Scala at LinkedIn: Distributed Computing with Norbert

Chris Conrad
Technical Lead, Social Graph

1. Why Norbert?

2. Tutorial

3. Why Scala?

Why?

Search Network Analytics

Lots of data
Lots of traffic

People You May Know Recommendation Engine

For partitioning data: Voldemort

For batch processing: Hadoop & Pig

People Search

15 million queries per day

250 queries per second peak

up to 100 tokens per query

Scatter Gather

Social Graph

65+ million nodes

680+ million edges

250+ million requests per day

Peer to Peer

For partitioning workloads: ???

Software Load Balancing

Asynchronous

Group Management / Cluster Aware

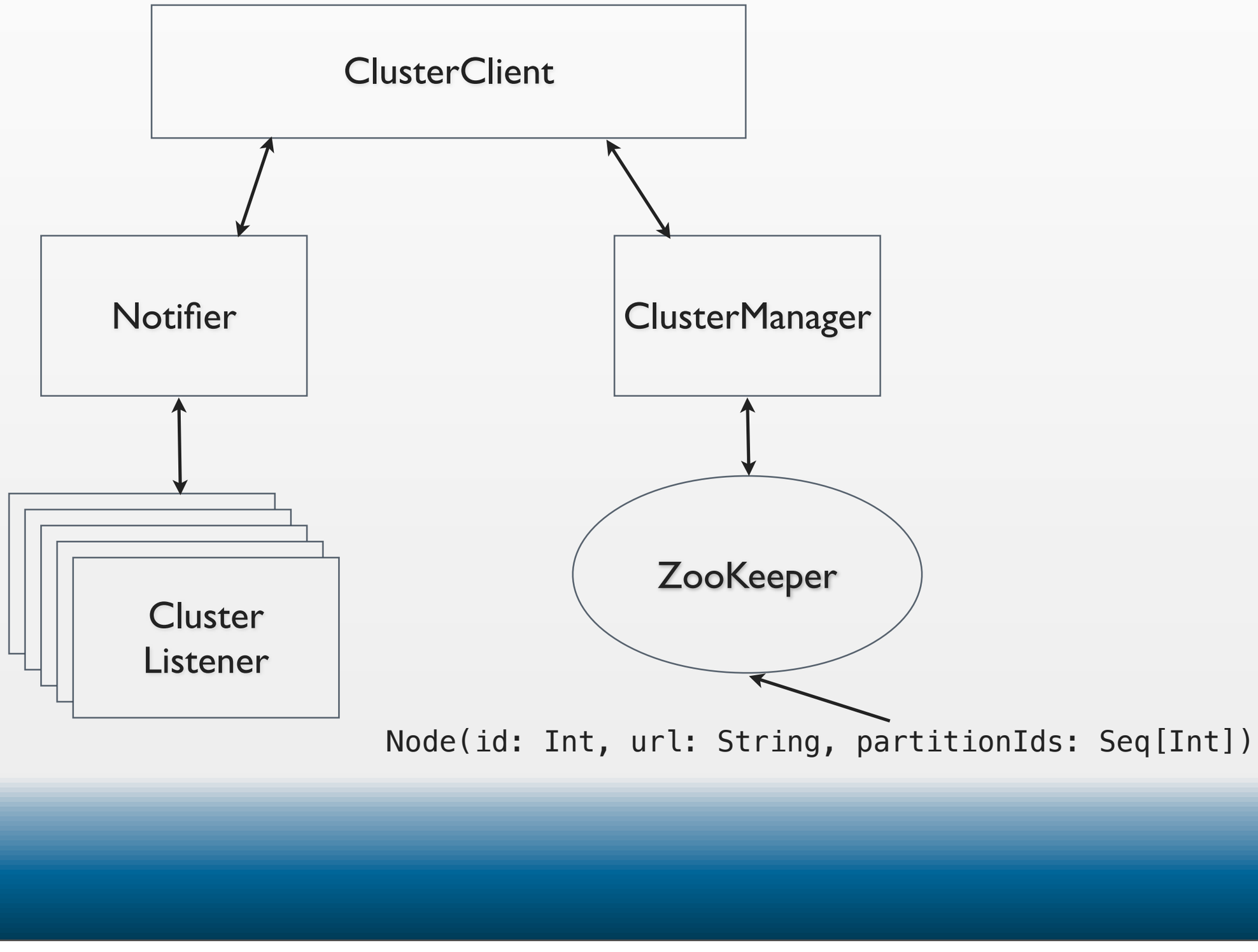
ZooKeeper
Netty
Protocol Buffers

Thus, Norbert

Using Norbert

1. Cluster Client

Cluster Client Design



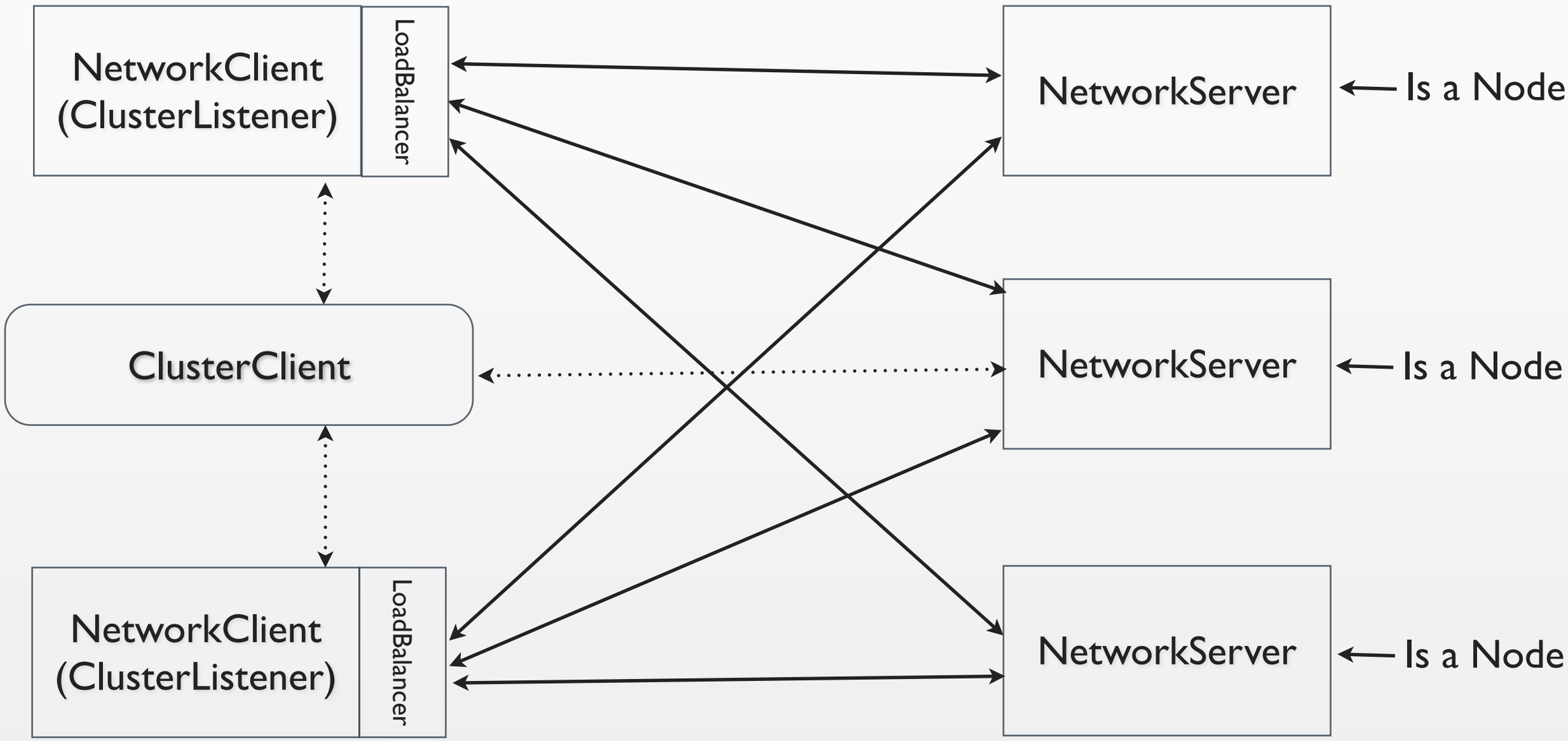
Cluster Manager

```
val cc = ClusterClient(“serviceName”,
    “zookeeperHost:2181”, 3000)
cc.awaitConnectionUninterruptibly

// Do stuff
cc.nodes
cc.addListener(new MyClusterListener)
cc.addNode(1, “hostname:31313”, Array(1, 2, 3))
```

2. Client / Server Framework

Client Server Design



Define your API

Protocol Buffer In Protocol Buffer Out

Balance your load

```
trait LoadBalancer {
  def nextNode: Option[Node]
}

trait LoadBalancerFactory {
  def newLoadBalancer(nodes: Seq[Node]):
      LoadBalancer
}
```

Or, partition it

```
trait PartitionedLoadBalancer[PartitionedId] {
  def nextNode(id: PartitionedId): Option[Node]
}

trait PartitionedLoadBalancerFactory[PartitionedId] {
  def newLoadBalancer(nodes: Seq[Node]):
      PartitionedLoadBalancer[PartitionedId]
}
```

Network Client

```
val nc = NetworkClient(config,
    new RoundRobinLoadBalancerFactory)

// Register your API
nc.registerRequest(
    MyRequestMessage.getDefaultInstance(),
    MyResponseMessage.getDefaultInstance())

// Make a call
val future = nc.sendMessage(myRequestMessage)
```

Partitioned Network Client

```
val nc = PartitionedNetworkClient(config,
    new IntConsistentHashLoadBalancerFactory)

// Register your API
...


// Make a call
val future = nc.sendMessage(1210, myRequestMessage)
val iter = nc.sendMessage(Array(1210, 1541, 890313),
    myRequestMessage)
```

Network Server

```
val ns = NetworkServer(config)

// Register your API
ns.registerHandler(
    MyRequestMessage.getDefaultInstance(),
    MyResponseMessage.getDefaultInstance(),
    myMessageHandler _)


// Bind
ns.bind(nodeId)

// Handler
def myMessageHandler(message: Message):
    Message = // your handler goes here
```

README, examples and API docs for more

So why Scala?

Multithreaded, asynchronous coding is hard

Actors makes it easy(er)

Code reuse with traits

```
trait BaseNetworkClient  // common code, abstract net
trait NetworkClient extends BaseNetworkClient
trait PartitionedNetworkClient
   extends BaseNetworkClient


abstract class NettyBaseNetworkClient extends
    BaseNetworkClient  // common code, Netty impl
class NettyNetworkClient extends
    NettyBaseNetworkClient with NetworkClient
class NettyPartitionedNetworkClient extends
    NettyBaseNetworkClient with
    PartitionedNetworkClient
```

+ Mixins

```
class NettyNetworkClient extends
    LocalMessageExecution

trait LocalMessageExecution extends
    BaseNetworkClient {
  override protected def doSendMessage(...) {
    if (local) msgExecutor.execute(...) // local
    else super.doSendMessage(...) // remote
}
```

Dependency Injection and the Cake Pattern

Self types to declare dependencies

```
trait ClusterIoClientComponent {
  val clusterIoClient: ClusterIoClient

  trait ClusterIoClient { ... }
}

trait BaseNetworkClient {
  this: ClusterIoClientComponent =>

  def send(...) = clusterIoClient.send(...)
}
```

Wire in dependencies

```
trait NettyClusterIoClientComponent extends
    ClusterIoClientComponent {
  val clusterIoClient = new NettyClusterIoClient

  class NettyClusterIoClient extends
      ClusterIoClient { ... }
}

abstract class NettyBaseNetworkClient extends
    BaseNetworkClient with
    NettyClusterIoClientComponent
```

Testability

```
class BaseNetworkClientSpec extends
    SpecificationWithJUnit with Mockito {
  val nc = new BaseNetworkClient
      with ClusterIoClientComponent {
    val clusterIoClient = mock[ClusterIoClient]
  }

  “should call clusterIoClient” in {
    nc.clusterIoClient.send(...) returns ...
    nc.send(...)
    nc.clusterIoClient.send was called
  }
}
```

Norbert.

Open Source

Apache License 2.0

(comes with a Java API)

The End

http://sna-projects.com

http://github.com/rhavyn/norbert