Item 282
git_comments:

1. Do nothing
2. Change node's parent to new_parent
3. delete all child nodes
4. Use stream id as initial point
5. TODO: K is a constant, 256 is temporal value.
6. * @file HTTP/2 Dependency Tree The original idea of Stream Priority Algorithm using Weighted Fair Queue (WFQ) Scheduling is invented by Kazuho Oku (H2O project). @section license License Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
7. __HTTP2_DEP_TREE_H__
8. Add node with exclusive flag
9. Activate B, C and D
10. NOTE, weight is actual weight - 1 node_a is unused
11. * * Reprioritization (exclusive) * * x x * | | * A D * / \ | * B C ==> A * / \ /|\ * D E B C F * | | * F E
12. * * Tree of Chrome 50 * * ROOT * / | \ * A(3) B(5) ... I(19) *
13. * * Simple Tree * ROOT * / * A(3) * / * B(5) *
14. * * Reprioritization (non-exclusive) * * x x * | | * A D * / \ / \ * B C ==> F A * / \ / \ * D E B C * | | * F E
15. * * Basic Tree * ROOT * / \ * A(3) D(9) * / \ * B(5) C(7) *
16. Activate A and B
17. argc ATS_UNUSED
18. * * Exclusive Dependency Creation * * A A * / \ => | * B C D * / \ * B C
19. argv ATS_UNUSED
20. atype ATS_UNUSED
21. * * Only One Node Tree * ROOT * / * A(1)
22. * @file Unit tests for Http2DependencyTree @section license License Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
23. Clear the highest bit for exclusive flag
24. [RFC 7540] 5.3.5 Default Priorities The RFC says weight value is 1 to 256, but the value in TS is between 0 to 255 to use uint8_t. So the default weight is 16 minus 1.
25. No response body to send
26. No node to send or no connection level window left
27. [RFC 7540] 5.3.3 Reprioritization
28. When no stream level window left, deactivate node once and wait window_update frame
29. Select appropriate payload length
30. A receiver MUST treat the receipt of a WINDOW_UPDATE frame with a flow control window increment of 0 as a connection error of type PROTOCOL_ERROR;
31. Copy into the payload buffer. Seems like we should be able to skip this copy step
32. Are we at the end? If we return here, we never send the END_STREAM in the case of a early terminating OS. OK if there is no body yet. Otherwise continue on to send a DATA frame and delete the stream
33. * [RFC 7540] 6.3 PRIORITY *
34. Send DATA frames directly

git_commits:

1. **summary:** TS-3535: Experimental Support of HTTP/2 Stream Priority feature
   message: TS-3535: Experimental Support of HTTP/2 Stream Priority feature - Add a option to enable this feature (disabled in default). `proxy.config.http2.stream_priority_enabled` - Parse priority parameters of HEADERS and PRIORITY frame correctly. - Add Http2DependencyTree and tests using `lib/ts/PriorityQueue.h`. - Create a dependency tree when clients send HEADERS frame with priority parameters or PRIORITY frame. - Separate `Http2ConnectionState::send_data_frame()` into `Http2ConnectionState::send_a_data_frame()` and `Http2ConnectionState::send_data_frames()`. - Schedule DATA frames using the WFQ algorithm. This closes #632 (cherry picked from commit 16172a4e79865d1201a51e85aeb72df8b0609986)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** TS-3535: Experimental Support of HTTP/2 Stream Priority feature
   body: This is an experimental support of HTTP/2 Stream Priority feature. - Add a option to enable this feature (disabled in default). `proxy.config.http2.stream_priority_enabled` - Parse priority parameters of HEADERS and PRIORITY frame correctly. - Add Http2DependencyTree and tests using `lib/ts/PriorityQueue.h`. - Create a dependency tree when clients send HEADERS frame with priority parameters or PRIORITY frame. - Separate `Http2ConnectionState::send_data_frame()` into `Http2ConnectionState::send_a_data_frame()` and `Http2ConnectionState::send_data_frames()`. - Schedule DATA frames using the WFQ algorithm.

github_pulls_comments:

1. This is second patch after the #525 and TS-4295.
2. **body:** This is running on docs.trafficserver right now.
   label: documentation
3. The new commit passed the build tests on the CI - https://ci.trafficserver.apache.org/view/github/job/Github-Linux/80/ - https://ci.trafficserver.apache.org/view/github/job/Github-FreeBSD/175/

github_pulls_reviews:

1. Why does this need TM restart? Shouldn't TS restart be enough? Looking at it, if you agree, then probably should change proxy.config.http2.enabled to be RESTART_TS as well?
2. Since `stream_priority_enabled` uses `REC_EstablishStaticConfigBool()`, changes will take effect immediately so it should be `RECU_DYNAMIC`.
3. **body:** You sure ConfigBool here is appropriate? I don't see us use it anywhere, and the configs in RecordsConfig.cc / records.config are always of integer type. Not opposed to using Bool's here, but seems a little bit inconsistent.
   label: code-design
4. Since you referenced the RFC here: "In both cases, streams are assigned a default weight of 16". So, why is it 15? :)
5. The RFC says below in 6.2 HEADERS and 6.3 PRIORITY > Weight: An unsigned 8-bit integer representing a priority weight for the stream (see Section 5.3). Add one to the value to obtain a weight between 1 and 256. We're not plus 1 to use `uint8_t`, so the default value become `16 - 1`. I'll add comments to describe this.
6. Huh, I'm merely glancing here, but it seems really strange that we'd need to hold a mutex just to set _scheduled to false? Could that not at least be done with just a CAS? Seeing this, also makes me wonder is if the send_data_frames_depends_on_priority() call should also be protected under the mutex?
7. Why does this need to be a template class? is it only for being able to make a test? Not a big deal, I'm just a hater. But as far as I can tell, it's only used as a streams dependency tree.
8. I agree with both of them should be `RECU_DYNAMIC`. (I just copied `proxy.config.http2.enabled` :p)
9. **body:** Just for easy to test this.
   label: test

10. What about if the stream is new and we already received a priority frame. It looks like below you are adding the priority in the tree if the stream doesn't exist. My understanding is that we should be using the priority that is in the tree.
11. I think it is helpful to have this comment in.
12. I think it is helpful to have this comment in.
13. **body:** This check and the duplicate one below can be moved up now since the assignment of size is done above now.
    label: code-design
14. I would call it "priority_node" or something better then "node".
15. OK, revert those.
16. You're right. The priority in the tree shouldn't be overwrote in here, if HEADERS frame don't have priority flag.
17. Do we really want to schedule the continuation again if the stream is closed? Should there be a return here?
18. You are taking the min of two ssize_t. I would think window_size should also be a ssize_t.
19. If window_size above is a ssize_t then you can use it here.
20. IMO, we need schedule the continuation again. Because it could be possible that another stream in the dependency tree is ready to send.
21. Makes sense. thx.

jira_issues:

1. **summary:** Add priority feature to the HTTP/2 implementation
   description: Prioritizes the responses back to the client based on the priority level specified by the HTTP/2 protocol.

jira_issues_comments:

1. So this is driven by client requests entirely? Are there configs for ATS to decide when to honor such priorities ?
2. Priority for HTTP/2 is set by the client. We could have a configuration option to turn it off or on, but it is part of the RFC.
3. I'll work for this
4. GitHub user masaori335 opened a pull request: https://github.com/apache/trafficserver/pull/525 TS-3535: Experimental Support of Stream Priority Feature in HTTP/2 [TS-3535] (https://issues.apache.org/jira/browse/TS-3535) Experimental Support of Stream Priority Feature. ## Approach - Basically I followed WFQ Scheduling Algorithm invented by Kazuho and introduced by Kazu and Tatsuhiro at [ATS Meetup in Tokyo] (https://cwiki.apache.org/confluence/display/TS/Meetup+Tokyo+2015#MeetupTokyo2015-PresentationSession) - Using MinHeap for PriorityQueue ## Issues - Currently overheads of Priority Feature is high, we need optimization. - Works fine with Chrome, FireFox, Safari (on MacOSX), but has troubles with h2spec and h2load. You can merge this pull request into a Git repository by running: $ git pull https://github.com/masaori335/trafficserver TS-3535 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/trafficserver/pull/525.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #525 ---- commit ab25d07ad25a77780b1c495f36877e4742d4dc7e Author: Masaori Koshiba <masaori@apache.org> Date: 2016-02-12T07:07:10Z TS-3535: Add Stream Priority Feature to HTTP/2 Component ----
5. Github user zwoop commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-196868807 Seeing that there are some pretty serious issues, should we leave it disabled for now, with a records.config option to enable it?
6. Github user jpeach commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-196875583 @zwoop see ``proxy.config.http2.stream_priority_enabled``?
7. Github user zwoop commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-196971215 Perfect!
8. Github user zwoop commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-199871552 @bryancall to review, if we

land this with off by default it seems harmless. We do want to avoid STL when possible, but it is what it is :).
9. Github user PSUdaemon commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-199873028 Is it possible to make the priority queue stuff more generic so it can be reusable?
10. Github user masaori335 commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-200325489 @zwoop Should we avoid STL in test code too? I used `ts/Vec.h` and implemented priority queue to avoid STL:D
11. Github user masaori335 commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-200337999 @PSUdaemon Yes, it is! It is one of reasons of implemented Http2PriorityQueue using template. Is it better to change name and move under `lib/ts/`?
12. Github user PSUdaemon commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-200348327 @masaori335, yes exactly. It does look very generic already and it seems valuable to have in lib/ts. I can't think of anything offhand that would need it, but if you are already adding it for H2 seems prudent to make it available to other things.
13. Github user zwoop commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-200354234 +1 on adding it to lib/ts. As for STL, the general rule is that if something runs on the critical path (e.g. as part of the HttpSM, or normal transaction handling), where it is performance sensitive, we should make every effort to avoid STL. As a subsidiary to that, we should really think hard before using std::string on said critical path as well (there's generally little reason to use std::string there). Now, the exception here are two things: 1) Plugins. We don't dictate anything for plugins, so use whatever you deem necessary to make it work. If someone doesn't like it, they can either fix it, or they can not use the plugin. 2) Anything not on the critical path. This includes configuration loading, traffic_manager/cop, periodic tasks, tests etc. That much said, don't use STL just for the hell of it. If there is a reasonable alternative to use in lib/ts, please use it. Dragging in STL does increase binary sizes, which could affect performance in other ways (such as worse L2/3 caches etc.). It also sets bad precedence, where someone seeing it used, makes the mistake and thinks it's ok to use everywhere.
14. Github user masaori335 commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-200705387 I spined out priority queue stuff as [TS-4295](https://issues.apache.org/jira/browse/TS-4295). I'm going to fix this patch to use it and fix conflicts.
15. Github user masaori335 commented on the pull request: https://github.com/apache/trafficserver/pull/525#issuecomment-211173098 Now, FetchSM and PluginVC is removed from HTTP/2 components by TS-3612. It look like I need more works to rebase this on latest master. I'm going to close this once and reopen new Pull-Requests for v7.0.0 to avoid blocking v6.2.0 release.
16. Github user masaori335 closed the pull request at: https://github.com/apache/trafficserver/pull/525
17. Github user masaori335 commented on the pull request: https://github.com/apache/trafficserver/pull/632#issuecomment-219591587 The new commit passed the build tests on the CI - https://ci.trafficserver.apache.org/view/github/job/Github-Linux/80/ - https://ci.trafficserver.apache.org/view/github/job/Github-FreeBSD/175/
18. Github user bryancall commented on a diff in the pull request: https://github.com/apache/trafficserver/pull/632#discussion_r63739375 --- Diff: proxy/http2/Http2ConnectionState.cc --- @@ -906,74 +957,140 @@ Http2ConnectionState::update_initial_rwnd(Http2WindowSize new_size) } void - Http2ConnectionState::send_data_frame(Http2Stream *stream) +Http2ConnectionState::schedule_stream(Http2Stream *stream) { - size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; - uint8_t payload_buffer[buf_len]; + DebugHttp2Stream(ua_session, stream->get_id(), "Scheduled"); - if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + DependencyTree::Node *node = stream->priority_node; + ink_release_assert(node != NULL); + + SCOPED_MUTEX_LOCK(lock, this->mutex, this_ethread()); + dependency_tree->activate(node); + + if (!_scheduled) { + _scheduled = true; + SET_HANDLER(&Http2ConnectionState::main_event_handler); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + } +} + +void +Http2ConnectionState::send_data_frames_depends_on_priority() +{ + DependencyTree::Node *node =

dependency_tree->top(); + + // No node to send or no connection level window left + if (node == NULL || client_rwnd <= 0) { return; } - for (;;) { - uint8_t flags = 0x00; + Http2Stream *stream = node->t; + ink_release_assert(stream != NULL); + DebugHttp2Stream(ua_session, stream->get_id(), "top node, point=%d", node->point); - size_t window_size = min(this->client_rwnd, stream->client_rwnd); - size_t send_size = min(buf_len, window_size); - size_t payload_length; - IOBufferReader *current_reader = stream->response_get_data_reader(); + size_t len = 0; + Http2SendADataFrameResult result = send_a_data_frame(stream, len); - // Are we at the end? - // If we break here, we never send the END_STREAM in the case of a - // early terminating OS. Ok if there is no body yet. Otherwise - // continue on to delete the stream - if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { - Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); - flags |= HTTP2_FLAGS_DATA_END_STREAM; - payload_length = 0; - } else { - // Select appropriate payload size - if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) - break; - // Copy into the payload buffer. Seems like we should be able to skip this - // copy step - payload_length = current_reader ? current_reader->read(payload_buffer, send_size) : 0; - - if (payload_length == 0 && !stream->is_body_done()) { - break; - } + if (result != HTTP2_SEND_A_DATA_FRAME_NO_ERROR) { + // When no stream level window left, deactivate node once and wait window_update frame + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - // Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); --- End diff -- Do we really want to schedule the continuation again if the stream is closed?

19. Github user bryancall commented on a diff in the pull request: https://github.com/apache/trafficserver/pull/632#discussion_r63740371 --- Diff: proxy/http2/Http2ConnectionState.cc --- @@ -906,74 +957,140 @@ Http2ConnectionState::update_initial_rwnd(Http2WindowSize new_size) } void - Http2ConnectionState::send_data_frame(Http2Stream *stream) +Http2ConnectionState::schedule_stream(Http2Stream *stream) { - size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; - uint8_t payload_buffer[buf_len]; + DebugHttp2Stream(ua_session, stream->get_id(), "Scheduled"); - if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + DependencyTree::Node *node = stream->priority_node; + ink_release_assert(node != NULL); + + SCOPED_MUTEX_LOCK(lock, this->mutex, this_ethread()); + dependency_tree->activate(node); + + if (!_scheduled) { + _scheduled = true; + SET_HANDLER(&Http2ConnectionState::main_event_handler); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + } +} + +void +Http2ConnectionState::send_data_frames_depends_on_priority() +{ + DependencyTree::Node *node = dependency_tree->top(); + + // No node to send or no connection level window left + if (node == NULL || client_rwnd <= 0) { return; } - for (;;) { - uint8_t flags = 0x00; + Http2Stream *stream = node->t; + ink_release_assert(stream != NULL); + DebugHttp2Stream(ua_session, stream->get_id(), "top node, point=%d", node->point); - size_t window_size = min(this->client_rwnd, stream->client_rwnd); - size_t send_size = min(buf_len, window_size); - size_t payload_length; - IOBufferReader *current_reader = stream->response_get_data_reader(); + size_t len = 0; + Http2SendADataFrameResult result = send_a_data_frame(stream, len); - // Are we at the end? - // If we break here, we never send the END_STREAM in the case of a - // early terminating OS. Ok if there is no body yet. Otherwise - // continue on to delete the stream - if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { - Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); - flags |= HTTP2_FLAGS_DATA_END_STREAM; - payload_length = 0; - } else { - // Select appropriate payload size - if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) - break; - // Copy into the payload buffer. Seems like we should be able to skip this - // copy step - payload_length = current_reader ? current_reader->read(payload_buffer, send_size) : 0; - - if (payload_length == 0 && !stream->is_body_done()) { - break; - } + if (result != HTTP2_SEND_A_DATA_FRAME_NO_ERROR) { + // When no stream level window left, deactivate node once and wait window_update frame + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - // Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); --- End diff -- Do we really want to schedule the continuation again if the stream is closed?

Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); + } else { + dependency_tree->update(node, len); + } + + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); +} + +Http2SendADataFrameResult +Http2ConnectionState::send_a_data_frame(Http2Stream *stream, size_t &payload_length) +{ + size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; + uint8_t payload_buffer[buf_len]; + uint8_t flags = 0x00; + size_t window_size = min(this->client_rwnd, stream->client_rwnd); --- End diff -- You are taking the min of two ssize_t. I would think window_size should also be a ssize_t.

20. Github user bryancall commented on a diff in the pull request: https://github.com/apache/trafficserver/pull/632#discussion_r63740460 --- Diff: proxy/http2/Http2ConnectionState.cc --- @@ -906,74 +957,140 @@ Http2ConnectionState::update_initial_rwnd(Http2WindowSize new_size) } void - Http2ConnectionState::send_data_frame(Http2Stream *stream) +Http2ConnectionState::schedule_stream(Http2Stream *stream) { - size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; - uint8_t payload_buffer[buf_len]; + DebugHttp2Stream(ua_session, stream->get_id(), "Scheduled"); - if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + DependencyTree::Node *node = stream->priority_node; + ink_release_assert(node != NULL); + + SCOPED_MUTEX_LOCK(lock, this->mutex, this_ethread()); + dependency_tree->activate(node); + + if (!_scheduled) { + _scheduled = true; + SET_HANDLER(&Http2ConnectionState::main_event_handler); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + } +} + +void +Http2ConnectionState::send_data_frames_depends_on_priority() +{ + DependencyTree::Node *node = dependency_tree->top(); + + // No node to send or no connection level window left + if (node == NULL || client_rwnd <= 0) { return; } - for (;;) { - uint8_t flags = 0x00; + Http2Stream *stream = node->t; + ink_release_assert(stream != NULL); + DebugHttp2Stream(ua_session, stream->get_id(), "top node, point=%d", node->point); - size_t window_size = min(this->client_rwnd, stream->client_rwnd); - size_t send_size = min(buf_len, window_size); - size_t payload_length; - IOBufferReader *current_reader = stream->response_get_data_reader(); + size_t len = 0; + Http2SendADataFrameResult result = send_a_data_frame(stream, len); - // Are we at the end? - // If we break here, we never send the END_STREAM in the case of a - // early terminating OS. Ok if there is no body yet. Otherwise - // continue on to delete the stream - if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { - Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); - flags |= HTTP2_FLAGS_DATA_END_STREAM; - payload_length = 0; - } else { - // Select appropriate payload size - if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) - break; - // Copy into the payload buffer. Seems like we should be able to skip this - // copy step - payload_length = current_reader ? current_reader->read(payload_buffer, send_size) : 0; - - if (payload_length == 0 && !stream->is_body_done()) { - break; - } + if (result != HTTP2_SEND_A_DATA_FRAME_NO_ERROR) { + // When no stream level window left, deactivate node once and wait window_update frame + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - // Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); + } else { + dependency_tree->update(node, len); + } + + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); +} + +Http2SendADataFrameResult +Http2ConnectionState::send_a_data_frame(Http2Stream *stream, size_t &payload_length) +{ + size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; + uint8_t payload_buffer[buf_len]; + uint8_t flags = 0x00; + size_t window_size = min(this->client_rwnd, stream->client_rwnd);

window_size = min(this->client_rwnd, stream->client_rwnd); + size_t send_size = min(buf_len, window_size); + IOBufferReader *current_reader = stream->response_get_data_reader(); + + SCOPED_MUTEX_LOCK(stream_lock, stream->mutex, this_ethread()); + // Are we at the end? + // If we break here, we never send the END_STREAM in the case of a + // early terminating OS. Ok if there is no body yet. Otherwise + // continue on to delete the stream + if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { + Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); + flags |= HTTP2_FLAGS_DATA_END_STREAM; + payload_length = 0; + } else { + // Select appropriate payload size + if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) --- End diff -- If window_size above is a ssize_t then you can use it here.

21. Github user masaori335 commented on a diff in the pull request: https://github.com/apache/trafficserver/pull/632#discussion_r63741679 --- Diff: proxy/http2/Http2ConnectionState.cc --- @@ -906,74 +957,140 @@ Http2ConnectionState::update_initial_rwnd(Http2WindowSize new_size) } void - Http2ConnectionState::send_data_frame(Http2Stream *stream) +Http2ConnectionState::schedule_stream(Http2Stream *stream) { - size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; - uint8_t payload_buffer[buf_len]; + DebugHttp2Stream(ua_session, stream->get_id(), "Scheduled"); - if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + DependencyTree::Node *node = stream->priority_node; + ink_release_assert(node != NULL); + + SCOPED_MUTEX_LOCK(lock, this->mutex, this_ethread()); + dependency_tree->activate(node); + + if (!_scheduled) { + _scheduled = true; + SET_HANDLER(&Http2ConnectionState::main_event_handler); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + } +} + +void +Http2ConnectionState::send_data_frames_depends_on_priority() +{ + DependencyTree::Node *node = dependency_tree->top(); + + // No node to send or no connection level window left + if (node == NULL || client_rwnd <= 0) { return; } - for (;;) { - uint8_t flags = 0x00; + Http2Stream *stream = node->t; + ink_release_assert(stream != NULL); + DebugHttp2Stream(ua_session, stream->get_id(), "top node, point=%d", node->point); - size_t window_size = min(this->client_rwnd, stream->client_rwnd); - size_t send_size = min(buf_len, window_size); - size_t payload_length; - IOBufferReader *current_reader = stream->response_get_data_reader(); + size_t len = 0; + Http2SendADataFrameResult result = send_a_data_frame(stream, len); - // Are we at the end? - // If we break here, we never send the END_STREAM in the case of a - // early terminating OS. Ok if there is no body yet. Otherwise - // continue on to delete the stream - if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { - Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); - flags |= HTTP2_FLAGS_DATA_END_STREAM; - payload_length = 0; - } else { - // Select appropriate payload size - if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) - break; - // Copy into the payload buffer. Seems like we should be able to skip this - // copy step - payload_length = current_reader ? current_reader->read(payload_buffer, send_size) : 0; - - if (payload_length == 0 && !stream->is_body_done()) { - break; - } + if (result != HTTP2_SEND_A_DATA_FRAME_NO_ERROR) { + // When no stream level window left, deactivate node once and wait window_update frame + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - // Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); --- End diff -- IMO, we need schedule the continuation again. Because it could be possible that another stream in the dependency tree is ready to send.

22. Github user bryancall commented on a diff in the pull request: https://github.com/apache/trafficserver/pull/632#discussion_r63748572 --- Diff: proxy/http2/Http2ConnectionState.cc --- @@ -906,74 +957,140 @@ Http2ConnectionState::update_initial_rwnd(Http2WindowSize new_size) } void - Http2ConnectionState::send_data_frame(Http2Stream *stream) +Http2ConnectionState::schedule_stream(Http2Stream *stream) { - size_t buf_len = BUFFER_SIZE_FOR_INDEX(buffer_size_index[HTTP2_FRAME_TYPE_DATA]) - HTTP2_FRAME_HEADER_LEN; - uint8_t payload_buffer[buf_len]; + DebugHttp2Stream(ua_session,

stream->get_id(), "Scheduled"); - if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + DependencyTree::Node *node = stream->priority_node; + ink_release_assert(node != NULL); + + SCOPED_MUTEX_LOCK(lock, this->mutex, this_ethread()); + dependency_tree->activate(node); + + if (!_scheduled) { + _scheduled = true; + + SET_HANDLER(&Http2ConnectionState::main_event_handler); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + } +} + +void +Http2ConnectionState::send_data_frames_depends_on_priority() +{ + DependencyTree::Node *node = dependency_tree->top(); + + // No node to send or no connection level window left + if (node == NULL || client_rwnd <= 0) { return; } - for (;;) { - uint8_t flags = 0x00; + Http2Stream *stream = node->t; + ink_release_assert(stream != NULL); + DebugHttp2Stream(ua_session, stream->get_id(), "top node, point=%d", node->point); - size_t window_size = min(this->client_rwnd, stream->client_rwnd); - size_t send_size = min(buf_len, window_size); - size_t payload_length; - IOBufferReader *current_reader = stream->response_get_data_reader(); + size_t len = 0; + Http2SendADataFrameResult result = send_a_data_frame(stream, len); - // Are we at the end? - // If we break here, we never send the END_STREAM in the case of a - // early terminating OS. Ok if there is no body yet. Otherwise - // continue on to delete the stream - if (stream->is_body_done() && current_reader && !current_reader->is_read_avail_more_than(0)) { - Debug("http2_con", "End of Stream id=%d no more data and body done", stream->get_id()); - flags |= HTTP2_FLAGS_DATA_END_STREAM; - payload_length = 0; - } else { - // Select appropriate payload size - if (this->client_rwnd <= 0 || stream->client_rwnd <= 0) - break; - // Copy into the payload buffer. Seems like we should be able to skip this - // copy step - payload_length = current_reader ? current_reader->read(payload_buffer, send_size) : 0; - - if (payload_length == 0 && !stream->is_body_done()) { - break; - } + if (result != HTTP2_SEND_A_DATA_FRAME_NO_ERROR) { + // When no stream level window left, deactivate node once and wait window_update frame + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - // Update window size - this->client_rwnd -= payload_length; - stream->client_rwnd -= payload_length; + // No response body to send + if (len == 0 && !stream->is_body_done()) { + dependency_tree->deactivate(node, len); + this_ethread()->schedule_imm_local((Continuation *)this, HTTP2_SESSION_EVENT_XMIT); + return; + } - if (stream->is_body_done() && payload_length < send_size) { - flags |= HTTP2_FLAGS_DATA_END_STREAM; - } + if (stream->get_state() == HTTP2_STREAM_STATE_CLOSED) { + dependency_tree->deactivate(node, len); + delete_stream(stream); --- End diff -- Makes sense. thx.

23. Commit 16172a4e79865d1201a51e85aeb72df8b0609986 in trafficserver's branch refs/heads/master from [~masaori] [https://git-wip-us.apache.org/repos/asf?p=trafficserver.git;h=16172a4] TS-3535: Experimental Support of HTTP/2 Stream Priority feature - Add a option to enable this feature (disabled in default). `proxy.config.http2.stream_priority_enabled` - Parse priority parameters of HEADERS and PRIORITY frame correctly. - Add Http2DependencyTree and tests using `lib/ts/PriorityQueue.h`. - Create a dependency tree when clients send HEADERS frame with priority parameters or PRIORITY frame. - Separate `Http2ConnectionState::send_data_frame()` into `Http2ConnectionState::send_a_data_frame()` and `Http2ConnectionState::send_data_frames()`. - Schedule DATA frames using the WFQ algorithm. This closes #632
24. Github user asfgit closed the pull request at: https://github.com/apache/trafficserver/pull/632