

FrontLoader

distributed load testing



What is it?

FrontLoader is a flexible distributed load tester with support for system monitoring and a full web interface. Originally designed for testing RTMP services, it is particularly suited for protocols that require long-lived connections. However, short-lived protocols like HTTP can be tested easily as well.

Features

- Includes a powerful monitoring framework that lets you collect real-time information from a remote server (e.g. CPU usage, memory usage) or even an entire cluster.
- An extensible connection system that makes testing almost any kind of service possible
- Configure and run your tests using the web interface. Then watch the results in real-time.
- Statistical analysis of the resulting data lets you see what factors affect your cluster's performance.

Scenarios

```
monitor do
  local          # use local monitoring (the default)
  role "web"     # monitor hosts in the web role using these settings
  role "mysql"  # ditto for mysql role
  host "12.42.44.12" # add a single host to monitor

  watch :cpu => 2 # run the CPU watch with priority 2
  watch :mem => 1
  watch :socket => 0.5
end

test :time => 120, :repeat => 1 do
  urls = ["http://site.com/page1", "http://site.com/page2"]
  500.times{|_|
    start :HttpConnector, :url => urls[rand % url.size]
    wait rand
  }
  wait_until_done
end
```

Watchers

```
class SocketWatcher < FrontLoader::Watcher
  name "socket"

  shell "ls -l /proc/*/fd | grep socket | wc -l", :sudo => true do |count|
    log :sockets => count.to_i unless count.to_i == 0
  end
end
```

Connectors

```
require 'em-http'

class HTTPConnector < FrontLoader::Connector
  option :url, :type => :text, :help => "The URL that will be
  accessed"

  def start opt
    @connections ||= []
    @counter ||= 0
    counter = (@counter += 1)
    http = EM::HttpRequest.new(opt[:url]).get
    log_started :id => counter
    http.callback do
      log_connected :id => counter
      log_disconnected :id => counter
      @connections.delete http
    end
    http.errback do
      log_failed :id => counter
    end
    @connections << http
  end

  def stop
    @connections.each{|c| c.close_connection} if @connections
  end
end
```

run

watch

analyze

advanced editor

scenarios

devices

new

general

name

time

max load

rate

repeat

monitoring

roles

hosts

connections

connector

url

run

run

watch

analyze

scenarios

devices

new

```
# Define your scenario here using the scenario DSL
```

```
monitor do
```

```
  host 'roomtrol-allb004.class'
```

```
  host 'roomtrol-allb103.class'
```

```
  host 'roomtrol-allb003.class'
```

```
  host 'roomtrol-allb017.class'
```

```
end
```

```
test :time => 120, :repeat => 1 do
```

```
  load :FrontLoader::HTTPConnector, :load => 10, :rate => 3, :url => "http://google.com"
```

```
end
```

name

devices

run

run

watch

analyze

scenario

test number

connections

avg length

failures

latency

time left

devices

1/1

0

0.01 s

0.01 s

00:01:05

stop

role

hosts

streams

cpu

load

mem

sockets

roomtrol-allb103.class

1

1.68%

0.86

149.21 MB

160.00

roomtrol-allb004.class

1

4.87%

0.54

536.66 MB

114.00

roomtrol-allb113.class

1

12.44%

0.91

94.59 MB

151.00

roomtrol-allb017.class

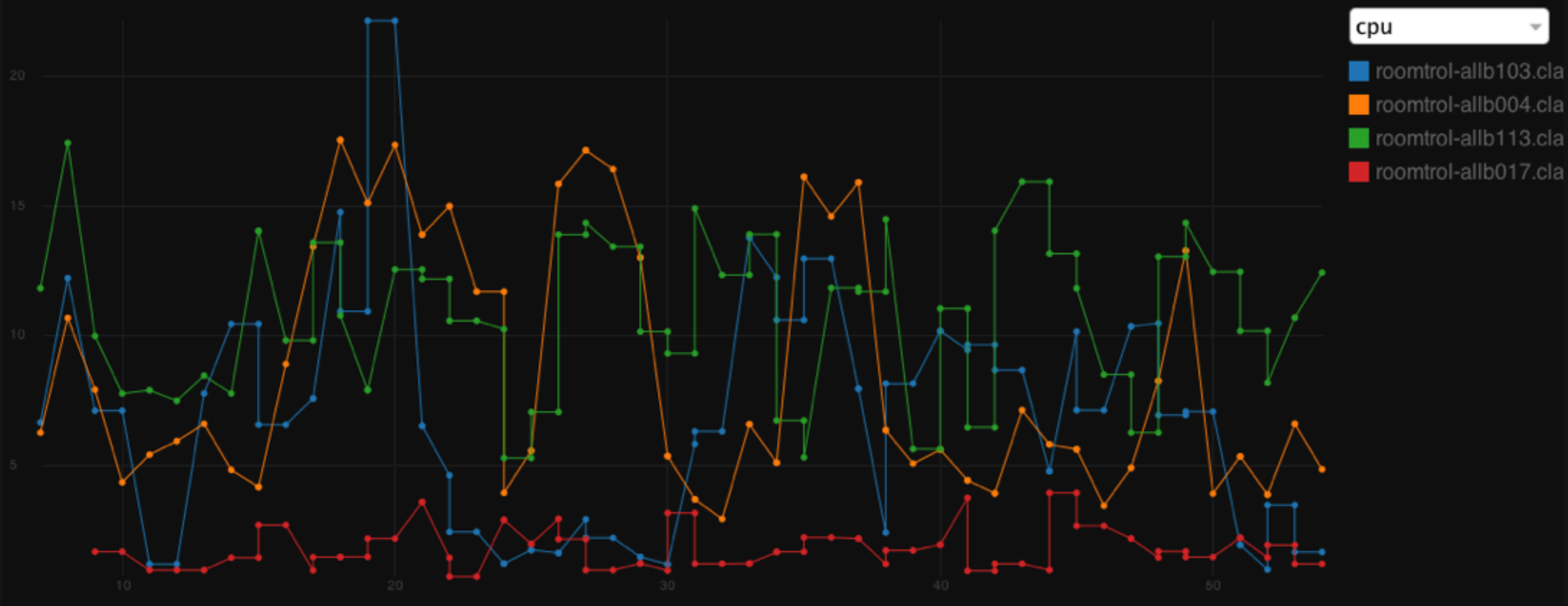
1

1.23%

0.37

58.54 MB

140.00



run

watch

analyze

10/10/2011 0:00

time

cpu:user

linear

all

JSON

CSV

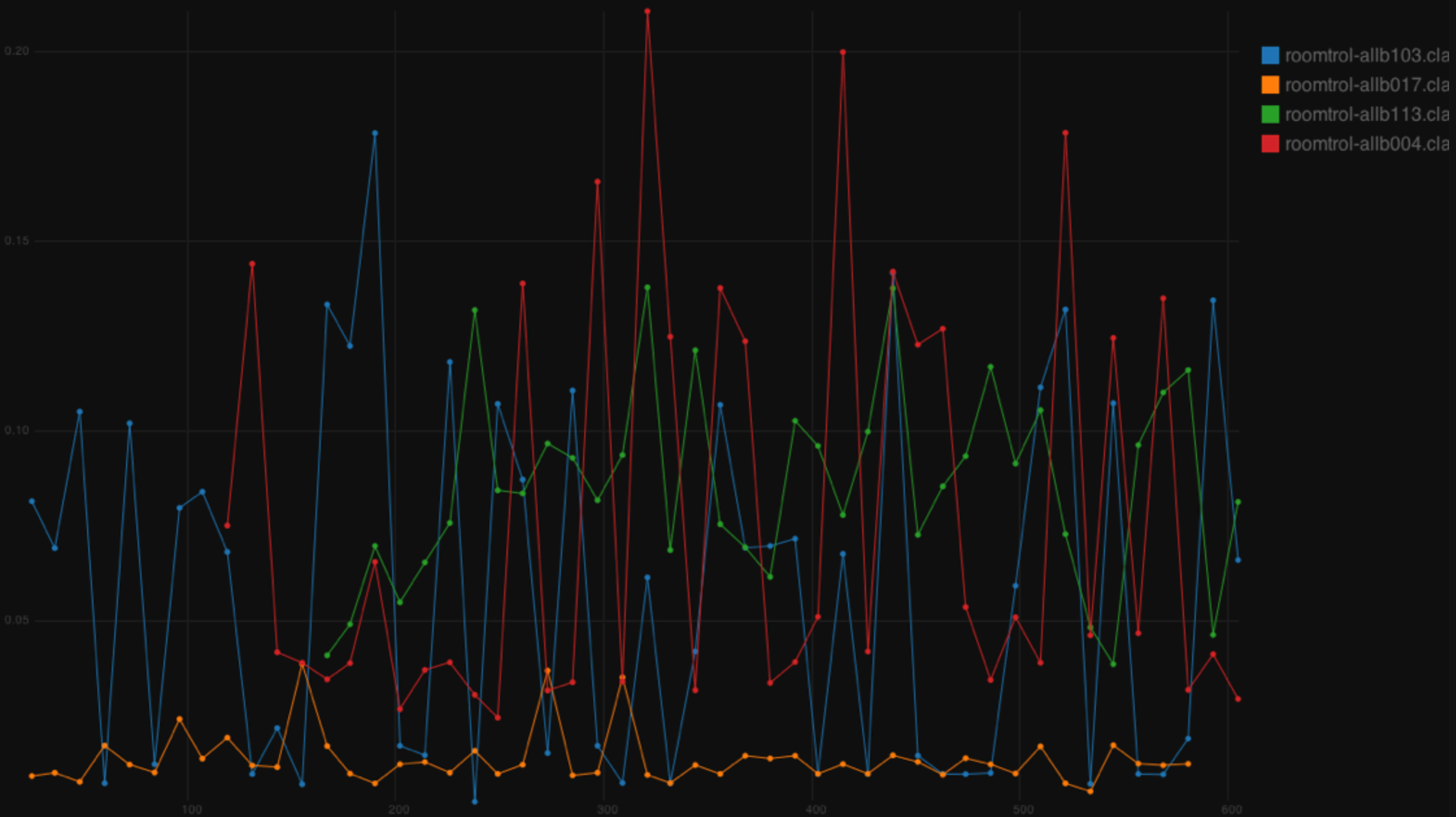
run

explanatory (x)

dependent (y)

model

scope



demo

thanks to



for sponsoring this work