Performance Comprehension at WiredTiger

Alexandra Fedorova, **Craig Mustard**, Ivan Beschastnikh, Julia Rubin,
Augustine Wong, Svetozar Miucin, Louis Ye
University of British Columbia

Performance is an important software feature for databases, key-value stores, and more.

Prior work on Performance Bugs

- Lots of tools to debug performance
- Prior Studies: How are performance bugs...
  - ... different from other bugs? [Zaman et al. 2012]
  - ... identified by developers and users? [Nistor et al. 2013, Song et al. 2014, Liu et al. 2014]
  - ... caused by developers? [Jin et al. 2012]

But no one has looked at...

How do developers come to understand the underlying cause of performance issues?

"Performance Comprehension":

The process of diagnosing and mitigating performance bottlenecks in software.

This study:

How do developers approach performance comprehension?

Why study performance comprehension?

Before we can build tools that truly help, we need to understand what developers are doing today.

Questions not previously asked:

- → What information do they look for?
- → What tools do they use?
- → What is their overall approach?

Research Questions

How do developers understand the underlying cause of performance issues?

RQ1 / Info: What information is needed and collected?

RQ2 / Tools: What tools are used?

RQ3 / Process: What processes do developers follow?

Subjects: WiredTiger

Software:

- ➔ High-performance Key-Value store (multithreaded C library)
- ➔ Good performance is crucial for success.
- ➔ Carefully designed to use all available CPU cores, RAM, and parallelism.
- ➔ Company acquired by MongoDB in 2014
  - ◆ Now default storage engine since 2015.

Team:

- ➔ Small (24) team, distributed across the world.
- ➔ Open source development practices.
- ➔ Allows us to study their development practices in depth.

JIRA tickets as the source of information

- Main written communication mechanism among developers
  - (confirmed by developers)
- All developer work becomes a JIRA ticket
- Exclusive. No Slack / IRC.
  - Email/Phone/Video conversations are summarized in JIRA.

Personally, I comment in tickets for a few purposes. Some of it is so that we have a history and can revisit "why did we do that?" at a future time after we've (or I've) forgotten the details. Some is so that others can reproduce what I'm doing and see (or not) the same results. Some is to validate or simply communicate how I'm attacking a problem.

- WiredTiger Developer

Methodology: Data Collection

- JIRA tickets that contained “performance” in subject or body.
  - Original issue often came from inspecting automated performance tests.
- Analyzed all comments in 44 JIRA tickets
  - Stopped after conceptual saturation

Tickets characteristics:

Several comments:	(min: 1, max: 39, median: 6.5)
Several developers:	(min: 1, max: 7, median: 3)
Issue word length:	(min: 13, max: 394, median: 86.5)
Comment word length:	(min: 3, max: 915, median: 60.5)

- Top 4 developers > 10 tickets, 16 developers overall.

Methodology: Data Analysis

- ➔ Open coding applied to comments on the tickets
  - ◆ At least two authors per ticket
- ➔ Codes linked directly to quotes from each ticket
  - ◆ Marked quotes per ticket: (min:1, max: 106, median: 14)
- ➔ Met weekly to reconcile codes and used tools
  - ◆ Tools aggregated code occurrences across tickets
- ➔ Began with 36 tickets, and then coded another 8 tickets to validate codes.

- ➔ Member check:

I'm frankly surprised how accurately you managed to capture the [performance issues] in the paper - it did an excellent job of defining the categories.

- *WT Dev*

Threats to Validity

- Analysis was done by the authors, different people might see different codes.
- Practices used by other companies and teams may differ.
  - Other teams may have better event tracing tools
- We studied tickets from 16 experienced developers
  - Who were developing high performance multithreaded software in C
- At least some of the typical practices for performance comprehension.

Main Findings

Main Findings

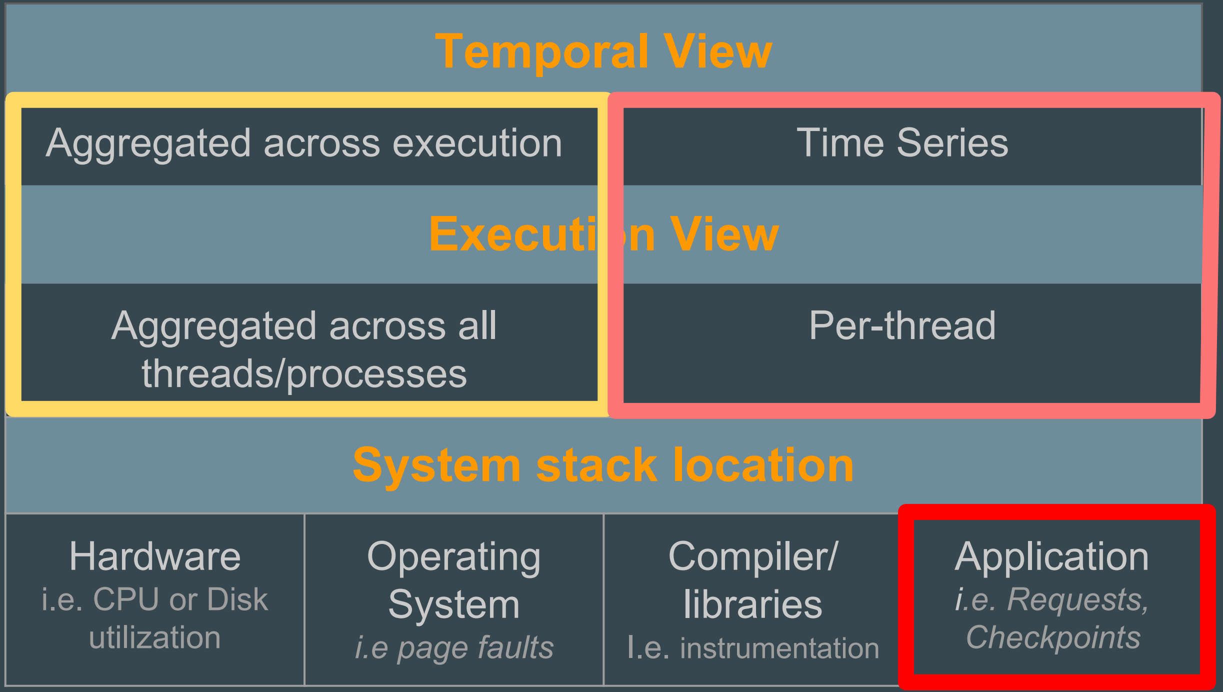
- ➔ (Info) Latency spikes and throughput variations are the main concern..
- ➔ (Tools) Developers correlate behavior between events over time.
- ➔ (Process) Tools used in exploratory fashion to find helpful information.

RQ1: What information is needed and collected?

- ➔ **17** / 36 codes appeared in >10% of tickets. (min: 6, max: 21)
- ➔ Categorized codes into three dimensions.

Aggregated, All threads (i.e. Profilers)

(4/17 codes, **12**/44 tickets)

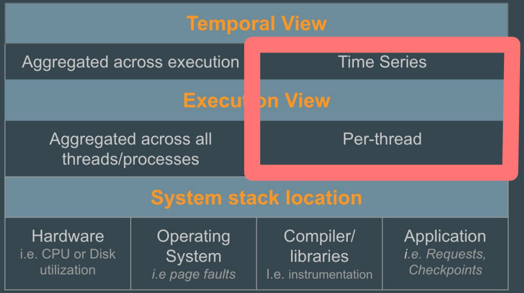


Time series Per-thread (i.e. Event logs)

(8/17 codes, **21**/44 tickets)

Info #1: Latency and Throughput Drops

(4/8 codes in Time series, Per Thread)



Temporal View	
Aggregated across execution	Time Series

Execution View	
Aggregated across all threads/processes	Per-thread

System stack location			
Hardware i.e. CPU or Disk utilization	Operating System *i.e page faults*	Compiler/ libraries I.e. instrumentation	Application *i.e. Requests, Checkpoints*

Main performance concern.

Rare instances of performance degradation that represent worst case software behaviour.

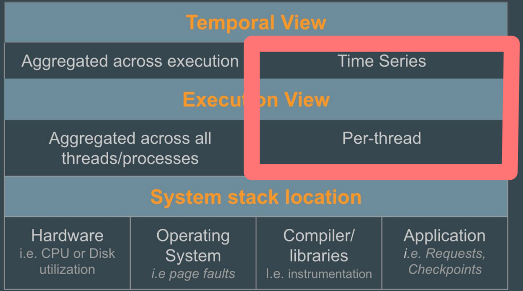
Both a symptom and cause of performance issues.

I got over 500 instances of slow operations on [branch X] and [commit Y], with maximum reported latencies over 1s. In running [workload Z], I see a number of multi-second latencies.

- WiredTiger developer trying to find diagnose throughput issues.

Info #2: Thread activity over time

(4/8 codes in Time series, Per Thread)



Temporal View

Aggregated across execution	Time Series

Execution View

Aggregated across all threads/processes	Per-thread

System stack location

Hardware i.e. CPU or Disk utilization	Operating System *i.e page faults*	Compiler/ libraries I.e. instrumentation	Application *i.e. Requests, Checkpoints*

A time-series trace of thread behaviour.

Used to correlate behaviour of threads to diagnose issues

Tools exist, but WT developers did not have access

There are two checkpoints, the first starts at about (40 secs) and it finishes quickly, before the next 10 second stat report. The next checkpoint starts at about (80 secs) and continues for the rest of the test.

- WiredTiger developer reasoning about a performance issue

What tools are used?

22 tool codes, 6 codes in more than 10% of tickets.

- ➔ Manual logging is prevalent
  - ◆ `print("thread %d cleanup started");`
    `… print("thread %d clean done");`
  - ◆ Reused event logs: 20/44 tickets
  - ◆ New event logs: 10/44 tickets

- ➔ Sampling profilers less common 8/44 tickets
  - ◆ Provides an aggregate view of an execution, not a time-series view

… perf [is] of limited use here. It is useful in giving an indication of where time is spent over the entire run which is good for general performance tuning but I don't know how to use the results to find what is causing outlier operations.

What processes do developers follow?

63 process codes, representing a largely exploratory process.

- ➔ Tool usage **preceded** need for specific information: 56/63 codes

  I collected WT statistics for all runs too. Maybe there is some trend ...

  [There are] lots of charts and numbers below. It is going to take a bit to digest them and determine if the ones I've chosen to display are useful and relevant.

- ➔ Frequently ran experiments to compare:
  - ◆ **code**: 20/44 tickets, **workloads**: 22/44 tickets

  I compared changeset 1234 (good performance) to 4567 (post-merge, bad performance)

Discussion and Opportunities

- Performance Comprehension is manual and exploratory
  - Developers gather information before knowing what will be useful.
  - Manual log analysis and experimentation processes.
  - Using only a few personal VMs and a single CI server. (In the paper)

- Opportunity to support experimentation + exploration
  - **Proactively** log information and **automatically** run experiments.
  - **Assist** developers in **exploring** collected information.
  - **Offload** work and tests from personal VMs to shared cloud resources.

Performance Comprehension: Takeaways

How do developers understand the underlying cause of performance issues?

RQ1 / Info: What information is needed and collected?
Time series per-thread information is the most frequent information needed

RQ2 / Tools: What tools are used?
Manual logging is most frequently used to acquire time series information.

RQ3 / Process: What processes do developers follow?
Tools often used in an exploratory fashion to find helpful information.

More findings in the paper!

Questions?

1. Are JIRA tickets really representative of developer tasks?
2. How well do the lessons here generalize?
3. Are there times when perf and other sampling profilers are useful?
4. Did developers use any tools to extract information from the log files?
5. Why aren’t these developers using [my favourite tool]?
6. ..
7. ..