GPU locality support for AI scheduling

Myeongjae Jeon, Qingcha Chen

Overview

Currently, there are efforts to add GPU support in Hadoop for AI job scheduling, one typical solution is YARN-3926. Those solution are general solution which treat GPU as countable resource, only calculate the number.
But in the Deep learning training, the GPU placement is also important for the job running efficiency. For example, a 2-GPU jobs run on gpu0-gpu1 is much fast than run on gpu0-gpu7.

This the patch add the GPU support to Hadoop 2.7.2. provides two type of GPU scheduling: GPU count scheduling is used for the scenario don't care the GPU locality. And GPU locality scheduling is used for the advance scenarios which take care of the GPU placement.

We implement it by adding a 64-bits bitmap into yarn Resource, this bitmap indicates the GPU usage information in a machine node. '1' means available and '0' means not available.

64-bits can represent a 64 GPUs node machine, this is enough in current and following a few years. We need change in the future if necessary.

Yarn GPU Interface

1. Add GPUs and GPUAttribute into yarn_protos as interface.

hadoop-yarn-project/hadoop-yarn/hadoop-yarn-api/src/main/proto/yarn_protos.proto

```
message ResourceProto {
                optional int32 memory = 1;
                optional int32 virtual_cores = 2;
                optional int32 GPUs = 3;
                optional int64 GPUAttribute = 4;
```

2. Interface to get/set the GPU and GPU attribute:

Currently, it only supports 64 GPU in a node, "long" is 64 bit in java, so the GPUAttribute is a long variable.

hadoop-yarn-project/hadoop-yarn/hadoop-yarn-api/src/main/java/org/apache/hadoop/yarn/api/records/Resource.java

```
1. public static Resource newInstance(int memory, int vCores, int GPUs, long GPUAttribute)
2. pubic abstract int getGPUs();
3. public abstract void setGPUs(int GPUs);
4. public abstract long getGPUAttribute();
5. public abstract void setGPUAttribute(long GPUAttribute);
```

3. Yarn config: yarn-default.xml

hadoop-yarn-project/hadoop-yarn/hadoop-yarn-common/src/main/resources/yarn-default.xml

below setting are help to config the yarn RM.

```
<property>
   <description>The minimum allocation for every container request at the RM,  in terms of GPUs. Requests lower than this will throw an InvalidResourceRequestException.
</description>
   <name>yarn.scheduler.minimum-allocation-gpus</name>
   <value>0</value>
 </property>
 <property>
   <description>The maximum allocation for every container request at the RM, in terms of GPUs. Requests higher than this will throw an InvalidResourceRequestException.
</description>
   <name>yarn.scheduler.maximum-allocation-gpus</name>
   <value>8</value>
 </property>
 <property>
<property>
   <description>Percentage of GPU that can be allocated  for containers. This setting allows users to limit the amount of  GPU that YARN containers use. Currently functional only  on Linux using cgroups. The default is to use 100% of GPU.
   </description>
   <name>yarn.nodemanager.resource.percentage-physical-gpu-limit</name>
   <value>100</value>
 </property>
 <property>
```

Yarn Client: GPU Resource Request

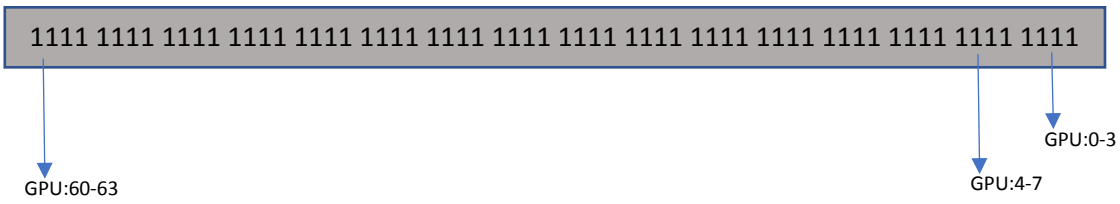
The GPU request information was sent to RM though the Resource object in `org.apache.hadoop.yarn.client.api.AMRMClient.ContainerRequest`, before call `org.apache.hadoop.yarn.client.api.YarnClientaddContainerRequest`

There are several GPU request scenarios:

1. Request GPU only by count:
   If use job only need the number of GPU, and don't care about the GPU placement. GPUAttribute must set to 0 in the request resource instance.

   ```
   Resource res = Resource.newInstance(requireMem, requiredCPU, requiredGPUs, 0)
   Or
   Res.setGPUAttribute((long)0)
   ```

2. Request GPU with locality(GPU attribute):
   The GPU locality information is stored in a 64 bits map(long), each bit represent a gpu. A 64 GPUs attribute with it's GPU mapping relation ship is:

   In GPU locality request, the request bit map is in Resource instance. For example, If we need 4 gpus, and the GPU locality is gpu0-3, we need create the request instance like:

   ```
   Resource res = Resource.newInstance(requireMem, requiredCPU, 4, 15)
   4 is the request GPU count.
   15 for "1111" is the GPU locality.
   ```

   If the GPU attribute is set to none-zero, The request GPU count must match with it. Otherwise the request will not success.

3. Request with Node/Label
   This is also supported in GPU, and the behavior is the same with the official Hadoop 2.7.2

4. Relax

   In GPU scheduling, the Relax is node level relax, Not GPU locality level relax.

   For example, If request Node1 with GPU count 2, GPU attribute 3 for gpu0-gpu1, once enable the Relax in container request.

If node1's gpu0 and gpu1 are available, yarn RM might relax to other node with gpu0-gpu1 available; yarn RM won't relax to other gpus in any node.

Resource manger

1. GPU allocation algorithm
   a. If request with GPU locality, the algorithm is just a simple "match" the request locality with Nodes' GPU attribute. If matched, RM will return a container with resource exact match the requirement, and subtract the allocated GPU.

   b. If request without GPU locality, the algorithm is only compare the request GPU, count with Nodes's free GPU count, if matched a node, RM will sequence assign free GPU in the node. And set the assigned GPU information into the assigned container resources.

   No matter the request with or without gpu locality, the RM will allocate and return container always with GPU attribute information in the Resource instance.

2. Scheduler

   CapacityScheduler, FairScheduler and FifoScheduler are all updated to support the the GPU attubute scheduling.

3. Resource Calculator

   the GPU count is considered in DominantResourceCalculator.

   Meantime, a new Calculator org.apache.hadoop.yarn.api.records.Resource.GPUResourceCalculator is created to only calculate resource by GPU.

Node Manager

Code change: org.apache.hadoop.yarn.util.LinuxResourceCalculatorPlugin

In node manger, the Node GPU hardware information is collected by running a nvidia-sim command when the Node Manger service start. In current version, only collect the GPU Capacity in the initialized.

Web apps

The GPU count and GPU attribute information are also displayed in the Hadoop web. User can check the overall capacity, used, free information in app information page. And use can also check each node's GPUs utilization information in bitmap format in Node Information page.