How to Do Website Performance Analysis and Tuning with Flood and Apache 2.0

Justin Erenkrantz *University of California, Irvine*

Aaron Bannert *Covalent Technologies*
No, not this type of flood

http://www.eng.uci.edu/~bfs/flood.jpg
Flood

- Subproject of the Apache HTTP Server Project
- Licensed under the [Apache Software License](http://www.apache.org/LICENSE.txt)
- *Mailing List*: test-dev-subscribe@httpd.apache.org

```
Updated slides at: http://www.clove.org/flood-presentation/
```

---

[^a]: [http://www.apache.org/LICENSE.txt](http://www.apache.org/LICENSE.txt)
Why should you care about performance?
Performance equals $$

✧ Meet expectations
✧ Economies of scale
✧ Bang for buck
Capacity Planning

- How far can you go?
- Average load
- Peak load
Learn to love the Slashdot effect
Learn to love the Slashdot effect (cont.)

- Publicity is good
- May miss opportunity
- Need to capitalize

/.ed, I will repost image later
Denial of Service

- Worms
- Malicious attacks
- Email harvesters
- Robots
Unforeseen factors

✧ Catastrophic events

✦ A spokeswoman for Keynote Europe, a firm that monitors Internet performance, said “the [9/11] slowdown was worse than ... Code Red.”

✧ May affect your ISP

✧ May affect potential client’s ISP

---

“How well does your site size up?”
Performance as design criteria

- Architecture can dictate performance
- Consider performance from the beginning
- Make it a part of your process
Performance as continuing practice

- Tuning
- Measurement
- Analysis
Approaching Performance Analysis

“A Recipe for Success”
1. Identify components

Does this represent your system?

✧ 1-tier

✧ 2-tier

✧ 3-tier

HTTP Servers (Apache 2.0)  App Servers (Tomcat 4.0)  Database (PostgreSQL 7)
What about the Nameserver?

HTTP Servers (Apache 2.0)  App Servers (Tomcat 4.0)  Database (PostgreSQL 7)

Nameserver
Identify components (cont.)

Do you have a file server?

HTTP Servers
(Apache 2.0)

App Servers
(Tomcat 4.0)

Database
(PostgreSQL 7)

Nameserver

File Server

The 4th O’Reilly Open Source Convention
Identify components (cont.)

How about “Single Signon”?
What’s your point?

✧ Small components can affect the performance.
✧ Therefore, we need to look all the components in detail.
What do we have?

- Web servers
- Application servers
- Custom modules
- Custom applications
- Databases
- ...

The 4th O’Reilly Open Source Convention
What else do we have?

- File servers
- Nameservers
- Directory services (LDAP)
- Authentication services
- SSL accelerators
- ...

The 4th O’Reilly Open Source Convention
What about the network?

- Hubs/Switches/Cables
- Routers
- Firewalls
- Transparent caches (HTTP caches)
- Load Balancers
- ...

The 4th O’Reilly Open Source Convention
2. How do components interact?

- Inter-component dependencies
- e.g. Setting up an account requires 10 table inserts in the DB.
3. Identify points of contention

- Single point(s) of failure
- Critical resources
- Failure conditions
- Recovery plans
4. What information do we already have?

- OS statistics
- Network statistics
- Cache performance
- DB statistics
- Homebrew monitors
What is wrong with this picture?
5. We need to think about the users

- The web browser

How do we emulate such a complex set of users?
Why we developed Flood

“Let’s generate some load!”
Scratch an itch

- Java-based web applications
- Performance requirements in the contract
Why we developed Flood? (cont.)

- Previous ad-hoc tools:
  - Shell scripts (*netcat*, *curl*, etc...)
  - Multiple clients via *ssh/rsh*
  - Coarse measurements
  - Difficult to control
Why we really open-sourced flood

- So others may benefit
  - Building a community
- So others may contribute
  - Further our work
Who wants flood?

Target Users:

✧ QA Engineer
✧ Performance Testers

Target Applications:

✧ Websites
✧ HTTP Apps
✧ “Web Services”
Features

- HTTP/1.1
- SSL
- Emulate users
- Emulate multiple users
- Emulate multiple complex users
- Emulate several different types of users
What can’t it do?

- Automated scripting
- Find and fix your performance problems

This means that flood is merely a load generating tool that can aid in the collection of data. Analysis and interpretation of that data must be done by a human familiar with the system. In short, we can not fix your problems, but we can help you to identify them.
Flood Design
Design Goals

✧ Modular
✧ Simple but flexible configuration
✧ Scalable
✧ Accurate
✧ Portable
Modularity

- Easily add new features
- Design a framework
  - Add the kitchen sink later
- Framework defines actions
- Modules define behavior
What Flood modules exist today?

- normal BSD sockets
- SSL
- Round-robin URL lists
- Simple reports
- Relative-timing reports
Why XML?

✧ Simple
✧ Flexible
✧ Machine verifiable \(^a\)
✧ Easily human generated

✧ Disadvantage:
  ✧ some input must be XML-encoded,
  (which is ugly)

\(^a\) so we can write fancy frontends later
Scalability

✧ Huge webserver farms
✧ Take advantage of multiple \(^a\)
  ✧ threads
  ✧ processes
  ✧ machines

\(^a\)in order to eliminate hardware constraints
Accuracy

- Minimal overhead
- Reproduceable results
- Accuracy vs. Precision
Portability

❖ APR \(^a\)

❖ Platforms (the short list):
   
   – Linux
   – FreeBSD, NetBSD, OpenBSD, Darwin (Mac OS X)
   – Solaris
   – Windows \(^b\)
   – ...

\(^a\)Apache Portable Runtime (http://apr.apache.org/)

\(^b\)Thanks to William Rowe
Running Flood
“Farm World”
What the heck is this wacko naming scheme?

Problem: Terms like

- *thread*
- *worker*
- *process*

are overused and ambiguous.
Solution: Come up with our own naming scheme. \(^a\)

\(^a\)Side-effect: Allows us to think outside of concepts like sequences and threads and remote processes.
You know what this is...
url\texttt{list}

- Group \texttt{urls} together
- Predelay and postdelay
Single thread

Single site visitor

Uses urllists in specific ways:

- Random order
- Round-robin (run in a loop)
- Keepalive
farm

✦ Group of farmers in parallel
✦ Controls ramp-up
✦ Creates/Controls farmers:
  ✦ looping
  ✦ reordering \(^a\)

\(^a\)fine-grain control is not implemented, we only have simple looping at the moment
collective

✦ A set of farms running on a single host in parallel

\footnote{The collective class hasn’t been implemented, so this syntax is preliminary.}
A set of collectives running on multiple host machines

Invokes remote instances (using rsh/ssh/etc..)

Central coordination

Central reporting

---

The megaconglomerate class hasn’t been implemented, so this syntax is preliminary.
Extensions to core flood functionality

Modules may override specific methods

The runtime configuration is defined in the “profile”

Examples of overridable methods:

- `socket (generic, ssl)`
- `verify.resp (200/OK)`
- `report (easy, simple, relative_times)`
“Practical Analysis”
Let’s focus on Apache’s handling of server-side includes (SSI)

✦ Compare Apache 1.3 and Apache 2.0
✦ Same application
✦ Same client characteristics
What does the site look like?

✧ Main page:
  ✦ SSI
  ✦ contains 2 static images
  ✦ links to secondary page and some big file

✧ Secondary page:
  ✦ static HTML
  ✦ contains 2 static images
What are the use cases?

3 typical uses of this site:

1. user just hits the main page
2. user hits main page then hits secondary page
3. user hits main page then downloads a big file
Gratuitous Flowcharts

User just hits the main page:
Gratuitous Flowcharts

Sample XML Configuration

```xml
<urllist>
    <name>First</name>
    <description>Use Case 1</description>
    <url>http://localhost:8080/site/</url>
    <url>http://localhost:8080/site/httpd_logo_wide.gif</url>
    <url>http://localhost:8080/icons/apache_pb.png</url>
</urllist>
```
More Gratuitous Flowcharts

_user hits main page then hits one secondary page:

1. User hits main page
2. Fetch Page
3. 10-20sec later
4. Fetch Image
5. Keepalive connection
6. Fetch Image
7. Fetch Page
8. Fetch Image
9. Keepalive connection
10. Fetch Image

The 4th O’Reilly Open Source Convention
Sample XML Configuration

```xml
<urllist>
  <name>Second</name>
  <description>Use Case 2</description>
  <url>http://localhost:8080/site/</url>
  <url>http://localhost:8080/site/httpd_logo_wide.gif</url>
  <url>http://localhost:8080/icons/apache_pb.png</url>
  <url predelay="30">http://localhost:8080/site/test2.html</url>
  <url>http://localhost:8080/site/oscon2002-lg_ad.gif</url>
  <url>http://localhost:8080/site/hotel_pool.jpg</url>
</urllist>
```
More Gratuitous Flowcharts

* User hits main page then downloads a big file:
Sample XML Configuration

```xml
<urlist>
  <name>Third</name>
  <description>Use Case 3</description>
  <url>http://localhost:8080/site/</url>
  <url>http://localhost:8080/site/httpd.logo_wide.gif</url>
  <url>http://localhost:8080/icons/apache_pb.png</url>
  <url predelay="15">http://localhost:8080/site/flood.pdf</url>
</urlist>
```
Set up a profile

- enable keepalive
- use the relative_times output format
- check for 200 OK

Sample XML Configuration

```xml
<profile>
  <name>SiteProfile</name>
  <description>Round Robin Configuration</description>
  <useurllist>Second</useurllist>
  <profiletype>round.robin</profiletype>
  <socket>keepalive</socket>
  <report>relative_times</report>
  <verify_resp>verify.200</verify_resp>
</profile>
```
A farmer to use the profile

-run for 300 seconds (5 minutes)

Sample XML Configuration

```xml
<famer>
  <name>John</name>
  <time>300</time>
  <useprofile>SiteProfile</useprofile>
</famer>
```
The one and only farm

10 Farmer Johns

Sample XML Configuration

```xml
<farm>
  <name>Bingo</name>
  <usefarmer count="10">John</usefarmer>
</farm>
```
Intermission
Demonstration
“The showdown: httpd-2.0 vs 1.3”

What are we testing?

✧ SSI performance
✧ static file performance (small images, html files)
✧ big file downloads
Building Apache httpd-2.0
Building Flood
Running Flood
Gathering realtime stats
Flood Output
What output does Flood produce?

- Raw format
- No automatic analysis
- Define custom output module
  - Flood is very extensible
What kinds of metrics does Flood capture?

✧ Just some basic metrics:
   ✦ Requests
   ✦ Time frame
   ✦ Requests per second

✧ Whatever metrics you want
   ✦ Flood is very extensible
What are the provided output formats?

 desteği kontrol edilen rapor etiketi profil bölmesinde

 relative_times

easy

 simple
Relative Times Report Format - Summary

<table>
<thead>
<tr>
<th></th>
<th>Connect</th>
<th>Write</th>
<th>Read</th>
<th>Close</th>
<th>Valid</th>
<th>ID</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>123123123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>456</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123123706</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>457</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124289</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>458</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124872</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>459</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123125455</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>460</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126038</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>461</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126621</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>462</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127204</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>463</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127787</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>464</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
</tbody>
</table>

- Prints out an entry for each URL that is hit
- Times given as microseconds from epoch
- All times relative to the first time listed
- If using HTTP keep-alive, connect and close times may be 0
Relative Times Report Format - Detail

<table>
<thead>
<tr>
<th>Start</th>
<th>Connect</th>
<th>Write</th>
<th>Read</th>
<th>Close</th>
<th>Valid</th>
<th>ID</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>123123123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>456</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123123706</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>457</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124289</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>458</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124872</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>459</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123125455</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>460</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126038</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>461</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126621</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>462</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127204</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>463</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127787</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>464</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
</tbody>
</table>

- Absolute time
- Microseconds since the epoch (Jan 1, 1970)
Relative Times Report Format - Detail (cont.)

<table>
<thead>
<tr>
<th>Start</th>
<th>Connect</th>
<th>Write</th>
<th>Read</th>
<th>Close</th>
<th>Valid</th>
<th>ID</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>123123123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>456</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123123706</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>457</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124289</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>458</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124872</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>459</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123125455</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>460</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126038</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>461</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126621</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>462</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127204</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>463</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127787</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>464</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
</tbody>
</table>

✧ Relative to start time in microseconds
✧ Time it took to connect
✧ Time to write the request
✧ Time to read the response
✧ Time to close the response
Relative Times Report Format - Detail (cont.)

<table>
<thead>
<tr>
<th>Start</th>
<th>Connect</th>
<th>Write</th>
<th>Read</th>
<th>Close</th>
<th>Valid</th>
<th>ID</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>123123123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>456</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123123706</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>457</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124289</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>458</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124872</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>459</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123125455</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>460</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126038</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>461</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126621</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>462</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127204</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>463</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127787</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>464</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
</tbody>
</table>

✎ Response valid
♦ OK or FAIL
♦ Indicates whether verification was successful
Unique client ID

- usually just Thread ID or Process ID
- Allows aggregate results for each virtual user
Relative Times Report Format - Detail (cont.)

<table>
<thead>
<tr>
<th>Start</th>
<th>Connect</th>
<th>Write</th>
<th>Read</th>
<th>Close</th>
<th>Valid</th>
<th>ID</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>123123123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>456</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123123706</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>457</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124289</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>458</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123124872</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>459</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123125455</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>460</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126038</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>461</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123126621</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>FAIL</td>
<td>462</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127204</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>463</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
<tr>
<td>123127787</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>OK</td>
<td>464</td>
<td><a href="http://www.example.com/">http://www.example.com/</a></td>
</tr>
</tbody>
</table>

- Request-URI
  - w/o query string
  - Allows aggregate results for each page
Easy report format

- Identical to the relative times format
- Except all times relative to the epoch
Simple report format

✦ For each URL hit
  ✦ Verification result: OK/FAIL
  ✦ Request-URI

✦ As a summary
  ✦ Tally of all response status codes
Measurement Tools
Performance analysis

- Highly OS-dependent
- Varies greatly by OS
- Linux sysstat tools: http://perso.wanadoo.fr/sebastien.godard/
Solaris has **truss**, Linux and others have **strace**

**Features:**
- Traces system calls
- Attach to running processes
- No kernel-level info

---

\[\text{FreeBSD and Darwin currently have no good way to trace system calls on a per-process basis except for at the kernel level with } \text{kdump and ktrace. This makes performance analysis quite challenging on these platforms.}\]
The stat tools

- Huge range of system statistics
- `vmstat`
  - Memory and paging metrics
- `iostat`
  - Disk performance metrics
- `nfsstat`
  - Network filesystem metrics
More stat tools

- **netstat**
  - Network subsystem information

- **mpstat**
  - Multi-Processor metrics
  - (locks, threads, semaphores)

- **systat**
  - Overall system metrics (FreeBSD)
sar

- long-running stats-gathering program
- collects database of information
- wide-range of system metrics
- disk I/O, network I/O, memory, etc.
snoop/tcpdump

- captures raw network traces
- ethereal\(^a\) can provide further high-level analysis of the raw trace
- SSL poses a problem since the payload is encrypted

\(^a\)http://www.ethereal.com/
tcptrace

- http://www.tcptrace.org
- Gives statistical information from network traces
- Tracks and graphs network metrics
- Can reassemble TCP sessions
- (useful for feeding back into flood

\(^a\)See: xplot
pstack

- Takes a process ID
- Dumps current stack from each thread
- Great for snapshots
- Solaris, FreeBSD...
JVM stack trace

✧ Profiles all JVM threads
✧ Useful with pstack
   ✧ Use thread ids to correlate
✧ Send your JVM a SIGQUIT signal, it dumps to stdout/stderr
**dummynet**

- Use to emulate real-world networks
- Can add:
  - Random packet loss
  - Random delays
  - Bandwidth limiting
  - Fine-grain statistics
- FreeBSD
Result Analysis
Hints for dealing with all the data

✧ perl/awk/sed/grep/etc... are your friends
✧ Look for trends
✧ Rely on the statistics
Processing the data

We’ve provided a couple scripts:

- analyze-relative
  
  *various averages for the output of a relative_times report*

- analyze-relative-ramp
  
  *same as above, only deals with a ramp-up period and helps isolate slow pages*
Visualizing the data

- gnuplot/xplot
- tcptrace\(^a\)

\(^a\)http://www.tcptrace.org/
Things to look for

❖ Bottlenecks/Capacity limits

“No matter how many webservers we add we can’t seem to handle any more SSL traffic.”

❖ Failure points

“As soon as we hit 100 concurrent users, our database fails.”

❖ Over/under-utilized resources
Trends

- Unbounded resource consumption
- Periodic failures
Iterative Tuning

1. Identify problems
2. Propose solutions
3. Test them
4. Rinse, repeat
Future
What happens now?

- Graphical and non-graphical frontends to generate XML configs
- Raw data processing and analysis
- Take advantage of multiple client machines
- Multiple verification routines in parallel
- Automated profile generation from things like
  - tcptrace
  - snoop/tcpdump raw traces
  - common log format
Thank You
Appendix A: Sample XML Configuration Snippets
Sample XML Configuration

```xml
<url>http://httpd.apache.org/test/flood/</url>
```
Sample XML Configuration

```xml
<urllist>
  <name>SSL Hosts</name>
  <description>A bunch of SSL hosts we want to hit</description>
  <url>https://www.modssl.org/example/test.phtml</url>
  <url>https://mozilla-crypto.ssleay.org/cryptocheck.php</url>
</urllist>
```
Sample XML Configuration

```xml
<farmer>
  <name>Joe</name>
  <count>5</count>
  <useprofile>RoundRobinProfile</useprofile>
</farmer>
```
Sample XML Configuration

```xml
<farm>
  <name>Bingo</name>
  <usefarmer count="25">Joe</usefarmer>
</farm>
```
Sample XML Configuration

```xml
<collective>
  <name>Borg</name>
  <usefarm count=4>Bingo</usefarm>
  <usefarm count=2>Pepperidge</usefarm>
</collective>
```

---

The `collective` class hasn’t been implemented, so this syntax is preliminary.
The megaconglomerate class hasn’t been implemented, so this syntax is preliminary.
Sample XML Configuration

```xml
<profile>
  <name>RoundRobinProfile</name>
  <description>Round Robin Configuration</description>
  <useurlist>Test Hosts</useurlist>
  <profiletype>round_robin</profiletype>
  <socket>keepalive</socket>
  <verify_resp>verify_200</verify_resp>
  <report>easy</report>
</profile>
```
Appendix B: Other Performance Measurement Tools
What other tools are out there?

✦ Commercial
  ✦ Microsoft Web Application Stress Tool
  ✦ Empirix e-TEST
  ✦ Mercury Interactive LoadRunner
  ✦ SPECweb99

✦ Open-Source
  ✦ ApacheBench (ab)
  ✦ httpperf
  ✦ S-Client Architecture
  ✦ JMeter
Microsoft Web Application Stress Tool

- http://webtool.rte.microsoft.com/

- **Pros**
  - Uses Internet Explorer as request engine
  - Automated recording
  - Centralized administration
  - Free

- **Cons**
  - Only available on Microsoft platforms
  - Poor SSL support
  - Not able to handle dynamic sites
Empirix e-TEST

✥ http://www.empirix.com/

✥ Pros
   ✦ Automated recording
   ✦ Centralized administration
   ✦ Server-side collection via SNMP and plugins
   ✦ Multi-platform support

✥ Cons
   ✦ Expensive
Mercury Interactive LoadRunner

✈ http://www.mercuryinteractive.com/products/loadrunner/

✈ Pros

✦ Automated recording
✦ Centralized administration
✦ Server-side collection via SNMP and plugins
✦ Multi-platform support

✈ Cons

✦ Expensive
SPECweb99


- Pros
  - Widely accepted by industry

- Cons
  - Not a test tool, just a benchmark
ApacheBench (ab)

✧ Included with Apache HTTP Server releases:
   http://httpd.apache.org/

✧ Pros
   ✦ Potentially higher concurrency
   ✦ Uses select()/poll() model
   ✦ Free

✧ Cons
   ✦ Only able to handle one URL
httperf


**Pros**
- SSL support
- Provides a framework
- Free

**Cons**
- Primitive multiple URL support
S-Client architecture

✦ http://www.cs.rice.edu/CS/Systems/Web-measurement/

✦ Pros
  ✦ Produces a reliable steady stream of requests
  ✦ Free

✦ Cons
  ✦ Only able to handle one URL
JMeter

- http://jakarta.apache.org/jmeter/

Pros

- GUI front-end
- Extensible
- Free

Cons

- Accuracy sacrificed as scales up
- Can not handle dynamic requests