Item 203
git_comments:

1. * The default limit of concurrently happening checkpoints: one
2. * The default timeout of a checkpoint attempt: 10 minutes
3. * * Gets the maximum time that a checkpoint may take before being discarded. * * @return The checkpoint timeout, in milliseconds.
4. --
5. * Periodic checkpoint triggering interval
6. * * Gets the minimal pause between checkpointing attempts. This setting defines how soon the * checkpoint coordinator may trigger another checkpoint after it becomes possible to trigger * another checkpoint with respect to the maximum number of concurrent checkpoints * (see {@link #getMaxConcurrentCheckpoints()}). * * @return The minimal pause before the next checkpoint is triggered.
7. * * Sets the interval in which checkpoints are periodically scheduled. * * <p>This setting defines the base interval. Checkpoint triggering may be delayed by the settings * {@link #setMaxConcurrentCheckpoints(int)} and {@link #setMinPauseBetweenCheckpoints(long)}. * * @param checkpointInterval The checkpoint interval, in milliseconds.
8. * * Gets the maximum number of checkpoint attempts that may be in progress at the same time. If this * value is <i>n</i>, then no checkpoints will be triggered while <i>n</i> checkpoint attempts are * currently in flight. For the next checkpoint to be triggered, one checkpoint attempt would need * to finish or expire. * * @return The maximum number of concurrent checkpoint attempts.
9. * * Checks whether checkpointing is enabled. * * @return True if checkpointing is enables, false otherwise.
10. * * Sets the maximum number of checkpoint attempts that may be in progress at the same time. If this * value is <i>n</i>, then no checkpoints will be triggered while <i>n</i> checkpoint attempts are * currently in flight. For the next checkpoint to be triggered, one checkpoint attempt would need * to finish or expire. * * @param maxConcurrentCheckpoints The maximum number of concurrent checkpoint attempts.
11. * * Checks whether checkpointing is forced, despite currently non-checkpointable iteration feedback. * * @return True, if checkpointing is forced, false otherwise. * * @deprecated This will be removed once iterations properly participate in checkpointing.
12. * Maximum time checkpoint may take before being discarded
13. * * Checks whether checkpointing is forced, despite currently non-checkpointable iteration feedback. * * @param forceCheckpointing The flag to force checkpointing. * * @deprecated This will be removed once iterations properly participate in checkpointing.
14. * Maximum number of checkpoint attempts in progress at the same time
15. * Flag to force checkpointing in iterative jobs
16. * The default minimum pause to be made between checkpoints: none
17. * * Gets the checkpointing mode (exactly-once vs. at-least-once). * * @return The checkpointing mode.
18. * The default checkpoint mode: exactly once
19. /** * Sets the minimal pause between checkpointing attempts. This setting defines how soon the * checkpoint coordinator may trigger another checkpoint after it becomes possible to trigger * another checkpoint with respect to the maximum number of concurrent checkpoints * (see {@link #setMaxConcurrentCheckpoints(int)}). * * <p>If the maximum number of concurrent checkpoints is set to one, this setting makes effectively sure * that a minimum amount of time passes where no checkpoint is in progress at all. * * @param minPauseBetweenCheckpoints The minimal pause before the next checkpoint is triggered. */ public void setMinPauseBetweenCheckpoints(long minPauseBetweenCheckpoints) { if (minPauseBetweenCheckpoints < 0) { throw new IllegalArgumentException("Pause value must be zero or positive"); } this.minPauseBetweenCheckpoints = minPauseBetweenCheckpoints; }
20. * Minimal pause between checkpointing attempts
21. * * Configuration that captures all checkpointing related settings.
22. * * Sets the maximum time that a checkpoint may take before being discarded. * * @param checkpointTimeout The checkpoint timeout, in milliseconds.
23. * 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 *

24. * Checkpointing mode (exactly-once vs. at-least-once).
25. * * Gets the interval in which checkpoints are periodically scheduled. * * <p>This setting defines the base interval. Checkpoint triggering may be delayed by the settings * {@link #getMaxConcurrentCheckpoints()} and {@link #getMinPauseBetweenCheckpoints()}. * * @return The checkpoint interval, in milliseconds.
26. * * Sets the checkpointing mode (exactly-once vs. at-least-once). * * @param checkpointingMode The checkpointing mode.
27. * The min time(in ms) to delay after a checkpoint could be triggered. Allows to * enforce minimum processing time between checkpoint attempts
28. * Map from checkpoint ID to the pending checkpoint
29. * The max time (in ms) that a checkpoint may take
30. make sure all prior timers are cancelled
31. * The base checkpoint interval. Actual trigger time may be affected by the * max concurrent checkpoints and minimum-pause values
32. * The number of consecutive failed trigger attempts
33. guard the map against concurrent modifications
34. if too many checkpoints are currently in progress, we need to mark that a request is queued
35. --
36. * Completed checkpoints. Implementations can be blocking. Make sure calls to methods * accessing this don't block the job manager actor and run asynchronously.
37. * The maximum number of checkpoints that may be in progress at the same time
38. abort if the coordinator has been shutdown in the meantime
39. * A list of recent checkpoint IDs, to identify late messages (vs invalid ones)
40. * Flag whether a triggered checkpoint should immediately schedule the next checkpoint. * Non-volatile, because only accessed in synchronized scope
41. shut down the thread that handles the timeouts and pending triggers
42. re-acquire the lock since we released the lock in the meantime, we need to re-check that the conditions still hold. this is clumsy, but it allows us to release the lock in the meantime while calls to external services are blocking progress, and still gives us early checks that skip work if no checkpoint can happen anyways
43. trigger the checkpoint from the trigger timer, to finish the work of this thread before starting with the next checkpoint
44. * Actor that receives status updates from the execution graph this coordinator works for
45. -- job status listener that schedules / cancels periodic checkpoints --
46. end of lock scope
47. * Class loader used to deserialize the state handles (as they may be user-defined)
48. * Flag marking the coordinator as shut down (not accepting any messages any more)
49. make some eager pre-checks
50. * * Triggers the queued request, if there is one. * * <p>NOTE: The caller of this method must hold the lock when invoking the method!
51. this must happen outside the locked scope, because it communicates with external services (in HA mode) and may block for a while.
52. we will actually trigger this checkpoint!
53. * Checkpoint ID counter to ensure ascending IDs. In case of job manager failures, these * need to be ascending across job managers.
54. * Flag whether a trigger request could not be handled immediately. Non-volatile, because only * accessed in synchronized scope
55. sanity check: there should never be more than one trigger request queued
56. create the coordinator that triggers and commits checkpoints and holds the state
57. sanity checks
58. periodic interval is 10 ms timeout is very long (200 s) no extra delay max two concurrent checkpoints
59. do the final check
60. validate that the pending checkpoints are there

61. now we acknowledge the second checkpoint, which should subsume the first checkpoint and allow two more checkpoints to be triggered now, once we acknowledge one checkpoint, it should trigger the next one
62. the coordinator should start checkpointing now
63. after a while, there should be exactly as many checkpoints as concurrently permitted
64. no checkpoint should have started so far
65. for 400 ms, no further calls may come. there may be the case that one trigger was fired and about to acquire the lock, such that after cancelling it will still do the remainder of its work
66. now, once we acknowledge one checkpoint, it should trigger the next one
67. this should have immediately triggered a new checkpoint
68. create some mock execution vertices and trigger some checkpoint
69. periodic interval is 10 ms timeout is very long (200 s)
70. -- Utilities --
71. for 400 ms, no further calls may come there may be the case that one trigger was fired and about to acquire the lock, such that after cancelling it will still do the remainder of its work
72. periodic interval is 10 ms timeout is very long (200 s) no extra delay
73. now move the state to RUNNING
74. no further checkpoints should happen
75. after a while, there should be exactly as many checkpoints as concurrently permitted
76. after a while, there should be the new checkpoints
77. start another sequence of periodic scheduling
78. * Settings that control the checkpointing behavior
79. * * Gets the checkpoint config, which defines values like checkpoint interval, delay between * checkpoints, etc. * * @return The checkpoint config.
80. currently, these are all vertices
81. the "at-least-once" input handler is slightly cheaper (in the absence of checkpoints),

git_commits:

1. **summary:** [FLINK-3051] [streaming] Add mechanisms to control the maximum number of concurrent checkpoints
   message: [FLINK-3051] [streaming] Add mechanisms to control the maximum number of concurrent checkpoints This closes #1408

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-3051] Control the maximum number of concurrent checkpoints
   body: This change introduces a parameter that lets users control at most how many checkpoints should be in progress at any given point in time. This is very useful for cases where the best checkpoint interval is not easy to determine, or where once in a while checkpoints may take long. Previously, such situations lead to checkpoint queue-up, where the system ended up being more busy with checkpoints than with processing. After this change, the checkpoint rate will by default sort of self-regulate: The checkpoint interval is the most frequent time at which checkpoints will occur, but they will occur slower if they take longer than that interval. ### Checkpoint Config This also introduces the CheckpointConfig class that holds all checkpointing related parameters to more simply pass them between environment, streamgraph, etc... ### Default number of concurrent checkpoints Previously, the maximum number of concurrent checkpoints was implicitly infinite. I would suspect that most people will want to run the system such that only one checkpoint is ever concurrently active, so I set the default value for this to 1. In some corner cases, it may be interesting to set that value higher. ### WIP for delay between checkpoints This also contains some WIP to define a minimum time between checkpoint attempts. That flag would tell the system not only to not do more than one checkpoint concurrently, but to leave at least a certain time between completion of one checkpoint, and the triggering of the next. It basically defines a guaranteed time that is "work only" between checkpoints.

github_pulls_comments:

1. Good addition! More control over our checkpointing is something people were asking me at talks. Could you add a few sentences to the documentation about this?
2. Where is the docs would be the best place for that? - A new entry in the "Programing Guides" menu? - Or a section in the streaming guide? I would vote for the first
3. I think it fits best into the "Fault Tolerance" section of the Streaming Guide: https://ci.apache.org/projects/flink/flink-docs-release-0.10/apis/streaming_guide.html#fault-tolerance If you think that section will grow to big, we could move it to a new page and link it from the streaming guide.
4. Should we pull that into a separate document? It becomes quite large...
5. +1
6. Will do this in a separate pull request as follow up. Any concerns about merging this?
7. Please give me an hour or so to look at this :)
8. Looks good and the minimum delay between checkpoints would be an extremely useful feature. What's missing for that?
9. Missing for that is a bit of code in the checkpoint coordinator that marks the time when checkpoints become possible again and adds that delay to the time when the next checkpoint is scheduled. I'd suggest to add that as a followup...

github_pulls_reviews:

jira_issues:

1. **summary:** Define a maximum number of concurrent inflight checkpoints
   description: The checkpoint coordinator should define an option to limit the maximum number of current inflight checkpoints, as well as the checkpoint timeouts.

jira_issues_comments:

1. As part of this feature, I also started adding a config object that allows to define more checkpoint properties, like timeouts, delays between checkpoint attempts, maximum number of in-flight checkpoints...
2. GitHub user StephanEwen opened a pull request: https://github.com/apache/flink/pull/1408 [FLINK-3051] Control the maximum number of concurrent checkpoints This change introduces a parameter that lets users control at most how many checkpoints should be in progress at any given point in time. This is very useful for cases where the best checkpoint interval is not easy to determine, or where once in a while checkpoints may take long. Previously, such situations lead to checkpoint queue-up, where the system ended up being more busy with checkpoints than with processing. After this change, the checkpoint rate will by default sort of self-regulate: The checkpoint interval is the most frequent time at which checkpoints will occur, but they will occur slower if they take longer than that interval. ### Checkpoint Config This also introduces the CheckpointConfig class that holds all checkpointing related parameters to more simply pass them between environment, streamgraph, etc... ### Default number of concurrent checkpoints Previously, the maximum number of concurrent checkpoints was implicitly infinite. I would suspect that most people will want to run the system such that only one checkpoint is ever concurrently active, so I set the default value for this to 1. In some corner cases, it may be interesting to set that value higher. ### WIP for delay between checkpoints This also contains some WIP to define a minimum time between checkpoint attempts. That flag would tell the system not only to not do more than one checkpoint concurrently, but to leave at least a certain time between completion of one checkpoint, and the triggering of the next. It basically defines a guaranteed time that is "work only" between checkpoints. You can merge this pull request into a Git repository by running: $ git pull https://github.com/StephanEwen/incubator-flink checkpoints_configure Alternatively you can review and apply these changes as the patch at: https://github.com/apache/flink/pull/1408.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 #1408 ---- commit e4d063867ac09cbec6666ecbca92816cfb6faf6c Author: Stephan Ewen <sewen@apache.org> Date: 2015-11-19T17:03:55Z [hotfix] Java-7-ify the ExecutionConfig class commit 79def44a2e973c8b6817821d5d7342731f7e7aa2 Author: Stephan Ewen <sewen@apache.org> Date: 2015-11-19T18:05:47Z [FLINK-3051] [streaming] Add mechanisms to control the maximum number of concurrent checkpoints ----

3. Github user rmetzger commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159851509 Good addition! More control over our checkpointing is something people were asking me at talks. Could you add a few sentences to the documentation about this?
4. Github user StephanEwen commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159866582 Where is the docs would be the best place for that? - A new entry in the "Programing Guides" menu? - Or a section in the streaming guide? I would vote for the first
5. Github user rmetzger commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159868163 I think it fits best into the "Fault Tolerance" section of the Streaming Guide: https://ci.apache.org/projects/flink/flink-docs-release-0.10/apis/streaming_guide.html#fault-tolerance If you think that section will grow to big, we could move it to a new page and link it from the streaming guide.
6. Github user StephanEwen commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159872110 Should we pull that into a separate document? It becomes quite large...
7. Github user rmetzger commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159872462 +1
8. Github user StephanEwen commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159876995 Will do this in a separate pull request as follow up. Any concerns about merging this?
9. Github user gyfora commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159910663 Please give me an hour or so to look at this :)
10. Github user gyfora commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159915515 Looks good and the minimum delay between checkpoints would be an extremely useful feature. What's missing for that?
11. Github user StephanEwen commented on the pull request: https://github.com/apache/flink/pull/1408#issuecomment-159932475 Missing for that is a bit of code in the checkpoint coordinator that marks the time when checkpoints become possible again and adds that delay to the time when the next checkpoint is scheduled. I'd suggest to add that as a followup...
12. Github user asfgit closed the pull request at: https://github.com/apache/flink/pull/1408
13. Fixed via 55fd5f32d7ef0292a01192ab08456fae49b91791