Monitoring Hadoop with Akka

Clint Combs
Senior Software Engineer at Collective

Collective - “The Audience Engine”

- Ad Technology Company
- Heavy Investment in Hadoop and Other Scalable Infrastructure
- Need to Monitor Hadoop Ecosystem
- Collect metrics with Riemann, Graphite, and Grafana

Hadoop Ecosystem

- Yarn REST API Provides Metrics for Cluster and Jobs
- Oozie - Workflow Scheduler
- Celos - Partial replacement for Oozie
- Ecosystem produces metrics
- Metrics processed as Messages by HadoopMetrics which is built with Akka

"Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications on the JVM."

http://akka.io/

Akka Platform

- Runs on the JVM
- Closely tied to the Scala language and runtime
- Provides a Java API, including experimental Java 8 lambda support

Akka Background

- Created by Jonas Bonér
- Typesafe Co-founder with Martin Odersky, the creator of Scala and author of javac for JDK 1.3
- Based on ideas Jonas learned from Erlang
- Actor Model originated with Carl Hewitt in 1973 paper, “A Universal Modular Actor Formalism for Artificial Intelligence.”

Akka Features

- Actors - simple high-level concurrency abstraction
- Fault Tolerance - supervisor hierarchy spans 1+ JVMs
- Location Transparency - distributed asynchronous message passing
- Persistence - messages optionally persisted and replayed on actor restart

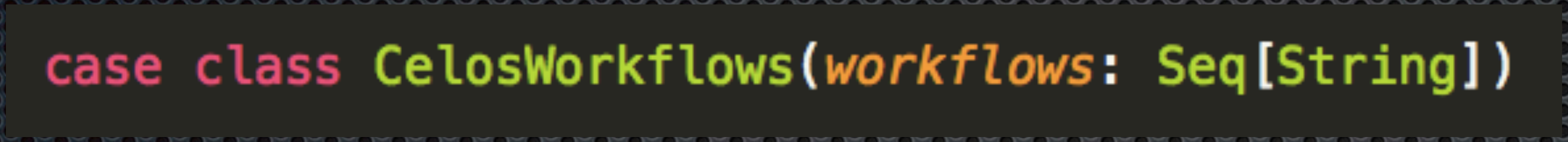
Actors

- Lightweight (~300 bytes per instance)
- Receive asynchronous messages via a mailbox
- Process messages one-at-a-time in the order they are received
- Maintain and should not directly share their own state

Messages

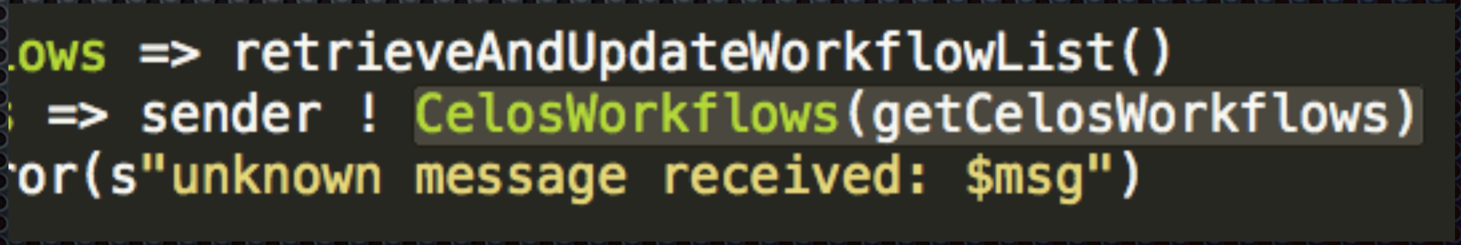
- Sent to Actors from other Actors
- Sometimes Sent from Non-Actors
- Immutable
  - Built-in types
  - Scala case classes are ideal

Case Classes as Messages



```
case class CelosWorkflows(workflows: Seq[String])
```

- Immutable by default
- Generated equals and toString with constructor parameters
- Can construct without “new” via companion object
- Pattern matching

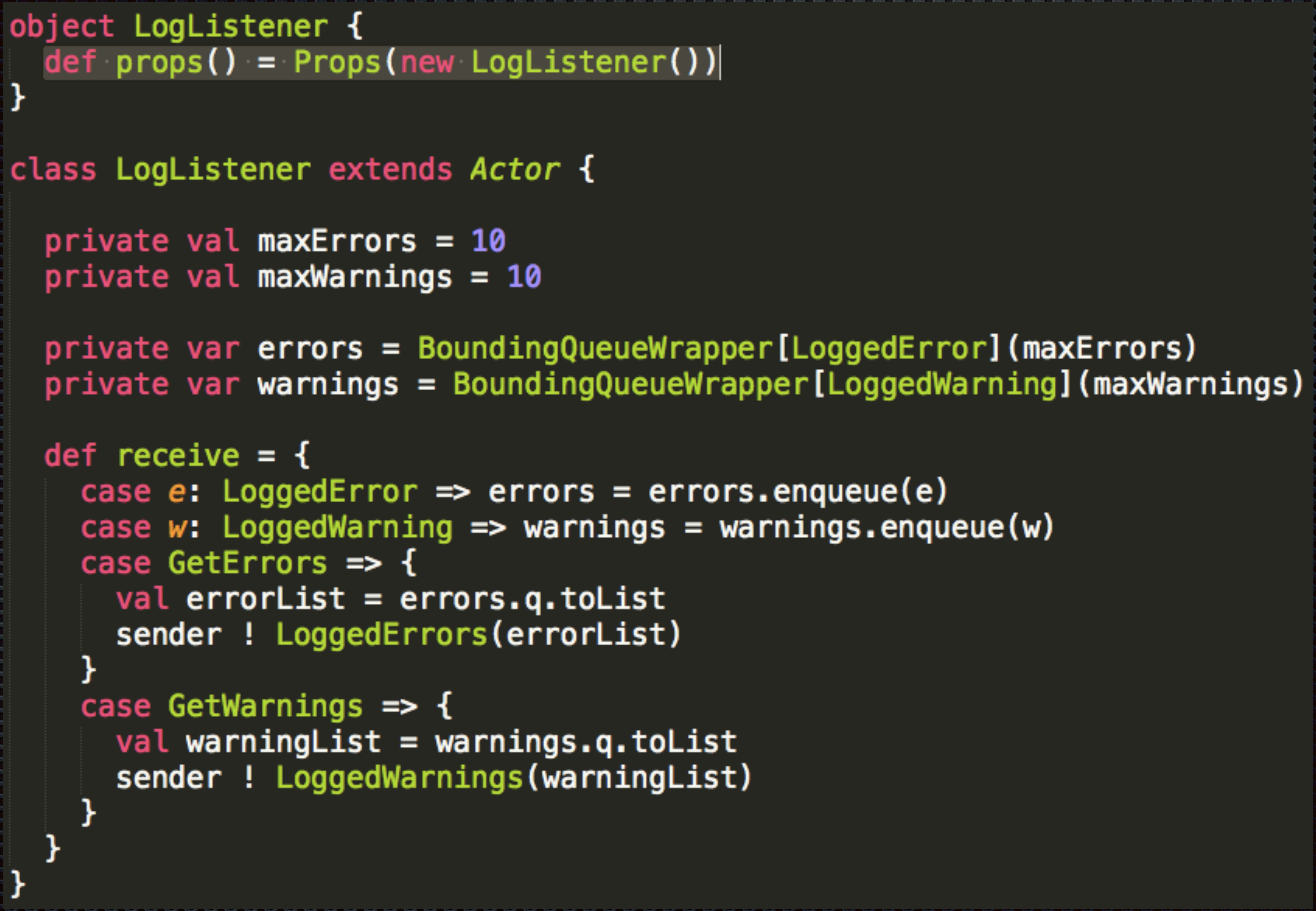


```
.ows => retrieveAndUpdateWorkflowList()
; => sender ! CelosWorkflows(getCelosWorkflows)
:or(s"unknown message received: $msg")
```

Creating Actors

- Don’t use new by itself
- Use Props - immutable (shareable) recipe for creating an actor
- ActorRef - used to send messages to actors
- Actor DSL

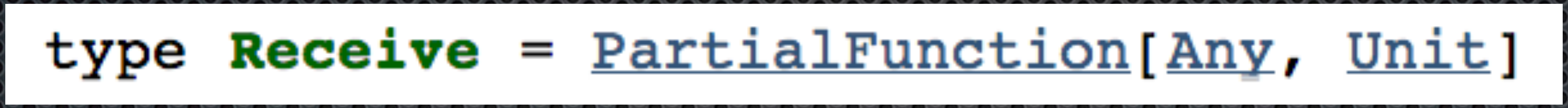
Props



```
object LogListener {
  def props() = Props(new LogListener())
}

class LogListener extends Actor {

  private val maxErrors = 10
  private val maxWarnings = 10

  private var errors = BoundingQueueWrapper[LoggedError](maxErrors)
  private var warnings = BoundingQueueWrapper[LoggedWarning](maxWarnings)

  def receive = {
    case e: LoggedError => errors = errors.enqueue(e)
    case w: LoggedWarning => warnings = warnings.enqueue(w)
    case GetErrors => {
      val errorList = errors.q.toList
      sender ! LoggedErrors(errorList)
    }
    case GetWarnings => {
      val warningList = warnings.q.toList
      sender ! LoggedWarnings(warningList)
    }
  }
}
```

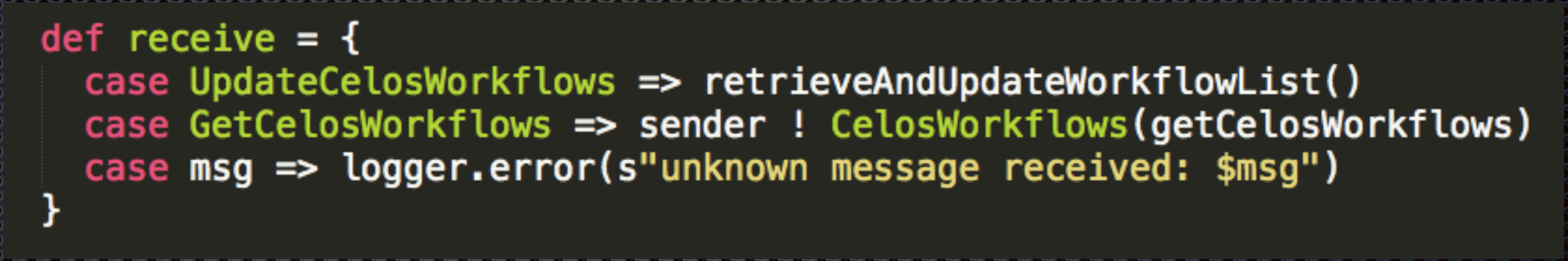
Receiving Messages

- Every actor has a receive function of type Receive



```
type Receive = PartialFunction[Any, Unit]
```

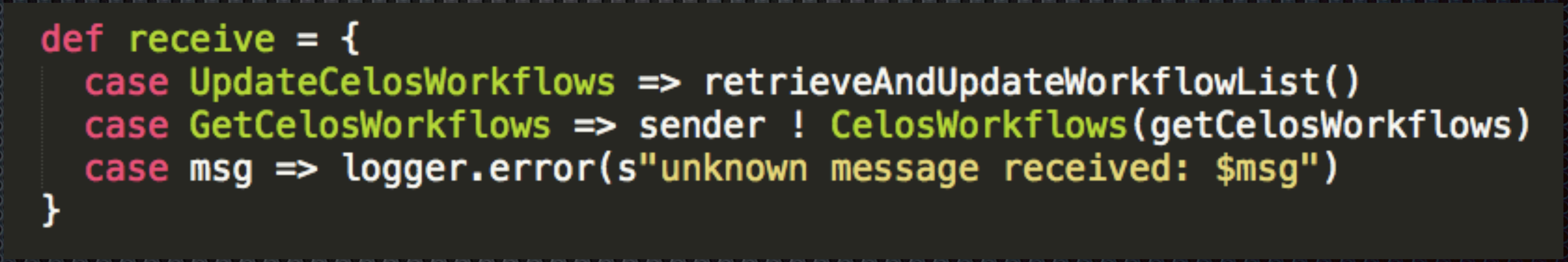
- Receive function processes messages from mailbox



```
def receive = {
  case UpdateCelosWorkflows => retrieveAndUpdateWorkflowList()
  case GetCelosWorkflows => sender ! CelosWorkflows(getCelosWorkflows)
  case msg => logger.error(s"unknown message received: $msg")
}
```

Reply to Messages

- Upon receiving a message an actor will often reply
- sender() ! replyMsg
- “tell, don’t ask”



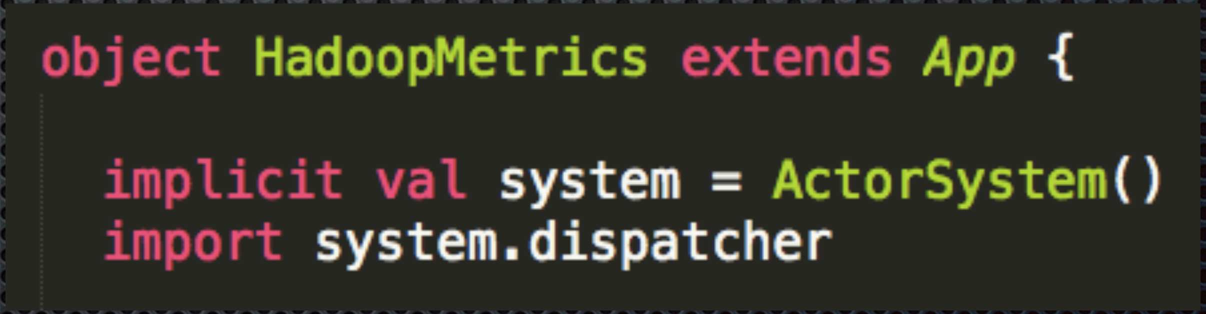
```
def receive = {
  case UpdateCelosWorkflows => retrieveAndUpdateWorkflowList()
  case GetCelosWorkflows => sender ! CelosWorkflows(getCelosWorkflows)
  case msg => logger.error(s"unknown message received: $msg")
}
```

Message Delivery Guarantees

- At-most-once
- Ordered per sender/receiver pair

Actor Systems

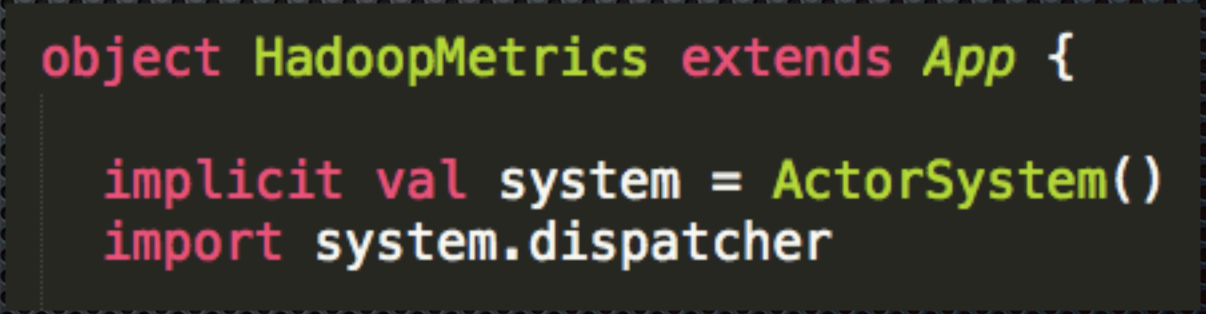
- One per application
- Manages resources and actor hierarchy
- Provide a default dispatcher



```
object HadoopMetrics extends App {

  implicit val system = ActorSystem()
  import system.dispatcher
```

ExecutionContext

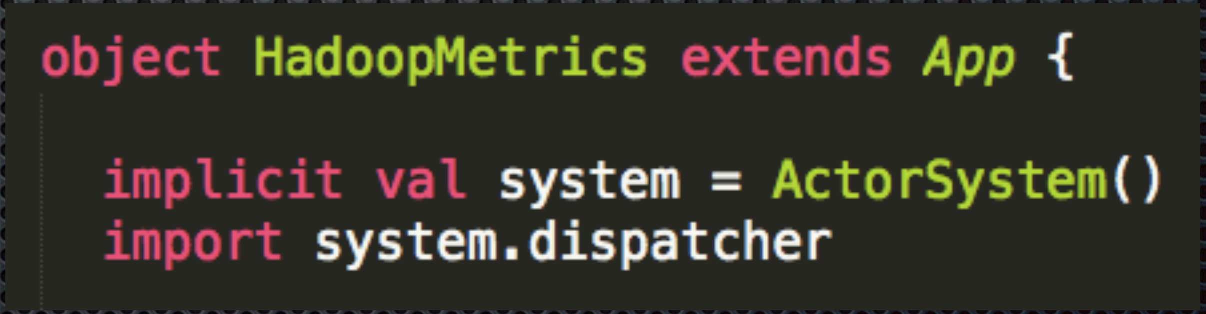
- Equivalent of java.util.concurrent.Executor
- Manages threads which execute Runnable and Future
- Provides default dispatcher



```
object HadoopMetrics extends App {

  implicit val system = ActorSystem()
  import system.dispatcher
```

MessageDispatcher

- An extension of ExecutionContext
- The “engine” that delivers messages to actors
- Default dispatcher is often replaced to scale application



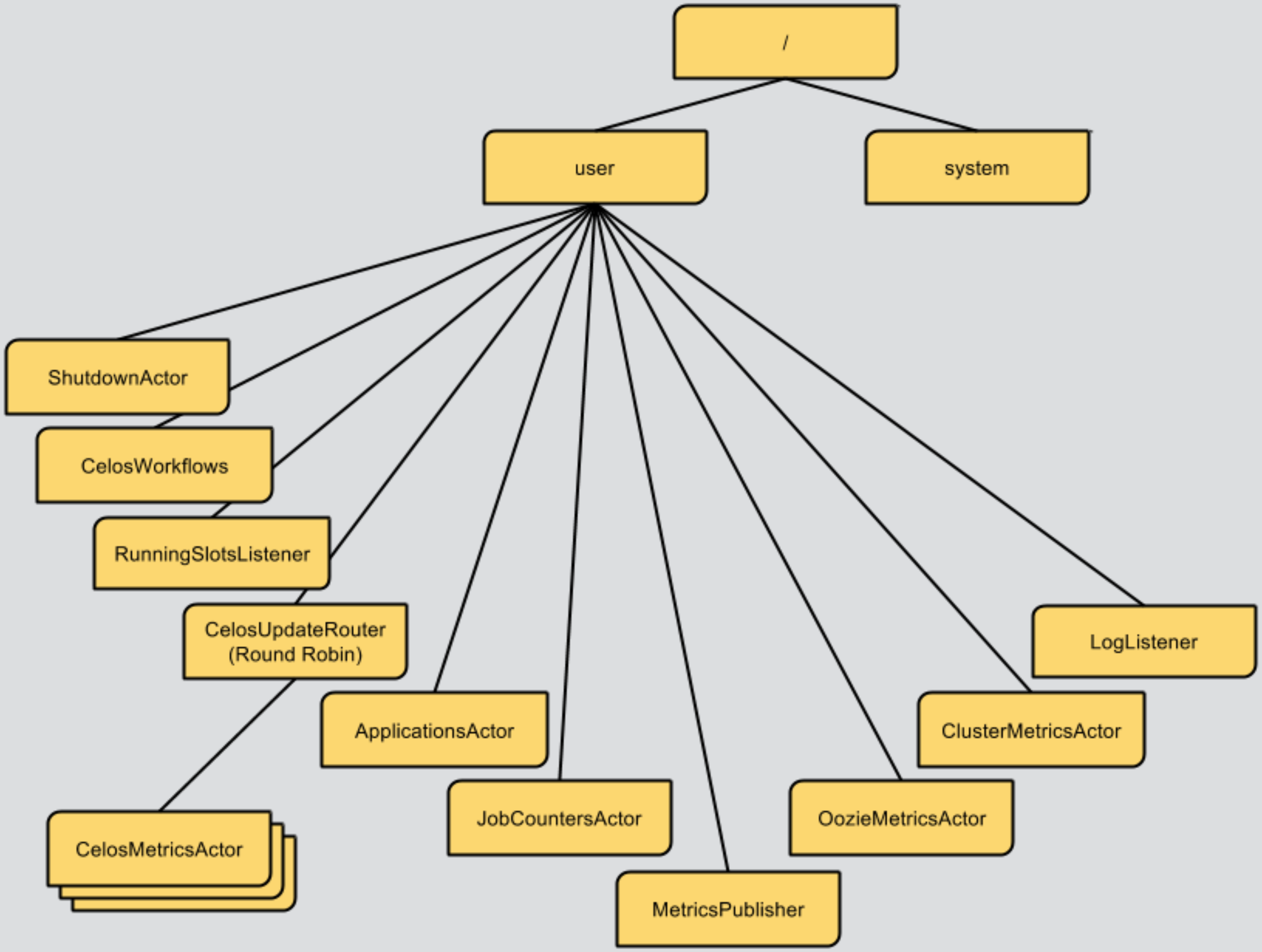
```
object HadoopMetrics extends App {

  implicit val system = ActorSystem()
  import system.dispatcher
```

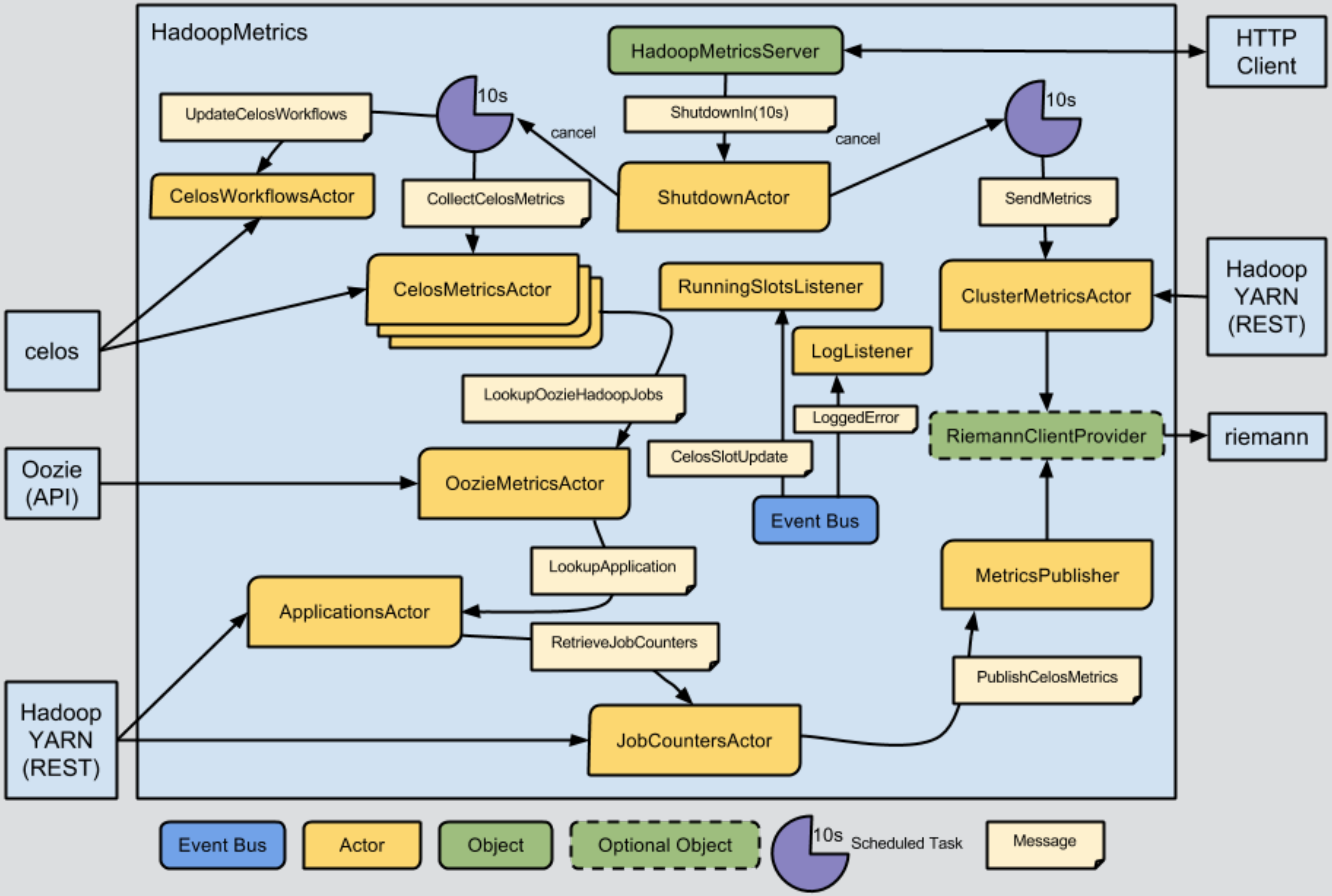
Performance

- 2.5 million actors per GB of heap (~300 bytes/actor)
- 50 million messages per second on a single machine
- Typesafe has tested a 2400 Node Akka Cluster on Google Compute Engine

Actor Hierarchy and Supervision

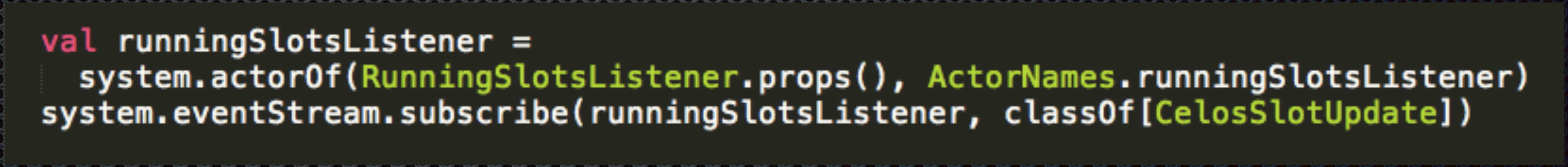
- Actors form a tree or Supervision Hierarchy
- Parents are notified of child failure and child restarted
- Failure is localized to a sub-branch or propagated up
- Routers act as supervisors (default to escalate)


/
user
system
ShutdownActor
CelosWorkflows
RunningSlotsListener
CelosUpdateRouter (Round Robin)
ApplicationsActor
JobCountersActor
MetricsPublisher
OozieMetricsActor
ClusterMetricsActor
LogListener
CelosMetricsActor


HadoopMetrics
HadoopMetricsServer
HTTP Client
UpdateCelosWorkflows
10s
cancel
ShutdownIn(10s)
cancel
10s
CelosWorkflowsActor
CollectCelosMetrics
ShutdownActor
SendMetrics
CelosMetricsActor
RunningSlotsListener
ClusterMetricsActor
Hadoop YARN (REST)
celos
LogListener
LookupOozieHadoopJobs
LoggedError
RiemannClientProvider
riemann
CelosSlotUpdate
Oozie (API)
OozieMetricsActor
Event Bus
LookupApplication
MetricsPublisher
ApplicationsActor
RetrieveJobCounters
PublishCelosMetrics
Hadoop YARN (REST)
JobCountersActor
Event Bus
Actor
Object
Optional Object
10s
Scheduled Task
Message

Event Bus

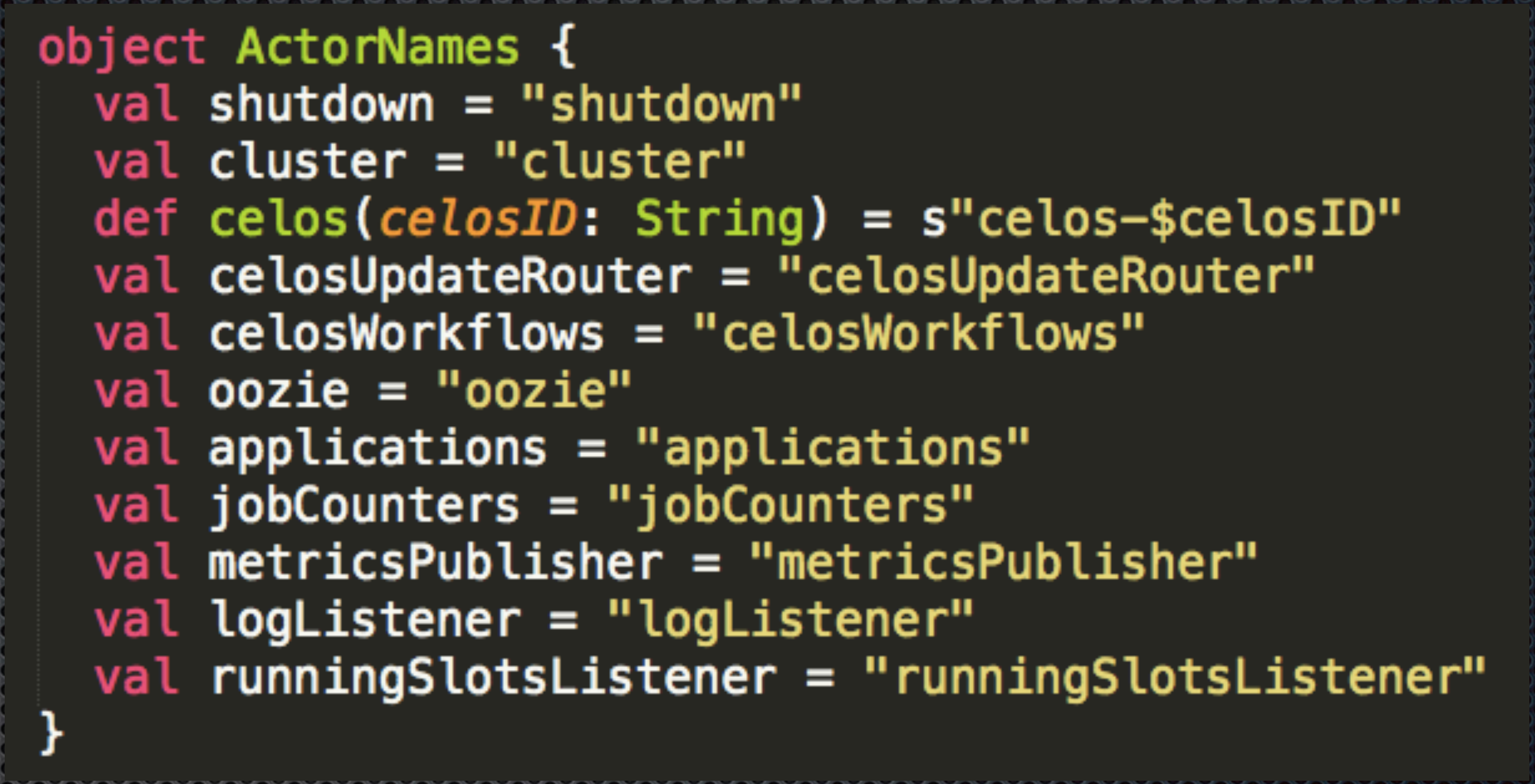
- Publish and Subscribe mechanism for messages
- Sender not preserved in message
- Actors subscribe and handle messages normally



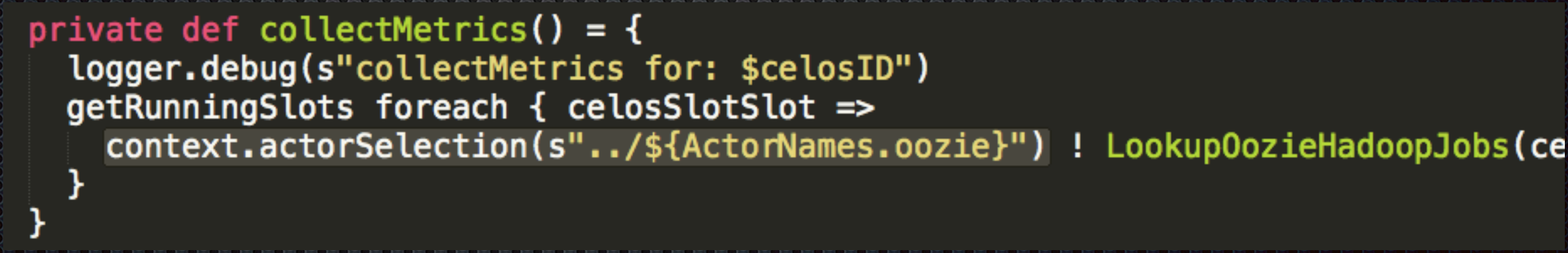
```
val runningSlotsListener =
  system.actorOf(RunningSlotsListener.props(), ActorNames.runningSlotsListener)
system.eventStream.subscribe(runningSlotsListener, classOf[CelosSlotUpdate])
```

Paths

- Actor path can be used to look up an ActorRef
- ActorRef is then used to send messages
- Paths may “select” multiple actors



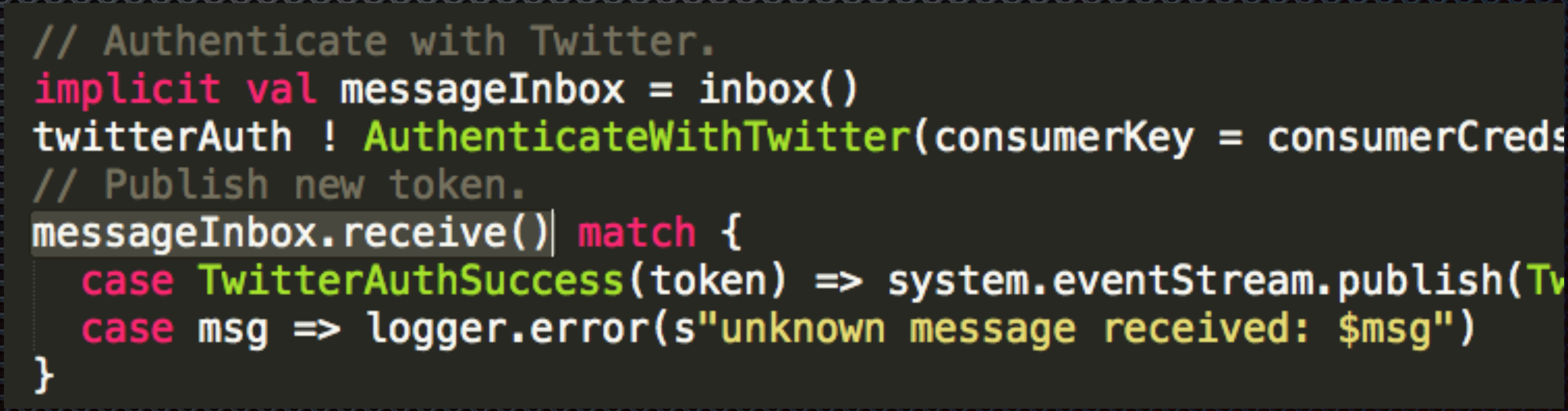
```scala
object ActorNames {
  val shutdown = "shutdown"
  val cluster = "cluster"
  def celos(celosID: String) = s"celos-$celosID"
  val celosUpdateRouter = "celosUpdateRouter"
  val celosWorkflows = "celosWorkflows"
  val oozie = "oozie"
  val applications = "applications"
  val jobCounters = "jobCounters"
  val metricsPublisher = "metricsPublisher"
  val logListener = "logListener"
  val runningSlotsListener = "runningSlotsListener"
}
```



```scala
private def collectMetrics() = {
  logger.debug(s"collectMetrics for: $celosID")
  getRunningSlots foreach { celosSlotSlot =>
    context.actorSelection(s"../${ActorNames.oozie}") ! LookupOozieHadoopJobs(ce
  }
}
```

Messaging Actors from Non-Actors

- Inbox
- the ask pattern
  - spawns temporary actor to handle reply
  - reply is handled with a future
- Await



```
// Authenticate with Twitter.
implicit val messageInbox = inbox()
twitterAuth ! AuthenticateWithTwitter(consumerKey = consumerCreds
// Publish new token.
messageInbox.receive() match {
  case TwitterAuthSuccess(token) => system.eventStream.publish(Tw
  case msg => logger.error(s"unknown message received: $msg")
}
```

The Scheduler and Cancellable

- Schedules
  - recurring or one-time messages to actors
  - recurring or one-time execution with futures
- Returned values are Cancellable

Shutdown

- Difficult problem in async systems
- Roll-your-own
- PoisonPill
- Graceful Stop

ShutdownActor

```scala
case class ShutdownIn(delay: FiniteDuration)
case class CancelOnShutdown(cancellable: Cancellable, actorRefOpt: Option[ActorRef])

class ShutdownActor extends Actor {

  private[ShutdownActor] case object ShutdownNow;

  implicit val system = context.system
  import system.dispatcher

  val logger = Logging(system, this.toString)

  var cancels = List[CancelOnShutdown]()

  def receive = {
    case ShutdownIn(delay) => {
      cancels foreach { c =>
        c.actorRefOpt match {
          case Some(a) => logger.info(s"cancelling ${c.cancellable} associated with ${a.path}")
          case None => logger.info(s"cancelling ${c.cancellable}")
        }
        c.cancellable.cancel
      }

      logger.info(s"shutting down actor system in ${delay}...")
      system.scheduler.scheduleOnce(delay, self, ShutdownNow)
    }
    case ShutdownNow => system.shutdown()
    case cancel @ CancelOnShutdown(c, a) => cancels = cancel :: cancels
    case msg => logger.error(s"unknown message received: $msg")
  }
}
```

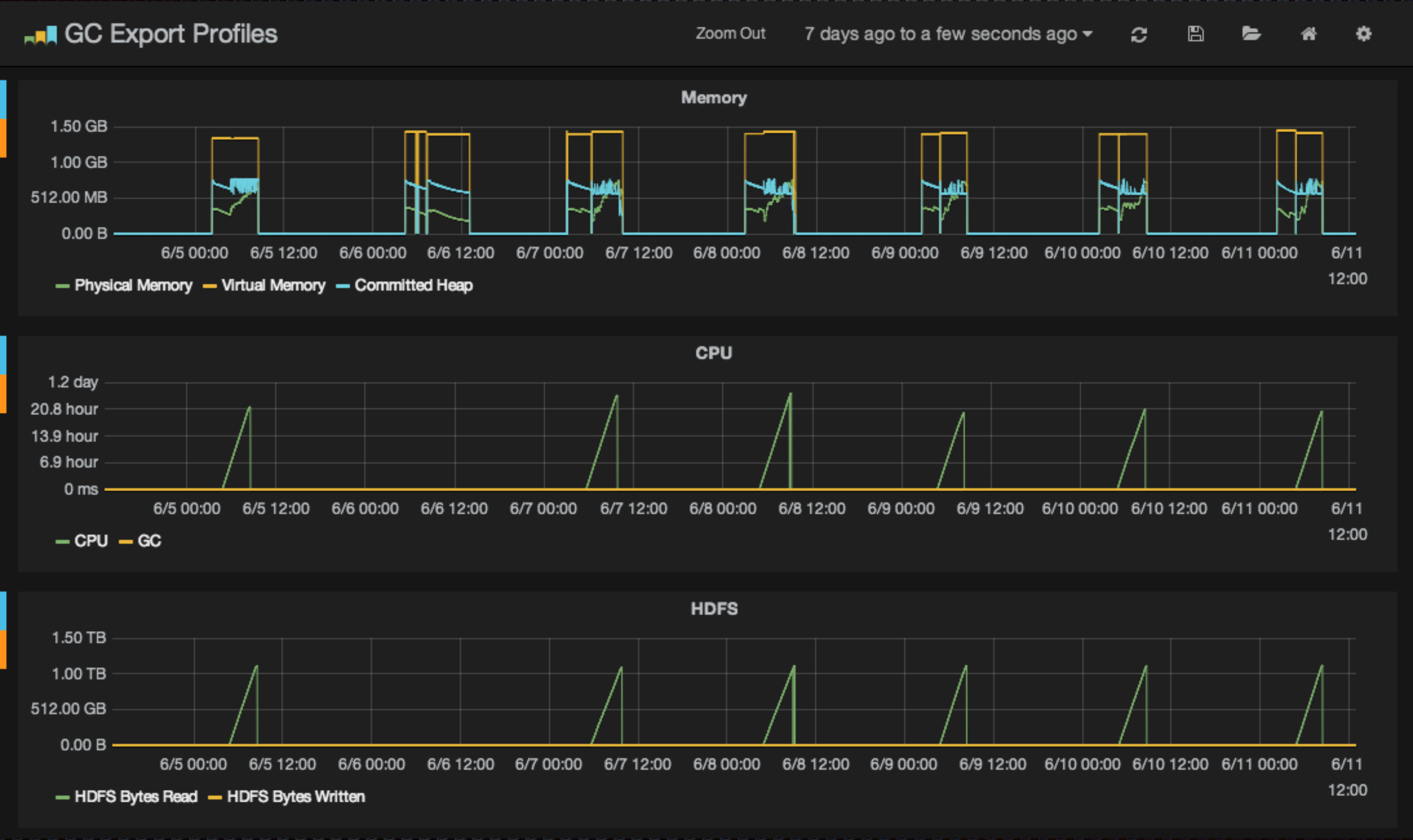
Other Akka Topics

- Testing framework closely integrated with ScalaTest
- Clustering
- Spray.io
  - Fully asynchronous HTTP becoming part of Akka
  - Used in this system for status and control

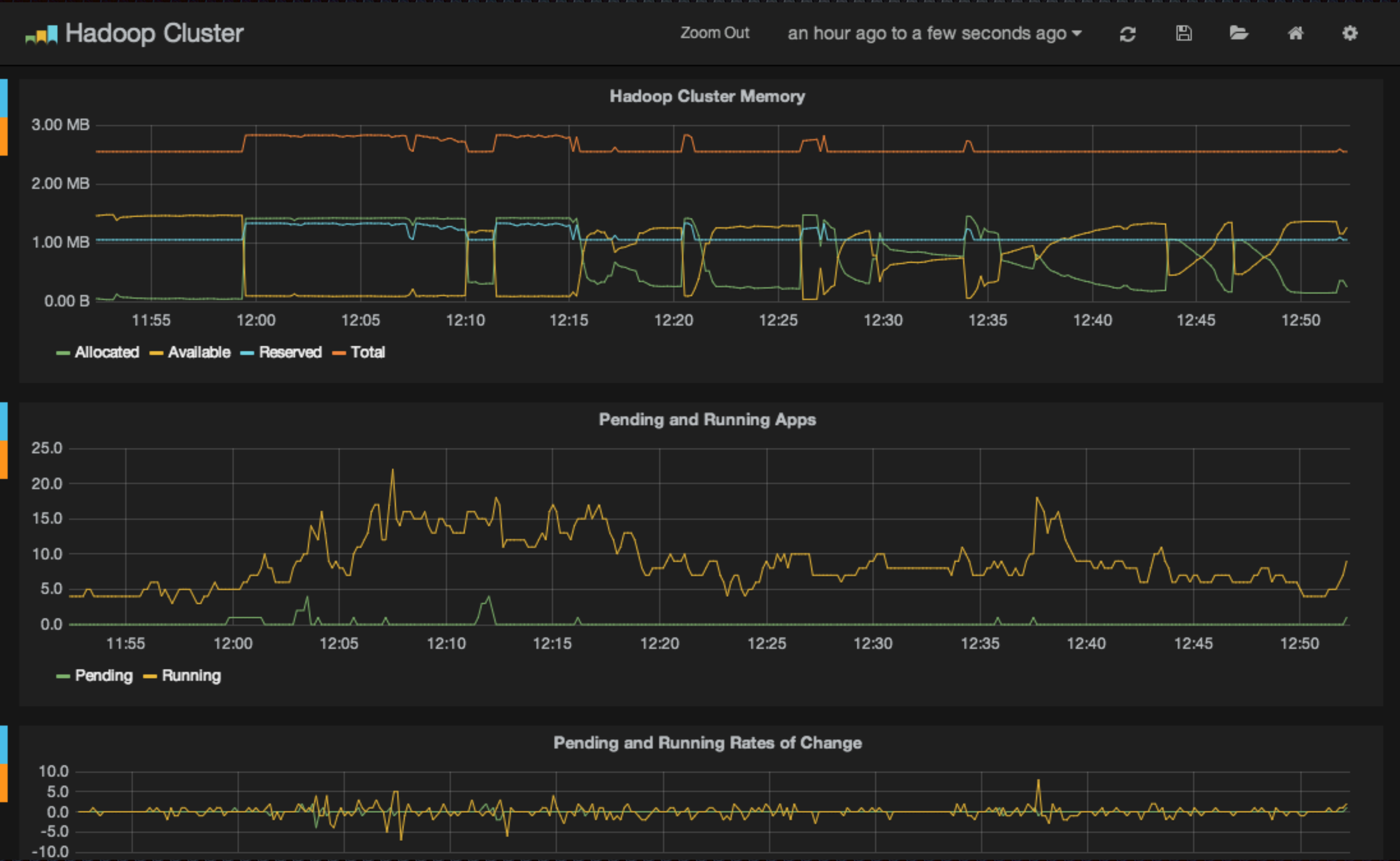
Results

- Job workflows are tracked through lifecycle:
  - Celos - workflows and metrics are configurable
  - Oozie - associated with Celos workflow slots
  - Hadoop - Jobs associated with Oozie workflow IDs
- Metrics are published to Grafana via Riemann

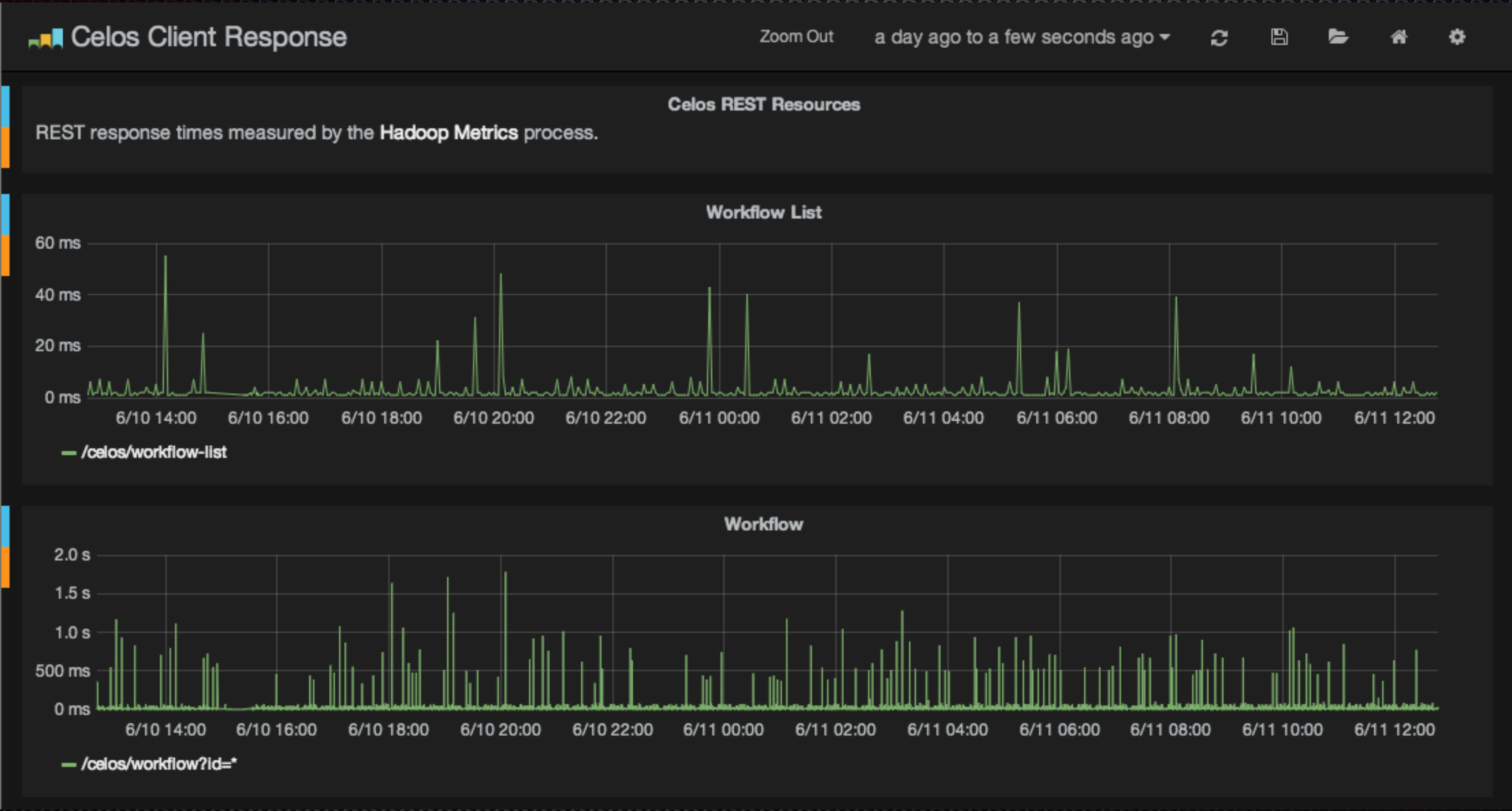
GrandCentral Export Profiles



Hadoop Cluster

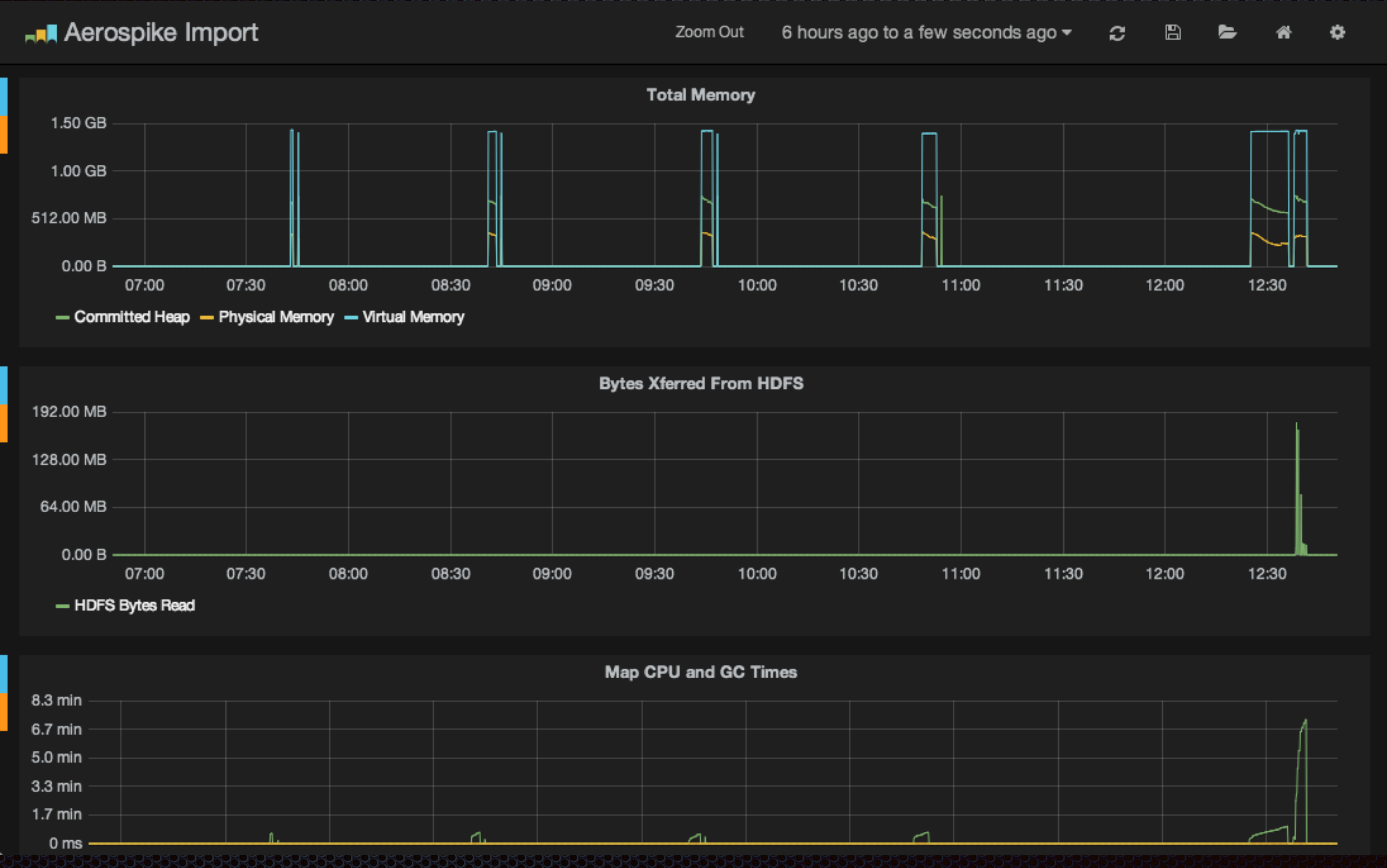


Celos Client Response



REST response times measured by the **Hadoop Metrics** process.

Aerospike Import



Aerospike Import
Zoom Out
6 hours ago to a few seconds ago
Total Memory
1.50 GB
1.00 GB
512.00 MB
0.00 B
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
Committed Heap
Physical Memory
Virtual Memory
Bytes Xferred From HDFS
192.00 MB
128.00 MB
64.00 MB
0.00 B
HDFS Bytes Read
Map CPU and GC Times
8.3 min
6.7 min
5.0 min
3.3 min
1.7 min
0 ms