Programação Distribuída & Processamento Assíncrono

Rodrigo Kochenburger <divoxx@gmail.com>

O que é isso?

Um sistema distribuído consiste de múltiplos nodos/programas autônomos que se comunicam entre si para alcançar um objetivo comum.

Sistemas assíncronos são aqueles que não trabalham em sincronia com outros sistemas, fazendo o seu processamento de forma independente e isolada, apenas se comunicando com outros processos para coletar ou enviar dados.

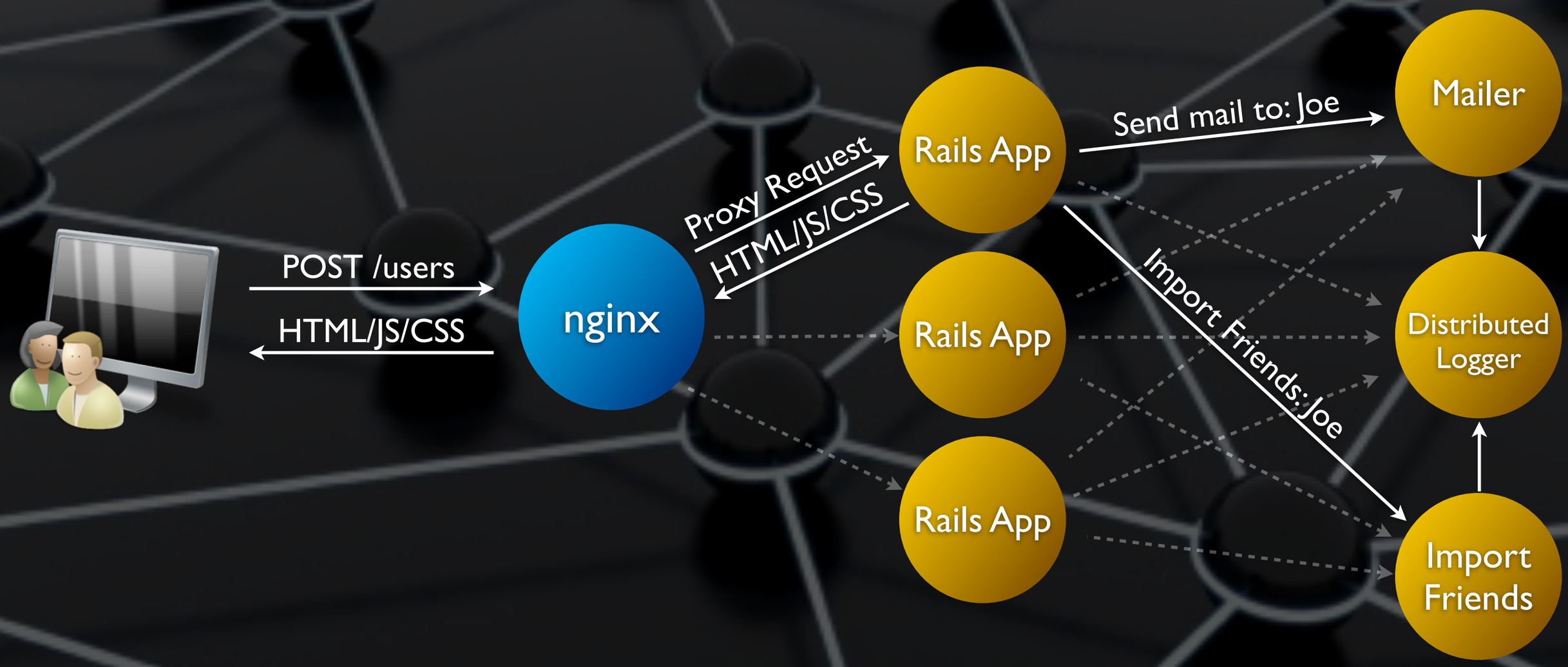
Porque?

- Performance
- Escalabilidade
- Baixo acoplamento entre sub-sistemas
- Facilidade de manutenção

Em Ruby...

- **Programação Distribuída**
  - DRb (Distributed Ruby)
  - Rinda (Linda implementation in Ruby, on top of DRb)
  - REST / SOAP
- **Message Queues**
  - Starling (Pure-ruby message-queue)
  - Redis/Resque
- **MapReduce**
  - Skynet

Programação Distribuída



Programação Distribuída

- Processamento síncrono
- Não escala transparentemente
- Cuidados com concorrência

DRb (Distributed Ruby)

DRb é um sistema de objetos distribuído. Permite um objeto em um processo Ruby invocar um método em outro objeto em outro processo Ruby, na mesma maquina ou em uma diferente.

- Criação fácil de serviços
- Chamada remota de métodos em objetos
- Compartilhamento distribuído de objetos

DistributedLogger (Usando DRb)

log_service.rb

```ruby
require "drb"
require "logger"

DRb.start_service("druby://127.0.0.1:61676", Logger.new(STDOUT))
puts "Distributed logger service started"
DRb.thread.join

```

log_consumer.rb

```ruby
require "drb"

logger = DRbObject.new_with_uri("druby://127.0.0.1:61676")

logger.info  "Client started"
logger.debug "I think I made something wrong, let's investigate"
logger.warn  "Oh yeah, watch out!"
logger.error "Something is definitely wrong!!"
logger.fatal "OMG! RUUUNNN!!"

```

```
Distributed logger service started
I, [2010-08-19T20:39:44.249443 #16162]  INFO -- : Client started
D, [2010-08-19T20:39:44.250495 #16162] DEBUG -- : I think I made something wrong, let's investigate
W, [2010-08-19T20:39:44.250916 #16162]  WARN -- : Oh yeah, watch out!
E, [2010-08-19T20:39:44.251358 #16162] ERROR -- : Something is definitely wrong!!
F, [2010-08-19T20:39:44.251794 #16162] FATAL -- : OMG! RUUUNNN!!
```

Object Reference (Using DRb)

service.rb

```ruby
require "drb"
require "date"

class User
  include DRbUndumped
  attr_reader :name, :birthdate

  def initialize(name, birthdate)
    @name, @birthdate = name, birthdate
  end
end

class Date; include DRbUndumped; end

class Database
  include DRbUndumped

  def initialize(users)
    @users = users
  end

  def each(&block)
    @users.each(&block)
  end
end

db = Database.new([
  User.new("Rodrigo", Date.civil(1985, 10, 31)),
  User.new("John", Date.civil(1980, 1, 22)),
  User.new("Mary", Date.civil(1990, 5, 4))
])

DRb.start_service("druby://127.0.0.1:61676", db)
DRb.thread.join
```

consumer.rb

```ruby
require "drb"

DRb.start_service
db = DRbObject.new_with_uri("druby://127.0.0.1:61676")

db.each do |user|
  p user
  puts "Name: #{user.name}, Birthdate: #{user.birthdate}"
end
```

```
#<DRb::DRbObject:0x00000100844b30 @uri="druby://127.0.0.1:61676", @ref=2152207344>
Name: Rodrigo, Birthdate: 1985-10-31
#<DRb::DRbObject:0x00000100840878 @uri="druby://127.0.0.1:61676", @ref=2152206644>
Name: John, Birthdate: 1980-01-22
#<DRb::DRbObject:0x0000010083cb38 @uri="druby://127.0.0.1:61676", @ref=2152205804>
Name: Mary, Birthdate: 1990-05-04
```

Rinda (Linda paradigm)

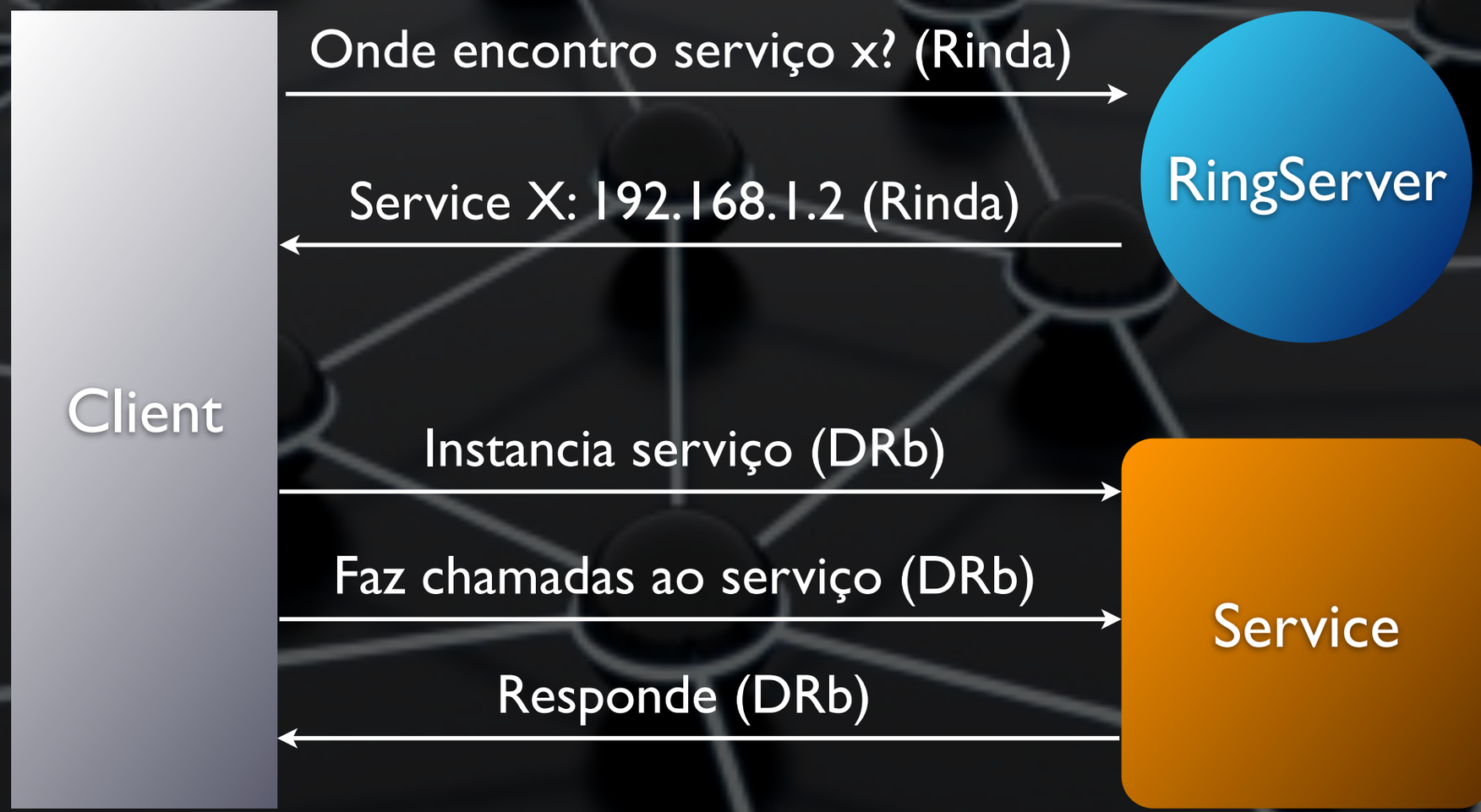
Sistema de coordenação para descobrimento de serviços distribuídos.

- Construído em cima de DRb
- Divulgação de um serviço para outros nodos
- Descobrimento de serviços distribuídos
- Melhora escalabilidade
- TupleSpaces / Tuples

Rinda (Linda paradigm)

TupleSpace

Rinda (Linda paradigm)



Rinda (Linda paradigm)

log_service.rb

```ruby
require "rinda/ring"
require "logger"

DRb.start_service
ring_server = Rinda::RingFinger.primary

ring_server.write([
  :primary_logger,
  :Logger,
  Logger.new(STDOUT),
  "The primary logger to be used by all systems"
], Rinda::SimpleRenewer.new)

puts "Distributed logger service started"
DRb.thread.join
```

log_consumer.rb

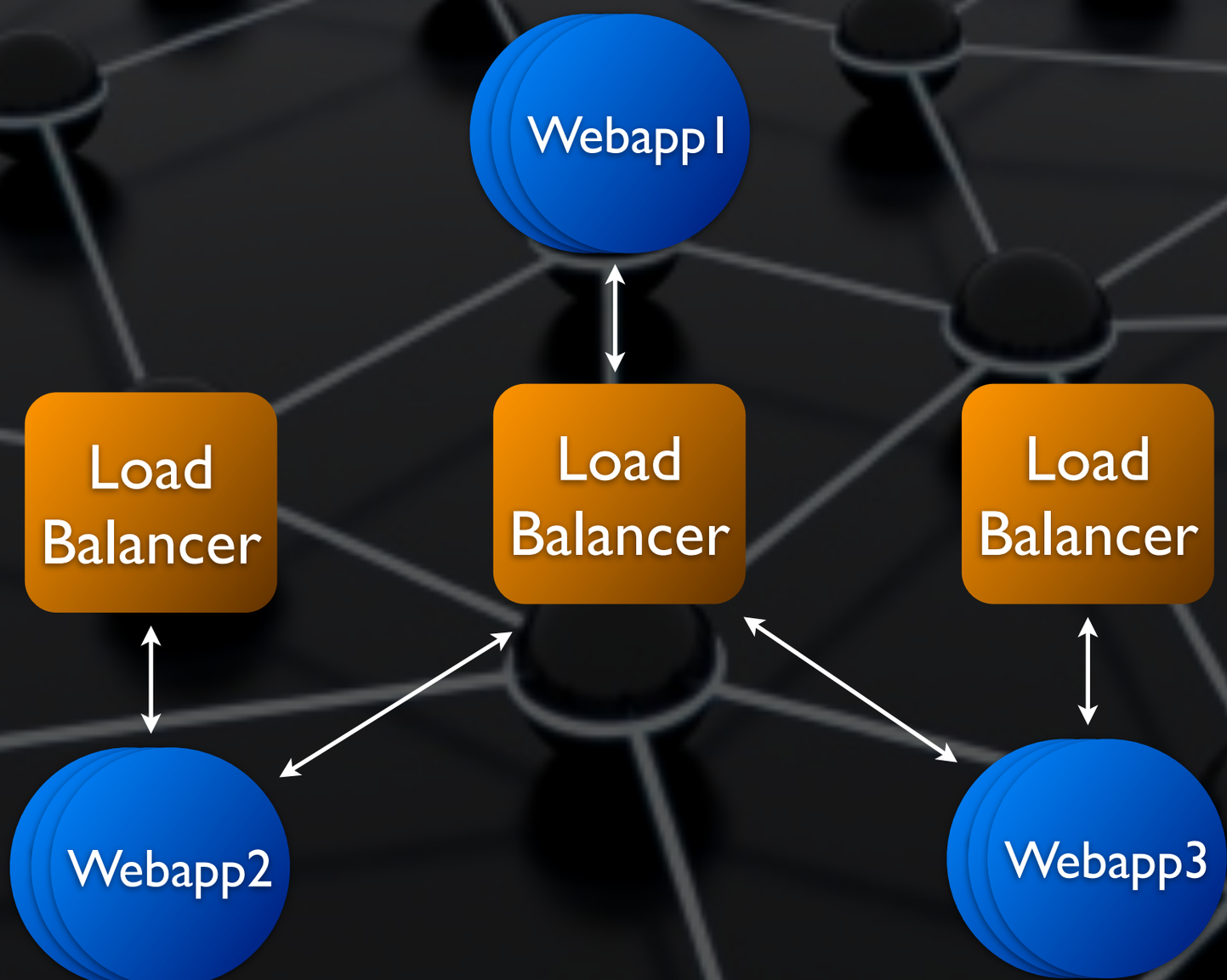
```ruby
require "rinda/ring"

DRb.start_service
ring_server = Rinda::RingFinger.primary

service = ring_server.read([:primary_logger, nil, nil, nil])
logger  = service[2]

logger.info  "Client started"
logger.debug "I think I made something wrong, let's investigate"
logger.warn  "Oh yeah, watch out!"
logger.error "Something is definitely wrong!!"
logger.fatal "OMG! RUUUNNN!!"
```

```
Distributed logger service started
I, [2010-08-19T22:40:38.512909 #16754]  INFO -- : Client started
D, [2010-08-19T22:40:38.513611 #16754] DEBUG -- : I think I made something wrong, let's investigate
W, [2010-08-19T22:40:38.514047 #16754]  WARN -- : Oh yeah, watch out!
E, [2010-08-19T22:40:38.514493 #16754] ERROR -- : Something is definitely wrong!!
F, [2010-08-19T22:40:38.514902 #16754] FATAL -- : OMG! RUUUNNN!!
```

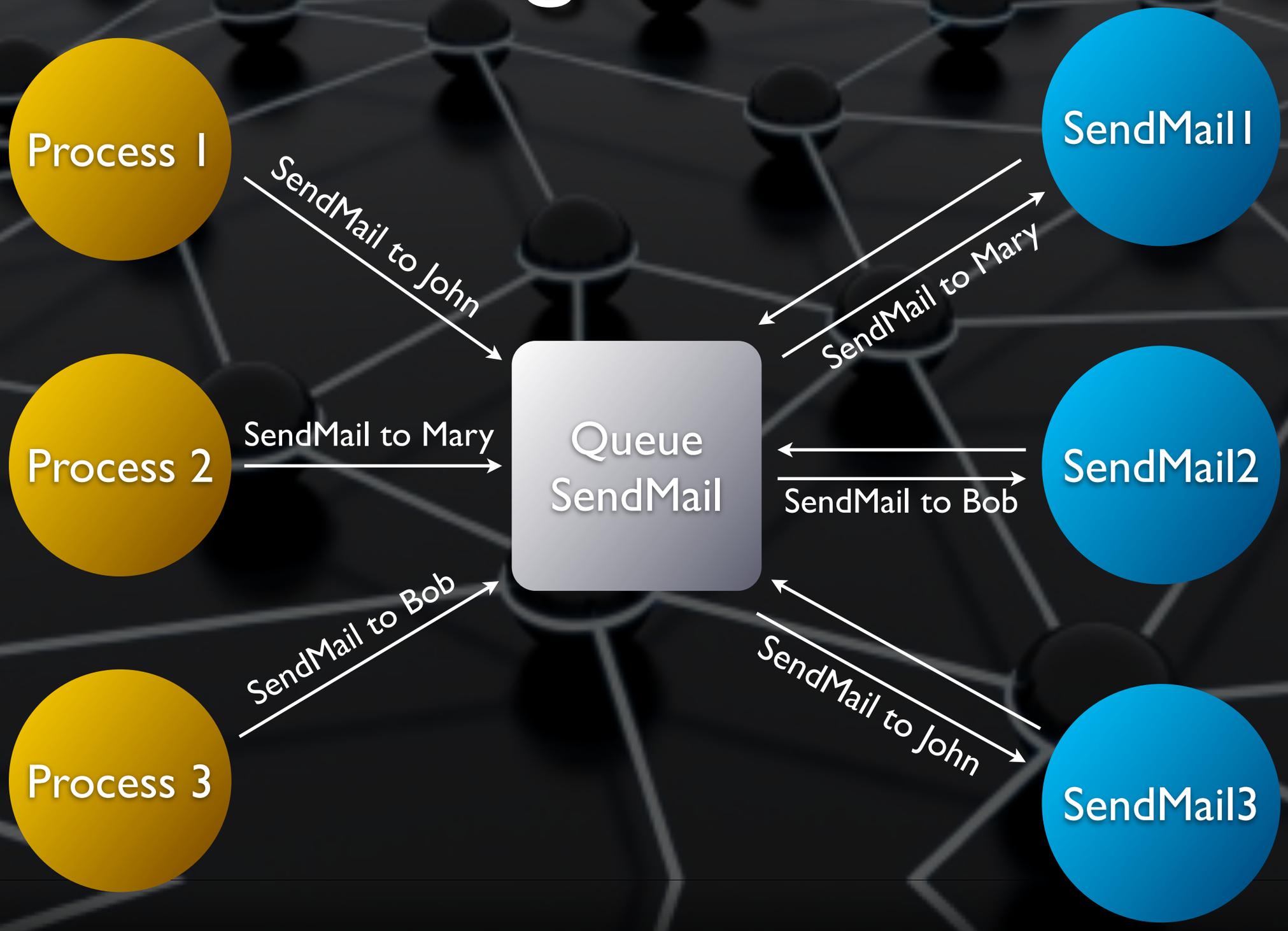
REST :) / SOAP :(

- Diversas aplicações web que se comunicam
- Chamada de “métodos” remotos usando HTTP
- Aplicações com responsabilidades bem definidas
- Facil de escalar usando um HTTP load balancer

REST :) / SOAP :(


Webapp1
Load Balancer
Load Balancer
Load Balancer
Webapp2
Webapp3

Message Queues



Message Queue

- Processamento assíncrono
- Fácil de escalar
- Simplifica concorrência

Starling

- Message queue (Fila de mensagens)
- Pure Ruby
- Simples de instalar e usar
- Protocolo MemCache
- Desenvolvido pelo Twitter

Starling

worker.rb

```ruby
require "rubygems"
require "starling"
require "logger"

starling = Starling.new("127.0.0.1:22222")
logger   = Logger.new(STDOUT)

loop do
  level, string = starling.get('log_queue')
  logger.send(level, string)
end

```

process.rb

```ruby
require "rubygems"
require "starling"

starling = Starling.new("127.0.0.1:22222")

starling.set("log_queue", [:info,  "Client started"])
starling.set("log_queue", [:debug, "I think something is wrong"])
starling.set("log_queue", [:warn,  "Oh yeah, watch out!"])
starling.set("log_queue", [:error, "Something is definitely wrong!!"])
starling.set("log_queue", [:fatal, "OMG! RUUUNNN!!"])
```

```
I, [2010-08-19T23:15:56.915671 #17360]  INFO -- : Client started
D, [2010-08-19T23:15:56.916331 #17360] DEBUG -- : I think I made something wrong, let's investigate
W, [2010-08-19T23:15:56.916753 #17360]  WARN -- : Oh yeah, watch out!
E, [2010-08-19T23:15:56.917166 #17360] ERROR -- : Something is definitely wrong!!
F, [2010-08-19T23:15:56.917584 #17360] FATAL -- : OMG! RUUUNNN!!
```

Redis

Redis is an advanced key-value store. It is similar to memcached but the dataset is not volatile, and values can be strings, exactly like in memcached, **but also lists, sets, and ordered sets**. All this data types can be manipulated with atomic operations to push/pop elements, add/remove elements, perform server side union, intersection, difference between sets, and so forth. Redis supports different kind of sorting abilities.

- Data Structure Server
- Suporta lista e sets
- Super Fast (110000 set/sec, 81000 get/sec)
- Dataset não é volátil / Failure recovery
- Operações atomicas sobre valores
- Replicação
- Escrito em ANSI C, suporta todos OS POSIX (*nix, *bsd, osx)
- Lib disponível para várias linguagens, incluindo Ruby

Redis

- GitHub
- Engine Yard
- The Guardian
- Craigslist

Redis

worker.rb

```ruby
require "rubygems"
require "redis"
require "logger"

redis  = Redis.new(:host => "127.0.0.1", :port => 6379)
logger = Logger.new(STDOUT)

loop do
  _, data = redis.brpop('log_queue', 0)
  level, string = Marshal.load(data)
  logger.send(level, string)
end
```

process.rb

```ruby
require "rubygems"
require "redis"

redis  = Redis.new(:host => "127.0.0.1", :port => 6379)

redis.lpush("log_queue", Marshal.dump([:info,
  "Client started"]))
redis.lpush("log_queue", Marshal.dump([:debug,
  "I think something is wrong"]))
redis.lpush("log_queue", Marshal.dump([:warn,
  "Oh yeah, watch out!"]))
redis.lpush("log_queue", Marshal.dump([:error,
  "Something is definitely wrong!!"]))
redis.lpush("log_queue", Marshal.dump([:fatal,
  "OMG! RUUUNNN!!"]))
```

```
I, [2010-08-21T08:49:16.576740 #20917]  INFO -- : Client started
D, [2010-08-21T08:49:16.577086 #20917] DEBUG -- : I think something is wrong
W, [2010-08-21T08:49:16.577352 #20917]  WARN -- : Oh yeah, watch out!
E, [2010-08-21T08:49:16.577566 #20917] ERROR -- : Something is definitely wrong!!
F, [2010-08-21T08:49:16.577805 #20917] FATAL -- : OMG! RUUUNNN!!
```

Resque

Resque is a Redis-backed library for creating background jobs, placing those jobs on multiple queues, and processing them later.

- Abstração de envio de mensagens
- Gerenciamento de *workers*
- Biblioteca para gerencia e consulta de *jobs*
- Sinatra app para monitoramento das filas, *workers* e *jobs*

Resque

- Workers são classes que respondem ao método perform
- Classe do worker é necessária em ambas partes
- Informações sobre workers e jobs são armazenadas Redis
- Master/slave fork para cada Job
- Chamada é armazenada na fila em formato JSON

Resque

```
class Archive
  @queue = :file_serve

  def self.perform(repo_id, branch = 'master')
    repo = Repository.find(repo_id)
    repo.create_archive(branch)
  end
end

Resque.enqueue(Archive, 10, “master”)
```

↓

```
resque:queue:log << {'class': 'Archive', 'args': [ 10, 'master' ]}
```

Resque

worker.rb

```ruby
require "rubygems"
require "resque"
require "singleton"
require "logger"

class LogService
  include Singleton

  @queue = :log

  def self.perform(level, string)
    self.instance.log(level, string)
  end

  def initialize
    @logger = Logger.new(STDOUT)
  end

  def log(level, string)
    @logger.send(level, string)
  end
end
```

process.rb

```ruby
require "rubygems"
require "resque"
require "worker"

Resque.enqueue(LogService, "info",  "Client started")
Resque.enqueue(LogService, "debug", "I think something is wrong")
Resque.enqueue(LogService, "warn",  "Oh yeah, watch out!")
Resque.enqueue(LogService, "error", "Something is definitely wrong!!")
Resque.enqueue(LogService, "fatal", "OMG! RUUUNNN!!")
```

```
divoxx@wizard distributed_log : QUEUE=log rake resque:work
(in /Users/divoxx/Development/Projects/OpenSource/DistribAsyncRuby/examples/resque/distributed_log)
I, [2010-08-21T09:44:56.933123 #21327]  INFO -- : Client started
D, [2010-08-21T09:44:56.939552 #21328] DEBUG -- : I think something is wrong
W, [2010-08-21T09:44:56.944662 #21329]  WARN -- : Oh yeah, watch out!
E, [2010-08-21T09:44:56.949360 #21330] ERROR -- : Something is definitely wrong!!
F, [2010-08-21T09:44:56.953370 #21331] FATAL -- : OMG! RUUUNNN!!
```

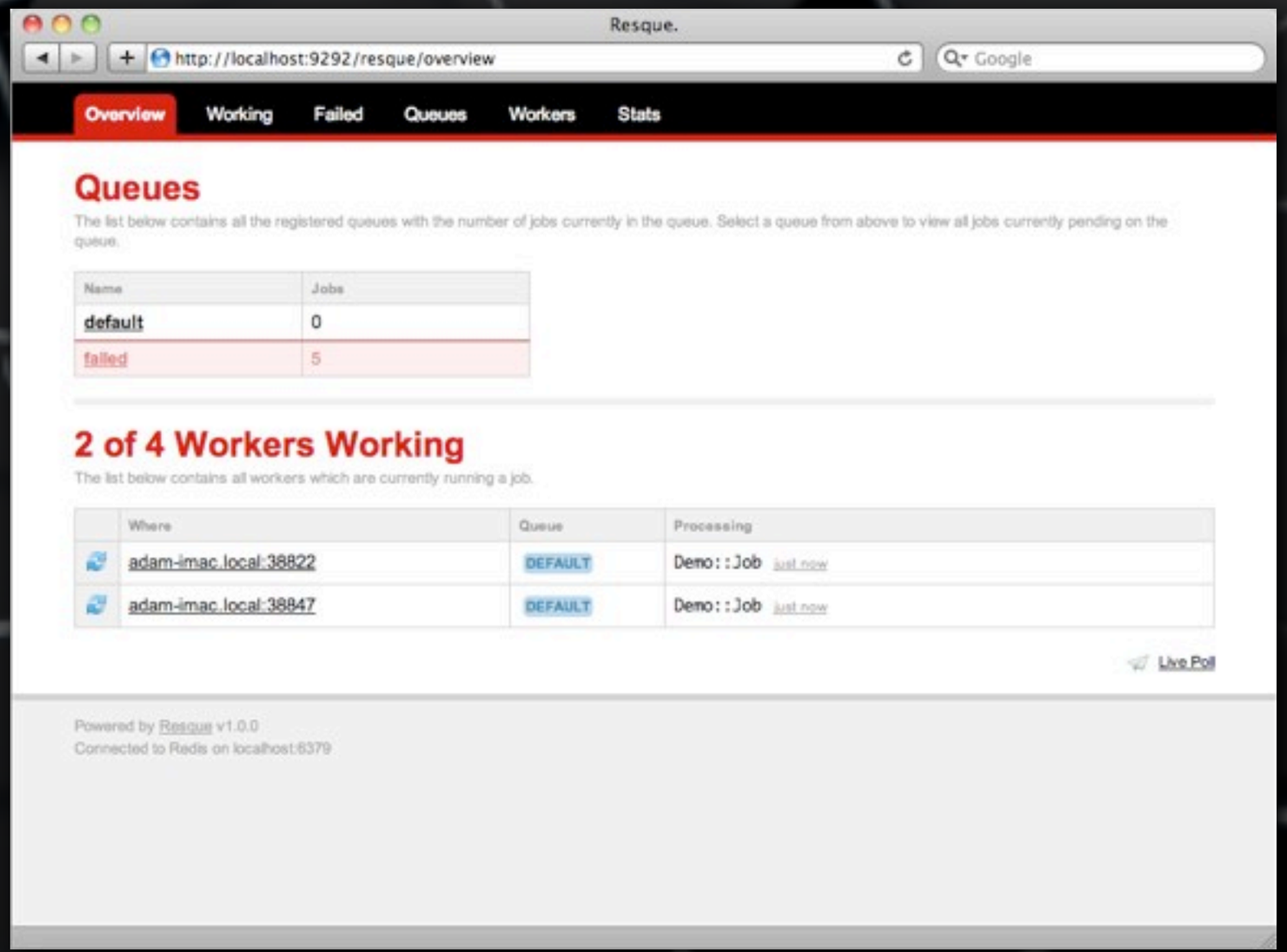
Resque

Processamento assíncrono GitHub:

Warming caches
Counting disk usage
Building tarballs
Building Rubygems
Firing off web hooks
Creating events in the db and pre-caching them
Building graphs
Deleting users
Updating our search index

Resque

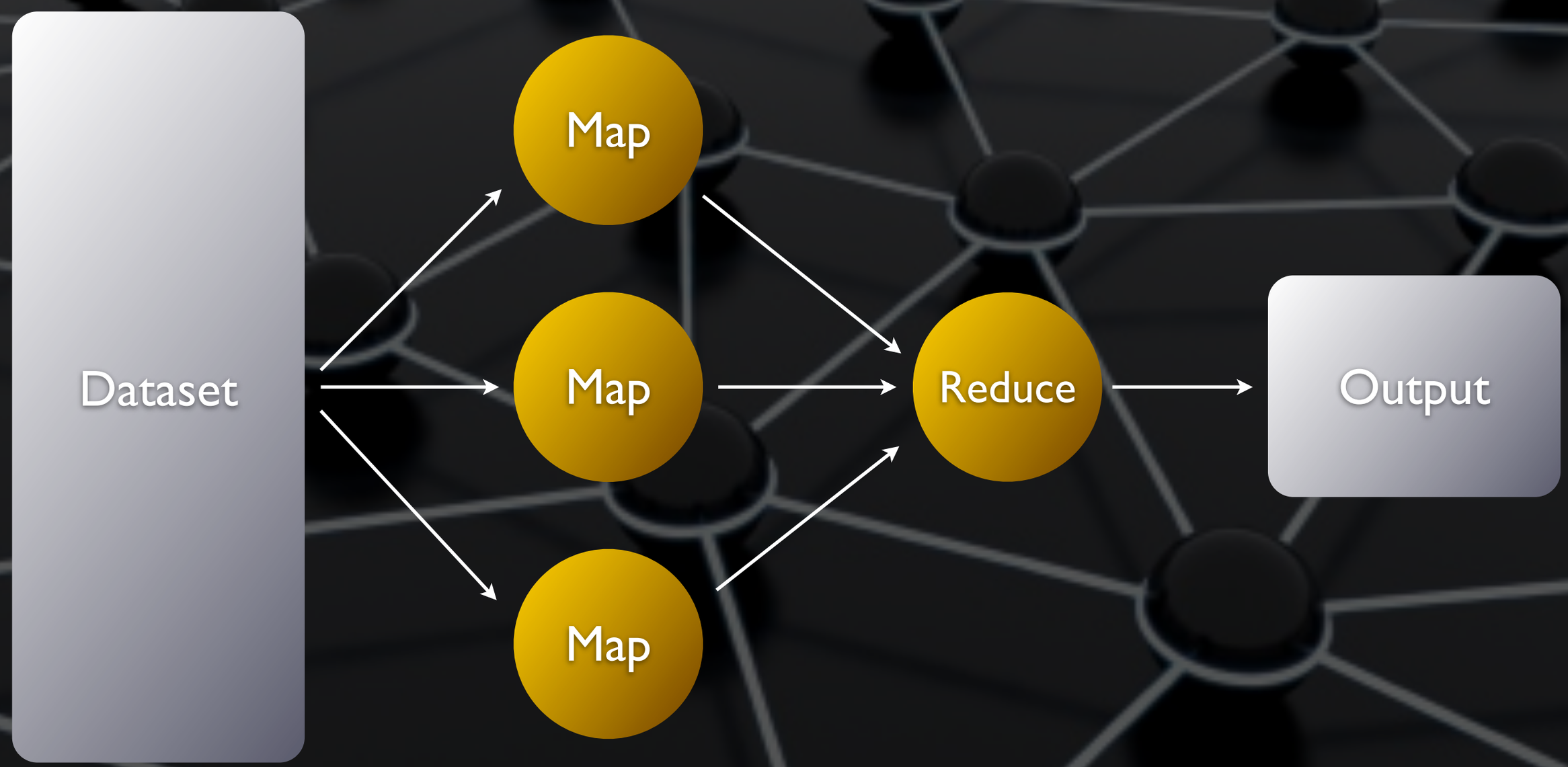
$ resque-web



Map Reduce

- Criado pela Google
- Computação distribuída on datasets grandes
- Altalmente escalável
- Rápido
- Complexidade de implementação

Skynet

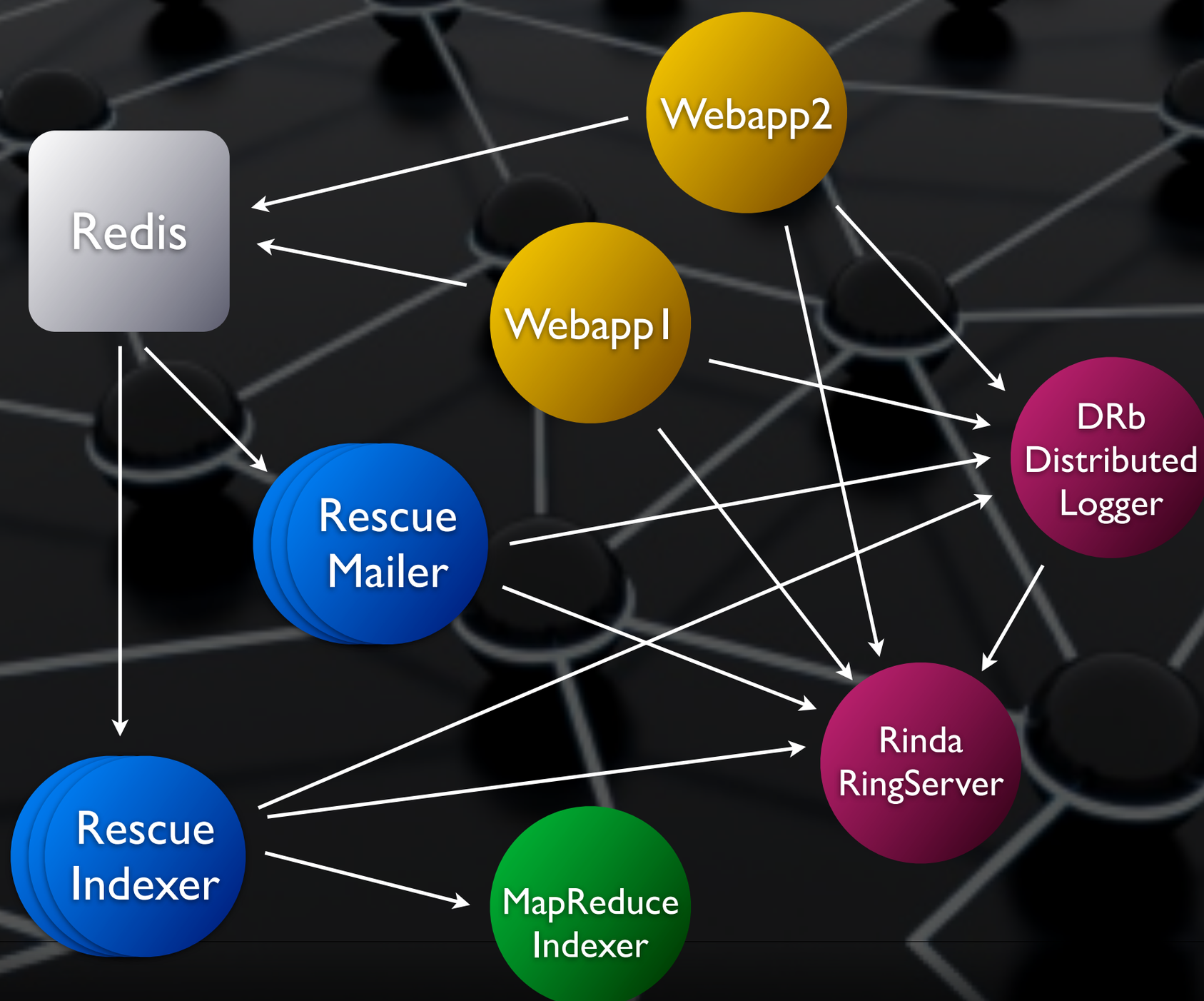


Skynet

```ruby
class WordCounter
  def self.map(lines)
    counts = []
    lines.each do |line|
      line.scan(/\w+/).each do |word|
        counts << [word.downcase.strip, 1]
      end
    end
    counts
  end

  def self.reduce(counts)
    totals = Hash.new(0)
    counts.each do |word, count|
      totals[word] += count
    end
    totals
  end
end
```

```ruby
require "rubygems"
require "skynet"
require "word_counter"
require "pp"

job = Skynet::Job.new(
  :mappers          => 8,
  :reducers         => 1,
  :map_reduce_class => WordCounter,
  :map_data         => File.readlines("text.txt")
)
pp job.run
```

Juntando Tudo



Other frameworks

RingyDingy
Starfish
Distribunaut
Politics
AMQP/RabbitMQ
BackgroundDRb
DelayedJob
.....

Perguntas?

Rodrigo Kochenburger <divoxx@gmail.com>
http://github.com/divoxx/distributed_talk