



## Нагрузочное тестирование с помощью Grinder



Илья Евлампиев  
“Exigen Services”

# Нагрузочное тестирование...

определение того, как быстро работает система при определенной нагрузке. Позволяет также определить такие параметры системы как расширяемость и надежность.

Вкратце, это:

- Имитация реальной нагрузки на систему
- Выполнение тестов
- Анализ результатов

# Цели нагрузочного тестирования

- Недопущение «падения» системы
- Обойтись без излишних затрат на «железо»
- Снизить стоимость продукта
- Облегчить обслуживание
- Продемонстрировать заказчику, что система удовлетворяет требованиям

# Стратегии нагрузочного тестирования

- Поиск «узкого горла»
- Оптимизация
- Сравнение альтернативных вариантов
- Определение максимального объема данных
- Чистый интерес

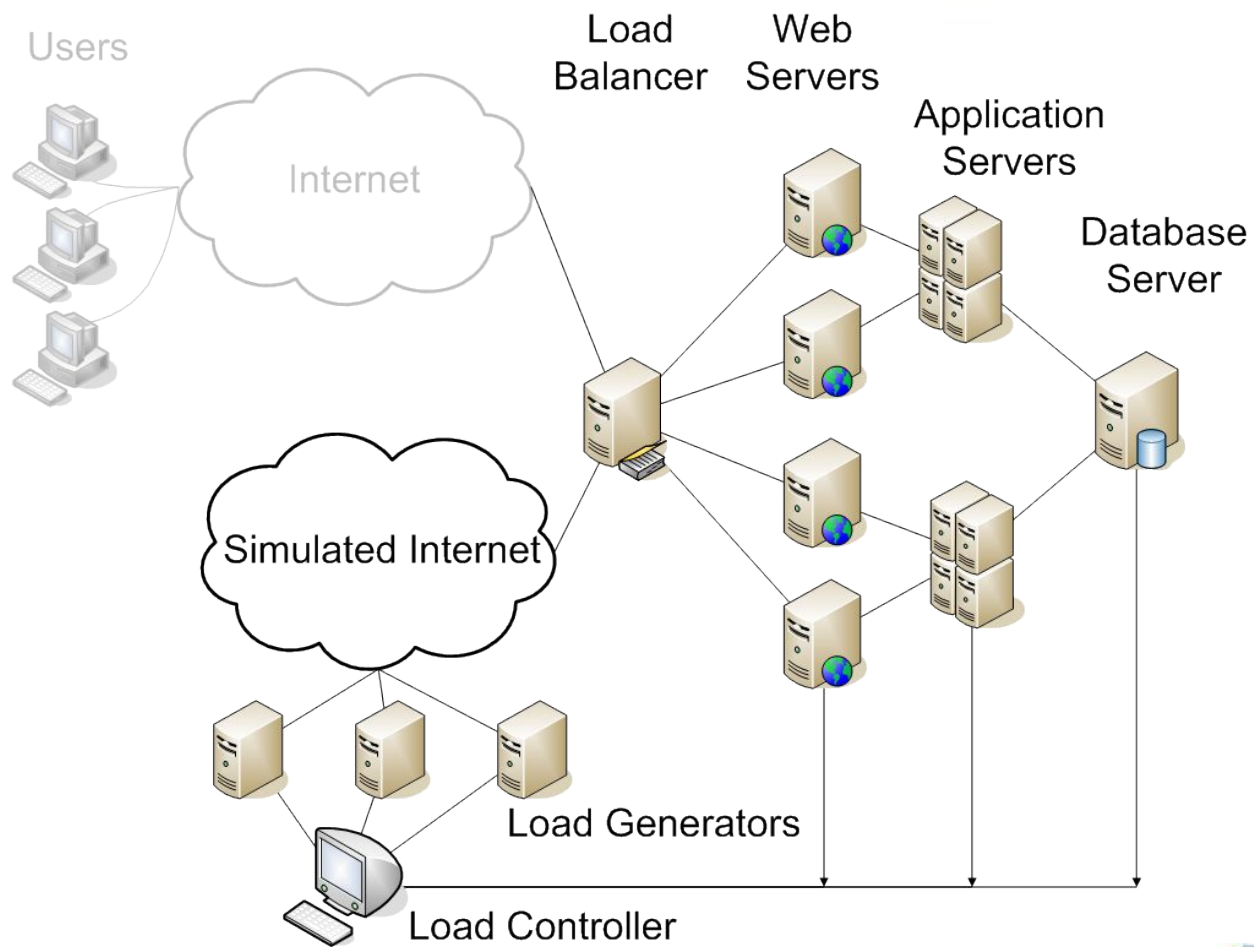
# Типы нагрузочного тестирования

- Нагрузочное тестирование (load testing)
- Тестирование отказоустойчивости (failover testing)
- Стресс-тестирование (stress testing)
- Целевые испытания (targeted infrastructure test)
- Объемное тестирование (volume testing)
- Тестирование стабильности (endurance testing)
- Тестирование производительности (performance testing)
- Тестирование пропускной способности сети (network sensitivity testing)

# Словарь

- Vuser - виртуальный пользователь
- Load generator (agent) - генератор нагрузки
- Process - процесс
- Thread - поток
- Run - прогон
- Ramp-up - «разброс»
- Load controller - контроллер нагрузки

# Схема



# Метрики

- Времена отклика +
- Построение графиков производительности +
- Пропускная способность +
- Надежность (MTBF - Mean Time Between Failures)
- Доступность (какой процент времени сервис лежал)
- Загрузка процессора
- Загрузка памяти
- Сетевой трафик +
- Особые показатели сервера приложений
- Особые показатели сервера баз данных
- Запросы в секунду +



# Что нужно знать о каждой метрике?

- Среднее значение
- Разброс
- Максимальное и минимальное достигнутое значение
- Перцентиль (уложившиеся около среднего 90% результатов)
- Все распределение
- Упавшие запросы (процент)

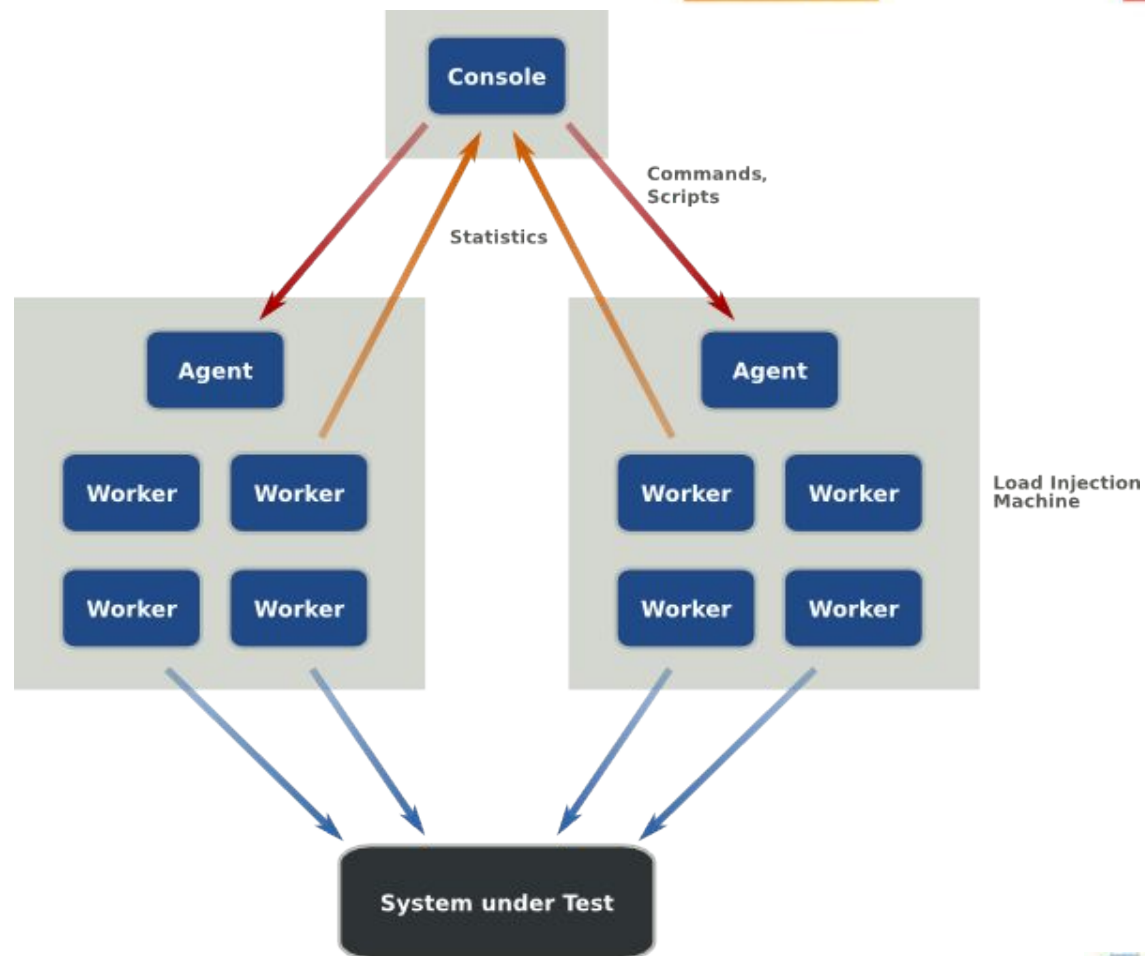
# The Grinder

- ❑ 100% Java (любая ОС с поддержкой J2SE)
- ❑ Протоколы
- ❑ HTTP, HTTPS (out of the box)
- ❑ SOAP, XML-RPC
- ❑ IIOP, RMI/IIOP, RMI/JRMP, and JMS.
- ❑ POP3, SMTP, FTP, and LDAP.
- ❑ Базы данных - JDBC
- ❑ Скрипт на Jython / Автозапись HTTP
- ❑ GrinderStone for Eclipse - дебаггинг
- ❑ Бесплатный 😊

# The Grinder: Настройка среды

- Java (JVM)
- Jython (add to PATH)
- Add external JARs to Grinder /lib/

# Схема нагрузочного тестирования



# The Grider: Property File

The screenshot displays the 'The Grinder Console' application window. The interface includes a menu bar with 'File', 'Action', and 'Distribute', and a toolbar with icons for file operations. On the left, there are controls for 'Sample interval: 1000 ms' with a slider, 'Ignore 0 samples' and 'Collect samples forever' both set to 0, and a status indicator 'Collecting samples: 52083'. A large display shows '0.00 TPS'. Below this, a 'Total' section lists statistics: 25.3 ms (mean), 0.154 TPS (mean), 969 TPS (peak), 8000 tests, and 0 errors. The main area is divided into 'Graphs', 'Results', 'Processes', and 'Script' tabs. The 'Script' tab is active, showing a list of files in a tree view on the left and a text editor on the right. The text editor contains the following properties:

```
grinder.processes 1
grinder.threads 100
grinder.runs 10
grinder.sleepTimeVariation 0.9
grinder.initialSleepTime 1000
grinder.numberOfOldLogs 0
grinder.script scripts/jdbc.py
```

# The Grider: Property File

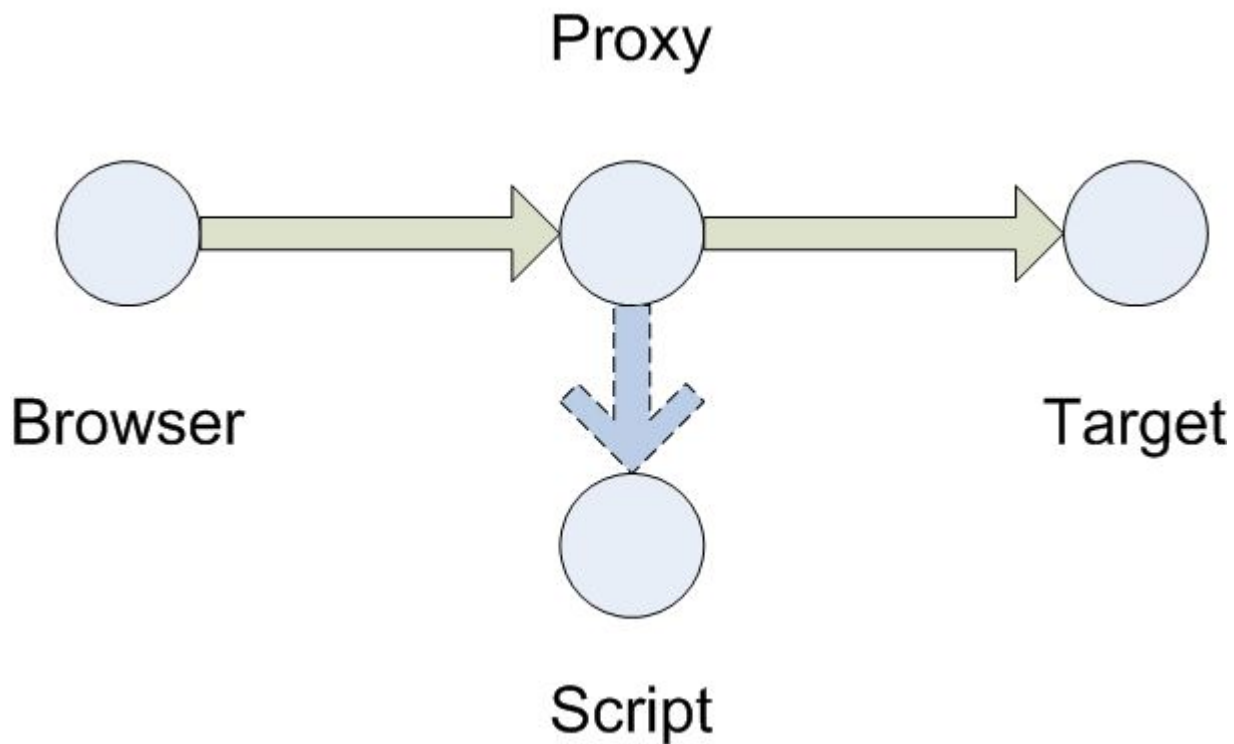
- ❑ grinder.processes
- ❑ grinder.threads
- ❑ grinder.runs
- ❑ grinder.processIncrement
- ❑ grinder.processIncrementInterval
- ❑ grinder.initialProcesses
- ❑ grinder.duration
- ❑ grinder.script
- ❑ grinder.initialSleepTime
- ❑ grinder.sleepTimeVariation
- ❑ grinder.sleepTimeFactor

# The Grinder: UI

The screenshot displays the 'The Grinder Console' application window. The interface includes a menu bar with 'File', 'Action', and 'Distribute' options, and a toolbar with icons for file operations. The main area is divided into several sections:

- Configuration:** 'Sample interval: 1000 ms' with a slider, 'Ignore 0 samples' and 'Collect samples forever' with input fields.
- Status:** 'Collecting samples: 52155' with a document icon.
- Summary:** A large display showing '0.00 TPS'.
- Total:** Summary statistics for the entire run: 27.4 ms (mean), 0.230 TPS (mean), 2430 TPS (peak), 12000 tests, 0 errors.
- Test Results:** Two test-specific sections, each with a histogram and summary statistics:
  - Test 1 (Just test1):** 26.7 ms (mean), 0.115 TPS (mean), 1220 TPS (peak), 6000 tests, 0 errors.
  - Test 2 (Database query):** 28.1 ms (mean), 0.115 TPS (mean), 1210 TPS (peak), 6000 tests, 0 errors.

# Workflow: Запись скрипта





# Пример скрипта

```
c:\Work\Tools\Grinder\grinder-3.1\scripts\google.py - Notepad++
Файл Правка Поиск Вид Кодировки Стиль Опции Макросы Запуск TextFX Дополнения Окна ?
google.py
1 # The Grinder 3.1
2 # HTTP script recorded by TCPProxy at Mar 26, 2009 4:08:58 PM
3
4 from net.grinder.script import Test
5 from net.grinder.script.Grinder import grinder
6 from net.grinder.plugin.http import HTTPPluginControl, HTTPRequest
7 from HTTPClient import NVPair
8 connectionDefaults = HTTPPluginControl.getConnectionDefaults()
9 httpUtilities = HTTPPluginControl.getHTTPUtilities()
10
11 # To use a proxy server, uncomment the next line and set the host and port.
12 # connectionDefaults.setProxyServer("localhost", 8001)
13
14 # These definitions at the top level of the file are evaluated once,
15 # when the worker process is started.
16
17 connectionDefaults.defaultHeaders = \
18 ( NVPair('Accept-Language', 'ru,en-us;q=0.7,en;q=0.3'),
19   NVPair('Accept-Charset', 'windows-1251,utf-8;q=0.7,*;q=0.7'),
20   NVPair('Accept-Encoding', 'gzip,deflate'),
21   NVPair('User-Agent', 'Mozilla/5.0 (Windows; U; Windows NT 5.1; ru; rv:1.9.0.7) Gecko/2009021910 Firefox/3.0.7'),
22   NVPair('Cache-Control', 'no-cache'), )
23
24 headers0= \
25 ( NVPair('Accept', 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'), )
26
27 headers1= \
28 ( NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
29   NVPair('Referer', 'http://www.google.ru/'), )
30
```

Python file      nb char : 168117      Ln : 7 Col : 30 Sel : 0      Dos\Windows      ANSI      INS

# Jython

- “an implementation of the Python programming language written in Java”
- Python Syntax + ability to use Java classes
- Python написанный на Java, т.е. с возможностью использовать синтаксис Python и библиотеки Java

# Содержимое скрипта

- Импорт Java библиотек
- Определение переменных (HTTP headers/connection settings)
- Определение прокси-объектов для Test
- Группировка отдельных тестов в функции Test group
- Вызов каждой тестовой функции из главной функции `__call__(self)`

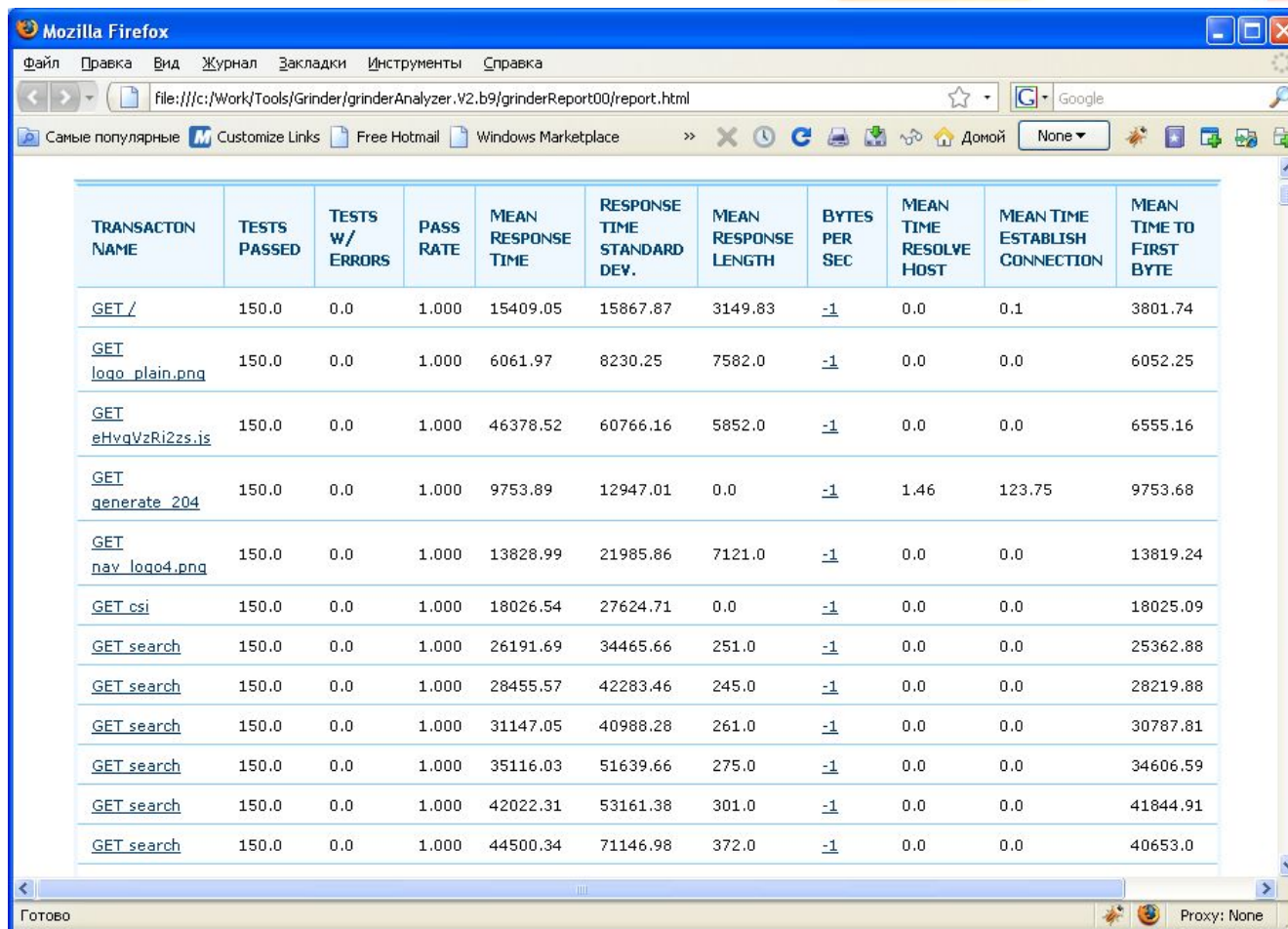
# Пример лог файла

```
Lister - [c:\Work\Tools\Grinder\grinder-3.1\data_iyevlampiev-0.log]
File Edit Options Help
Thread, Run, Test, Start time (ms since Epoch), Test time, Errors, HTTP response
code, HTTP response length, HTTP response errors, Time to resolve host, Time to
establish connection, Time to first byte
117, 0, 101, 1238081078220, 110, 0, 200, 3145, 0, 0, 0, 110
0, 0, 101, 1238081078220, 219, 0, 200, 3158, 0, 0, 0, 110
88, 0, 101, 1238081078392, 125, 0, 200, 3152, 0, 0, 0, 125
117, 0, 102, 1238081078392, 125, 0, 200, 7582, 0, 0, 0, 125
101, 0, 101, 1238081078423, 125, 0, 200, 3152, 0, 0, 0, 125
0, 0, 102, 1238081078470, 110, 0, 200, 7582, 0, 0, 0, 110
88, 0, 102, 1238081078548, 110, 0, 200, 7582, 0, 0, 0, 110
101, 0, 102, 1238081078595, 94, 0, 200, 7582, 0, 0, 0, 94
88, 0, 103, 1238081078658, 203, 0, 200, 5852, 0, 0, 0, 109
88, 0, 100, 1238081078392, 469, 0, 0, 0, 0, 0, 0, 0
139, 0, 101, 1238081078798, 94, 0, 200, 3147, 0, 0, 0, 94
101, 0, 103, 1238081078689, 234, 0, 200, 5852, 0, 0, 0, 125
101, 0, 100, 1238081078423, 500, 0, 0, 0, 0, 0, 0, 0
117, 0, 103, 1238081078517, 516, 0, 200, 5852, 0, 0, 0, 110
117, 0, 100, 1238081078220, 798, 0, 0, 0, 0, 0, 0, 0
139, 0, 102, 1238081078939, 94, 0, 200, 7582, 0, 0, 0, 94
43, 0, 101, 1238081079002, 93, 0, 200, 3152, 0, 0, 0, 93
0, 0, 103, 1238081078580, 531, 0, 200, 5852, 0, 0, 0, 125
0, 0, 100, 1238081078220, 891, 0, 0, 0, 0, 0, 0, 0
43, 0, 102, 1238081079127, 93, 0, 200, 7582, 0, 0, 0, 93
139, 0, 103, 1238081079033, 203, 0, 200, 5852, 0, 0, 0, 109
139, 0, 100, 1238081078798, 438, 0, 0, 0, 0, 0, 0, 0
69, 0, 101, 1238081079173, 110, 0, 200, 3150, 0, 0, 0, 110
79, 0, 101, 1238081079173, 125, 0, 200, 3147, 0, 0, 0, 125
119, 0, 101, 1238081079205, 93, 0, 200, 3151, 0, 0, 0, 93
103, 0, 101, 1238081079220, 94, 0, 200, 3151, 0, 0, 0, 94
101, 0, 201, 1238081079064, 266, 0, 204, 0, 0, 63, 78, 266
101, 0, 200, 1238081079064, 266, 0, 0, 0, 0, 0, 0, 0
88, 0, 201, 1238081079033, 312, 0, 204, 0, 0, 94, 109, 312
88, 0, 200, 1238081079033, 312, 0, 0, 0, 0, 0, 0, 0
117, 0, 201, 1238081079158, 203, 0, 204, 0, 0, 0, 0, 203
117, 0, 200, 1238081079158, 203, 0, 0, 0, 0, 0, 0, 0
66, 0, 101, 1238081078877, 500, 0, 200, 3148, 0, 0, 0, 93
```

# Grinder Analyzer

- Специальный питоновский скрипт для анализа логов, записанных с помощью Grinder HTTP Plugin
- необходима установка Jython/Python

# Пример отчета



The screenshot shows a Mozilla Firefox browser window displaying a performance report. The address bar shows the file path: file:///c:/Work/Tools/Grinder/grinderAnalyzer.V2.b9/grinderReport00/report.html. The report is presented as a table with 11 columns: TRANSACTION NAME, TESTS PASSED, TESTS W/ ERRORS, PASS RATE, MEAN RESPONSE TIME, RESPONSE TIME STANDARD DEV., MEAN RESPONSE LENGTH, BYTES PER SEC, MEAN TIME RESOLVE HOST, MEAN TIME ESTABLISH CONNECTION, and MEAN TIME TO FIRST BYTE. The table contains 12 rows of data, each representing a different transaction type and its performance metrics.

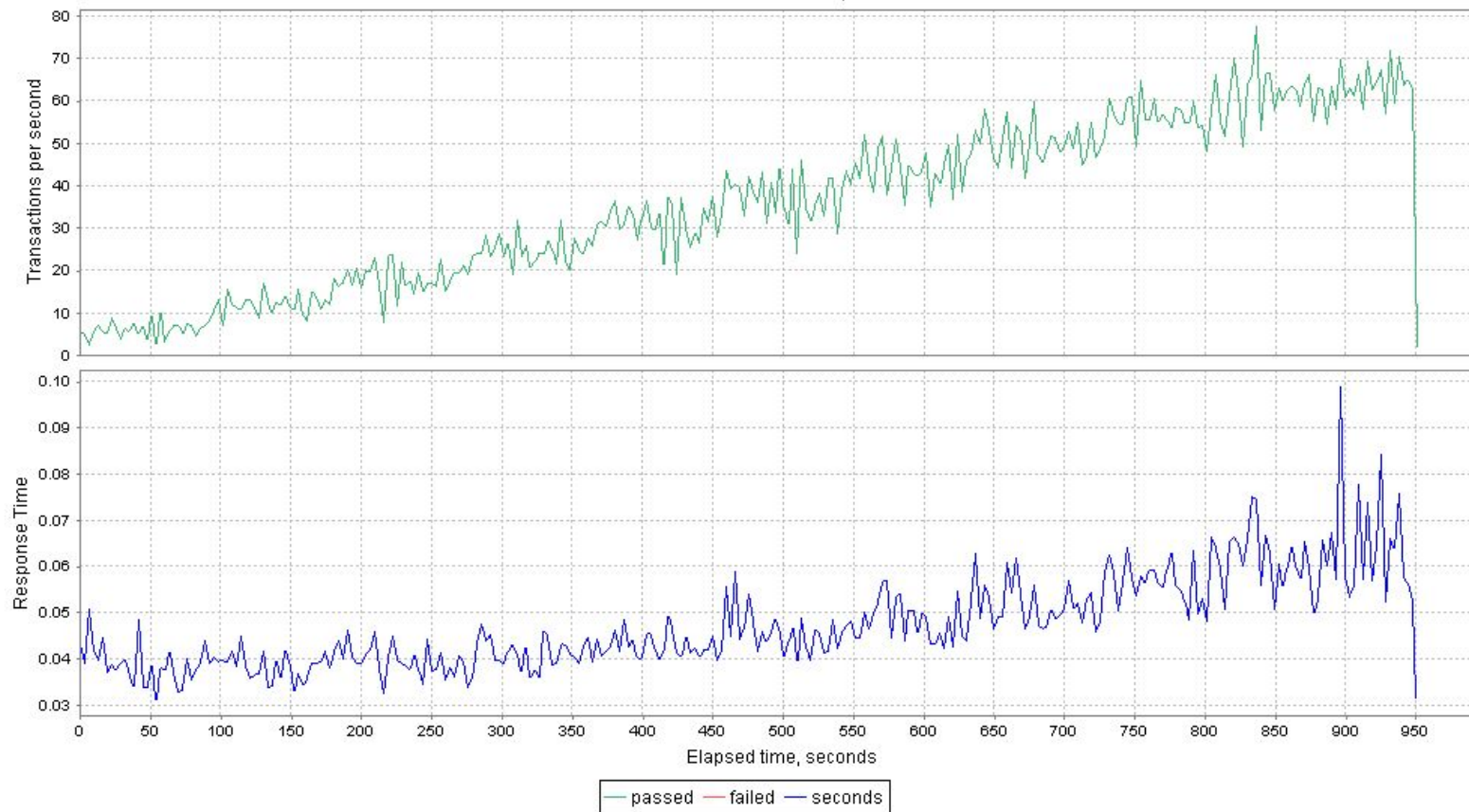
TRANSACTION NAME	TESTS PASSED	TESTS W/ ERRORS	PASS RATE	MEAN RESPONSE TIME	RESPONSE TIME STANDARD DEV.	MEAN RESPONSE LENGTH	BYTES PER SEC	MEAN TIME RESOLVE HOST	MEAN TIME ESTABLISH CONNECTION	MEAN TIME TO FIRST BYTE
<a href="#">GET /</a>	150.0	0.0	1.000	15409.05	15867.87	3149.83	-1	0.0	0.1	3801.74
<a href="#">GET logo_plain.png</a>	150.0	0.0	1.000	6061.97	8230.25	7582.0	-1	0.0	0.0	6052.25
<a href="#">GET eHvqVzRi2zs.js</a>	150.0	0.0	1.000	46378.52	60766.16	5852.0	-1	0.0	0.0	6555.16
<a href="#">GET generate_204</a>	150.0	0.0	1.000	9753.89	12947.01	0.0	-1	1.46	123.75	9753.68
<a href="#">GET nav_logo4.png</a>	150.0	0.0	1.000	13828.99	21985.86	7121.0	-1	0.0	0.0	13819.24
<a href="#">GET csi</a>	150.0	0.0	1.000	18026.54	27624.71	0.0	-1	0.0	0.0	18025.09
<a href="#">GET search</a>	150.0	0.0	1.000	26191.69	34465.66	251.0	-1	0.0	0.0	25362.88
<a href="#">GET search</a>	150.0	0.0	1.000	28455.57	42283.46	245.0	-1	0.0	0.0	28219.88
<a href="#">GET search</a>	150.0	0.0	1.000	31147.05	40988.28	261.0	-1	0.0	0.0	30787.81
<a href="#">GET search</a>	150.0	0.0	1.000	35116.03	51639.66	275.0	-1	0.0	0.0	34606.59
<a href="#">GET search</a>	150.0	0.0	1.000	42022.31	53161.38	301.0	-1	0.0	0.0	41844.91
<a href="#">GET search</a>	150.0	0.0	1.000	44500.34	71146.98	372.0	-1	0.0	0.0	40653.0

# Пример отчета

## All Transactions

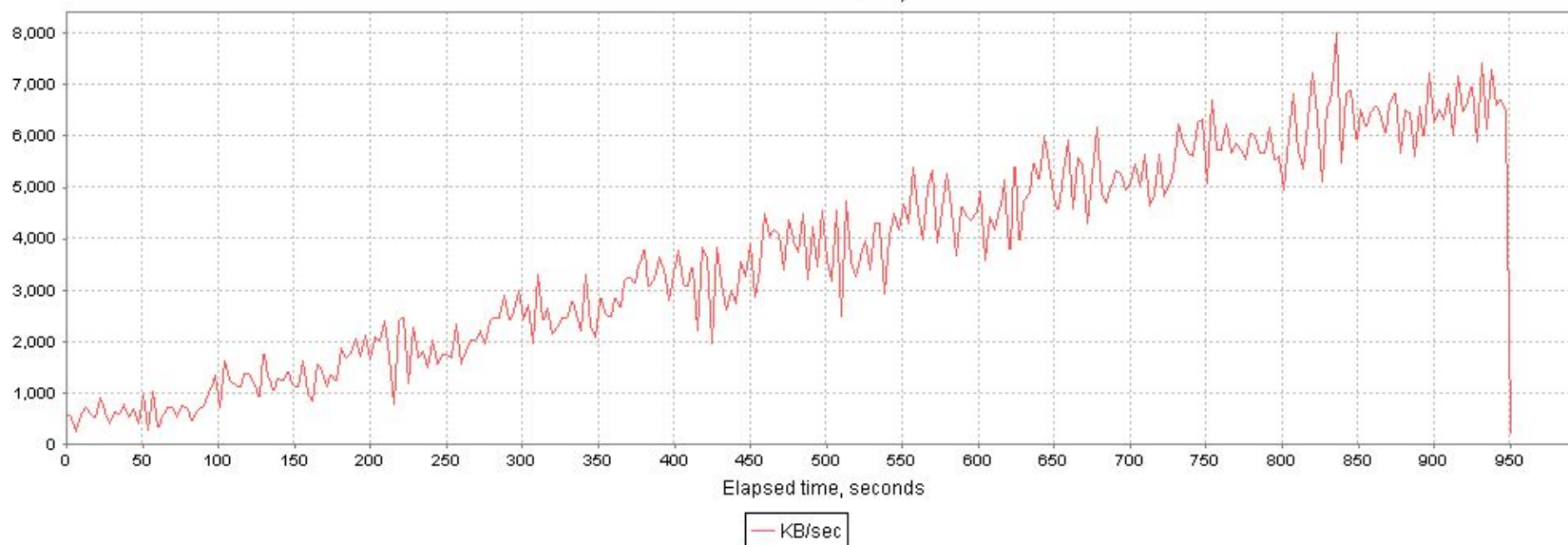
Performance

Test executed March 13 2009, 23:03



# Пример отчета

**All Transactions**  
Bandwidth Used  
Test executed March 13 2009, 23:03







Вопросы?

> End