NodeManager生命周期介绍

ResourceManager 内维护了 NodeManager 的生命周期；对于每个 NodeManager 在 ResourceManager 中都有一个 RMNode 与其对应；除了 RMNode ，ResourceManager 中还定义了 NodeManager 的状态（states）以及触发状态转移的事件（event）。具体如下：

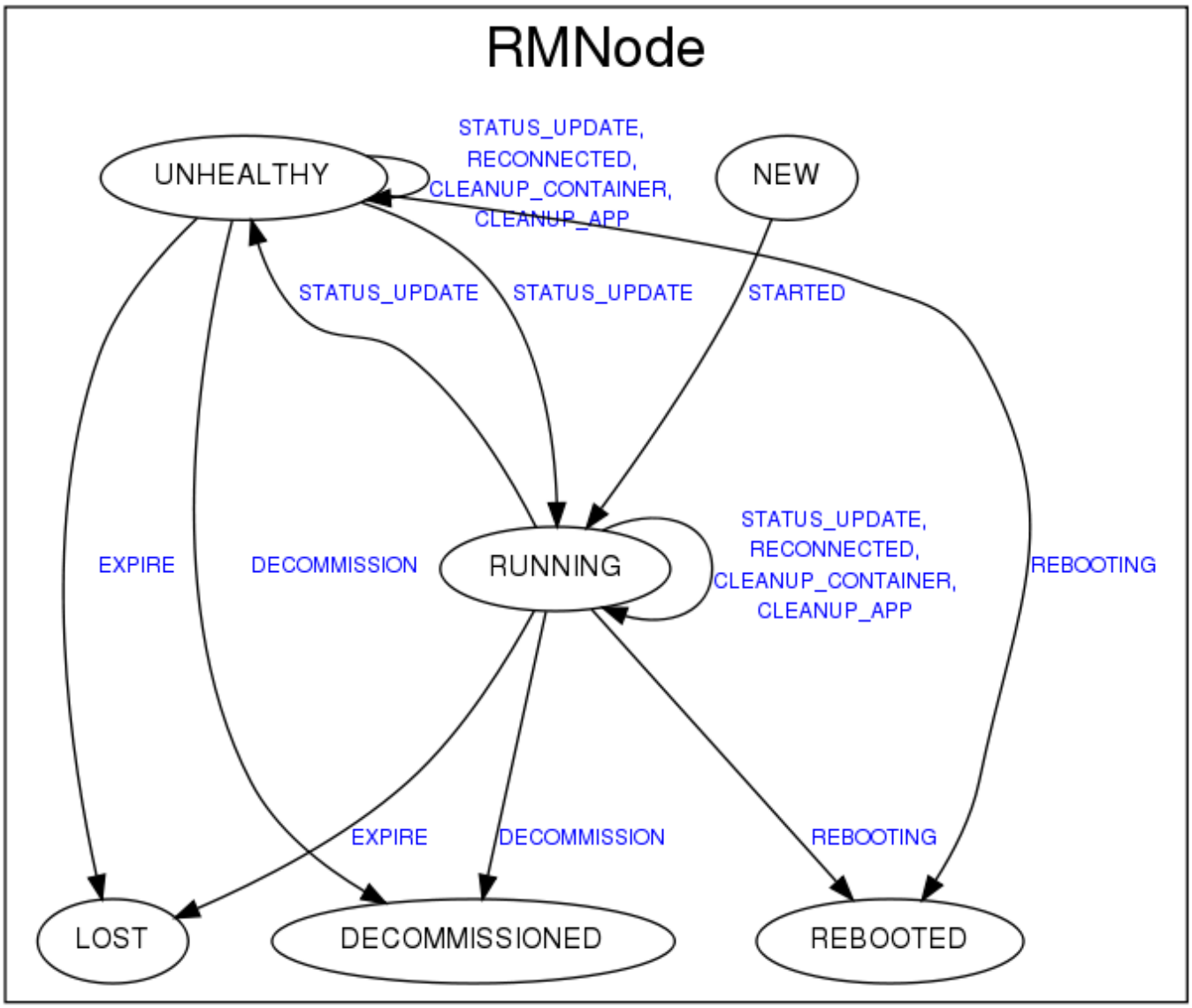
- org.apache.hadoop.yarn.server.resourcemanager.rmnode.RMNode：这是一个接口，每个 NodeManager 都与 RMNode 对应，这个接口主要维护 NodeManager 的可用资源（主要是内存及CPU）以及其他一些静态信息（比如NodeManager 的ID、hostname、Http端口、健康状况、机架名称等）；
- org.apache.hadoop.yarn.server.resourcemanager.rmnode.RMNodeImpl：这个类实现了 RMNode 接口，其记录了当前 NodeManager 中所有运行的 applications/containers；并且这里面还定义了 NodeManager 的状态转移以及其处理的类。
- org.apache.hadoop.yarn.server.resourcemanager.rmnode.RMNodeEventType：这是个枚举类，定义了 NodeManager 所有的事件类型。
- org.apache.hadoop.yarn.api.records.NodeState：这也是个枚举类，其定义了 NodeManager 所有可能的状态。

RMNodeImpl 中定义了 NodeManager 的状态转移情况，如下：

```
private static final StateMachineFactory<RMNodeImpl,
                                         NodeState,
                                         RMNodeEventType,
                                         RMNodeEvent> stateMachineFactory
                 = new StateMachineFactory<RMNodeImpl,
                                         NodeState,
                                         RMNodeEventType,
                                         RMNodeEvent>(NodeState.NEW)

  //Transitions from NEW state
  .addTransition(NodeState.NEW, NodeState.RUNNING,
      RMNodeEventType.STARTED, new AddNodeTransition())

  //Transitions from RUNNING state
  .addTransition(NodeState.RUNNING,
      EnumSet.of(NodeState.RUNNING, NodeState.UNHEALTHY),
      RMNodeEventType.STATUS_UPDATE, new StatusUpdateWhenHealthyTransition())
  .addTransition(NodeState.RUNNING, NodeState.DECOMMISSIONED,
      RMNodeEventType.DECOMMISSION,
      new DeactivateNodeTransition(NodeState.DECOMMISSIONED))
  .addTransition(NodeState.RUNNING, NodeState.LOST,
      RMNodeEventType.EXPIRE,
```

```
    new DeactivateNodeTransition(NodeState.LOST))
  .addTransition(NodeState.RUNNING, NodeState.REBOOTED,
    RMNodeEventType.REBOOTING,
    new DeactivateNodeTransition(NodeState.REBOOTED))
  .addTransition(NodeState.RUNNING, NodeState.RUNNING,
    RMNodeEventType.CLEANUP_APP, new CleanUpAppTransition())
  .addTransition(NodeState.RUNNING, NodeState.RUNNING,
    RMNodeEventType.CLEANUP_CONTAINER, new CleanUpContainerTransition())
  .addTransition(NodeState.RUNNING, NodeState.RUNNING,
    RMNodeEventType.RECONNECTED, new ReconnectNodeTransition())

  //Transitions from UNHEALTHY state
  .addTransition(NodeState.UNHEALTHY,
    EnumSet.of(NodeState.UNHEALTHY, NodeState.RUNNING),
    RMNodeEventType.STATUS_UPDATE, new StatusUpdateWhenUnHealthyTransition())
  .addTransition(NodeState.UNHEALTHY, NodeState.DECOMMISSIONED,
    RMNodeEventType.DECOMMISSION,
    new DeactivateNodeTransition(NodeState.DECOMMISSIONED))
  .addTransition(NodeState.UNHEALTHY, NodeState.LOST,
    RMNodeEventType.EXPIRE,
    new DeactivateNodeTransition(NodeState.LOST))
  .addTransition(NodeState.UNHEALTHY, NodeState.REBOOTED,
    RMNodeEventType.REBOOTING,
    new DeactivateNodeTransition(NodeState.REBOOTED))
  .addTransition(NodeState.UNHEALTHY, NodeState.UNHEALTHY,
    RMNodeEventType.RECONNECTED, new ReconnectNodeTransition())
  .addTransition(NodeState.UNHEALTHY, NodeState.UNHEALTHY,
    RMNodeEventType.CLEANUP_APP, new CleanUpAppTransition())
  .addTransition(NodeState.UNHEALTHY, NodeState.UNHEALTHY,
    RMNodeEventType.CLEANUP_CONTAINER, new CleanUpContainerTransition())

  // create the topology tables
  .installTopology();
```

上面定义了 NodeManager 每种状态之间的转移及其触发的事件。比如从 NodeState.NEW 状态转移到 NodeState.RUNNIN 的触发事件为 RMNodeEventType.STARTED，其状态转移处理类为 AddNodeTransition 。根据上面的定义我们可以画出 NodeManager 整个生命周期转换图：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

从上图可以看出，NodeManager 生命周期的开始状态和最终状态：

- 开始状态: NEW
- 最终状态: DECOMMISSION / REBOOTED / LOST

所有的 NodeManager 向 ResourceManager 注册的初始化状态都为 NodeState.NEW，一旦向 ResourceManager 注册成功其状态会更新为 NodeState.RUNNIN。还会处理以下两件事：

- 如果这个 NodeManager 之前在 ResourceManager 中的 inactive 节点列表里面，说明这个节点之前处于 LOST 、REBOOTED、DECOMMISSIONED 或 UNHEALTHY 的状态，但是现在这个节点正常了，所以我们需要把该 NodeManager 从 inactive 节点列表移除，并且更新集群的 Metrics 信息（增加 Active Node 的个数，减少处于 LOST 、REBOOTED、DECOMMISSIONED 或 UNHEALTHY

状态节点的个数）；
- 否则直接更新集群的 Metrics 信息，并增加 Active Node 的个数。

NodeManager 在启动之后会默认每隔 1s（由 yarn.resourcemanager.nodemanagers.heartbeat-interval-ms 参数配置）向 ResourceManager 发送心跳信息。在 ResourceManager 端会启动 PingChecker 线程默认每隔 200s（yarn.nm.liveness-monitor.expiry-interval-ms 参数值得三分之一）检测所有注册到 ResourceManager 的节点，一旦发现有节点超过 600s（由 yarn.nm.liveness-monitor.expiry-interval-ms 参数配置）没有发送心跳信息，则认为这个节点出问题了，这时候将会把这个节点从 running 列表中移除，并发送 RMNodeEventType.EXPIRE 事件。RMNodeImpl 接收到这个事件会通过 DeactivateNodeTransition 类来处理，这个类会将此节点从 ResourceManager 可用节点列表中移除，并加入到 inactiveNodes 列表里面，此时 NodeManager 状态由 NodeState.RUNNING 或 NodeState.UNHEALTHY 状态转移到 NodeState.LOST 状态，而且该节点被标记为 NodesListManagerEventType.NODE_UNUSABLE。所有在这个节点上运行的 containers 将会标记为失败，并且会分配到新的 NodeManager 上运行。

NodeManager 其他状态之间的转移及其触发的事件这里就不再详细阐述。关于 NodeManager 自身的状况监测机制，请点击这里。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）