Progress Report May 2015

MATTHEW KHOUZAM,
ERICSSON

PRESENTER

- Matthew Khouzam
  - Software Developer at Ericsson
  - Former Research Associate at Polytechnique
  - Fan of tracing!

AGENDA

1. Trace Compass Update
2. How Kernel Tracing Accelerated a Java Application
3. Q&A

TRACE COMPASS

- Framework to build trace visualization and analysis tools
- Scalable: handle traces exceeding memory
- Extensible for any trace or log format
  - Binary, text, XML etc.
- Reusable views and widgets
- Available as standalone application or set of plug-in

COLLABORATIONS

- Contributors for Trace Compass
  - Polytechnique Montreal
  - EfficiOS
  - Kalray in France
  - CEA in France (Papyrus – Modeling integration)
  - Windriver
  - And more!

COLLABORATIONS

- Trace Research Project
  - Academia: Polytechnique Montreal, Concordia, others
  - Industry: Ericsson, DRDC, EfficiOS, others
  - Government: NSERC, others
- Contributions from the Trace Research Project
  - Generic State System
  - Synchronization of traces from multiple nodes
  - Data driven analyses and views
  - Analysis of traces in virtualized environment
- Upcoming contributions
  - Dependency analysis
  - Critical Path

NEW IN TRACE COMPASS

- Trace Compass!
- Backwards compatibility
- Stand alone application
- Initial Support of Perf CTF Traces
- Import improvements (remote/ from an archive)
- Events table enhancements (search/colors/fonts...)
- Control Flow view filter inactive processes
- Aligned Time Axis
- Constant time offsetting
- Aspects
- CTF Parser improvements

MORE COLLABORATION WITH EFFICIOS

- Work towards integrating Python analysis

STABILITY IMPROVEMENT

- Unit test improvement
- Static analysis improved general code quality
- UI functional/integration testing improved

PERFORMANCE

- Various CTF parser performance improvements
- State System performance improvements

THANK YOU

- Thank you all
- Thank you Alexandre
- Thank you Geneviève
- Thank you Francis

PART 2:

How Kernel Tracing Helped Solve Performance Issues in a COMPLICATED Java Application

CTF IMPROVEMENTS

- Work with profilers was useful
- Code is mostly single threaded
- I know the code well
- Tracing will not yield immediate benefits*

*Please prove me wrong, I want to be wrong

TRACE COMPASS

- 120 KLOC, can anyone know all of it?
- Profiling yielded some low hanging fruits

  BUT!

SOME ODD BEHAVIOUR

- Thread to write to disk was bottleneck
- Assumption that we were IO bound as the writing code is so “simple”.

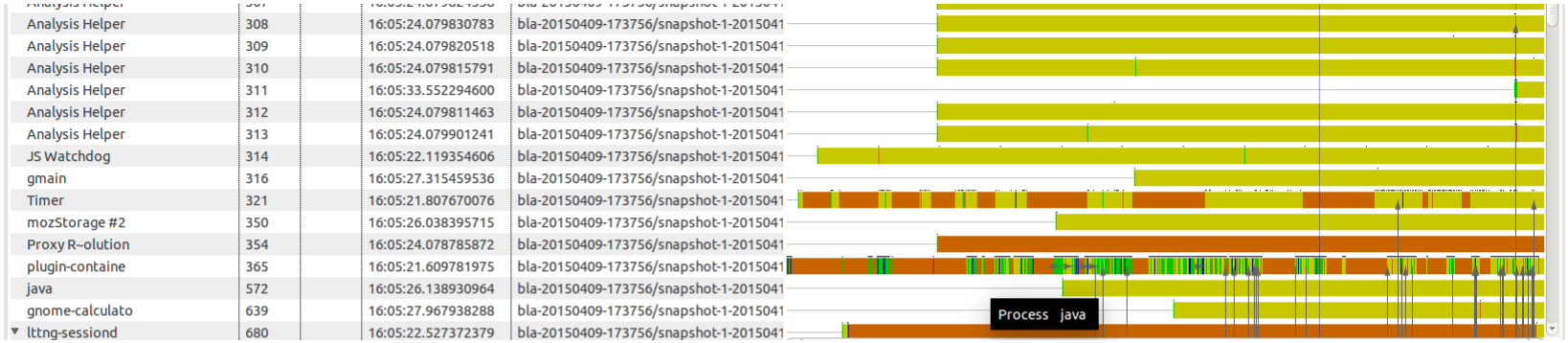
  BUT

LET'S TRACE IT!

- Kernel trace showed only 1% of time spent writing to disk.
- This needs more investigation

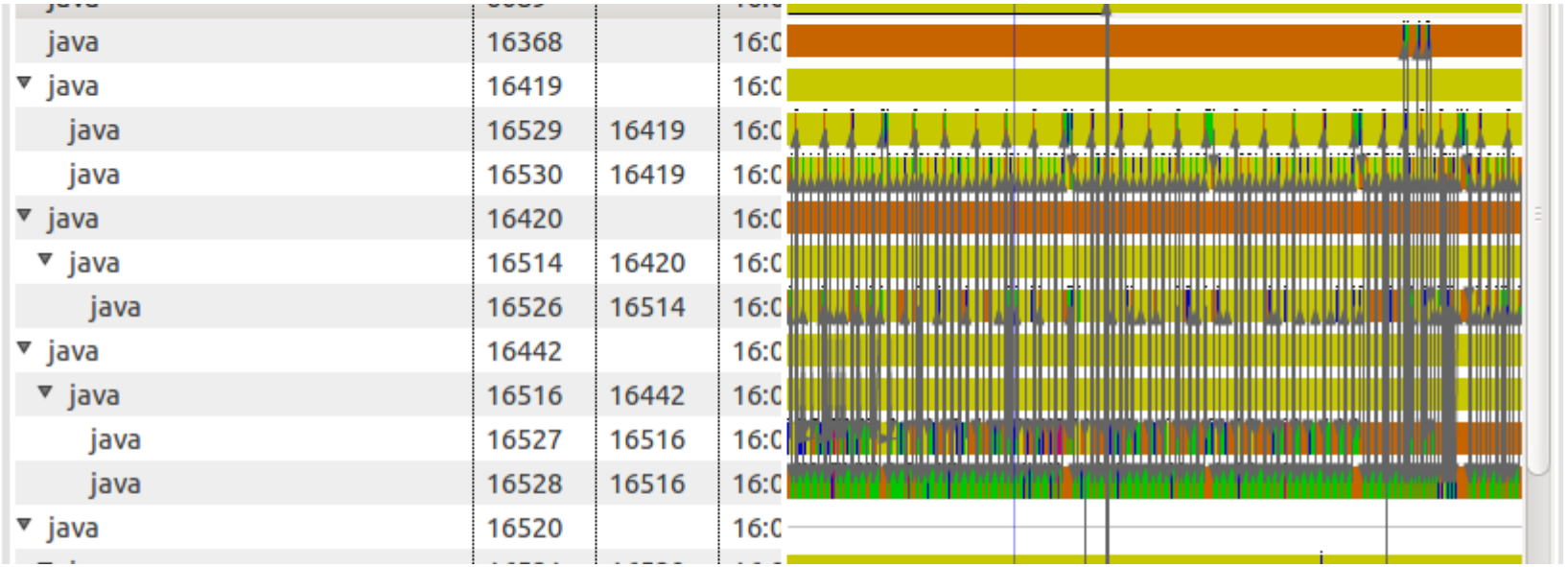
TRACE ANALYSIS

- Eclipse is strongly multi-threaded
- In the kernel, every thread is called “java”
- The Control Flow view is not very useful unless you know what you're looking for.



THANK YOU FRANCIS

- The critical path view on the other hand isolated the problem in 4 minutes
- We isolated the disk writer thread and saw

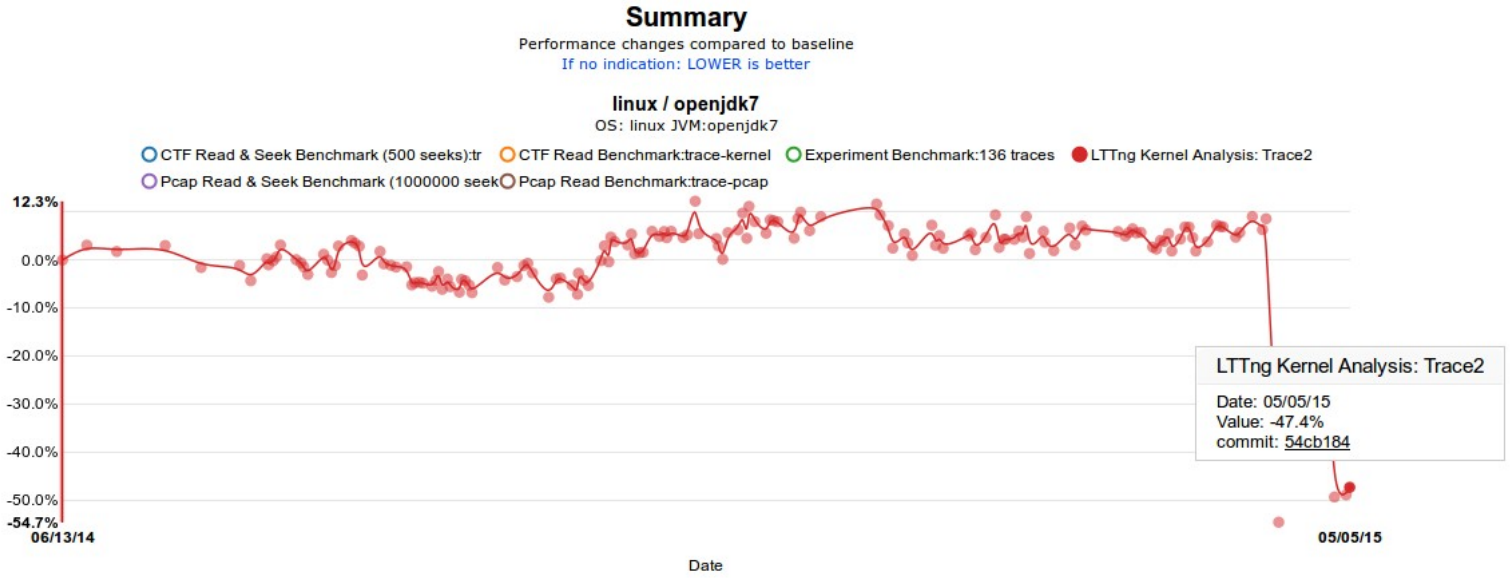


ROOT CAUSE

- Every interval (~20 bytes) is placed in a buffer and blocks the thread. This takes a futex.
- >70% of the time of the state system is spent in that futex!
- Solution: aggregate the intervals before putting it in the queue
  - Originally 20 loc yielded 40% performance improvement in trace compass
  - Now ~1600 loc (generalized solution)

IS TRACE COMPASS FAST YET?

- No, but we're getting better



TAKE AWAY

- Profilers are good.
- Tracing, is good too!
- They are different tools and should be considered complimentary
- Trace Compass has started drinking it's own champagne
- A quick win in very threaded systems would be to check context switches/second

REFERENCES

- Project pages
  - Trace Compass Project Page
  - Is Trace Compass Fast Yet?
  - LTTng Project
  - Polarsys
- Download
  - Trace Compass Downloads
- Documentation
  - Trace Compass User Guide
  - Trace Compass Developer Guide

Q&A