# Architecture



# Detail

1. Metrics: Collect every certain period
   1. Latency
   2. Queue Usage
   3. Requests
2. Broker:
   1. Idle. Release reserved resource.
   2. Normal. Stay unchanged.
   3. Warn. Send reservation request if there is no pending one.
   4. Critical. Force sending reservation request.
3. Llama Client: Implemented on Thrift.
   1. Reserve: handler ID, resource, location, queue, user, etc.
   2. Release: reservation ID
   3. Register: Register to Llama and get handler ID.

# Deploy

Make a Jar and put into hbase/lib directory together with llama.jar. (In case hbase-server.jar is sealed, you have to extract all packages and add into hbase-server.jar)

Add following configurations:

<property>

<name>hbase.region.server.rpc.scheduler.factory.class</name>

<value>org.apache.hadoop.hbase.ipc.MonitoredSimpleRpcSchedulerFactory</value>

</property>

<property>

<name>hbase.schedule.llama.host</name>

<value>llama-host</value>

</property>

Add a queue called root.hbase(default, you can change it with configurations). Allocate reasonable resources.

Start HBase and run jobs. You should see resource usage in above queue.

# Configuration

|  |  |
| --- | --- |
| Conf | Default |
| hbase.ipc.queue.dump.interval | 10 \* 1000 |
| hbase.schedule.broker | LlamaResourceBrokerFactory.class |
| hbase.ipc.latency.threshold | 1000 |
| hbase.schedule.llama.cpu\_adjust | 1 |
| hbase.schedule.llama.mem\_adjust\_mb | 1024 |
| hbase.schedule.llama.min\_cpu | 1 |
| hbase.schedule.llama.min\_mem | 1024 |
| hbase.schedule.llama.user | hbase |
| hbase.schedule.llama.queue | root.hbase |
| hbase.schedule.llama.host |  |
| hbase.schedule.llama.port | 15000 |
| hbase.schedule.llama.client\_port | 55555 |

# Future Work

1. Extract code to make a monitor layer.
2. Better schedule logic.
3. What to do with allocated resources.
4. User based schedule.
5. Other solutions?