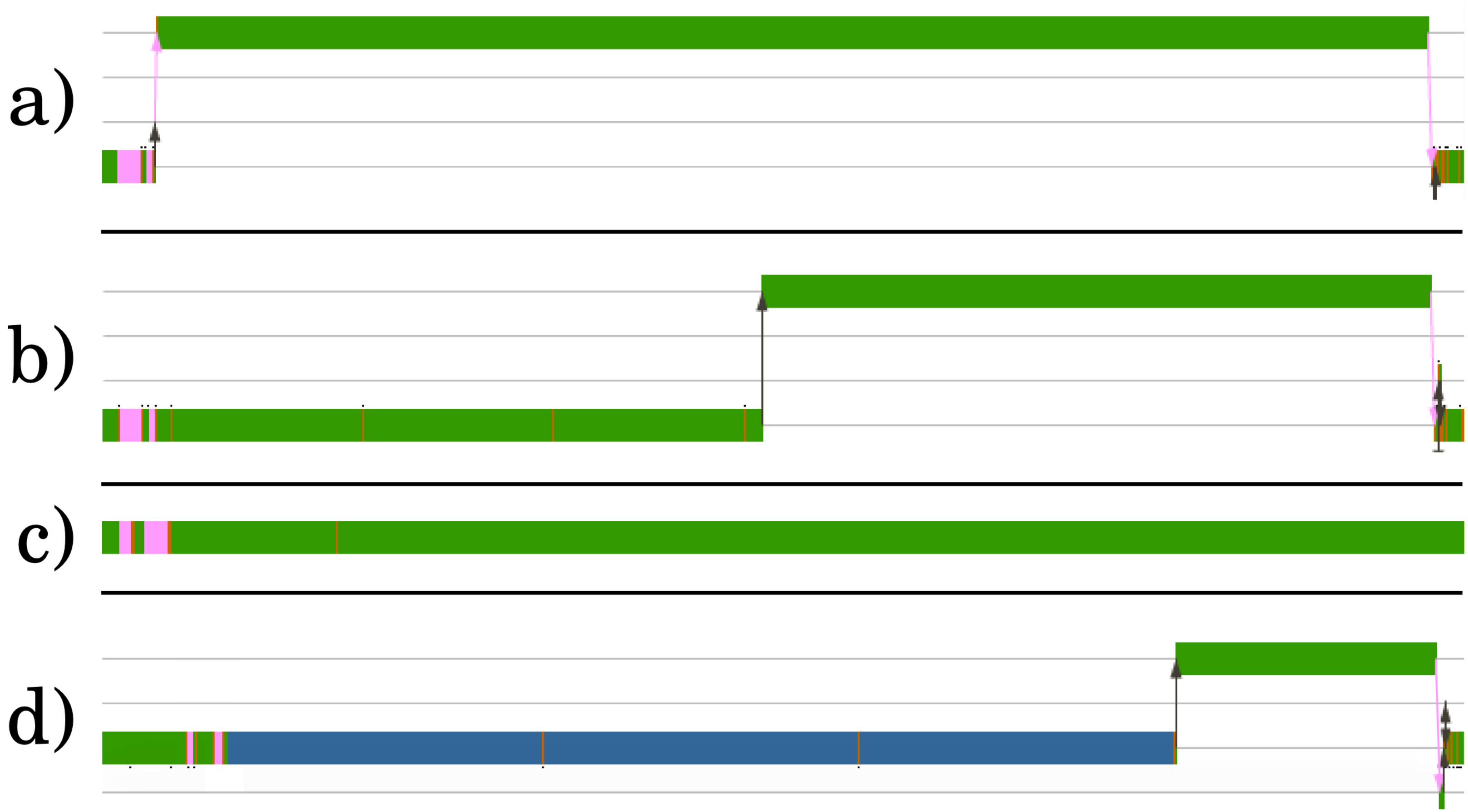
wk-rpc



Async processing

wk-rpc

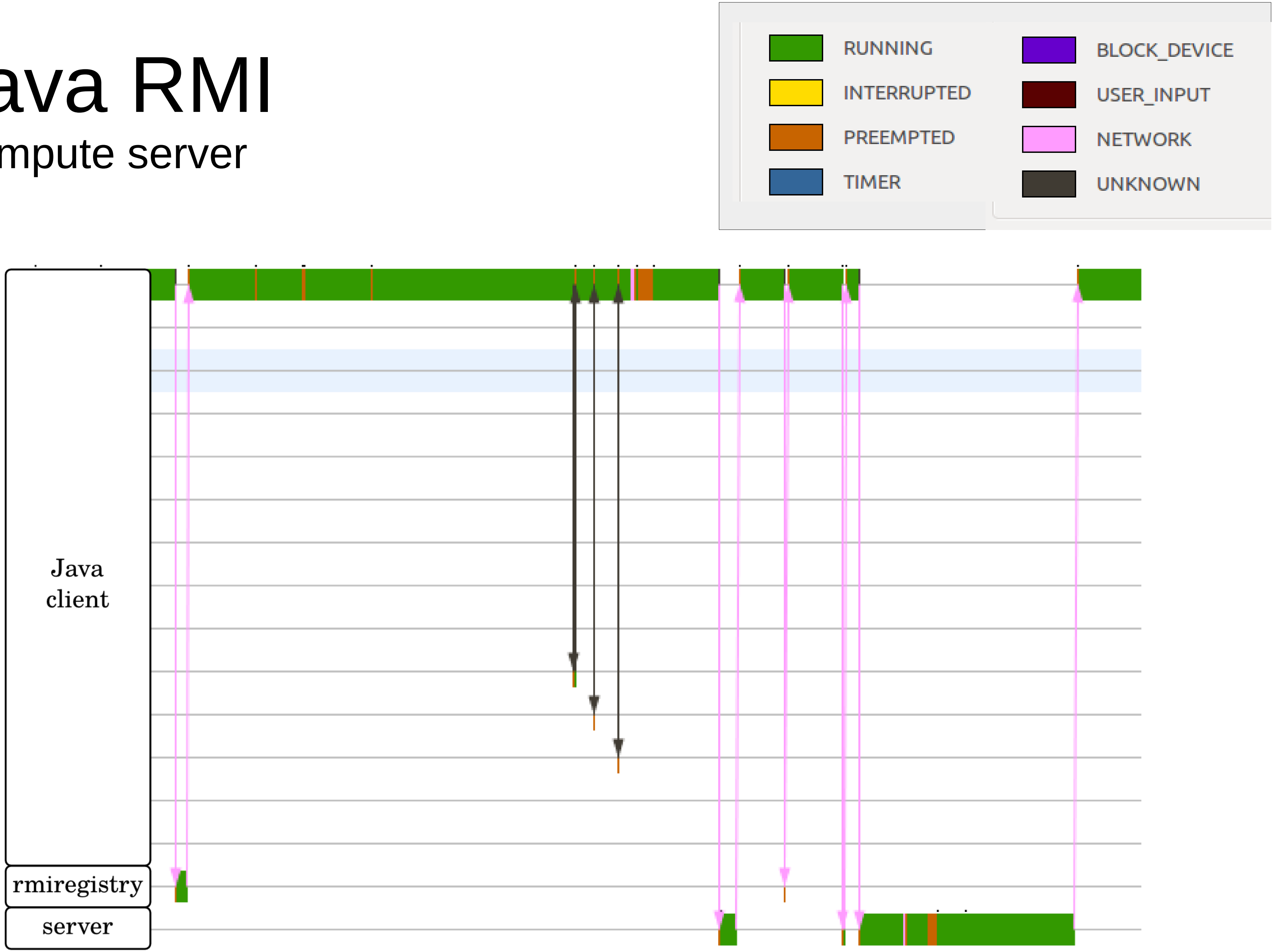




Use-cases

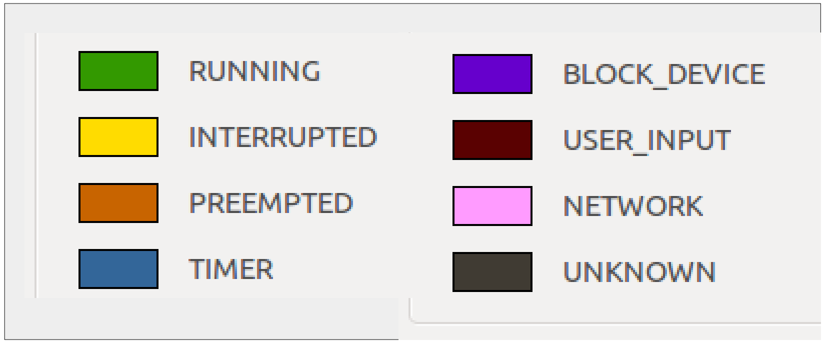
Java RMI

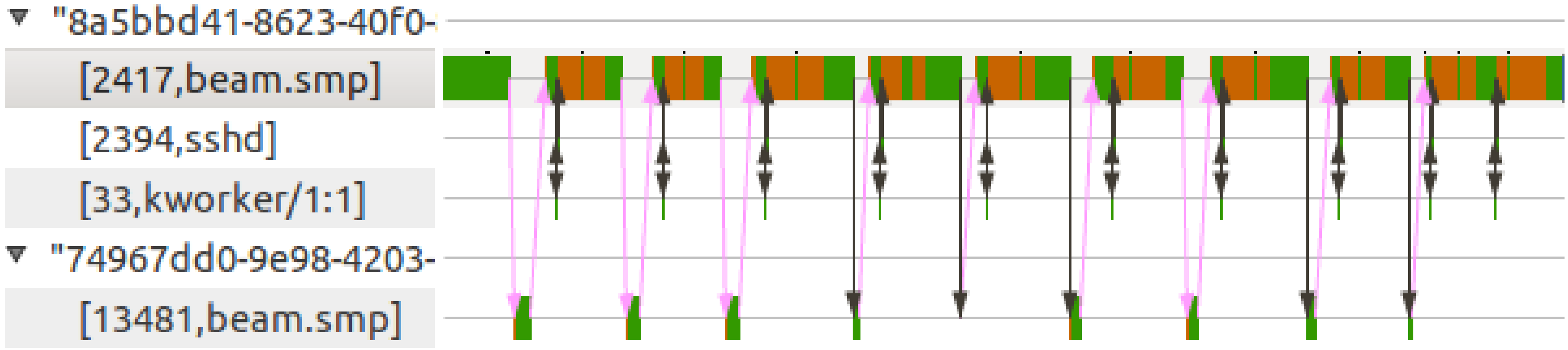
Compute server



Erlang

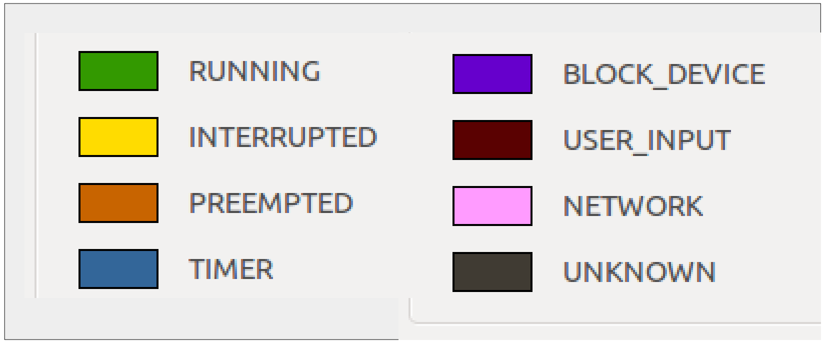
ping-pong

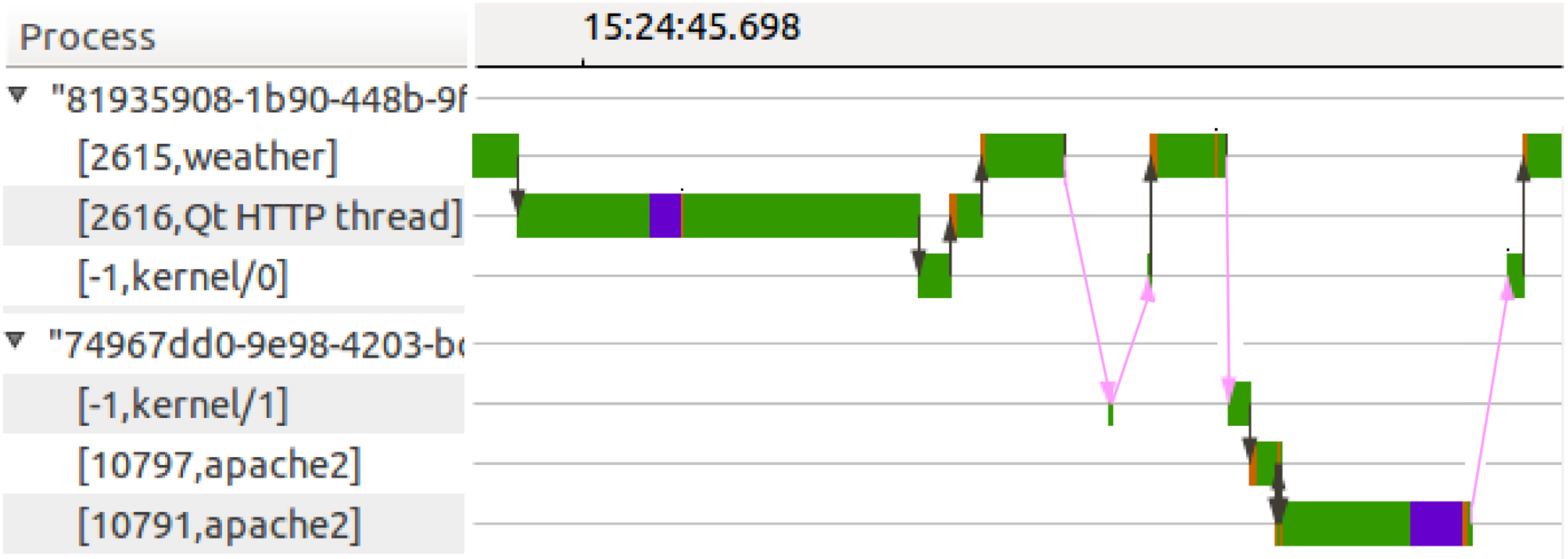




Qt5 Weather app

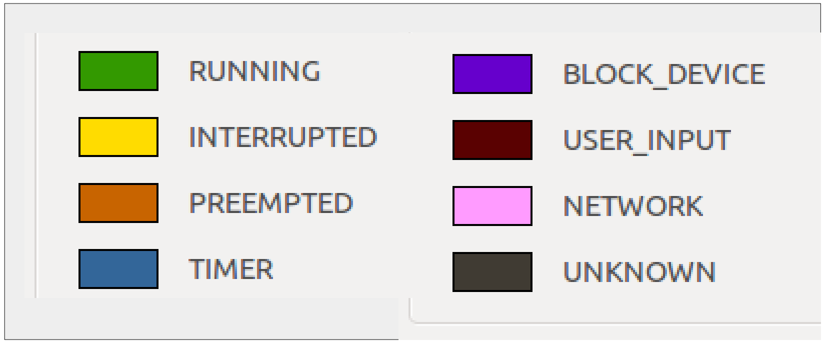
UI getting content with HTTP

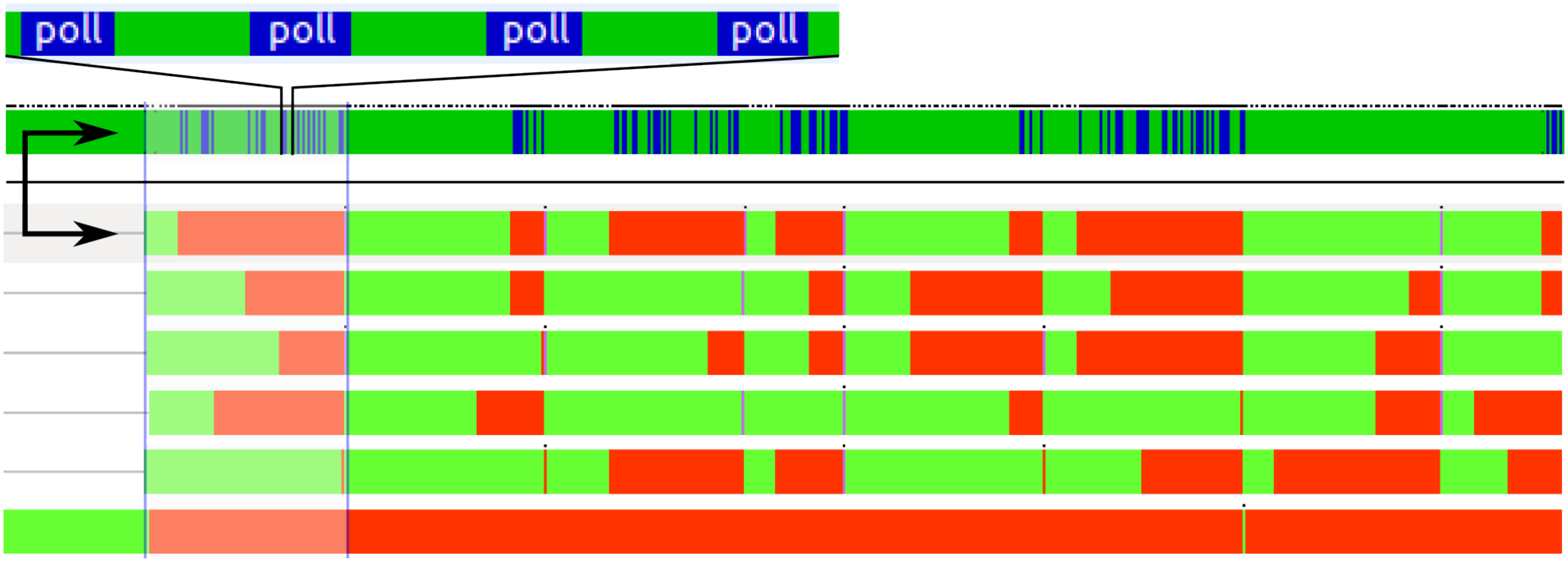




MPI Imbalance

Uneven MPI computation

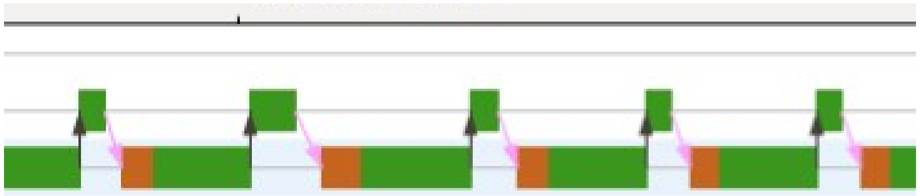




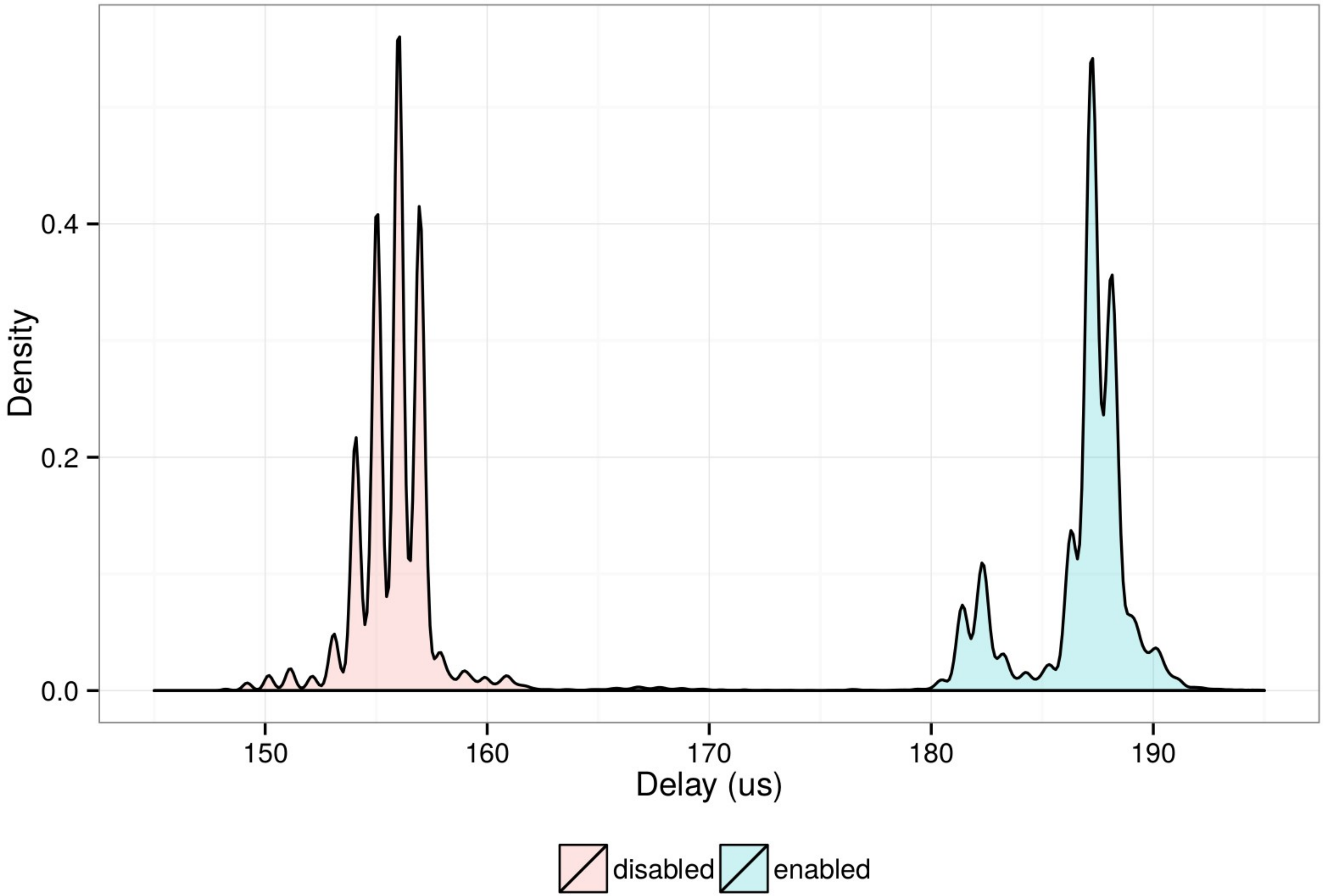
Analysis cost

Runtime overhead

- Effect on throughput
  - Saturate network link with iperf/netperf
  - Impact non-statistically significant
  - 0.1Mb/s, t(10)=0.591, p-value = 0.55
- Effect on latency
  - Design wk-rpc with tight-loop, small message
  - Simulate RPC at the highest rate possible
  - Experiment on two physical machines
  - tracing increased request latency by 18.3%, from 155.9us to 190.8us (mean difference in change, 34.9 [95% CI, 34.7 to 35.1]; p<0.01)

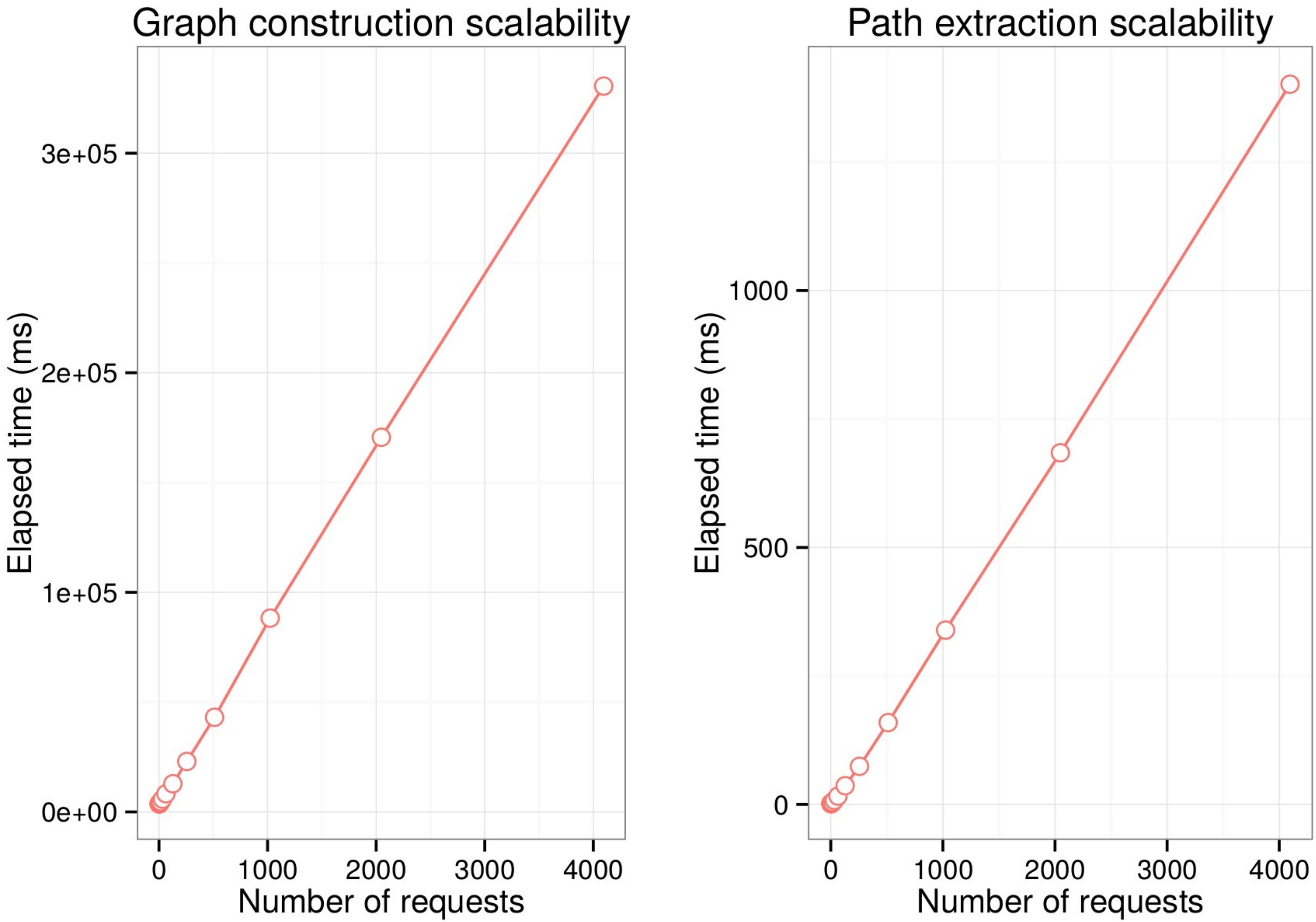


Effect of tracing on RPC delay density



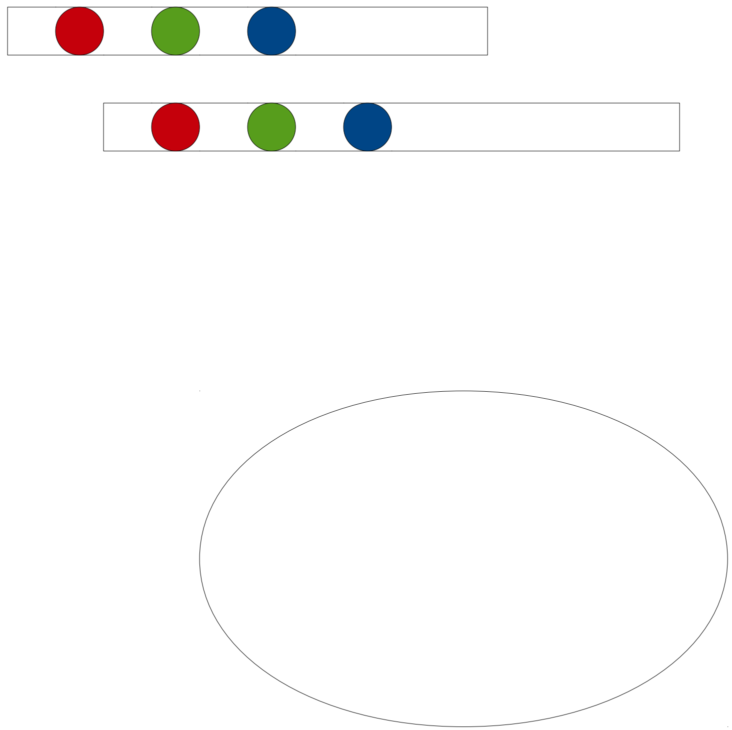
Effect of tracing on Django web app

- Poll vote simulation, automated with mechanize
- Three-tier bare-metal cluster with Apache and PostgreSQL
- Request latency increased by 5.1%, from 116.3,ms to 122.5,ms (mean difference in change, 6.3 [95% CI, 4.8 to 7.7]; $p<0.01$)
- Average CPU usage of the trace consumer daemon is comprised between 0.8% and 8.2% of one CPU, and is proportional to the event production rate ($R^2=0.91$)


Graph construction scalability
Elapsed time (ms)
3e+05
2e+05
1e+05
0e+00
0
1000
2000
3000
4000
Number of requests
Path extraction scalability
Elapsed time (ms)
1000
500
0
0
1000
2000
3000
4000
Number of requests

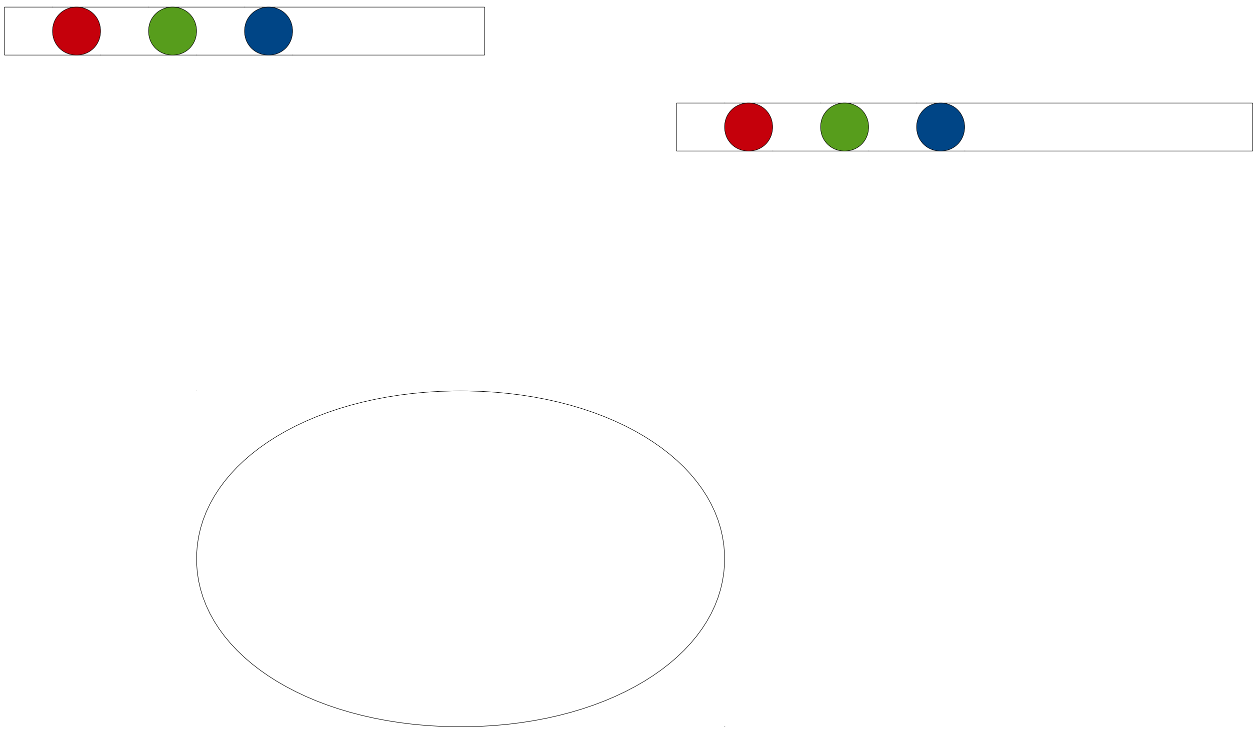
KEEP
CALM
AND
START
OPTIMIZING

Packet matching memory usage



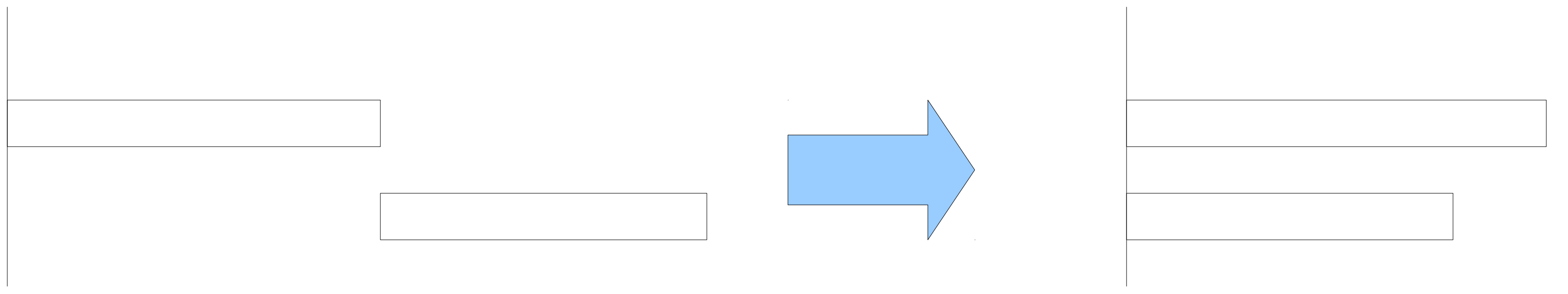
Unmatched packets

Packet matching memory usage

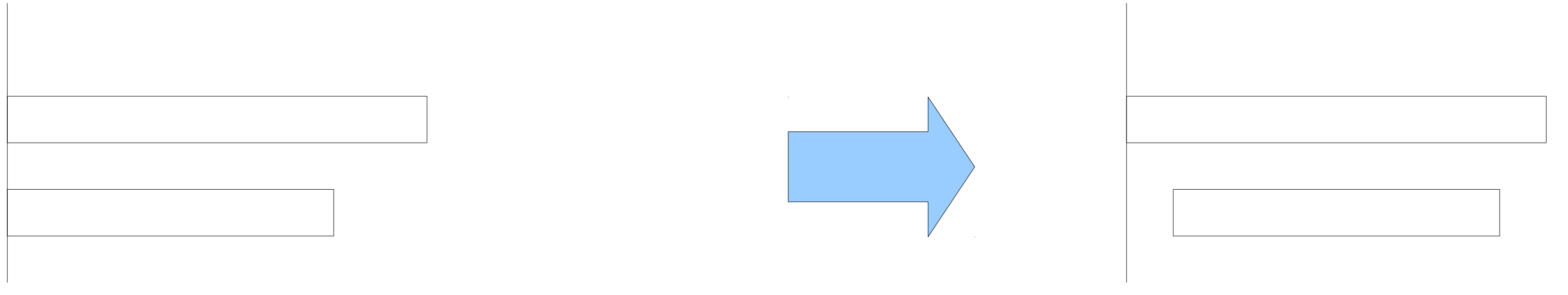


Unmatched packets

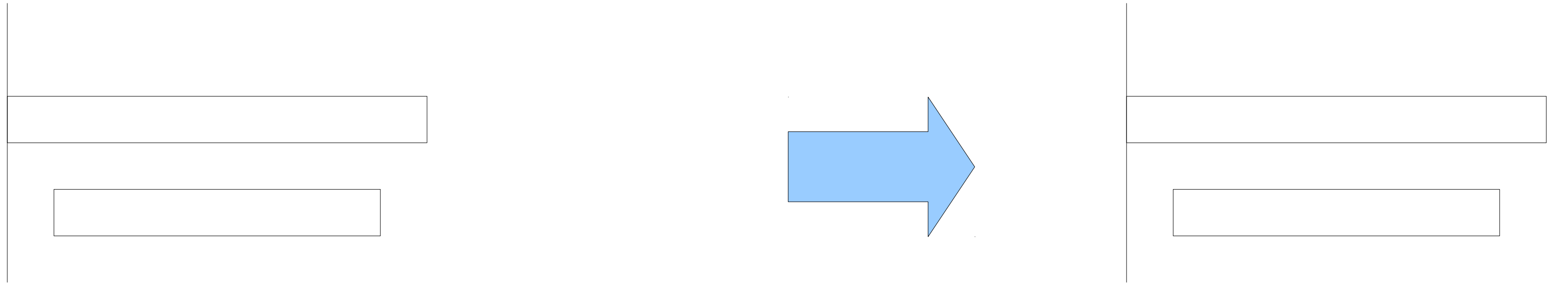
Step 1: Alignment sync



Step 2: Partial, coarse sync (1%)



Step 3: Normal sync



Partial coarse sync

- Detect when all nodes are connected
- When sync error < 1%, create edge
- Uses weighted quick-union find algorithm
- Stops when number of partitions reaches 1

Results

- Traces: django trace on three hosts
- Max memory usage reduction: 99%
- Usage nearly constant v.s. linear in worst case
  - Assumption: traces start (almost) at the same time
- Increased runtime: 16.8% [95% CI, 14.5% to 19.1%]; $t(10)=14.1$, $P<0.01$
  - Small portion of the trace is read twice

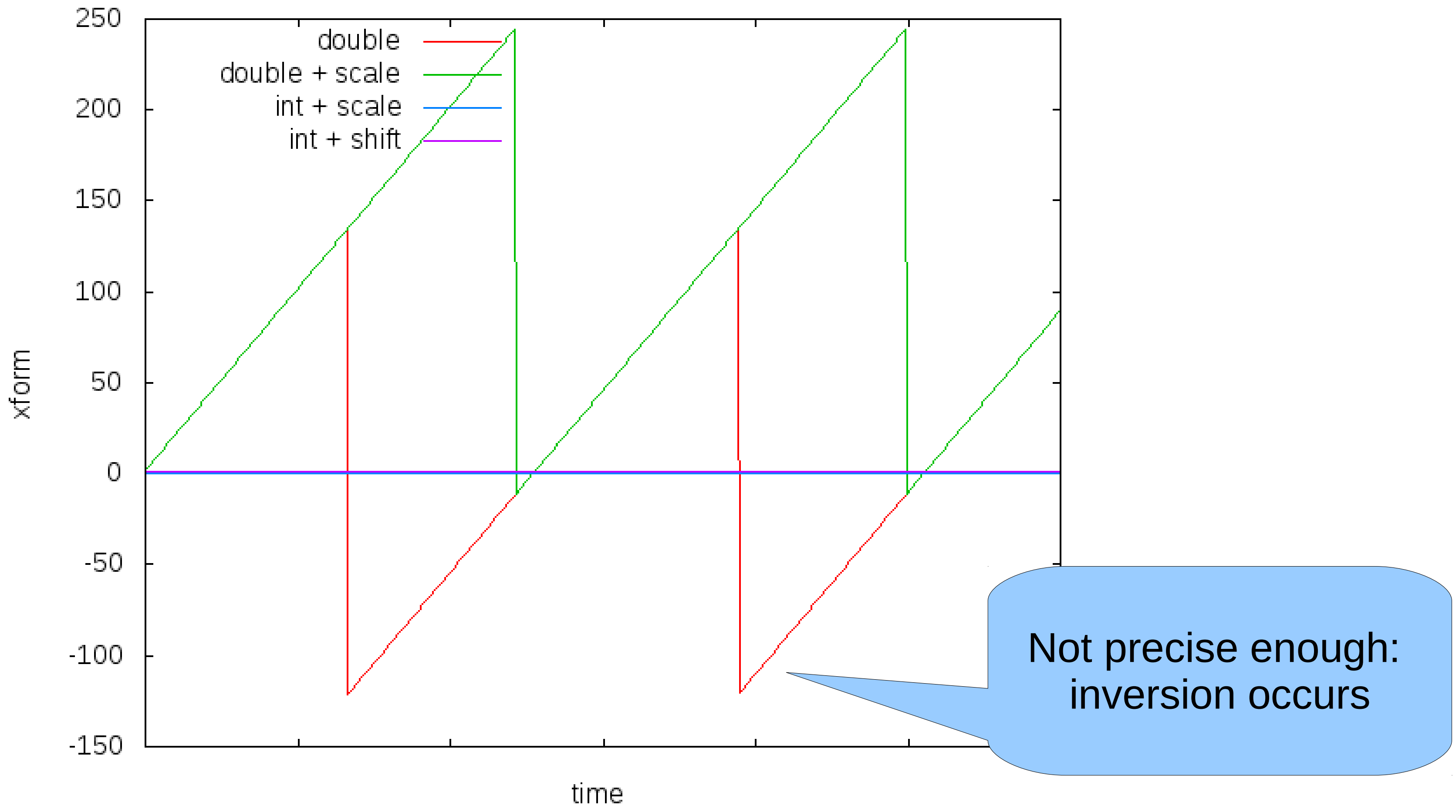
Timestamp transform

Tx = alpha * ts + beta

~ 1.0

~1.0E9

Error of floating point arithmetic according to timestamps



Fast timestamp transform

$$
\begin{aligned}
f(t) &= \frac{mc(t - t_0)}{c} + (mt_0 + b) \\
&= m(t - t_0) + mt_0 + b \\
&= mt - mt_0 + mt_0 + b \\
&= mt + b
\end{aligned}
$$

Algorithm 3 Fast timestamp transform

Input: slope m, offset b, timestamp t
Output: transformed timestamp t_x

$t_0 \leftarrow 0$ ▷ initialization
$c \leftarrow (1 \ll 30)$
$cst \leftarrow 0$
$m_{int} \leftarrow (int)(m \times c)$
$b_{int} \leftarrow (int)b$
function FAST_TRANSFORM(t)
  if ABS$(t - t_0) > c$ **then** ▷ Rescale
    $cst \leftarrow t \times m + b_{int}$
    $t_0 \leftarrow t$
  end if
  $t_{tmp} \leftarrow (m_{int} \times \text{ABS}(ts - t_0)) \gg 30$ ▷ Transform
  if $ts < start$ **then** ▷ Rectify sign
    $t_{tmp} \leftarrow -t_{tmp}$
  end if
  return $t_{tmp} + cst$
end function

Results

- Constant error (2-3ns) because of rounding
- Micro-benchmark: 155x faster than BigDecimal
- Trace reading: -20.8% [95% CI, -16.4% to -25.3%]; $t(10)=9.2$, $P<0.01$

Conclusion

- New use-cases for number of runtimes
- Sync memory usage reduced by 99%
- Fast timestamps transforms reduces analysis time

Thanks to Professor Michel Dagenais and our partners EfficiOS and Ericsson.

Special thanks to Geneviève Bastien for her excellent work on Trace Compass.

Software:

http://secretaire.dorsal.polymtl.ca/~fgiraldeau/workload-kit/

http://secretaire.dorsal.polymtl.ca/~fgiraldeau/traceset/

https://github.com/giraldeau