Hadoop Yarn FifoScheduler Container 资源分配 WorkFlow

FifoScheduler

```
Public void handle(SchedulerEvent event) {
    …
    case NODE_UPDATE: {
        NodeUpdateSchedulerEvent nodeUpdatedEvent =
      (NodeUpdateSchedulerEvent)event;
        nodeUpdate(nodeUpdatedEvent.getRMNode());
    }
    break;
    …
}
private synchronized void nodeUpdate(RMNode rmNode) {
        ..
        // Processing the newly launched containers
        // process completed containers
        if (Resources.greaterThanOrEqual(resourceCalculator, clusterResource,
                node.getAvailableResource(),minimumAllocation)) {
                assignContainers(node);
        }
        metrics.setAvailableResourcesToQueue(
          Resources.subtract(clusterResource, usedResource));
}

private void assignContainers(FiCaSchedulerNode node) {
        for_each_app {
          synchronized(app) {
                for_each_priority {
                    // ensure the application needs containers of this priority
                    int assignedContainers =
                            assignContainersOnNode(node, application, priority);
                }
          }
        }
}

private int assignContainersOnNode(FiCaSchedulerNode node,
        FiCaSchedulerApp application, Priority priority) {
    // Data-local
    int nodeLocalContainers = assignNodeLocalContainers(node, application, priority);

    // Rack-local
    int rackLocalContainers = assignRackLocalContainers(node, application, priority);
```

```
        // Off-switch
        int offSwitchContainers = assignOffSwitchContainers(node, application, priority);
        return (nodeLocalContainers + rackLocalContainers + offSwitchContainers);
}

private int assignNodeLocalContainers(FiCaSchedulerNode node,
            FiCaSchedulerApp application, Priority priority) {
            …
            int assignableContainers =
              Math.min(
                    getMaxAllocatableContainers(application, priority, node,
                          NodeType.NODE_LOCAL),
                          request.getNumContainers());
            assignedContainers =
              assignContainer(node, application, priority,
                    assignableContainers, request, NodeType.NODE_LOCAL);
            return assignedContainers;
}

private int assignContainer(FiCaSchedulerNode node, FiCaSchedulerApp application,
            Priority priority, int assignableContainers,
            ResourceRequest request, NodeType type) {
            int availableContainers_mem =
            node.getAvailableResource().getMemory() / capability.getMemory();
            int avilableContainers_cpu =
              node.getAvailableResource().getVirtualCores() / capability.getVirtualCores();

            int availableContainers =
              Math.min(availableContainers_mem, avilableContainers_cpu);

        /*
         * @note GPU resource is not a must, so no modification about this upper bound
         */
            int assignedContainers = Math.min(assignableContainers, availableContainers);

            if (assignedContainers > 0) {
            for (int i=0; i < assignedContainers; ++i) {
              NodeId nodeId = node.getRMNode().getNodeID();
              ContainerId containerId = BuilderUtils.newContainerId(application
                    .getApplicationAttemptId(), application.getNewContainerId());
              Token containerToken = null;
              /*
                * if this node has gpu resource and this request requests gpu, then assign it to this
                * container, otherwise, no gpu
```

```
            */
            if ( capability.getGPUCores() > node.getAvailableResource().getGPUCores())
            {
                capability=Resource.newInstance(capability);
                capability.setGPUCores(0);    //set the require gpu as 0.
                capability.setGPUId(null);
            }

            /*
             * set the allocated gpu id, here we use only one gpu. 2013-12
             */
            else if(capability.getGPUCores()>0 && capability.getGPUCores()
                    <= node.getAvailableResource().getGPUCores())
            {
                capability=Resource.newInstance(capability);
                int allocate_gpu_id=node.getAvailableResource().getGPUId().get(0);
                List<Integer> gpuId = new ArrayList<Integer>();
                gpuId.add(allocate_gpu_id);
                capability.setGPUId(gpuId);
            }
            containerToken = this.rmContext.getContainerTokenSecretManager()
                    .createContainerToken(containerId, nodeId, application.getUser(),
                        capability);
            if (containerToken == null) {
               return i; // Try again later.
            }

            // Create the container
            Container container =
                BuilderUtils.newContainer(containerId, nodeId, node.getRMNode()
                    .getHttpAddress(), capability, priority, containerToken);

            // Inform the application
            RMContainer rmContainer =
                application.allocate(type, node, priority, request, container);

            // Inform the node
            node.allocateContainer(application.getApplicationId(), rmContainer);

            // Update usage for this container
            Resources.addTo(usedResource, capability);
        }
    }
```