Item 187
git_comments:

1. * * Returns the {@link AllocationID} of active {@link TaskSlot}s attached to the job with the given {@link JobID}. * * @param jobId The {@code JobID} of the job for which the {@code AllocationID}s of the attached active * {@link TaskSlot}s shall be returned. * @return A set of {@code AllocationID}s that belong to active {@code TaskSlot}s having the passed {@code JobID}.

git_commits:

1. **summary:** [FLINK-17341][runtime] Add TaskSlotTable.getActiveTaskAllocationIdsPerJob
   message: [FLINK-17341][runtime] Add TaskSlotTable.getActiveTaskAllocationIdsPerJob Introduced new method for retrieving AllocationIDs of active TaskSlots based on the JobID in a Set. This newly introduced method replaces getActiveSlots(JobID) returning an Iterator to avoid ConcurrentModificationExceptions. This closes #13564.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-17341][runtime] Fixed ConcurrentModificationException in TaskExecutor
   body: ## What is the purpose of the change The fix is necessary since there is a possibility that a `ConcurrentModificationException` is thrown as described in FLINK-17341. ## Brief change log The solution includes switching from `Iterator` to a `Set` of `AllocationID`s: - Introduced `TaskSlotTable.getActiveTaskAllocationIdsPerJob(JobID)` - Replaced usage of `TaskSlotTable.getActiveSlots(JobID)` by `TaskSlotTable.getActiveTaskAllocationIdsPerJob(JobID)` - Removed `TaskSlotTable.getActiveSlots(JobID)` and `TaskSlotTable.AllocationIDIterator` ## Verifying this change This change is already covered by existing tests, such as `TaskSlotTableImplTest.testTryMarkSlotActive`. ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): no - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: no - The serializers: no - The runtime per-record code paths (performance sensitive): no - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Kubernetes/Yarn/Mesos, ZooKeeper: no - The S3 file system connector: no ## Documentation - Does this pull request introduce a new feature? no - If yes, how is the feature documented? not applicable
   label: code-design

github_pulls_comments:

1. Thanks a lot for your contribution to the Apache Flink project. I'm the @flinkbot. I help the community to review your pull request. We will use this comment to track the progress of the review. ## Automated Checks Last check on commit c8deaa314d1ca5f93f66c45391a5ba0cb8d9fd22 (Thu Oct 08 15:14:53 UTC 2020) **Warnings:** * No documentation files were touched! Remember to keep the Flink docs up to date! _{Mention the bot in a comment to re-run the automated checks.} ## Review Progress * ❓ 1. The [description] looks good. * ❓ 2. There is [consensus] that the contribution should go into to Flink. * ❓ 3. Needs [attention] from. * ❓ 4. The change fits into the overall [architecture]. * ❓ 5. Overall code [quality] is good. Please see the [Pull Request Review Guide](https://flink.apache.org/contributing/reviewing-prs.html) for a full explanation of the review process.<details> The Bot is tracking the review progress through labels. Labels are applied according to the order of the review items. For consensus, approval by a Flink committer of PMC member is required <summary>Bot commands</summary> The @flinkbot bot supports the following commands: - `@flinkbot approve description` to approve one or more aspects (aspects: `description`, `consensus`, `architecture` and `quality`) - `@flinkbot approve all` to approve all aspects - `@flinkbot approve-until architecture` to approve everything until `architecture` - `@flinkbot attention @username1 [@username2 ..]` to require somebody's attention - `@flinkbot disapprove architecture` to remove an approval you gave earlier </details>
2. <!-- Meta data { "version" : 1, "metaDataEntries" : [{ "hash" : "29eb9920d726629cb5d7c3b0a1093e46393b65eb", "status" : "DELETED", "url" : "https://dev.azure.com/apache-flink/98463496-1af2-4620-8eab-a2ecc1a2e6fe/_build/results?buildId=7303", "triggerID" : "29eb9920d726629cb5d7c3b0a1093e46393b65eb", "triggerType" : "PUSH" }, { "hash" : "26b91f610ae3adfa35b1f6b9ac5d60d6ff598cfc", "status" : "UNKNOWN", "url" : "TBD", "triggerID" : "26b91f610ae3adfa35b1f6b9ac5d60d6ff598cfc", "triggerType" : "PUSH" }, { "hash" : "e50feab6bbfe6bd193c5830cbb31d9899df7f2af", "status" : "SUCCESS", "url" : "https://dev.azure.com/apache-flink/98463496-1af2-4620-8eab-a2ecc1a2e6fe/_build/results?buildId=7322", "triggerID" : "e50feab6bbfe6bd193c5830cbb31d9899df7f2af", "triggerType" : "PUSH" }] }--> ## CI report: * 26b91f610ae3adfa35b1f6b9ac5d60d6ff598cfc UNKNOWN * e50feab6bbfe6bd193c5830cbb31d9899df7f2af Azure: [SUCCESS](https://dev.azure.com/apache-flink/98463496-1af2-4620-8eab-a2ecc1a2e6fe/_build/results?buildId=7322) <details> <summary>Bot commands</summary> The @flinkbot bot supports the following commands: - `@flinkbot run travis` re-run the last Travis build - `@flinkbot run azure` re-run the last Azure build </details>

github_pulls_reviews:

1. **body:** ```suggestion return Sets.newHashSet(new TaskSlotIterator(jobId, TaskSlotState.ACTIVE); ``` might be a bit shorter.
   label: code-design
2. **body:** ```suggestion return Sets.newHashSet(new TaskSlotIterator(jobId, TaskSlotState.ACTIVE)) .stream() .map(TaskSlot::getAllocationId) .collect(Collectors.toSet()); ``` The only other shorter way I can imagine is the one above. I just hesitated because it would mean iterating over the collection twice just to save a few lines. ```suggestion return StreamSupport.stream(Spliterators.spliteratorUnknownSize(new TaskSlotIterator(jobId, TaskSlotState.ACTIVE), Spliterator.ORDERED), false) .map(TaskSlot::getAllocationId) .collect(Collectors.toSet()); ``` Creating a stream right out of the Iterator does not make the code more readable either, IMHO.
   label: code-design
3. Yes, you are completely right. Forget about my comment. I overlooked that we are only interested in the `AllocationIDs`.

jira_issues:

1. **summary:** freeSlot in TaskExecutor.closeJobManagerConnection cause ConcurrentModificationException
   description: TaskExecutor may freeSlot when closeJobManagerConnection. freeSlot will modify the TaskSlotTable.slotsPerJob. this modify will cause ConcurrentModificationException. {code:java} Iterator<AllocationID> activeSlots =

taskSlotTable.getActiveSlots(jobId); final FlinkException freeingCause = new FlinkException("Slot could not be marked inactive."); while (activeSlots.hasNext()) { AllocationID activeSlot = activeSlots.next(); try { if (!taskSlotTable.markSlotInactive(activeSlot, taskManagerConfiguration.getTimeout())) { freeSlotInternal(activeSlot, freeingCause); } } catch (SlotNotFoundException e) { log.debug("Could not mark the slot {} inactive.", jobId, e); } } {code} error log： {code:java} 2020-04-21 23:37:11,363 ERROR org.apache.flink.runtime.rpc.akka.AkkaRpcActor - Caught exception while executing runnable in main thread. java.util.ConcurrentModificationException at java.util.HashMap$HashIterator.nextNode(HashMap.java:1437) at java.util.HashMap$KeyIterator.next(HashMap.java:1461) at org.apache.flink.runtime.taskexecutor.slot.TaskSlotTable$TaskSlotIterator.hasNext(TaskSlotTable.java:698) at org.apache.flink.runtime.taskexecutor.slot.TaskSlotTable$AllocationIDIterator.hasNext(TaskSlotTable.java:652) at org.apache.flink.runtime.taskexecutor.TaskExecutor.closeJobManagerConnection(TaskExecutor.java:1314) at org.apache.flink.runtime.taskexecutor.TaskExecutor.access$1300(TaskExecutor.java:149) at org.apache.flink.runtime.taskexecutor.TaskExecutor$JobLeaderListenerImpl.lambda$jobManagerLostLeadership$1(TaskExecutor.java:1726) at org.apache.flink.runtime.rpc.akka.AkkaRpcActor.handleRunAsync(AkkaRpcActor.java:397) at org.apache.flink.runtime.rpc.akka.AkkaRpcActor.handleRpcMessage(AkkaRpcActor.java:190) at org.apache.flink.runtime.rpc.akka.AkkaRpcActor.handleMessage(AkkaRpcActor.java:152) at akka.japi.pf.UnitCaseStatement.apply(CaseStatements.scala:26) at akka.japi.pf.UnitCaseStatement.apply(CaseStatements.scala:21) at scala.PartialFunction$class.applyOrElse(PartialFunction.scala:123) {code}

jira_issues_comments:

1. Thanks for reporting this issue [~huwh]. I believe that the problem still exists in the current master.
2. [~trohrmann] I would like to fix this issue, could you assign this to me?
3. How do you want to solve the problem [~huwh]?
4. I think we can put the slots that need to be free in a list, and then free them out of the loop. [~trohrmann]
5. Hi [~huwh], thanks for volunteering to fix it. Unfortunately, I took over yesterday already. But I followed the same approach you're proposing. Feel free to review [PR #13564|https://github.com/apache/flink/pull/13564].
6. Fixed via master: 832a4c19541d9f23017425f53e7d1eb884806c31 1.11.3: 5fd19275cc0b0d1f7d57aa12b2e02cede8b6c0c1 1.10.3: 0d78e806691cd04ac54713fc64372b7aeead6919