Software Design Document

Bridging Java Example Framework

# 

# 

# 

# Section 1: Imports

Example framework code

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import com.google.protobuf.ByteString;

import org.apache.mesos.\*;

import org.apache.mesos.Protos.\*;

Example list of imports

#include <fstream>

#include <iostream>

#include <string>

#include <vector>

#include <unordered\_map>

#include <list>

#include <map>

//Our protobuffs

#include "curated.pb.h"

#include mesos::Resources;

using namespace mesos;

# 

# 

# Section 2: Class Implementation

Example framework code

public class TestFramework {

static class TestScheduler implements Scheduler {

public TestScheduler(boolean implicitAcknowledgements,

ExecutorInfo executor) {

this(implicitAcknowledgements, executor, 5);

}

public TestScheduler(boolean implicitAcknowledgements,

ExecutorInfo executor,

int totalTasks) {

this.implicitAcknowledgements = implicitAcknowledgements;

this.executor = executor;

this.totalTasks = totalTasks;

}

//Sections 3-6 here

private final boolean implicitAcknowledgements;

private final ExecutorInfo executor;

private final int totalTasks;

private int launchedTasks = 0;

private int finishedTasks = 0;

}//end of testScheduler class

Analysis of code

Here is the code for the class and it’s constructor. These will be necessary to recreate in C++ with almost identical functionality.

# 

# 

# Section 3: Registration

Example framework code

@Override

public void registered(SchedulerDriver driver,

FrameworkID frameworkId,

MasterInfo masterInfo) {

System.out.println("Registered! ID = " + frameworkId.getValue());

}

@Override

public void reregistered(SchedulerDriver driver, MasterInfo masterInfo) {}

@Override

public void disconnected(SchedulerDriver driver) {}

Analysis of code

Here we have the registration code for the framework. These functions check states and confirm that the framework had a successful registration. We will need to replicate these functions as well as flesh out the empty ones.

# 

# 

# Section 4: Launching Tasks

Example framework code

@Override

public void resourceOffers(SchedulerDriver driver, List < Offer > offers) {

double CPUS\_PER\_TASK = 1;

double MEM\_PER\_TASK = 128;

for (Offer offer: offers) {

Offer.Operation.Launch.Builder launch = Offer.Operation.Launch.newBuilder();

double offerCpus = 0;

double offerMem = 0;

for (Resource resource: offer.getResourcesList()) {

if (resource.getName().equals("cpus")) {

offerCpus += resource.getScalar().getValue();

} else if (resource.getName().equals("mem")) {

offerMem += resource.getScalar().getValue();

}

}

Analysis of code

This code runs through all of the resource offers from mesos and takes note of them.

Example framework code

System.out.println(

"Received offer " + offer.getId().getValue() + " with cpus: " + offerCpus +

" and mem: " + offerMem);

double remainingCpus = offerCpus;

double remainingMem = offerMem;

while (launchedTasks < totalTasks &&

remainingCpus >= CPUS\_PER\_TASK &&

remainingMem >= MEM\_PER\_TASK) {

TaskID taskId = TaskID.newBuilder()

.setValue(Integer.toString(launchedTasks++)).build();

System.out.println("Launching task " + taskId.getValue() +

" using offer " + offer.getId().getValue());

TaskInfo task = TaskInfo.newBuilder()

.setName("task " + taskId.getValue())

.setTaskId(taskId)

.setSlaveId(offer.getSlaveId())

.addResources(Resource.newBuilder()

.setName("cpus")

.setType(Value.Type.SCALAR)

.setScalar(Value.Scalar.newBuilder().setValue(CPUS\_PER\_TASK)))

.addResources(Resource.newBuilder()

.setName("mem")

.setType(Value.Type.SCALAR)

.setScalar(Value.Scalar.newBuilder().setValue(MEM\_PER\_TASK)))

.setExecutor(ExecutorInfo.newBuilder(executor))

.build();

launch.addTaskInfos(TaskInfo.newBuilder(task));

remainingCpus -= CPUS\_PER\_TASK;

remainingMem -= MEM\_PER\_TASK;

}

// NOTE: We use the new API `acceptOffers` here to launch tasks. The

// 'launchTasks' API will be deprecated.

List < OfferID > offerIds = new ArrayList < OfferID > ();

offerIds.add(offer.getId());

List < Offer.Operation > operations = new ArrayList < Offer.Operation > ();

Offer.Operation operation = Offer.Operation.newBuilder()

.setType(Offer.Operation.Type.LAUNCH)

.setLaunch(launch)

.build();

operations.add(operation);

Filters filters = Filters.newBuilder().setRefuseSeconds(1).build();

driver.acceptOffers(offerIds, operations, filters);

}

}

@Override

public void offerRescinded(SchedulerDriver driver, OfferID offerId) {}

Analysis of code

This code runs through all of the resource offers from mesos and uses the ones that are acceptable to launch tasks as needed. For our framework, the initial version will need to hoard offers until it has enough to launch all necessary ZooKeeper nodes at the same time.

# Section 5: Task monitoring

Example framework code

@Override

public void statusUpdate(SchedulerDriver driver, TaskStatus status) {

System.out.println("Status update: task " + status.getTaskId().getValue() +

" is in state " + status.getState().getValueDescriptor().getName());

if (status.getState() == TaskState.TASK\_FINISHED) {

finishedTasks++;

System.out.println("Finished tasks: " + finishedTasks);

if (finishedTasks == totalTasks) {

driver.stop();

}

}

if (status.getState() == TaskState.TASK\_LOST ||

status.getState() == TaskState.TASK\_KILLED ||

status.getState() == TaskState.TASK\_FAILED) {

System.err.println("Aborting because task " + status.getTaskId().getValue() +

" is in unexpected state " +

status.getState().getValueDescriptor().getName() +

" with reason '" +

status.getReason().getValueDescriptor().getName() + "'" +

" from source '" +

status.getSource().getValueDescriptor().getName() + "'" +

" with message '" + status.getMessage() + "'");

driver.abort();

}

if (!implicitAcknowledgements) {

driver.acknowledgeStatusUpdate(status);

}

}

Analysis of code

This code keeps track of all running tasks. It will be necessary to replicate this functionality within our framework. If time permits we will be implementing task reconciliation as well. This will ensure task consistency between master and frameworks.

# 

# 

# Section 6: Helper Functions

Example framework code

@Override

public void frameworkMessage(SchedulerDriver driver,

ExecutorID executorId,

SlaveID slaveId,

byte[] data) {}

@Override

public void slaveLost(SchedulerDriver driver, SlaveID slaveId) {}

@Override

public void executorLost(SchedulerDriver driver,

ExecutorID executorId,

SlaveID slaveId,

int status) {}

public void error(SchedulerDriver driver, String message) {

System.out.println("Error: " + message);

}

Analysis of code

These functions are there to raise errors for administrators. They are necessary to monitor the framework. As such will be replicated with useful logging within our framework.

# 

# Section 7: Main & Stretch Goals

Example framework code

private static void usage() {

String name = TestFramework.class.getName();

System.err.println("Usage: " + name + " master <tasks>");

}

public static void main(String[] args) throws Exception {

if (args.length < 1 || args.length > 2) {

usage();

System.exit(1);

}

String uri = new File("./test-executor").getCanonicalPath();

ExecutorInfo executor = ExecutorInfo.newBuilder()

.setExecutorId(ExecutorID.newBuilder().setValue("default"))

.setCommand(CommandInfo.newBuilder().setValue(uri))

.setName("Test Executor (Java)")

.setSource("java\_test")

.build();

FrameworkInfo.Builder frameworkBuilder = FrameworkInfo.newBuilder()

.setUser("") // Have Mesos fill in the current user.

.setName("Test Framework (Java)")

.setCheckpoint(true);

boolean implicitAcknowledgements = true;

if (System.getenv("MESOS\_EXPLICIT\_ACKNOWLEDGEMENTS") != null) {

System.out.println("Enabling explicit acknowledgements for status updates");

implicitAcknowledgements = false;

}

Scheduler scheduler = args.length == 1 ? new TestScheduler(implicitAcknowledgements, executor) : new TestScheduler(implicitAcknowledgements, executor, Integer.parseInt(args[1]));

MesosSchedulerDriver driver = null;

if (System.getenv("MESOS\_AUTHENTICATE\_FRAMEWORKS") != null) {

System.out.println("Enabling authentication for the framework");

if (System.getenv("DEFAULT\_PRINCIPAL") == null) {

System.err.println("Expecting authentication principal in the environment");

System.exit(1);

}

Credential.Builder credentialBuilder = Credential.newBuilder()

.setPrincipal(System.getenv("DEFAULT\_PRINCIPAL"));

if (System.getenv("DEFAULT\_SECRET") != null) {

credentialBuilder.setSecret(System.getenv("DEFAULT\_SECRET"));

}

frameworkBuilder.setPrincipal(System.getenv("DEFAULT\_PRINCIPAL"));

driver = new MesosSchedulerDriver(

scheduler,

frameworkBuilder.build(),

args[0],

implicitAcknowledgements,

credentialBuilder.build());

} else {

frameworkBuilder.setPrincipal("test-framework-java");

driver = new MesosSchedulerDriver(scheduler, frameworkBuilder.build(), args[0], implicitAcknowledgements);

}

int status = driver.run() == Status.DRIVER\_STOPPED ? 0 : 1;

// Ensure that the driver process terminates.

driver.stop();

// For this test to pass reliably on some platforms, this sleep is

// required to ensure that the SchedulerDriver teardown is complete

// before the JVM starts running native object destructors after

// System.exit() is called. 500ms proved successful in test runs,

// but on a heavily loaded machine it might not.

// TODO(greg): Ideally, we would inspect the status of the driver

// and its associated tasks via the Java API and wait until their

// teardown is complete to exit.

Thread.sleep(500);

System.exit(status);

Analysis of code

It will be necessary to replicate most of the functionality of this code segment. Some of it is JVM specific and will not be required.