APACHECON
NORTH AMERICA

Hadoop Applications on High Performance Computing

Devaraj Kavali

devaraj@apache.org

intel

About Me

- Apache Hadoop Committer
- Yarn/MapReduce Contributor
- Senior Software Engineer @Intel Corporation

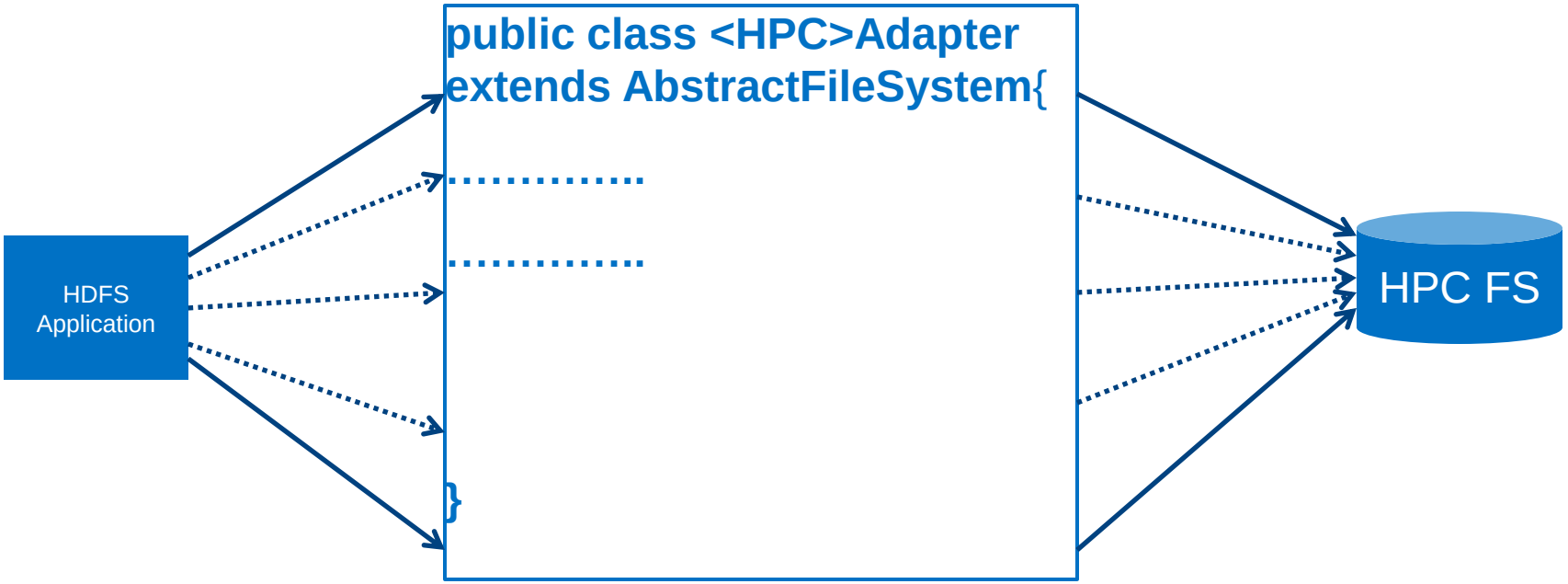
Agenda

- Objectives
- HDFS Applications with HPC File Systems
- Yarn Application
- Mapreduce Job
- HPC Schedulers
- Yarn Protocols
- Log Aggregation
- Shuffle Implementation
- Q&A

Objectives

- Use existing HPC Cluster for running Hadoop Applications
- Use any of the HPC File Systems like Lustre, PVFS, IBRIX Fusion, etc.
- Use any of the HPC schedulers like Slurm, Moab, PBS Pro, etc.
- Combine Hadoop workloads with HPC workloads
- No code changes to existing Hadoop(HDFS/YARN/MR) applications
- Minimal Hadoop configuration changes

HDFS Applications Using HPC File Systems

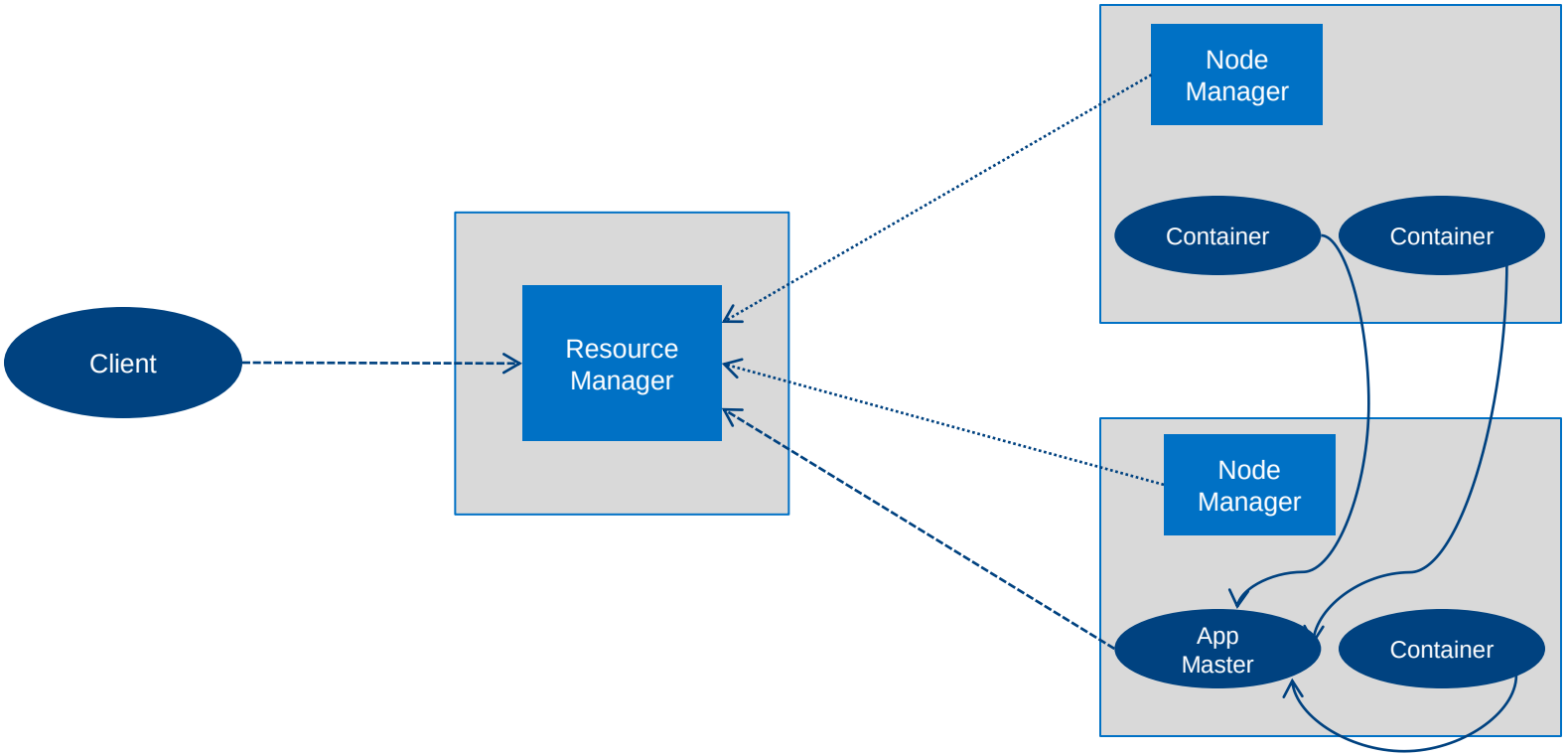


Hadoop Configurations for File System

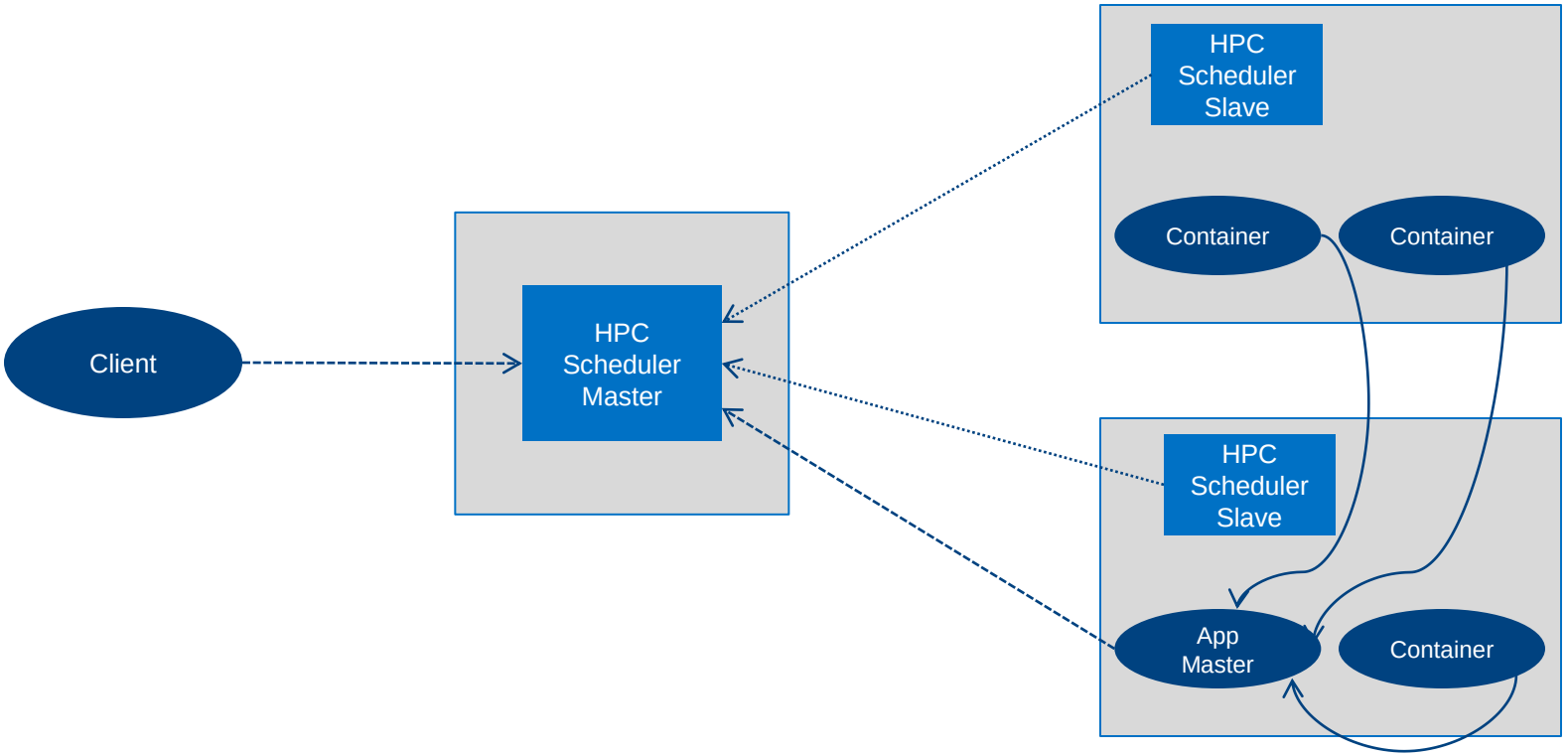
```
<property>
  <name>fs.defaultFS</name>
  <value>${hpc-uri}:///</value>
</property>
```

```
<property>
  <name>fs.AbstractFileSystem.${hpc-uri}.impl</name>
  <value>HPCFileSystemAdapter</value>
</property>
```

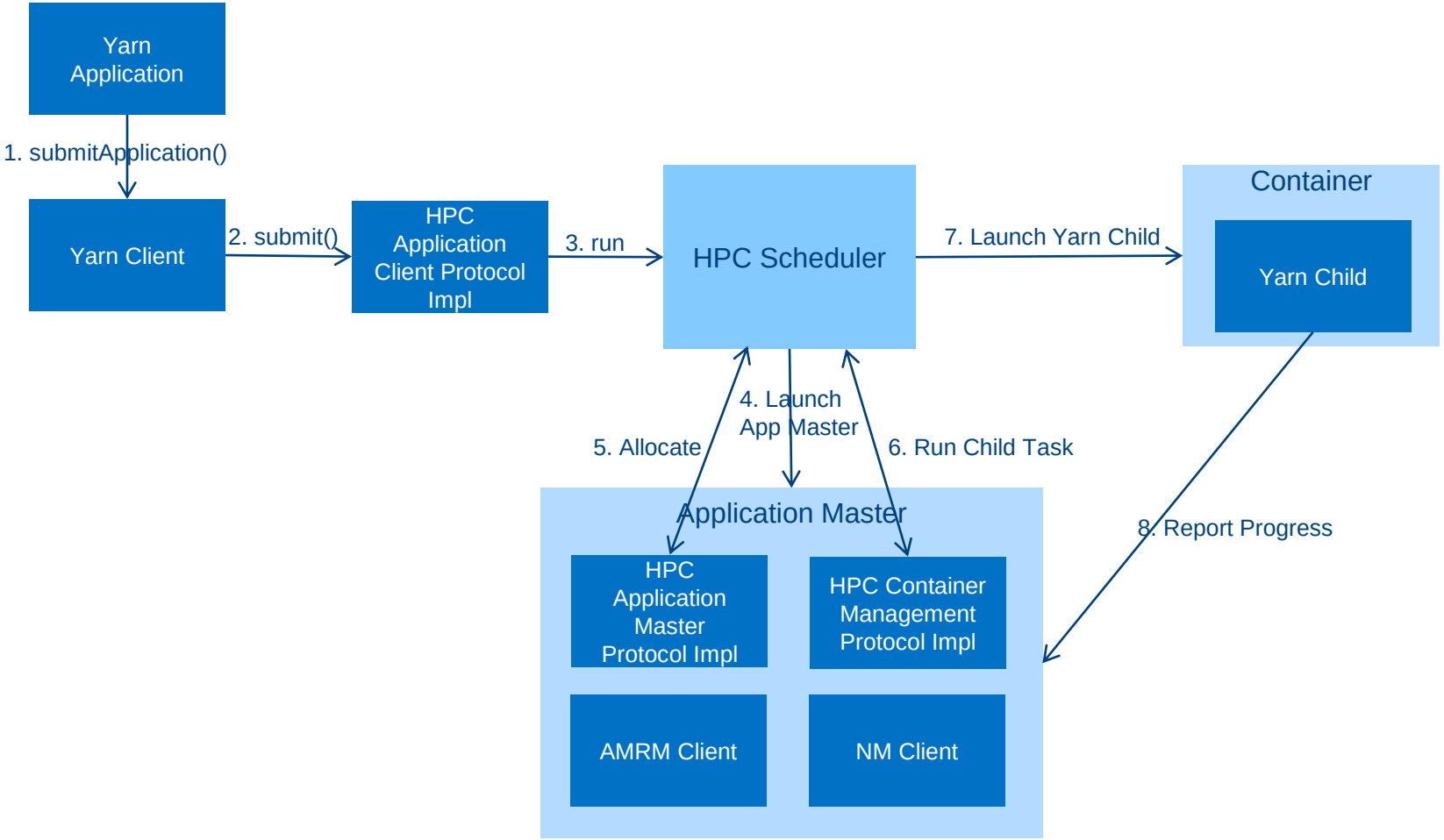
YARN Application



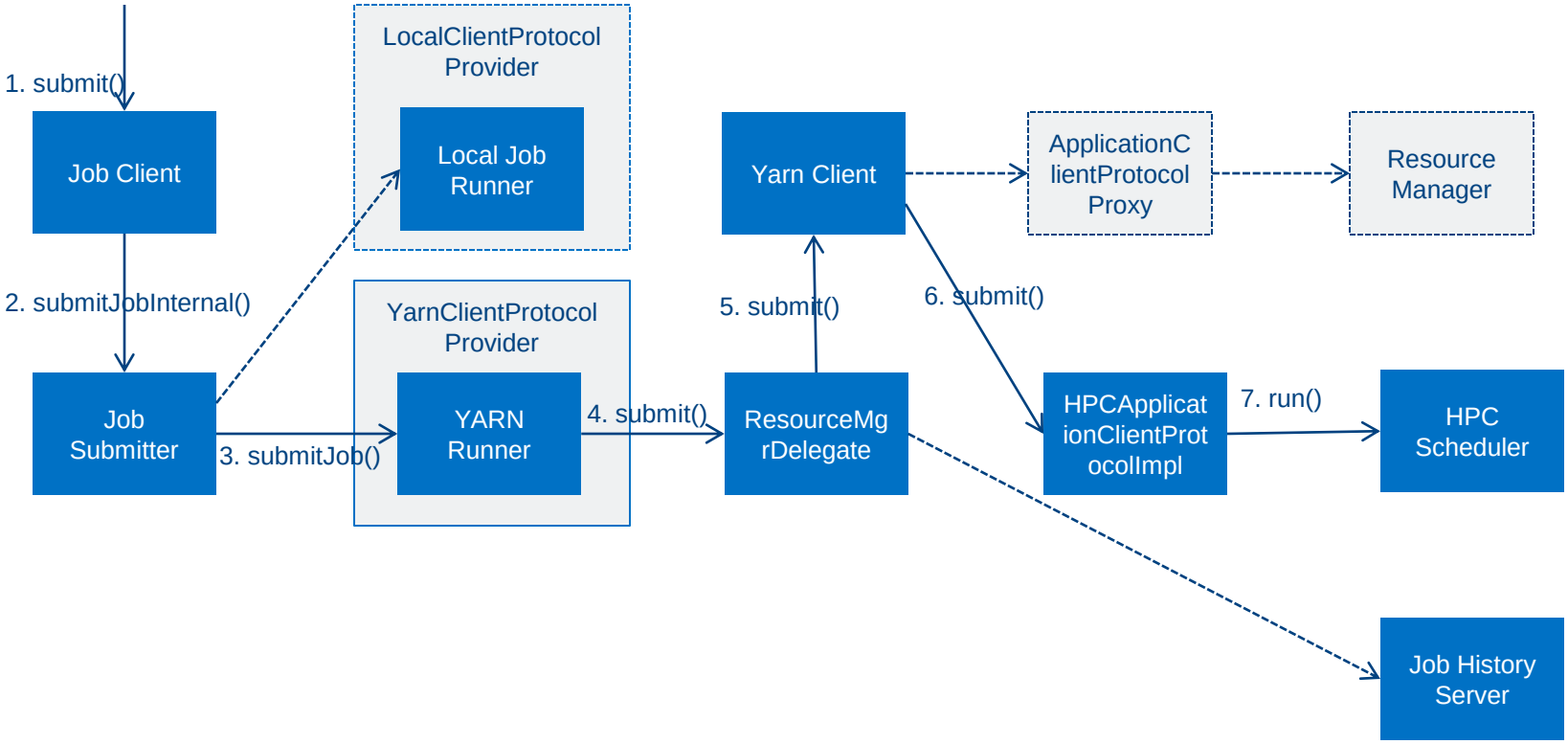
YARN Application with HPC Scheduler



Yarn Application Submission with HPC Scheduler



Mapreduce Job with HPC Scheduler



Yarn Protocols Configurations

RPC class Configuration

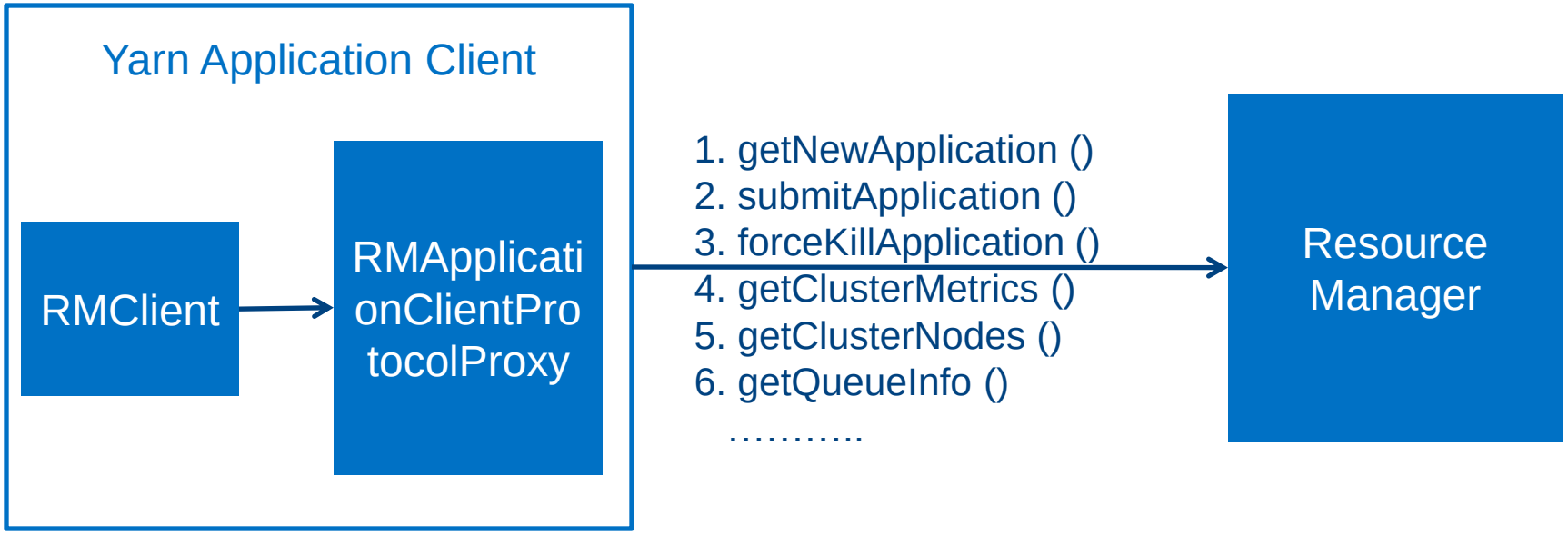
```
<property>
  <description>RPC class implementation</description>
  <name>yarn.ipc.rpc.class</name>
  <value>HadoopYarnHPCRPC</value>
 </property>
```

Yarn Protocols Configurations

```
public class HadoopYarnHPCRPC extends HadoopYarnProtoRPC {
 @Override
  public Object getProxy(Class protocol, InetSocketAddress address, Configuration conf) {
  Object proxy;
  if (protocol == ApplicationClientProtocol.class) {
    proxy = new HPCApplicationClientProtocolImpl(conf);
  } else if (protocol == ApplicationMasterProtocol.class) {
    proxy = new HPCApplicationMasterProtocolImpl(conf);
  } else if (protocol == ContainerManagementProtocol.class) {
    proxy = new HPCContainerManagementProtocolImpl(conf);
   } else {
    proxy = super.getProxy(protocol, address, conf);
  }
  return proxy;
}
```

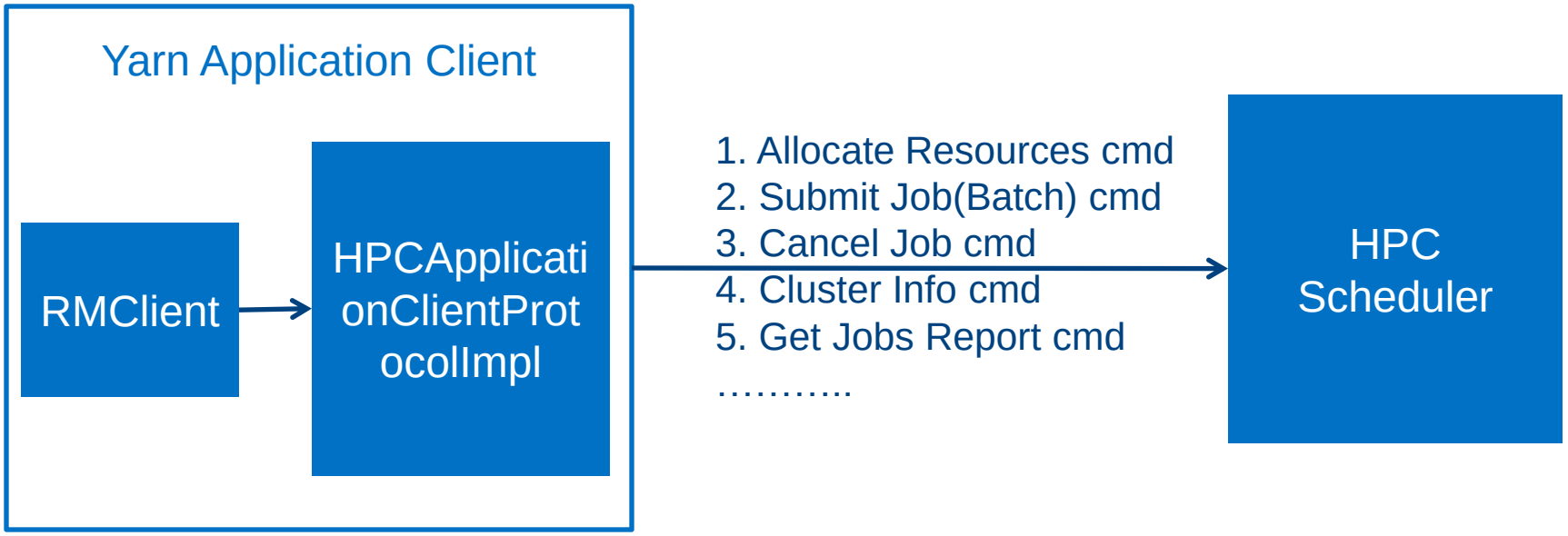
Application Client Protocol

Yarn Application Client Protocol Flow



Application Client Protocol

Yarn Application Client HPC Scheduler Flow



Application Client Protocol

API's for interaction

1. getNewApplication()

The interface used by clients to obtain a new ApplicationId for submitting new applications.

2. submitApplication()

The interface used by clients to submit a new application to the ResourceManager.

3. forceKillApplication()

The interface used by clients to request the ResourceManager to abort submitted application.

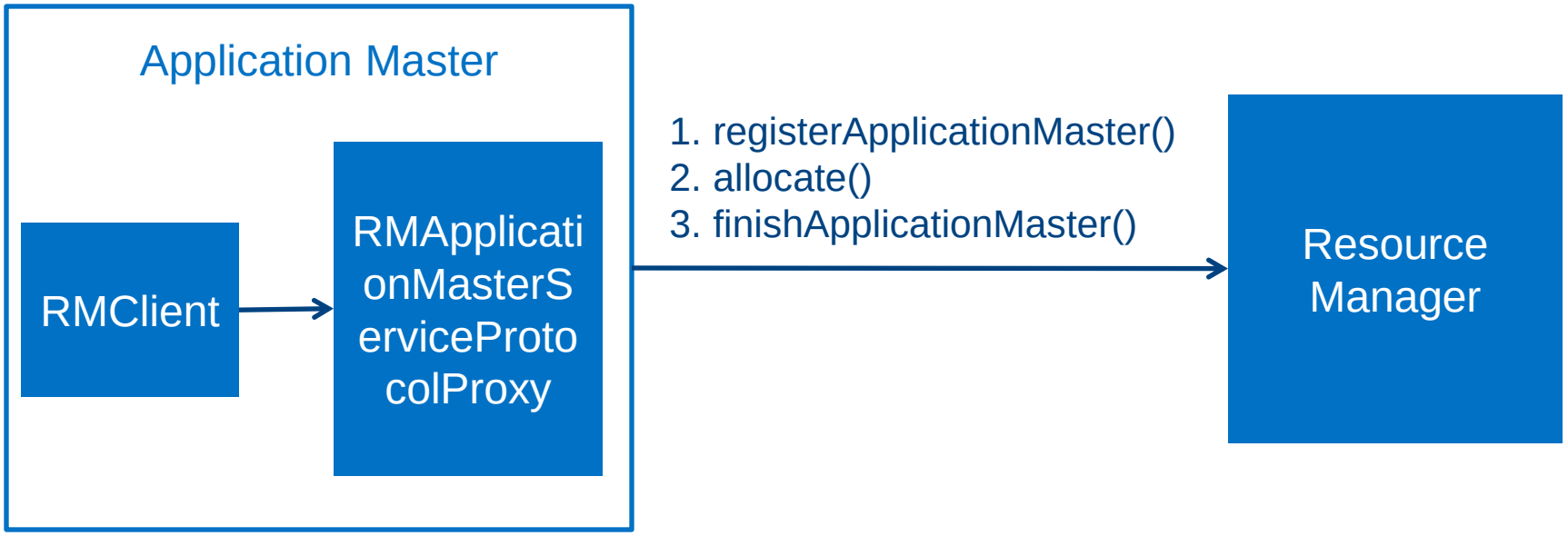
4. getClusterMetrics()

5. getClusterNodes()

6. getQueueInfo()

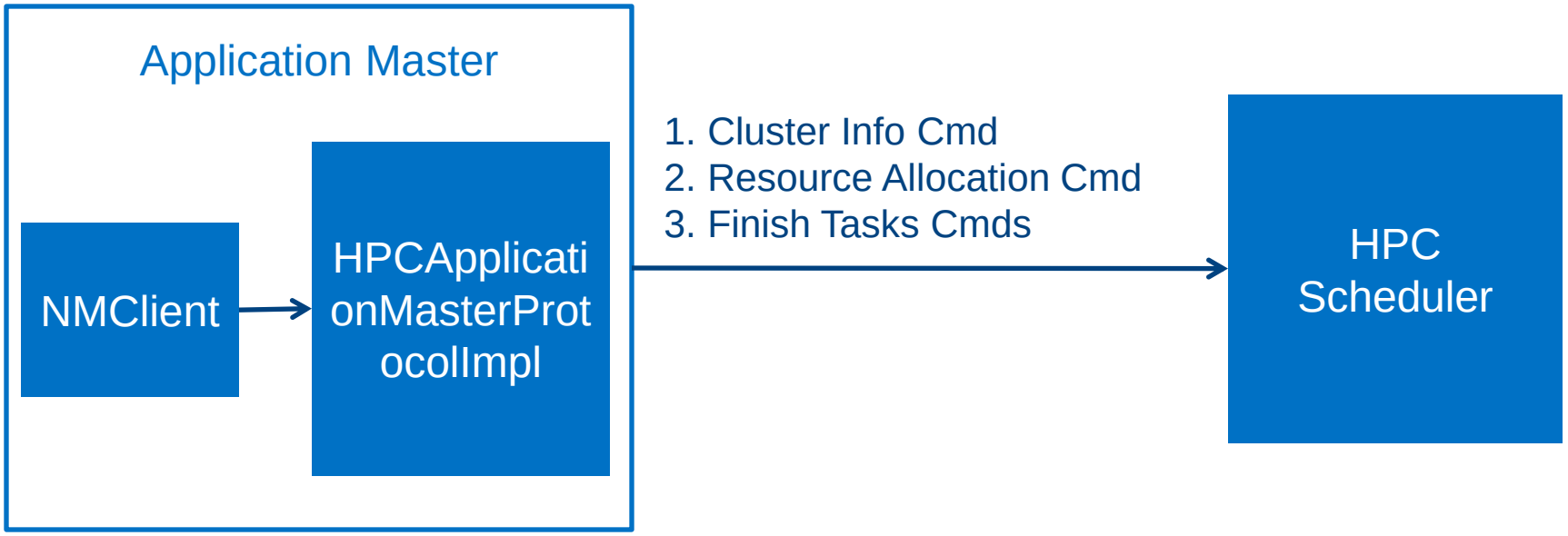
Application Master Protocol

Yarn Application Master Flow Diagram



Application Master Protocol

HPC Scheduler Application Master Flow Diagram



Application Master Protocol

API's for interaction

1. registerApplicationMaster()

The interface used by a new ApplicationMaster to register with the ResourceManager.

2. allocate()

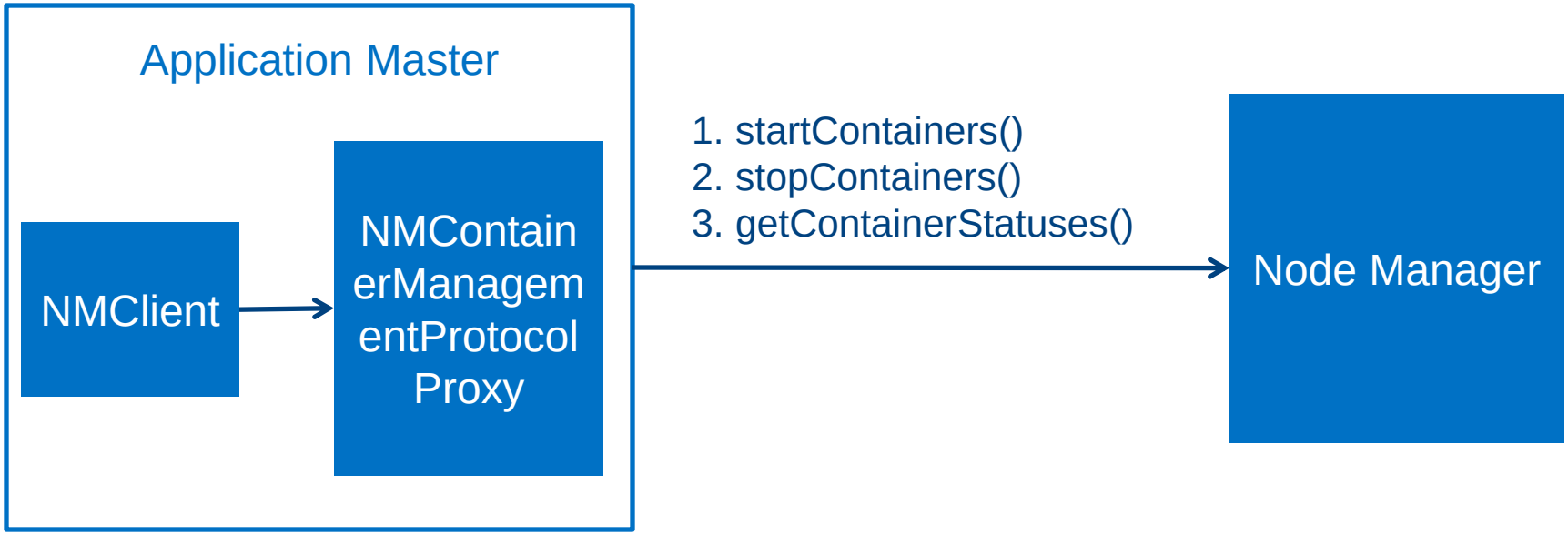
The main interface between an ApplicationMaster and the ResourceManager.

3. finishApplicationMaster()

The interface used by an ApplicationMaster to notify the ResourceManager about its completion (success or failed).

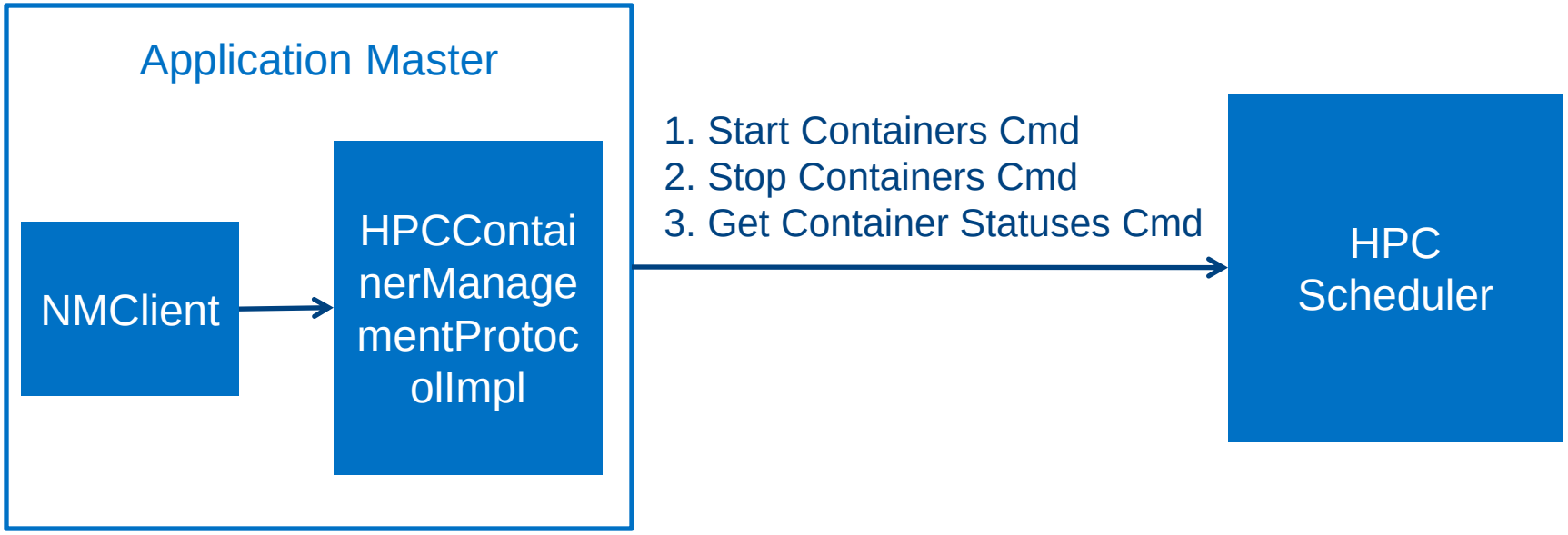
Container Management Protocol

Yarn Container Management Flow



Container Management Protocol

HPC Scheduler Task Management Flow



Container Management Protocol

API's for interaction

1. startContainers()

The ApplicationMaster provides a list of StartContainerRequest's to a NodeManager to start Container's allocated to it using this interface.

2. stopContainers()

The ApplicationMaster requests a NodeManager to stop a list of Container's allocated to it using this interface.

3. getContainerStatuses()

The API used by the ApplicationMaster to request for current statuses of Container's from the NodeManager.

Yarn Log Aggregation

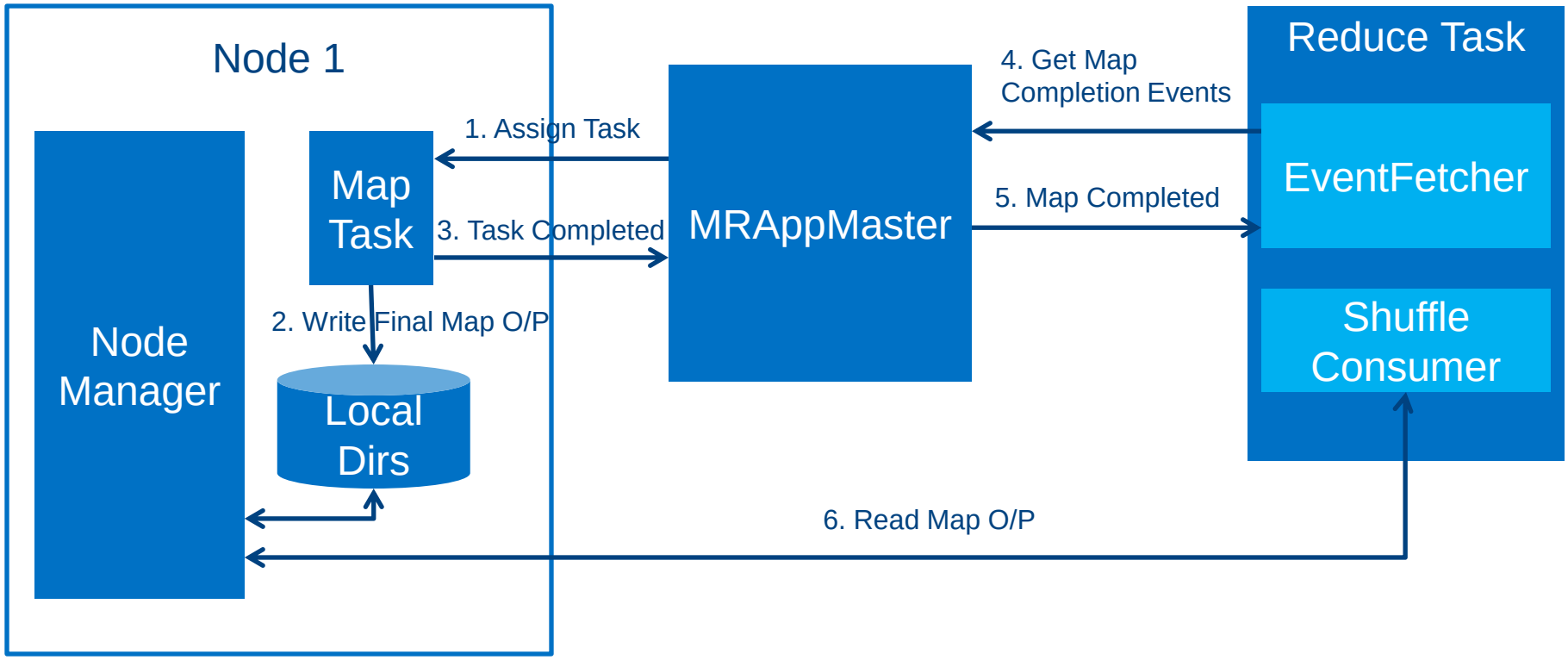
➢ Log Aggregation by Node Manager

```
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
 </property>
```

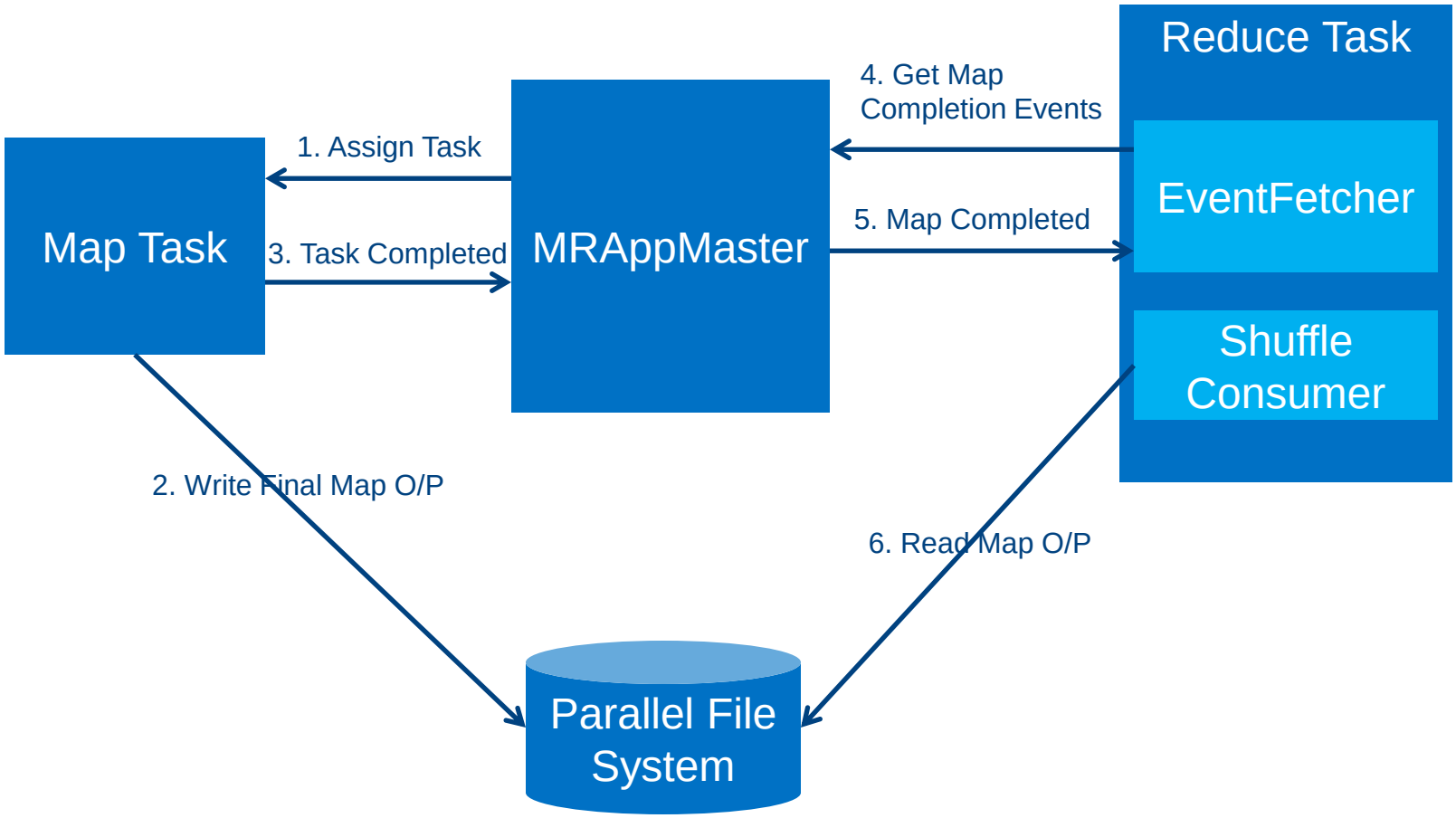
➢ Log Aggregation with HPC Scheduler

- Issue an HPC scheduler command to execute in all nodes(where application tasks executed) as part of ApplicationMasterProtocol.finishApplicationMaster() for aggregating the application logs.

Shuffle Handling – Hadoop



Shuffle Handling – HPC File Systems



Shuffle Handling

Shuffle Handler

```
<property>
  <name>mapreduce.job.map.output.collector.class</name>
<value>org.apache.hadoop.mapred.MapTask$MapOutputBuffer</value>
  <description>
    The MapOutputCollector implementation(s) to use. This may be a comma-separated list of class names, in which case the map task will try to initialize each of the collectors in turn. The first to successfully initialize will be used.
  </description>
</property>
```

Shuffle Handling

Shuffle Consumer

```
<property>
  <name>mapreduce.job.reduce.shuffle.consumer.plugin.class</name>
  <value>org.apache.hadoop.mapreduce.task.reduce.Shuffle</value>
  <description>
    Name of the class whose instance will be used to send shuffle
requests by reduce tasks of this job. The class must be an instance of
org.apache.hadoop.mapred.ShuffleConsumerPlugin.
  </description>
</property>
```

Summary

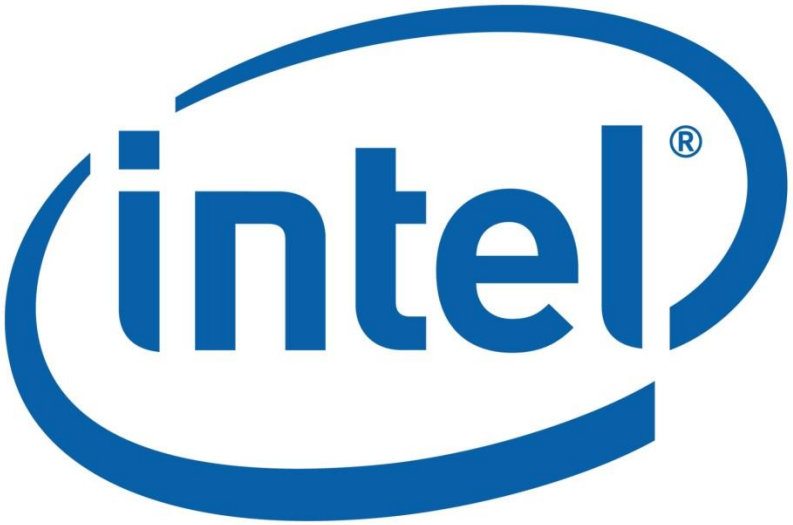
- ✓ HDFS configuration for new File System
- ✓ HPC Schedulers
- ✓ YARN Protocols
- ✓ M/R Shuffle Implementation
- ✓ Yarn Log Aggregation

Q & A

Thank You…

devaraj@apache.org

Notices and Disclaimers


intel