TAJO-1397

hyunsik

2015-03-12

Contents

Chapter 1

Root issue TAJO-1397

1.1 Summary

Resource allocation should be fine grained.

1.2 Description

See the comment:

https://issues.apache.org/jira/browse/TAJO-540?focusedCommentId=14359478&page=com.atlassian.jira.plugin.system
tabpanel#comment-14359478

From the discussion in TAJO-540

{quote}
In general, query (or job) scheduler aims at the maximum resource utilization. For multi-tenancy, we also need to consider the fairness for multiple users (or queries). BTW, the maximum resource utilization and fairness are usually conflict to each other in many cases. To mitigate this problem, many scheduler seems to use preemption approach.

In this point, our resource and scheduler system has the following problems:
* A query exclusively uses allocated resources at the first time until the query is completed or failed.
* There is no mechanism to deallocate resources during query processing.
* Preempt is also not allowed.

To achieve the multi tenancy, we should change our resource circulation. Especially, resource allocation must be fine grained instead of per query.
So, I'll create a jira issue to change the resource circulation. We have to do this issue firstly in my opinion. If we achieve this, implementing multi-tenant scheduler would be much easier than now. It would be a good starting point of this issue.
{quote}

1.3 Attachments

1. old_resource_circuit.png
2. ResoruceSequence.jpg

3. resource_circuit.png
4. TAJO-1397_2.patch
5. TAJO-1397_3.patch
6. TAJO-1397_4.patch
7. TAJO-1397.patch

1.4 Commits

1. Commit **5a02873** by **Jinho Kim** (2015-07-20): TAJO-1397: Resource allocation should be fine grained. (jinho)

   Closes #608

1.5 Comments

1. **hyunsik:** This figure represents the current resource circulation.
2. **hyunsik:** I'll post the figure of my proposed circulation and its description soon.
3. **hyunsik:** Here is my proposed resource circulation. See resource_circuit.png file.

   * Node sends a heartbeats with its resource and status periodically.
   * ResourceTracker maintains a global view of node resources.

   When a query master is launched, the query master requests necessary resource to ResourceTracker. This is repeatedly performed while a query is executing. The request will includes the following information:
   * query id
   * user
   * A set of resource requests, each of which will include
   * priority
   * the number of containers and resource capacity for containers
   * desired host names
   * type of task (leaf or intermediate)
   * ...

   When the query master receives the resource requests, it schedules tasks to nodes. In some cases where a node cannot accept the task schedule, the node can reject the request. It can be occur when the available resources that ResourceTracker and actual nodes keep are different in some cases.

   In the proposed resource circulation, the resource allocation will be performed in a task level. So, it will give even more flexibility to adjust the resource capacity per user or query during query processing.

4. **jhkim:** [~hyunsik]
   Your proposal looks great to large cluster.I have a question. If a tasks is fast(50~ 1000 ms) finished, minimum scheduling costs will be heartbeat delay of node.
   Have you an idea?

5. **jhkim:** [~hyunsik]
   I've attach the sequence diagram from your proposal.
   I think If a node dynamically handle period of heartbeat, delay will not be a problem.

6. **jhkim:** If you don't start this issue yet, I will create a sub task for this.

7. **hyunsik:** Please go ahead. You can also feel free to assign itself if you want. I don't have any free slot right now.

8. **jhkim:** OK, I will start initial work

9. **jhkim:** These issues does not cause in this patch

10. **jhkim:** fix jdk6 compilation failure

11. **jhkim:** committed it to master branch
    Thanks!

12. **hudson:** ABORTED: Integrated in Tajo-master-CODEGEN-build #393 (See [https://builds.apache.org/job/Tajo-master-CODEGEN-build/393/])
    TAJO-1397: Resource allocation should be fine grained. (jinho) (jhkim: rev 5a02873d5f0bd8fedd8527808bd3d4ecbe7
    * tajo-core/src/main/java/org/apache/tajo/worker/TaskHistory.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/TajoResourceTracker.java
    * tajo-core/src/main/java/org/apache/tajo/querymaster/Repartitioner.java
    * tajo-core/src/main/java/org/apache/tajo/worker/NodeResourceManager.java
    * tajo-core/src/main/proto/QueryCoordinatorProtocol.proto
    * tajo-core/src/test/java/org/apache/tajo/worker/TestHistory.java
    * tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestFullOuterMergeJoinExec.java
    * tajo-core/src/main/java/org/apache/tajo/querymaster/Task.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/TajoRMContext.java
    * tajo-core/src/test/java/org/apache/tajo/master/TestRepartitioner.java
    * tajo-core/src/main/resources/webapps/admin/cluster.jsp
    * tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptToSchedulerEvent.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerContainer.java
    * tajo-core/src/test/java/org/apache/tajo/querymaster/TestKillQuery.java
    * tajo-core/src/main/java/org/apache/tajo/master/event/GrouppedContainerAllocatorEvent.java
    * tajo-core/src/main/java/org/apache/tajo/worker/event/QMResourceAllocateEvent.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/NodeStatus.java
    * tajo-core/src/main/java/org/apache/tajo/master/scheduler/SimpleScheduler.java
    * tajo-core/src/main/proto/ContainerProtocol.proto
    * tajo-core/src/main/resources/webapps/admin/querytasks.jsp
    * tajo-common/src/main/java/org/apache/tajo/util/StringUtils.java
    * tajo-core/src/main/java/org/apache/tajo/engine/planner/enforce/Enforcer.java
    * tajo-rpc/tajo-rpc-protobuf/src/main/java/org/apache/tajo/rpc/AsyncRpcServer.java
    * tajo-core/src/main/java/org/apache/tajo/master/event/ContainerEvent.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerResourceManager.java
    * tajo-common/src/main/java/org/apache/tajo/util/TUtil.java
    * tajo-core/src/test/java/org/apache/tajo/worker/TestTaskExecutor.java
    * tajo-core/src/main/proto/QueryMasterProtocol.proto
    * tajo-core/src/main/java/org/apache/tajo/session/Session.java
    * tajo-core/src/main/java/org/apache/tajo/master/event/LocalTaskEvent.java
    * tajo-core/src/main/java/org/apache/tajo/master/rm/NodeEvent.java
    * tajo-core/src/main/resources/webapps/worker/queryplan.jsp
    * tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockStopEvent.java
    * tajo-core/src/main/java/org/apache/tajo/worker/event/TaskManagerEvent.java

* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskExecutorEvent.java
* tajo-core/src/main/resources/webapps/admin/index.jsp
* tajo-core/src/main/java/org/apache/tajo/master/LaunchTaskRunnersEvent.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Query.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Stage.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockTaskManager.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerStopEvent.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockTaskExecutor.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskFatalErrorEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryManager.java
* tajo-core/src/test/java/org/apache/tajo/master/rm/TestTajoResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerLivelinessMonitor.java
* tajo-core/src/main/proto/TajoWorkerProtocol.proto
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskRequestEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerReconnectEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/TaskRunnerGroupEvent.java
* tajo-core/src/main/resources/webapps/admin/query.jsp
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/dictionary/ClusterTableDescriptor.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/AbstractQueryScheduler.java
* tajo-core/src/test/java/org/apache/tajo/util/metrics/TestSystemMetrics.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainerId.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskStartEvent.java
* tajo-core/src/main/java/org/apache/tajo/engine/query/TaskRequestImpl.java
* tajo-core/src/main/java/org/apache/tajo/resource/DefaultResourceCalculator.java
* tajo-core/src/main/java/org/apache/tajo/worker/NodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeEventType.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockExecutionBlock.java
* tajo-core/src/test/java/org/apache/tajo/engine/query/TestOuterJoinQuery.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerEvent.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMasterManagerService.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestLeftOuterHashJoinExec.java
* .travis.yml
* tajo-core/src/main/java/org/apache/tajo/ws/rs/resources/ClusterResource.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QuerySchedulingInfo.java
* CHANGES
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QueueInfo.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceDeallocateEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoContainerProxy.java
* tajo-core/src/main/resources/webapps/admin/query_executor.jsp
* tajo-common/src/main/java/org/apache/tajo/util/NumberUtil.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskManager.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryInProgress.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptStatusUpdateEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/TajoResourceScheduler.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestPhysicalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockErrorEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/SchedulingAlgorithms.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMaster.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerEvent.java
* tajo-core/src/test/java/org/apache/tajo/querymaster/TestQueryState.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestRightOuterMergeJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/util/history/HistoryWriter.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockNodeResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/worker/LegacyTaskImpl.java

* tajo-core/src/main/java/org/apache/tajo/master/event/QueryStartEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainerIdPBImpl.java
* tajo-core/src/test/java/org/apache/tajo/master/scheduler/TestSimpleScheduler.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeStatusEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunnerHistory.java
* tajo-core/src/main/java/org/apache/tajo/engine/parser/SQLAnalyzer.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerContainerId.java
* tajo-cli/src/main/java/org/apache/tajo/cli/tools/TajoAdmin.java
* tajo-core/src/main/resources/webapps/worker/taskhistory.jsp
* tajo-core/src/test/java/org/apache/tajo/TajoTestingCluster.java
* tajo-core/src/main/resources/webapps/worker/index.jsp
* tajo-core/src/main/java/org/apache/tajo/master/event/StageContainerAllocationEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerEventType.java
* tajo-core/src/main/resources/webapps/admin/task.jsp
* tajo-core/src/test/java/org/apache/tajo/worker/TestFetcher.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestNodeResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerResourceManager.java
* tajo-common/src/main/java/org/apache/tajo/SessionVars.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeReconnectEvent.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/DataChannel.java
* tajo-core/src/main/resources/webapps/worker/task.jsp
* tajo-core/src/main/java/org/apache/tajo/worker/ExecutionBlockContext.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/PhysicalPlannerImpl.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeState.java
* tajo-core/src/main/java/org/apache/tajo/master/event/StageEventType.java
* tajo-core/src/main/java/org/apache/tajo/worker/ResourceAllocator.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockStartEvent.java
* tajo-core/src/main/proto/ResourceProtos.proto
* tajo-core/src/main/resources/webapps/worker/querydetail.jsp
* tajo-core/src/test/java/org/apache/tajo/worker/TestNodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/Worker.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoWorkerManagerService.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainer.java
* tajo-core/src/main/java/org/apache/tajo/master/event/StageShuffleReportEvent.java
* tajo-core/src/main/proto/ResourceTrackerProtocol.proto
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestBNLJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/metrics/WorkerResourceMetricsGaugeSet.java
* tajo-core/src/main/java/org/apache/tajo/util/JSPUtil.java
* tajo-core/src/main/resources/webapps/worker/tasks.jsp
* tajo-core/src/main/resources/webapps/worker/taskdetail.jsp
* tajo-core/src/main/java/org/apache/tajo/engine/planner/physical/ScanExec.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoResourceAllocator.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/MasterPlan.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptEventType.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerStartEvent.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestMergeJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/QueryStopEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskCompletionEvent.java
* tajo-core/pom.xml
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/FunctionDesc.java
* tajo-core/src/main/java/org/apache/tajo/util/history/HistoryReader.java
* tajo-core/src/main/java/org/apache/tajo/master/event/ContainerAllocatorEventType.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/SchedulerEventType.java

* tajo-core/src/main/java/org/apache/tajo/master/TajoMaster.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/CatalogServer.java
* tajo-core/src/test/java/org/apache/tajo/engine/query/TestJoinOnPartitionedTables.java
* tajo-project/pom.xml
* tajo-core/src/main/java/org/apache/tajo/master/ContainerProxy.java
* tajo-core/src/main/resources/webapps/admin/querydetail.jsp
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptAssignedEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskContainer.java
* tajo-core/src/main/java/org/apache/tajo/ws/rs/responses/WorkerResponse.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/master/exec/NonForwardQueryResultSystemScanner.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockWorkerContext.java
* tajo-docs/src/main/sphinx/configuration/worker_configuration.rst
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestHashJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMasterTask.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMasterClientService.java
* tajo-core/src/main/java/org/apache/tajo/worker/FetchImpl.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryCoordinatorService.java
* tajo-core/src/main/java/org/apache/tajo/worker/WorkerHeartbeatService.java
* tajo-core/src/main/resources/webapps/worker/taskcontainers.jsp
* tajo-core/src/test/java/org/apache/tajo/master/TestNonForwardQueryResultSystemScanner.java
* tajo-core/src/main/java/org/apache/tajo/master/TaskRunnerLauncher.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/ResourceReserveSchedulerEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeLivelinessMonitor.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/TaskAttempt.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerStatusEvent.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestTaskManager.java
* tajo-core/src/main/java/org/apache/tajo/engine/query/TaskRequest.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoWorker.java
* tajo-common/src/main/java/org/apache/tajo/conf/TajoConf.java
* tajo-core/src/test/java/org/apache/tajo/QueryTestCaseBase.java
* tajo-client/src/main/proto/ClientProtos.proto
* tajo-core/src/main/java/org/apache/tajo/master/event/ContainerAllocationEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerState.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceAllocateEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/AbstractResourceAllocator.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/AbstractTaskScheduler.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockNodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskImpl.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunnerManager.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestFullOuterHashJoinExec.java
* tajo-plan/src/main/java/org/apache/tajo/plan/util/PlannerUtil.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunner.java
* tajo-core/src/main/java/org/apache/tajo/resource/NodeResource.java
* tajo-core/src/main/java/org/apache/tajo/master/cluster/WorkerConnectionInfo.java
* tajo-plan/src/main/proto/Plan.proto
* tajo-core/src/main/java/org/apache/tajo/worker/Task.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/SchedulerEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskExecutor.java
* tajo-core/src/main/resources/webapps/worker/querytasks.jsp
* tajo-core/src/main/java/org/apache/tajo/querymaster/DefaultTaskScheduler.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/builder/DistinctGroupbyBuilder.java
* tajo-core/src/test/java/org/apache/tajo/master/scheduler/TestFifoScheduler.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QueueState.java

* tajo-core/src/main/java/org/apache/tajo/master/container/TajoConverterUtils.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/SimpleFifoScheduler.java

13. **hudson:** SUCCESS: Integrated in Tajo-master-build #753 (See [https://builds.apache.org/job/Tajo-master-build/753/])
TAJO-1397: Resource allocation should be fine grained. (jinho) (jhkim: rev 5a02873d5f0bd8fedd8527808bd3d4ecbe7
* tajo-core/src/main/proto/ResourceProtos.proto
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerStartEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QueueState.java
* tajo-core/src/main/java/org/apache/tajo/master/TaskRunnerLauncher.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoResourceManager.java
* tajo-plan/src/main/proto/Plan.proto
* tajo-core/src/main/resources/webapps/admin/cluster.jsp
* tajo-core/src/main/java/org/apache/tajo/engine/query/TaskRequest.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskStartEvent.java
* tajo-core/src/main/proto/ResourceTrackerProtocol.proto
* tajo-common/src/main/java/org/apache/tajo/util/TUtil.java
* tajo-core/src/test/java/org/apache/tajo/QueryTestCaseBase.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeLivelinessMonitor.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/DefaultTaskScheduler.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerContainerId.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockNodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/util/history/HistoryReader.java
* tajo-core/src/main/java/org/apache/tajo/worker/NodeResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/master/event/LocalTaskEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptToSchedulerEvent.java
* tajo-rpc/tajo-rpc-protobuf/src/main/java/org/apache/tajo/rpc/AsyncRpcServer.java
* tajo-common/src/main/java/org/apache/tajo/conf/TajoConf.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunner.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerEvent.java
* tajo-core/src/main/proto/QueryCoordinatorProtocol.proto
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoRMContext.java
* tajo-core/src/main/java/org/apache/tajo/resource/DefaultResourceCalculator.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskHistory.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Query.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunnerHistory.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskRequestEvent.java
* tajo-common/src/main/java/org/apache/tajo/util/StringUtils.java
* tajo-core/src/test/java/org/apache/tajo/TajoTestingCluster.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/PhysicalPlannerImpl.java
* tajo-core/src/main/java/org/apache/tajo/engine/query/TaskRequestImpl.java
* tajo-core/src/main/resources/webapps/worker/taskdetail.jsp
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/builder/DistinctGroupbyBuilder.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Repartitioner.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestPhysicalPlanner.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/SchedulerEventType.java
* tajo-project/pom.xml
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestHashJoinExec.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestRightOuterMergeJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/util/history/HistoryWriter.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/SchedulerEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoWorker.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryCoordinatorService.java

* tajo-core/src/main/java/org/apache/tajo/master/event/TaskCompletionEvent.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestLeftOuterHashJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/AbstractTaskScheduler.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMasterManagerService.java
* tajo-core/src/test/java/org/apache/tajo/querymaster/TestQueryState.java
* tajo-core/src/main/resources/webapps/worker/queryplan.jsp
* tajo-core/src/test/java/org/apache/tajo/master/rm/TestTajoResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerContainer.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockStartEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeEventType.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeState.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QuerySchedulingInfo.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/enforce/Enforcer.java
* tajo-core/src/main/java/org/apache/tajo/master/cluster/WorkerConnectionInfo.java
* tajo-core/src/main/java/org/apache/tajo/resource/NodeResource.java
* tajo-core/src/main/java/org/apache/tajo/worker/WorkerHeartbeatService.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoConverterUtils.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestHistory.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockNodeResourceManager.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockTaskManager.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestBNLJoinExec.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/CatalogServer.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskExecutor.java
* tajo-core/src/main/resources/webapps/worker/querytasks.jsp
* tajo-core/src/test/java/org/apache/tajo/worker/TestFetcher.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMasterTask.java
* tajo-core/src/main/java/org/apache/tajo/master/event/ContainerEvent.java
* tajo-core/src/main/resources/webapps/admin/querydetail.jsp
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptEventType.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoWorkerManagerService.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerEventType.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerReconnectEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerLivelinessMonitor.java
* tajo-core/src/main/java/org/apache/tajo/master/TaskRunnerGroupEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/ContainerAllocationEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryManager.java
* tajo-core/src/main/java/org/apache/tajo/worker/FetchImpl.java
* tajo-core/src/main/resources/webapps/worker/querydetail.jsp
* tajo-core/src/main/java/org/apache/tajo/master/event/StageContainerAllocationEvent.java
* tajo-catalog/tajo-catalog-server/src/main/java/org/apache/tajo/catalog/dictionary/ClusterTableDescriptor.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/SchedulingAlgorithms.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerState.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockWorkerContext.java
* tajo-core/src/main/java/org/apache/tajo/worker/AbstractResourceAllocator.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/TaskAttempt.java
* tajo-core/src/main/proto/ContainerProtocol.proto
* tajo-core/src/main/java/org/apache/tajo/worker/ResourceAllocator.java
* tajo-core/src/main/proto/QueryMasterProtocol.proto
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeReconnectEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeEvent.java
* tajo-core/src/test/java/org/apache/tajo/master/TestRepartitioner.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskContainer.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptAssignedEvent.java

* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceAllocateEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoWorkerResourceManager.java
* tajo-core/src/main/proto/TajoWorkerProtocol.proto
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/SimpleScheduler.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoContainerProxy.java
* CHANGES
* tajo-core/src/test/java/org/apache/tajo/worker/TestNodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/engine/parser/SQLAnalyzer.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainerId.java
* tajo-core/src/test/java/org/apache/tajo/util/metrics/TestSystemMetrics.java
* tajo-core/src/main/resources/webapps/admin/querytasks.jsp
* tajo-core/src/main/java/org/apache/tajo/util/JSPUtil.java
* tajo-core/src/main/resources/webapps/admin/task.jsp
* tajo-cli/src/main/java/org/apache/tajo/cli/tools/TajoAdmin.java
* tajo-core/src/main/resources/webapps/worker/taskcontainers.jsp
* tajo-core/src/main/java/org/apache/tajo/worker/LegacyTaskImpl.java
* tajo-core/src/test/java/org/apache/tajo/engine/query/TestJoinOnPartitionedTables.java
* tajo-docs/src/main/sphinx/configuration/worker_configuration.rst
* tajo-core/src/main/java/org/apache/tajo/worker/NodeStatusUpdater.java
* tajo-core/src/main/java/org/apache/tajo/worker/TajoResourceAllocator.java
* tajo-core/src/main/java/org/apache/tajo/ws/rs/resources/ClusterResource.java
* tajo-core/src/main/java/org/apache/tajo/master/ContainerProxy.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestMergeJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/master/event/StageShuffleReportEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/ExecutionBlockContext.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/QueryStopEvent.java
* .travis.yml
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerStatusEvent.java
* tajo-common/src/main/java/org/apache/tajo/SessionVars.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskImpl.java
* tajo-core/src/main/java/org/apache/tajo/master/event/StageEventType.java
* tajo-core/src/main/resources/webapps/worker/task.jsp
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/AbstractQueryScheduler.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/QueryMaster.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestTaskManager.java
* tajo-core/src/test/java/org/apache/tajo/master/scheduler/TestSimpleScheduler.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/QMResourceAllocateEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/GrouppedContainerAllocatorEvent.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockExecutionBlock.java
* tajo-core/src/main/java/org/apache/tajo/master/LaunchTaskRunnersEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainerIdPBImpl.java
* tajo-core/src/test/java/org/apache/tajo/engine/query/TestOuterJoinQuery.java
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestFullOuterMergeJoinExec.java
* tajo-core/src/test/java/org/apache/tajo/master/TestNonForwardQueryResultSystemScanner.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMasterClientService.java
* tajo-core/src/main/java/org/apache/tajo/session/Session.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestTaskExecutor.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockErrorEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/TajoResourceScheduler.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerEvent.java
* tajo-core/src/main/resources/webapps/admin/query.jsp
* tajo-core/src/main/java/org/apache/tajo/master/rm/WorkerResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/physical/ScanExec.java
* tajo-core/src/test/java/org/apache/tajo/master/scheduler/TestFifoScheduler.java

* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskRunnerStopEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskManagerEvent.java
* tajo-core/src/main/resources/webapps/worker/taskhistory.jsp
* tajo-core/pom.xml
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeStatus.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskAttemptStatusUpdateEvent.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/TaskExecutorEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/event/TaskFatalErrorEvent.java
* tajo-core/src/test/java/org/apache/tajo/worker/MockTaskExecutor.java
* tajo-core/src/main/java/org/apache/tajo/worker/Task.java
* tajo-core/src/main/java/org/apache/tajo/master/TajoMaster.java
* tajo-core/src/main/java/org/apache/tajo/metrics/WorkerResourceMetricsGaugeSet.java
* tajo-core/src/main/java/org/apache/tajo/master/event/QueryStartEvent.java
* tajo-core/src/main/resources/webapps/admin/index.jsp
* tajo-core/src/main/java/org/apache/tajo/master/rm/Worker.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/TajoResourceTracker.java
* tajo-core/src/main/resources/webapps/worker/tasks.jsp
* tajo-core/src/test/java/org/apache/tajo/querymaster/TestKillQuery.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Stage.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskManager.java
* tajo-core/src/main/java/org/apache/tajo/master/event/ContainerAllocatorEventType.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/DataChannel.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/NodeResourceDeallocateEvent.java
* tajo-core/src/main/java/org/apache/tajo/engine/planner/global/MasterPlan.java
* tajo-core/src/main/java/org/apache/tajo/master/exec/NonForwardQueryResultSystemScanner.java
* tajo-core/src/main/resources/webapps/worker/index.jsp
* tajo-core/src/test/java/org/apache/tajo/engine/planner/physical/TestFullOuterHashJoinExec.java
* tajo-core/src/main/java/org/apache/tajo/ws/rs/responses/WorkerResponse.java
* tajo-core/src/main/resources/webapps/admin/query_executor.jsp
* tajo-catalog/tajo-catalog-common/src/main/java/org/apache/tajo/catalog/FunctionDesc.java
* tajo-core/src/test/java/org/apache/tajo/worker/TestNodeResourceManager.java
* tajo-core/src/main/java/org/apache/tajo/querymaster/Task.java
* tajo-core/src/main/java/org/apache/tajo/worker/event/ExecutionBlockStopEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/QueueInfo.java
* tajo-plan/src/main/java/org/apache/tajo/plan/util/PlannerUtil.java
* tajo-core/src/main/java/org/apache/tajo/worker/TaskRunnerManager.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/event/ResourceReserveSchedulerEvent.java
* tajo-common/src/main/java/org/apache/tajo/util/NumberUtil.java
* tajo-core/src/main/java/org/apache/tajo/master/QueryInProgress.java
* tajo-core/src/main/java/org/apache/tajo/master/rm/NodeStatusEvent.java
* tajo-core/src/main/java/org/apache/tajo/master/scheduler/SimpleFifoScheduler.java
* tajo-client/src/main/proto/ClientProtos.proto
* tajo-core/src/main/java/org/apache/tajo/master/container/TajoContainer.java

14. **hudson:** ABORTED: Integrated in Tajo-master-CODEGEN-build #394 (See [https://builds.apache.org/job/Tajo-master-CODEGEN-build/394/])
close related issues by TAJO-1397 (jhkim: rev a06b5e56490116e345afc737de46bf1a4ed256a9)
* BUILDING

15. **hudson:** SUCCESS: Integrated in Tajo-master-build #754 (See [https://builds.apache.org/job/Tajo-master-build/754/])
close related issues by TAJO-1397 (jhkim: rev a06b5e56490116e345afc737de46bf1a4ed256a9)

* BUILDING

16. **hudson:** ABORTED: Integrated in Tajo-master-CODEGEN-build #395 (See [https://builds.apache.org/job/Tajo-master-CODEGEN-build/395/])
close related issues by TAJO-1397 (jhkim: rev 4bfd693acb1a294c6d52fa28e7c61b77f0a2034f)
* dev-support/findbugs-exclude.xml

17. **hudson:** SUCCESS: Integrated in Tajo-master-build #755 (See [https://builds.apache.org/job/Tajo-master-build/755/])
close related issues by TAJO-1397 (jhkim: rev 4bfd693acb1a294c6d52fa28e7c61b77f0a2034f)
* dev-support/findbugs-exclude.xml

1.6 Pull requests

1.6.1 Pull request 608

Title: TAJO-1397: Resource allocation should be fine grained.
Author: jinossy
Date: 2015-06-21
Status: closed
Comments:

1. **jinossy** (2015-06-21): This PR is not ready to review. please don't start review.
I just test the Travis-CI

2. **jinossy** (2015-07-14): This patch is ready, please review this PR

3. **jihoonson** (2015-07-15): @jinossy, really nice work!
I'm still reviewing and not finished yet. Here, I left some comments first, and will finish soon.
- StringUtils.java
- The comment of isPureAscii() is incomplete.
- QueryCoordinatorProtocol.proto
- As you know, @hyunsik is working on refactoring our error propagation system. According to the guide at https://cwiki.apache.org/confluence/display/TAJO/RPC+Protocol+Design+Guide, please separate QueryCoordinatorProtocolService from this file.
- Also, please add some comments for the methods of QueryCoordinatorProtocolService.
- QueryInProgress.java
- Please change the comments of allocateToQueryMaster() into the formal java doc format.
- Please add some comments for submitToQueryMaster().
- QueryManager.java
- The name of startQueryJob() looks not intuitive. How about isQueryExecutable()?
- TajoResourceManager.java
- Please remove the unused parameter, 'systemConf', from the constructor.
- QuerySchedulingInfo.java
- Please add some comments for the class and its variables.
- SimpleScheduler.java
- Please add some comments for the class.
- Would you please explain what PARALLEL_QUERY_LIMIT means?

4. **jinossy** (2015-07-16): @jihoonson
   Thank you for your review.
   PARALLEL_QUERY_LIMIT is maximum running QM. I will change to MAXIMUM_RUNNING_QM_RATE
   I'm going to update the patch that reflect your comments

5. **jinossy** (2015-07-16): @jihoonson
   I've update the patch but startQueryJob is not condition method

6. **jinossy** (2015-07-17): I've rename **Worker** to **NodeStatus**

7. **jinossy** (2015-07-18): @hyunsik
   Thank you for your review
   I've update the patch that reflects your comments

8. **jihoonson** (2015-07-19): The latest patch looks good to me.

9. **hyunsik** (2015-07-20): LGTM. Here is my +1.

10. **jihoonson** (2015-07-20): +1 ship it!

Chapter 2

Connected issue TAJO-1146

2.1 Summary

TajoWorkerResources are occasionally not released after the query completion relating to semaphore condition.

2.2 Description

When executing a bunch of (about >100) queries sequentially on my own JDBC batch program, Tajo frequently hangs its query allocation with notification of 'No available worker resource', after successful completion of the recent query. For instance,
{noformat:title = <Master Node>}
2014-10-28 15:57:50,481 INFO org.apache.tajo.master.rm.TajoWorkerResourceManager: Release Resource: 0.5,512
...
2014-10-28 15:57:50,489 INFO org.apache.tajo.master.rm.TajoWorkerResourceManager: Release Resource: 0.5,512
2014-10-28 15:57:50,511 INFO org.apache.tajo.catalog.CatalogServer: relation "lw_nps_2013.table201_c23_relation_hpi
is added to the catalog (192.168.0.70:26005)
2014-10-28 15:57:50,512 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: allocateWorkerResources:q_1414475997452_0013, requiredMemory:512~512, requiredContainers:32, requiredDiskSlots:0.5~0.5, queryMasterRequest=false, liveWorkers=8
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.querymaster.QueryInProgress: Received QueryMaster heartbeat:q_1414475997452_0013,state=QUERY_SUCCEEDED,progress=1.0, queryMaster=2.linewalks.local
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.querymaster.QueryInProgress:==========
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.querymaster.QueryInProgress: Stop query:q_1414475997452_0013
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.rm.TajoWorkerResourceManager: Release Resource: 0.0,512
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.rm.TajoWorkerResourceManager: Released QueryMaster (q_1414475997452_0013) resource.
2014-10-28 15:57:50,512 INFO org.apache.tajo.master.querymaster.QueryInProgress: q_1414475997452_0013 QueryMaster stopped
2014-10-28 15:57:50,513 WARN org.apache.tajo.master.TajoAsyncDispatcher: Interrupted Exception while stopping
2014-10-28 15:57:50,513 INFO org.apache.tajo.master.querymaster.QueryJobManager: Stop QueryInProgress:q_1414475997452_0013
2014-10-28 15:57:50,513 INFO org.apache.tajo.master.TajoAsyncDispatcher: AsyncDispatcher stopped:QueryInProgress:
2014-10-28 15:57:50,829 INFO org.apache.tajo.master.GlobalEngine: Query: create table table201_c23_relation_valid_h

as ...
2014-10-28 15:57:50,886 INFO org.apache.tajo.master.GlobalEngine: Non Optimized Query: ...
2014-10-28 15:57:50,888 INFO org.apache.tajo.master.GlobalEngine: ==================================
2014-10-28 15:57:50,888 INFO org.apache.tajo.master.GlobalEngine: Query is forwarded to :0
2014-10-28 15:57:50,888 INFO org.apache.tajo.master.TajoAsyncDispatcher: AsyncDispatcher started:QueryInProgress:q
2014-10-28 15:57:50,889 INFO org.apache.tajo.master.querymaster.QueryInProgress: Initializing QueryInProgress for QueryID=q_1414475997452_0014
2014-10-28 15:57:50,889 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: allocateWorkerResources:q_1414475997452_0014, requiredMemory:512~512, requiredContainers:1, requiredDiskSlots:0.0~0.0, queryMasterRequest=false, liveWorkers=8
2014-10-28 15:57:50,889 INFO org.apache.tajo.master.querymaster.QueryInProgress: Connect to QueryMaster:5.linewalks.local/192.168.0.75:28093
2014-10-28 15:57:50,890 INFO org.apache.tajo.master.querymaster.QueryInProgress: Call executeQuery to :5.linewalks.local:28093,q_1414475997452_0014
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: allocateWorkerResources:q_1414475997452_0014, requiredMemory:512~512, requiredContainers:10, requiredDiskSlots:0.5~0.5, queryMasterRequest=false, liveWorkers=8
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: ===================
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: Available Workers
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,589 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,590 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: ===================
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: allocateWorkerResources:q_1414475997452_0014, requiredMemory:512~512, requiredContainers:10, requiredDiskSlots:0.5~0.5, queryMasterRequest=false, liveWorkers=8
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: ===================
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: Available Workers
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
2014-10-28 15:57:51,690 DEBUG org.apache.tajo.master.rm.TajoWorkerResourceManager: org.apache.tajo.master.rm.Wo
... (continuously same log repetition)

```
{noformat:title = <Worker Node 5>}
014-10-28 15:57:22,048 INFO org.apache.tajo.master.querymaster.QueryMasterManagerService
Receive executeQuery request:q_1414475997452_0014
2014-10-28 15:57:22,050 INFO org.apache.tajo.master.querymaster.QueryMaster:
Start QueryStartEventHandler:q_1414475997452_0014
2014-10-28 15:57:22,091 INFO org.apache.tajo.master.querymaster.QueryMasterTask:
The staging dir 'hdfs://0.linewalks.local:9000/tmp/tajo-hadoop/staging/q_1414475997452_0
is created.
2014-10-28 15:57:22,398 INFO org.apache.tajo.engine.planner.global.GlobalPlanner:
...
```

```
2014-10-28 15:57:22,412 INFO org.apache.tajo.master.TajoAsyncDispatcher:
AsyncDispatcher started:q_1414475997452_0014
2014-10-28 15:57:22,412 INFO org.apache.tajo.master.querymaster.Query: Processing
q_1414475997452_0014 of type START
2014-10-28 15:57:22,417 INFO org.apache.tajo.master.querymaster.Query: q_1414475997452_(
Query Transitioned from QUERY_NEW to QUERY_RUNNING
2014-10-28 15:57:22,421 INFO org.apache.tajo.master.querymaster.SubQuery:
org.apache.tajo.master.DefaultTaskScheduler is chosen for the task scheduling
for eb_1414475997452_0014_000003
2014-10-28 15:57:22,434 INFO org.apache.tajo.storage.StorageManager: Total
input paths to process : 10
2014-10-28 15:57:22,498 INFO org.apache.tajo.storage.StorageManager: # of
splits with volumeId 10
2014-10-28 15:57:22,498 INFO org.apache.tajo.storage.StorageManager: Total
# of splits: 10
2014-10-28 15:57:22,504 INFO org.apache.tajo.storage.StorageManager: Total
input paths to process : 48
2014-10-28 15:57:22,575 INFO org.apache.tajo.storage.StorageManager: # of
splits with volumeId 48
2014-10-28 15:57:22,575 INFO org.apache.tajo.storage.StorageManager: Total
# of splits: 48
2014-10-28 15:57:22,576 INFO org.apache.tajo.master.querymaster.Repartitioner:
[Distributed Join Strategy] : Broadcast Join, base_table=lw_nps_2013.table201_individual
base_volume=976581946
2014-10-28 15:57:22,580 INFO org.apache.tajo.storage.StorageManager: Total
input paths to process : 10
2014-10-28 15:57:22,600 INFO org.apache.tajo.storage.StorageManager: # of
splits with volumeId 10
2014-10-28 15:57:22,600 INFO org.apache.tajo.storage.StorageManager: Total
# of splits: 10
2014-10-28 15:57:22,608 INFO org.apache.tajo.storage.StorageManager: Total
input paths to process : 48
2014-10-28 15:57:22,689 INFO org.apache.tajo.storage.StorageManager: # of
splits with volumeId 48
2014-10-28 15:57:22,689 INFO org.apache.tajo.storage.StorageManager: Total
# of splits: 48
2014-10-28 15:57:22,707 INFO org.apache.hadoop.yarn.util.RackResolver: Resolved
2.linewalks.local to /linewalks/rack1
2014-10-28 15:57:22,712 INFO org.apache.hadoop.yarn.util.RackResolver: Resolved
3.linewalks.local to /linewalks/rack2
2014-10-28 15:57:22,715 INFO org.apache.hadoop.yarn.util.RackResolver: Resolved
0.linewalks.local to /linewalks/rack1
2014-10-28 15:57:22,718 INFO org.apache.hadoop.yarn.util.RackResolver: Resolved
1.linewalks.local to /linewalks/rack1
2014-10-28 15:57:22,718 INFO org.apache.tajo.master.querymaster.SubQuery:
10 objects are scheduled
2014-10-28 15:57:22,718 INFO org.apache.tajo.master.DefaultTaskScheduler:
Start TaskScheduler
2014-10-28 15:57:22,718 INFO org.apache.tajo.worker.TajoResourceAllocator:
CalculateNumberRequestContainer - Number of Tasks=10, Number of Cluster
Slots=127
2014-10-28 15:57:22,723 INFO org.apache.hadoop.yarn.util.RackResolver: Resolved
4.linewalks.local to /linewalks/rack2
2014-10-28 15:57:22,723 INFO org.apache.tajo.master.querymaster.SubQuery:
```

```
Request Container for eb_1414475997452_0014_000003 containers=10
2014-10-28 15:57:22,727 INFO org.apache.tajo.worker.TajoResourceAllocator:
Start TajoWorkerAllocationThread
2014-10-28 15:57:25,734 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:28,735 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:31,735 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:34,736 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:37,736 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:40,736 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:43,737 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
2014-10-28 15:57:46,737 INFO org.apache.tajo.worker.TajoResourceAllocator:
No available worker resource for eb_1414475997452_0014_000003
....
```

After investigating the symptom in DEBUG mode, I concluded that TajoWorkerResourceManager could sometimes not release semaphore for resourceRequest callback due to if-else condition, especially here:

{noformat:title=tajo-core/.../master/rm/TajoWorkerResourceManager.java: 298}
// TajoWorkerResourceManager can't return allocated disk slots occasionally.
// Because the rest resource request can remains after QueryMaster stops.
// Thus we need to find whether QueryId stopped or not.
if (!rmContext.getStoppedQueryIds().contains(resourceRequest.queryId)) {
List<AllocatedWorkerResource> allocatedWorkerResources = chooseWorkers(resourceRequest);

if(allocatedWorkerResources.size() > 0) {
List<WorkerAllocatedResource> allocatedResources =
new ArrayList<WorkerAllocatedResource>();

....

//// Semaphore is currently released here!
resourceRequest.callBack.run(WorkerResourceAllocationResponse.newBuilder()
.setQueryId(resourceRequest.request.getQueryId())
.addAllWorkerAllocatedResource(allocatedResources)
.build()
);

} else {
if(LOG.isDebugEnabled()) {
LOG.debug("==");
LOG.debug("Available Workers");
...
{noformat}

Is there anybody knowing solution to avoid the race condition? This problem have existed since

0.8.x, where I started to use Tajo. :(Thanks a lot for your support in advance!

2.3 Attachments

No attachments

2.4 Commits

No related commits

2.5 Comments

1. **hyunsik:** Thank you for your bug report. I'll dig into this problem.
2. **jhkim:** solved by TAJO-1397

2.6 Pull requests

No pull requests

Chapter 3

Connected issue TAJO-1399

3.1 Summary

TajoResourceAllocator might hang on network error

3.2 Description

```
1  CallFuture<WorkerResourceAllocationResponse> callBack = new CallFuture<
       WorkerResourceAllocationResponse>();
2
3  ...
4
5  RpcConnectionPool connPool = RpcConnectionPool.getPool();
6  NettyClientBase tmClient = null;
7  try {
8    ServiceTracker serviceTracker = queryTaskContext.getQueryMasterContext().
         getWorkerContext().getServiceTracker();
9    tmClient = connPool.getConnection(serviceTracker.getU
10 mbilicalAddress(), QueryCoordinatorProtocol.class, true);
11   QueryCoordinatorProtocolService masterClientService = tmClient.getStub();
12   masterClientService.allocateWorkerResources(null, request, callBack);
13 } catch (Throwable e) {
14   LOG.error(e.getMessage(), e);
15 } finally {
16   connPool.releaseConnection(tmClient);
17 }
18
19 WorkerResourceAllocationResponse response = null;
20 while(!stopped.get()) {
21   try {
22
23    response = callBack.get(3, TimeUnit.SECONDS);
24     ...
```

If "callBack" is not registered properly in netty by failed connection, etc., allocator thread would block on empty future forever, possibly making thread leakage.

3.3 Attachments

No attachments

3.4 Commits

No related commits

3.5 Comments

1. **jhkim:** Solved by TAJO-1563, TAJO-1584

3.6 Pull requests

3.6.1 Pull request 420

Title: TAJO-1399 TajoResourceAllocator might hang on network error
Author: navis
Date: 2015-03-13
Status: closed
Comments:

1. **dongjoon-hyun** (2015-03-13): +1

2. **navis** (2015-03-17): I've misunderstood some part of codes and fixed that. And also amended resource leakage on timeout of allocating query master resource.

3. **navis** (2015-03-17): Confirmed all test passing in local environment.

4. **jinossy** (2015-04-28): This issue was solve by TAJO-1563
Could you test hangs on network error?

5. **jinossy** (2015-05-07): Solved by TAJO-1563, TAJO-1584
please close this PR.

Chapter 4

Connected issue TAJO-1622

4.1 Summary

UniformRangePartition occasionally causes IllegalStateException

4.2 Description

```
2015-05-26 20:15:47,544 INFO org.apache.tajo.querymaster.Repartitioner:
eb_1432630858987_0005_000006, Try to divide [(12ìŢij,0), (íđĹíŔňíĄňëİijíĚŇìŁd',íİňëğİ,1
into 10 sub ranges (total units: 10)
2015-05-26 20:15:47,551 ERROR org.apache.tajo.querymaster.Stage: Stage (eb_143263085898
ERROR:
java.lang.IllegalStateException
at com.google.common.base.Preconditions.checkState(Preconditions.java:129)
at org.apache.tajo.engine.planner.UniformRangePartition.isOverflow(UniformRangePartitio
at org.apache.tajo.engine.planner.UniformRangePartition.increment(UniformRangePartition
at org.apache.tajo.engine.planner.UniformRangePartition.partition(UniformRangePartition
at org.apache.tajo.querymaster.Repartitioner.scheduleRangeShuffledFetches(Repartitioner
at org.apache.tajo.querymaster.Repartitioner.scheduleFragmentsForNonLeafTasks(Repartiti
at org.apache.tajo.querymaster.Stage$InitAndRequestContainer.schedule(Stage.java:998)
at org.apache.tajo.querymaster.Stage$InitAndRequestContainer.access$900(Stage.java:776)
at org.apache.tajo.querymaster.Stage$InitAndRequestContainer$1.run(Stage.java:802)
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
at java.util.concurrent.FutureTask.run(FutureTask.java:262)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
at java.lang.Thread.run(Thread.java:745)
```

4.3 Attachments

No attachments

4.4 Commits

No related commits

4.5 Comments

1. **jhkim:** This issue caused by race condition in StringUtils.isPureAscii. CharsetEncoder is not thread-safe.
   I was fix this in TAJO-1397

4.6 Pull requests

No pull requests

Chapter 5

Connected issue TAJO-1536

5.1 Summary

Fix minor issues in QueryMaster

5.2 Description

1. queryMasterTasks/finishedQueryMasterTasks are not locked properly
2. heartbeat does not releases connection

5.3 Attachments

No attachments

5.4 Commits

No related commits

5.5 Comments

1. **jhkim:** contained by TAJO-1397

5.6 Pull requests

5.6.1 Pull request 519

Title: TAJO-1536 Fix minor issues in QueryMaster
Author: navis
Date: 2015-04-07
Status: closed
Comments: No comments

Chapter 6

Connected issue TAJO-1551

6.1 Summary

Reuse allocated resources in next execution block if possible

6.2 Description

Reuse resources acquired for previous execution block

6.3 Attachments

No attachments

6.4 Commits

No related commits

6.5 Comments

1. **jhkim:** resource allocation is changed by TAJO-1397

6.6 Pull requests

6.6.1 Pull request 534

Title: TAJO-1551 Reuse allocated resources in next execution block if possible
Author: navis
Date: 2015-04-14
Status: closed
Comments: No comments

Chapter 7

Connected issue TAJO-540

7.1 Summary

(Umbrella) Implement Tajo Query Scheduler

7.2 Description

Currently, there is no Tajo query scheduler. So, all queries launched simultaneously compete cluster resource which is managed by TajoResourceManager.

In this issue, we will investigate, design, and implement a Tajo query scheduler. This is an umbrella issue for that. We will create subtasks for them.

7.3 Attachments

No attachments

7.4 Commits

No related commits

7.5 Comments

1. **blrunner:** +1 for the idea.
2. **sirpkt:** +1 for the idea

   I think this is essential feature to support concurrent accesses of multiple users to the Tajo cluster.
3. **charsyam:** +1 for the idea.
   I think this is very cool feature.
4. **jihoonson:** +1 for this issue.
   It is mandatory for multi-query processing environments.
5. **coderplay:** As discussed with Hyunsik before, I suggest using sparrow. Sparrow scheduler is one of the best choices of this kind of low-latency schedulers.
   References:

http://www.cs.berkeley.edu/~matei/papers/2013/sosp_sparrow.pdf
https://github.com/radlab/sparrow

6. **coderplay:** Ok, I got time to write a more detailed plan for this ticket.

   Historically, the first scheduler exists in hadoop ecosystem is the JobTracker in mapreduce. JobTracker actually plays two roles of a mapreduce cluster, one is resource management and the other is job tasks scheduling. Because of JobTracker's playing those two roles, the job response time and scalability of JobTracker is not good. This kind of issue also came across the ancestor of mapreduce - Google, which later start a projected named Borg with one of the goal to address this problem. Borg become a cluster resource management scheduler in Google, and its current version name from their paper is Omega.(see https://medium.com/large-scale-data-processing/a7a81f278e6f)

   Later this kind of resource scheduler appears into our vision. That's Mesos and Hadoop Yarn. The different between this 2 is mesos support gang scheduling and yarn support incremental scheduling. Both of them divided cluster scheduling into 2 layers, the higher layer is resource management, which is the responsibility of those two. They control the resource for each application/framework/job. Meanwhile, the other role for job tasks scheduling of a JobTracker is put down into a lower layer - Each application/framework/job's master coordinates the tasks for one application/framework/job.

   From our benchmarking, a job with 10 sleep zero ms tasks in hadoop 1.0 costed about 20 seconds because of JobTracker's scheduling. And Hadoop Yarn take the same level time as well. What we need here is not a scheduler as MRAppMaster, it's a low-latency scheduler. From Jeff Dean's paper (http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/abstract), we get a knowledge that Google is always beyond us. They developed a so called tied request technology to solve the low-latency requirements. please see the tied request section in http://static.googleusercontent.com/media/research.google.com/en//people/jeff/MIT_BigData_S if you can't download the acm paper.

   What we need here is actually a google's tied request like scheduling. Fortunately, we have a good candidate, sparrow, which actually was the scheduler of the first version of Impala (c++ version), and will be plugged into spark.
   I'd like to port sparrow into Tajo, but before that I think we need to discuss something first , cuz the structure will be radically changed.

   to be continuted.

7. **jihoonson:** Thanks, Min.
   It's really interesting. I'll investigate, too.

8. **sirpkt:** +1 for the idea,
   and your comment is really useful, Min.
   I'll check the papers you suggested.

9. **coderplay:** For more information, I'd like to add some references

   Mesos:
   http://www.wired.com/wiredenterprise/2013/03/google-borg-twitter-mesos/2/
   http://www.cs.berkeley.edu/~matei/papers/2011/nsdi_mesos.pdf
   http://mesos.berkeley.edu/mesos_tech_report.pdf

Yarn:
https://issues.apache.org/jira/browse/MAPREDUCE-279
https://issues.apache.org/jira/secure/attachment/12486023/MapReduce_NextGen_Architecture.pdf

Sparrow on spark
https://github.com/kayousterhout/spark/tree/sparrow/core/src/main/scala/spark/scheduler/sparrow

10. **coderplay:** Continue my previous 2 comments. Sparrow improves "The power of two Choices" algorithm on 2 issues: 1) queued assignment can't accurately measure the real cost time of a task 2) the concurrent scheduling problem. You can check the sparrow paper for the details.

    As I mentioned, If we leverage a low-latency scheduler in an interactive or real-time system, we need radically change current design of tajo's scheduling.

    Firstly, the way we use Yarn is quite different from Spark and Impala. The resource requests are issued by Tajo workers, one container for one task/queryunit attempt. While spark and impala uses yarn as a higher layer scheduler for resource management. They use sparrow(-like) as their own internal scheduler in a lower layer for the purpose of low latency. Yarn is used for allocate the resources for a whole spark/impala cluster, not for a task. For example, if a spark cluster has 1 master and 10 slaves. The master need 10GB memory, and each of the slaves need 20GB memory. Yarn allocate a 10GB container for master daemon, and 20GB container for a slave daemon. Because those daemons are long-lived process, those resource are long time occupied by the spark cluster. Yarn revoke the resource only if one slave get decommissioned from the cluster. Here is my thought on tajo query scheduler, we can use yarn as higher layer resource management, yarn allocate cpu/memory resources to tajo master/querymaster/worker daemons. Sparrow-like scheduler coordinate query with those resources in a lower layer.

    Secondly, directly use sparrow is not proper. There are 3 reasons: 1) Sparrow need to start a scheduler daemon on each machine, is not convenient to operate 2) Sparrow support multitenancy, in another words, sparrow has user authentication, which tajo don't support yet. 3) sparrow can't kill a job currently. But the algorithm behind sparrow is quite suitable for tajo.

11. **hyunsik:** Min,
    Thank you for sharing informative materials, history of resource manager and scheduler, and your ideas.
    There are many design considerations. Now, I need more time to digest them. I'll leave some comments soon :)

12. **coderplay:** I'd like to write more about the code implementation.

    Currently, tajo has 2 threads for sheduling. One which is standalone mode schedules queries by tajo itself, the other is yarn mode where queries resources are allocated by a yarn cluster.

    Regarding to the standalone mode. When TajoMaster receives a query, it firstly use TajoWorkerResourceManager to choose a idler QueryMaster for this query. Then the query will be sent to that QueryMaster. QueryMaster breaks the query into several execution blocks, each execution block consists of several tasks. TajoResourceAllocator reside in that QueryMaster

send a rpc call TajoMasterProtocol.allocateWorkerResources() to TajoMaster. TajoMaster then use TajoWorkerResourceManager again to allocate workers for execution block. Yarn mode is quite similar.

What we want to do some change is for the standalone mode, right?

13. **jihoonson:** I also think so. Actually, in my opinion, the current implementation of yarn mode is not useful because its scheduling latency is too high. I think that we should adopt the way that Spark and Impala use yarn.

14. **coderplay:** Go ahead. From a deep investigation of mine, we can keep the yarn thread, only need some refactoring in order to keep the same interface as standalone mode scheduling.

    Currently, standalone mode scheduling is something like FIFO centralized scheduling, if the previous query occupies all of the slots of workers, the succeeding query will be blocked. We have 2 choices, the first one is change the FIFO strategy into another one, like fair share. But this hadoop jobtracker like scheduling can't achieve a very low latency and good scalalibity. The second one is porting sparrow into tajo.
    If we want to port sparrow, we need to do one thing in advance. That is , due to sparrow is a decentralized algorithm, typically every node has a scheduling service deployed. Those schedulers need to know every node's status. Actually, Impala has a Statestore daemon to offer this kind of service. see *The Impala Statestore* section at http://www.cloudera.com/content/cloudera-content/cloudera-docs/Impala/latest/Installing-and-Using-Impala/ciiu_concepts.html This is also called service discovery. Facebook presto has such component as well.
    see https://github.com/facebook/presto/blob/master/presto-main/src/main/java/com/facebook/presto/metadata

    For long term purpose, we need add a service discovery component not only for scheduling, but also for high availability. Fortunately, we needn't build a service discovery from scratch. There a lot of open source projects for this. One of the most famous is zookeeper.see http://www.javacodegeeks.com/2013/11/coordination-and-service-discovery-with-apache-zookeeper.html
    A better library built on the top of zookeeper https://github.com/Netflix/curator/wiki/Service-Discovery
    see http://blog.palominolabs.com/2012/08/14/using-netflix-curator-for-service-discovery/

    For short term. I think TajoMaster already hold the status of all workers. Each worker can fetch all workers' address through a rpc send to TajoMaster. If we have such information in the worker side, we can embed sparrow like scheduler as a optional service into worker.

15. **coderplay:** Here is the document describe how impala integrate with yarn. http://cloudera.github.io/llama/.

16. **coderplay:** [~jihoonson]

    Seems TajoWorkerResourceManager.WorkerResourceAllocationThread.chooseWorkers() try to solve the problem which kind of resource, memory or disk, is more critical to the container's requirement, right?
    Do you know how yarn solve it? Here is a paper which is used in yarn's all kind of schedulerhttp://www.cs.berkeley.edu/~matei/papers/2011/nsdi_drf.pdf

17. **jihoonson:** Min, really appreciate for sharing various articles and papers.
    For your questions, you are right. TajoWorkerResourceManager manages and schedules the worker resources.

    Actually, I don't have sufficient backgrounds for query scheduling. So, I'm still investigating for the deep understanding of the query scheduling. I'll read the DRF paper, too.

Many thanks,
Jihoon

18. **coderplay:** Here is a comment from [~mafish] on TAJO-611

    {quote}
    Hi There,
    I've done some basic investigation on the discovery service. Min give a very detailed discussion about the resource managements on the comment section of Tajo-540. That's very useful for me. Thanks Min. Now I have some questions to discuss.
    What's the current resource management mechanism in Tajo and what are related classes?
    Based on your previous discussion, it seems Tajo uses Yarn, but not at CPU/Memory level. Do we need a resource management with more granularity? It seem this question is more related to Tajo-540.
    {quote}

    Yarn support cpu/memory kind of resource management. Tajo right now can't accurately calculate the exact cpu/memory resource a task should acquire for, it instead ask for a rough number of those resource from Yarn. Nevertheless, at least tajo support resource management in both Yarn and Standalone modes, we can improve it after statistics get more affluent.

19. **jihoonson:** Hi, [~coderplay].

    Apologize for late reply. It took so long because I don't have much backgrounds for query scheduling.
    Actually, I'm still investigating more researches about query scheduling, but I've just read the Sparrow paper.
    I'm very impressed by Sparrow, and agree on that we should adopt it for low latency scheduling.
    Please go on this issue, and share the progress occasionally.

    Thanks,
    Jihoon

20. **coderplay:** Hi Jihoon,

    Sorry that havenot update for so long a time. I will reply soon.

    Min

21. **hyunsik:** I'm sorry too for commenting lazily. I'll comment soon.

22. **hyunsik:** Hi [~coderplay],

    Thank you for sharing nice articles. I'm very sorry for late response due to preparing the 0.8.0 release work and the work as an employee. Also, for one month, I've investigated the background of this area that you listed.

    I'd like to discuss some questions and suggestions that you threw. First of all, I agree with your suggestion for porting sparrow to Tajo. Now, we need the query scheduler to well support multiple users and multiple running queries. As you mentioned, sparrow is proper to these requirements while it is low latency.

    Also, you concerned with the way we use Yarn, and you propose two ways. One of them is that we use yarn as higher layer resource management and sparrow-like scheduler as a lower

layer scheduler. I'd like to throw +1 for this suggestion.

One of problems you concerned was that this approach will cause the radical change. Especially, there are two-types schedulers. The many dependent modules make this work very hard.

So, I suggest to comment out all TajoYarnResourceManager and the scheduler code related to Yarn. Then, we only focus on the sparrow-like scheduler for standalone scheduler. Later, we can recover the yarn scheduler. I think that this way makes our work more faster.

What do you think about my proposal? After we get some agreement, we can discuss several stages for this work. I'm looking forward to your response.

23. **hsaputra:** If not one object, I would like to add subtask for integration with YARN as resource manager for this ticket.

24. **hyunsik:** Hi [~hsaputra],

    Sounds nice. I think that it would be nice to create another top-level Jira issue for Yarn integration. In my opinion, the integration is related to this issue, but it would not be the subtasks of this issue.

    Regards,
    Hyunsik

25. **hsaputra:** Ah ok, thx Hyunsik, will do

26. **coderplay:** [~hyunsik]

    Sure. Since no one use that kind of yarn mode, commenting out is a agile way. In fact, I was lost in those yarn related code.
    Nice!

    Min

27. **hyunsik:** Hi Min,

    After TAJO-752, I'd like to comment out Yarn related code in a subtask.

    In addition to TAJO-603, I'd like to actively participate in this work. Also, I propose making another branch for this work and its subtasks. In the branch, we can freely do this work and we would not concern with the broken or completeness of features while we doing it.

    It is just a question. Will you reuse some of the exiting Sparrow source code or implement it in the from-the-scratch manner?

    Also, I think that this issue may include as follows:
    * Refactoring Worker containers
    ** Implementing Task Queues/node monitor in Worker
    * Refactoring RPC APIs among QueryMaster, TajoMaster, and Workers
    * Implementing Sparrow-like scheduler in QueryMaster

    Especially, Worker and RPC APIs are deeply related to other parts, and their codes are somewhat messy. If you are Ok, I'd like to firstly start the refactoring of worker and RPC APIs in order to help you easily do this work. What do you think about that?

Best regards,
Hyunsik

28. **coderplay:** I think implement it from the scratch is more reasonable because tajo has its own specificity. For example resource aware scheduling, disk volume id aware, etc.
That's would be great to me you will do this, thank you very much! Really appreciate it!.

    Best regards,
    Min

29. **hyunsik:** I got your point. Let's go head. I'll start the removal of yarn part. Thanks!

    Best regards,
    Hyunsik

30. **jihoonson:** Sorry.
The assignment was my mistake.

31. **jhkim:** Guys, Could you share the progress on this issue ? if not, I will start the TajoResourceManager
Because we need physical memory management

32. **hyunsik:** Hi [~jhkim],

    Probably, there is no progress. I already know that you made the fair scheduler and adopted the scheduler to some production clusters in some company. I think that we can continue this work from your work.

    But, it is very huge work. Before staring implementation, we need enough discussion and need to plan the details.

33. **jhkim:** Hi [~hyunsik]
Actually, I have no detailed plan yet.
I did implement trunk-based fair-scheduler and It does not scale out on big cluster
In my opinion, we must change the ResourceManager.

    Here is my simple plan:
    1. Add ResourceManager in TajoWorker (consider the physical memory and cpu)
    2. Add WorkerMonitor and TaskSchedulers in TajoWorker (sparrow like)
    3. Add TaskExecutor in TajoWorker
    4. Add TaskLoadBalancer in QueryMaster (probe the workers and throttling the task)
    5. Add FrontendQueryScheduler in TajoMaster

    I will attach the sequence diagram.If you have any idea, please feel free to give your opinion.

34. **jihoonson:** [~jhkim], thanks for sharing your plan.
However, IMHO, we need to start from the drawing a big picture of this issue.
The big picture may contain your goal, challenges, problem solving approaches, etc.
Do you plan to implement a Sparrow-like query scheduler?

    In addition, it would be great if you share some materials of the background as [~coderplay] did.

Thanks,
Jihoon

35. **jhkim:** [~jihoonson]
What is mean the Big picture ? You means that âĂŸfault toleranceâĂŹ , âĂŸspeculative executionâĂŹ âĂę ?I was brief my simple plan. I think we should discuss the skeleton for scheduler in TAJO-540

    IâĂŹve just refer to sparrow paper and source code. Because @coderplay was describe materials enoughhttp://www.cs.berkeley.edu/~matei/papers/2013/sosp_sparrow.pdf
    https://github.com/radlab/sparrow

36. **jihoonson:** [~jhkim], thanks for your reply.
I meant, we need to talk about implementation goals, problems which can be introduced in the implementation, and solutions of those problems.
If you want to implement a Sparrow-like query scheduler, the implementation goals will be low-latency and load balancing. (It would be much greater if the SLA scheme is supported.)Since you take over this issue from [~coderplay] after quite a long time has passed, I wondered your intention.

    Anyway, you seem to implement a Sparrow-like query scheduler.
    Is it a just porting of Sparrow to Tajo?
    Even though it is, it would be great for us to understand your implementation if you share your problem solving approach at a logical level, instead of implementation level.

37. **jihoonson:** I had a discussion with [~jhkim] in offline.I misunderstood his simple plan which is described above as his implementation plan, but it was not.
So, I'll wait for his sequence diagram that describes the overall behavior of his suggestion.
Thanks for valuable discussion, [~jhkim].

38. **hyunsik:** [~jhkim],

    Sounds nice. I'm looking forward to seeing your sequence diagram. We can start the discussion from your initial work.
    The main goal of is to support multi-tenancy and capacity scheduling for queries. I missed to mention it in the description. I'll rewrite the issue description.

39. **jhkim:** Thanks [~jihoonson], [~hyunsik]
Actually, this issue is huge. So I focus on task scheduling in this time.

40. **hyunsik:** Task scheduling and query scheduling mostly are related to each other. Don't worry about it. I can help this work, and I'd like participate in this work. Let's do it together.

41. **jhkim:** Hello guys

    Recently IâĂŹm looking a solution for capacity scheduler on WorkerNode. But it is very difficult and many challenges to me.Jihoon suggested omega scheduler to me. It very interesting and will be help to implement our scheduler
    http://research.google.com/pubs/pub41684.html

    As first, I suggest to borrow yarn scheduler queue into Tajo. It will be easy to add the multi-tenant query scheduler
    What do you think about to borrow the yarn scheduler queue for initial implementation?

42. **jihoonson:** [~jhkim]

It looks good to start with providing a multi-tenant query scheduler by borrowing the Yarn scheduler.
However, as you know, the Yarn scheduler has some problems such as high scheduling latency.
Do you have any good idea?

43. **jhkim:** [~jihoonson]

    I think that refactoring the current resource manager is a good approach. This model shows reasonable low latency (< 100ms).But, we must fix the leak of resource problem we've found.

44. **hyunsik:** Basically, our query scheduler is different from yarn's objective. In my opinion, we just need to refer to the multi-queue fair scheduler policy.

45. **jihoonson:** I got you point.Please go ahead.

46. **jhkim:** Thank you for the comments
    [~hyunsik] You're right. IâĂŹve just to refer to the scheduler policy
    Firstly, I will do start simulation about resource manager with scheduler policy. (fifo and fair)

47. **hyunsik:** In general, query (or job) scheduler aims at the maximum resource utilization. For multi-tenancy, we also need to consider the fairness for multiple users (or queries). BTW, the maximum resource utilization and fairness are usually conflict to each other in many cases. To mitigate this problem, many scheduler seems to use preemption approach.

    In this point, our resource and scheduler system has the following problems:
    * A query exclusively uses allocated resources at the first time until the query is completed or failed.
    * There is no mechanism to deallocate resources during query processing.
    * Preempt is also not allowed.

    To achieve the multi tenancy, we should change our resource circulation. Especially, resource allocation must be fine grained instead of per query.

    So, I'll create a jira issue to change the resource circulation. We have to do this issue firstly in my opinion. If we achieve this, implementing multi-tenant scheduler would be much easier than now. It would be a good starting point of this issue.

48. **hyunsik:** I created the above issue at TAJO-1397.

49. **hyunsik:** I changed sparrow-like scheduler to 'WISH' rather than sub task because we do not decide to use it yet.

50. **jhkim:** [~hyunsik]
    This is a good approach for the multi tenancy.

7.6 Pull requests

No pull requests