APACHECON
ASIA 2021

Web server / Tomcat

Debugging Complex Issues In Web Applications

Mark Thomas

Committer, Apache Tomcat

Introductions

- Mark Thomas
- Tomcat committer since 2003
- Tomcat PMC member since 2005
- markt@apache.org
- ASF, Eclipse, VMware

Agenda

APACHECON
ASIA 2021

- What do I mean by "Complex Issues"?
- Statistics
- Uses cases
  - Concurrent HTTP/2 bulk transfers
  - Terminating connections without a response
- Summary of techniques
- Questions

Complex Issues

Complex Issues?

APACHECON
ASIA 2021

- Subjective
- Not 100% repeatable
  - Less repeatable typically means harder to debug
- Only occurs under load
- Concurrency

Statistics

Statistics

APACHECON
ASIA 2021

- Need more samples than you think
- Test lots
- Keep notes
  - What changed
  - Test results
- May have multiple root causes
  - A fix, or the fix?

Use case 1
Current HTTP/2 large responses

Original report

- Originated on Tomcat users mailing list
  - 2021-06-16
  - Trouble with HTTP/2 during bulk data transfer (server -> client)
  - https://tomcat.markmail.org/thread/texcre345tmyn337
- Well-written bug report

Original report

APACHECON
ASIA 2021

- Multiple HTTP/2 steams on same connection are blocked indefinitely
- Described the scenario in sufficient detail for a test case to be coded
  - Writing large files
  - Three or more concurrent streams
- Described working configurations (HTTP/1.1) and non-working configurations (HTTP/2)
- Described what they tried to vary (non-blocking, IOUtils)

Original report

APACHECON
ASIA 2021

- Provided all relevant version numbers
- Provided test case with source code
- Provided some analysis
  - HTTP/2 streams were waiting for a semaphore
- The only thing they didn't mention was repeatability
- They responded to this question in ~60 mins

Review relevant source code

APACHECON
ASIA 2021

- HTTP/2 Connections are multiplexed
  - Multiple streams trying to write
  - Use a semaphore to ensure they write one at a time
- HTTP/2 uses an internal async writes with a CompletionHandler
  - If write can't complete, socket is added to the Poller
  - An writeOperation field holds an OperationState instance that tracks the state of the async write

Narrow focus of investigation

APACHECON
ASIA 2021

- Recreated within ~90 mins of original report
  - Indicator of the quality of the original report
  - Enables us to quickly include / exclude functionality
- Rémy suggested disabling asyncIO
  - This provided a workaround (~4 hours)
  - Brief discussion on merits of using asyncIO
- Confirmed NIO2 was unaffected

Identify root cause

APACHECON
ASIA 2021

- Could see threads waiting for semaphore
  - Semaphore released by Poller indicating ready for write
- Start with code review
  - Possible root cause – non-volatile interestOps flag
  - Initial testing was positive, larger sample size ruled it out
- Decide to debug Socket / Poller interactions

Identify root cause

APACHECON
ASIA 2021

- Debug logs change the timing
  - Issue a lot less repeatable
- Need to change logging strategy
  - Copy relevant information to local variables
  - Log them AFTER the failure / event of interest
  - Much less likely to affect timing
- After a lot of debug logging
  - Poller was working correctly

Identify root cause

- Poller was signaling write was possible
  - Trace the write notification
  - OperationState was null so event wasn't processed
- Why was OperationState null
  - Code review
  - Found (potential) root cause

Fix & confirm

- Applied fix
  - Local testing confirmed fix (0 failures in 20 runs)
- Explain fix on mailing list
  - Other committers can check my reasoning
- Check if same error exists elsewhere
  - Read also affected

Explanation

- For those that want to understand the problem
- You'll need the code in front of you
- The write
  - https://github.com/apache/tomcat/blob/main/java/org/apache/tomcat/util/net/SocketWrapperBase.java#L1364
- The associated completion handler
  - https://github.com/apache/tomcat/blob/main/java/org/apache/tomcat/util/net/SocketWrapperBase.java#L1044

Explanation

- T1 obtains the write semaphore (L1366)
- T1 creates an OperationState and sets writeOperation (L1390)
- the async write for T1 completes and the completion handler is called
- T1's completion handler releases the semaphore (L1046)
- T2 obtains the write semaphore (L1366)

Explanation

- T2 creates an OperationState and sets writeOperation (L1390)
- T1's completion handler clears writeOperation (L1050)
- the async write for T2 does not complete and the socket is added to the Poller
- The Poller signals the socket is ready for write
- The Poller finds writeOperation is null so performs a normal dispatch for write
- The async write times out as it never receives the notification from the Poller

Fix

- The fix is to swap the clearing of writeOperation and the releasing of the semaphore
  - https://github.com/apache/tomcat/commit/92b91857

APACHECON
ASIA 2021

Use case 2
End connection before response

Original report

- Originated on Tomcat users mailing list
  - 2020-10-16
  - Weirdest Tomcat Behaviour Ever
  - https://tomcat.markmail.org/thread/bf6oz7ibxccvodd2
- Well-written bug report

Original report

- Very occasionally Tomcat does not send a response
- The access log shows a response
- No exceptions in the logs
- Wireshark shows the GET request followed by a TCP FIN from Tomcat
  - Indicates normal TCP close
- All version information provided

Narrow focus of investigation

APACHECON
ASIA 2021

- Asked various questions to try and eliminate features and/or possible failure modes
- The response was small ~1k
  - Small enough to be buffered entirely at various points in the network stack
- Typical response time was 60ms
  - Not going to be timeout related

Narrow focus of investigation

APACHECON
ASIA 2021

- The FIN was sent ~100µs after the request was received
  - Further confirmation it isn't timeout related
- The request is fully sent
  - Not waiting for the rest of the request
- User agent -> Firewall -> Nginx -> Tomcat
  - Might have been relevant if timeouts were suggested

Narrow focus of investigation

APACHECON
ASIA 2021

- HTTP/1.0 request
  - Rules out HTTP/2 code
- Network traces obtained from both nginx and Tomcat
  - Great to confirm behaviour
  - Nothing that indicates a possible root cause
- Application has unique request IDs that aid correlation across logs

Narrow focus of investigation

APACHECON
ASIA 2021

- Another user suggests using strace
  - OP didn't see the suggestion
  - It struck me as too low level at this point
  - In hindsight, it might have saved some time
- Switching from BIO to NIO didn't fix it
  - Issue not in the endpoint specific code
  - JVM issue less likely

Narrow focus of investigation

APACHECON
ASIA 2021

- Custom debug code tricky, but not impossible
- Issue started in the last month or so
  - No obvious changes
- Systems are lightly loaded
  - 20-60 requests a second
- Review of network traces (off-list)
  - Confirmed previous obsevations

Narrow focus of investigation

APACHECON
ASIA 2021

- Timings suggest JSP is generating the response
  - Adding %b to the access log confirms this
- Happens with BIO so sendfile isn't a factor
- No compression so GZIP isn't a factor
- No obvious explanation
  - Add custom debug logging to help narrow down search
  - https://github.com/markt-asf/tomcat/tree/debug-7.0.72

Narrow focus of investigation

APACHECON
ASIA 2021

- Debug logging v1
  - Response was written
  - Socket was closed before this
  - Correct objects were used
- Debug logging v2
  - Socket closed long before Tomcat tries to write
  - Neither Tomcat nor the application are closing the socket

Narrow focus of investigation

APACHECON
ASIA 2021

- Debug logging v3
  - Swallowed exception message "Bad file descriptor"
  - Exception swallowed because it was assumed to be a dropped client connection
  - Tomcat changed to log all such exceptions at debug
- Possibly running out of file descriptors?
  - No

Narrow focus of investigation

APACHECON
ASIA 2021

- Debug logging v4
  - No other active connections between nginx and Tomcat when the issue occurs
- Debug logging v5
  - No indication of JRE mis-handling file descriptors
- Back to strace

Narrow focus of investigation

APACHECON
ASIA 2021

- strace shows that the socket close came from somewhere in the JRE
- Try to correlate with thread dumps to identify where the close is occurring
- Possibility it was database related
  - False alarm

Identify root cause

APACHECON
ASIA 2021

- strace showed a native library incorrectly managing file descriptors associated with a fork
- The native library closed a file descriptor twice
- In same cases, that descriptor has already been re-used for the network connection
- When this happened, the network connection was closed

Resolution

- The vendor accepted the native library was at fault
  - PDFTron
- The vendor provided instructions to disable the use of the library that was triggering the issue
- We recommended a switch to HTTP/1.1 for the nginx / Tomcat connection
  - Fewer new connections, less chances of file descriptor reuse

Techniques

Other debugging techniques

APACHECON
ASIA 2021

- Logging / Wireshark
  - use a 5 minute rolling window
  - copy current logs when issue occurs
- Use ERROR level logs
  - simple filtering
- Issues that depend on network latency
  - Run Tomcat in the cloud
  - Simulate latency in the hypervisor

Other debugging techniques

APACHECON
ASIA 2021

- Choose you load generator carefully
  - Is it really multi-threaded?
  - Can it keep up?
- telnet
- Multiple physical machines
  - Pull out network cable to simulate lost connection
- Multiple platforms
  - VM or bare metal seems to be less of an issue

THANK YOU

QUESTIONS?

markt@apache.org

https://tomcat.apache.org

https://github.com/apache/tomcat