Scalable Internet Architectures

George Schlossnagle
Theo Schlossnagle

<george@omniti.com>
<theo@omniti.com>
Agenda

- Choosing Hardware
- Choosing an Application/Availability Architecture
- Deciding between Third-Party and Custom-Built Software
- Case Study I: Building Fast Scalable Web Forums
- Case Study II: Distributed Logging
Choosing Hardware

- 'Enterprise' Hardware
  - Expensive
  - Reliable

- Commodity Hardware
  - Cheap
  - Fast
  - Unreliable
Setting Up Apache

- Turn KeepAlives off
- As cluster grows, keep MaxClients tuned to avoid excessive database connections
- For dynamic content consider using a local proxy instance
Configuring A Local Proxy

- **Run 2 Apache Instances on a Single Host**
- **Public Instance handles high-latency clients using mod_rewrite/mod_proxy.**
- **Local Instance handles dynamic content - only makes low-latency connections to the public instance**
Configuring A Local Proxy

**Exterior (proxy) Instance**

```html
<IfDefine PROXY
  DocumentRoot /var/apache/htdocs
  Listen myexternal_ip:80
  MaxSpareServers 32
  MaxClients 128
  MaxRequestsPerChild 100000
  KeepAlive off
  LoadModule proxy_module libexec/libproxy.so
  LoadModule rewrite_module libexec/mod_rewrite.so
  AddModule mod_proxy.c
  AddModule mod_rewrite.c
  ProxyRequests on
  NoCache
  ProxyPassReverse / http://127.0.0.1
  RewriteRule ^proxy: - [F]
  RewriteRule ^/(http|ftp): - [F]
  RewriteRule ^/(.*\.html)$ http://127.0.0.1/$1 [P,L,T]
</IfDefine>
```
Configuring A Local Proxy

• **INTERIOR INSTANCE**

```
<IfDefine DYNAMIC>
  DocumentRoot /var/apache/htdocs
  Listen                localhost:80
  MaxClients            40
  MaxRequestsPerChild   0
  KeepAlive             off
  LoadModule perl_module libexec/libperl.so
  AddModule mod_perl.c
  <Files *.asp>
    SetHandler perl-script
  </Files>
</IfDefine>
```
Designing a HA/LB scheme that's right for you

- Recognize the difference between replicatable data and non-replicatable data
  - Replicatable data needs marginal protection. Use commodity hardware.
  - Non-Replicatable data needs single-point reliability, consider Enterprise hardware.
- Bring the data to the session, not vice-versa.
  - Leverage distributed systems technology
  - Avoid creating artificial points of failure
Typical Three Tier Architecture

- The Internet
- exterior load balancers
- webserver tier
- interior load-balancers (physical or logical)
- application server tier
- database

- easily replicateable (really acts just as a proxy)
- unnecessary single point of failure (data is client session data, generated 'on demand')
- natural single point of failure (difficult to replicate - requires long-term data integrity)
Modern Two Tier Model

The Internet

webserver tier
using distributed session cache and db connection pooling

database

head tier servers all disposable
(think about RAID)

reduced to one tier in which all hosts are mission critical

fast static server for images/static content
Choosing between Custom and Commercial Software

**Commercial**
- Code 'maturity'
- Dedicated Support

**Homegrown**
- Designed for your particular needs
- In-house support
Case Study I: Caching Web Objects

- How well does your data match the original design goals of any commercial products being considered?
  - Is the data static?
  - Is the data static for a short period of time?
  - Is the data static for a short period of time for each client?
  - Does the data contain components which are static for each client for a short period of time?
Detailed Example: Web Forums
Original Implementation

- Every page is generated by a database query which returns a sorted list of all messages which are returned to the user.
  - Inefficient, database intensive, scales poorly as message volume increases.
  - Takes no advantage of select/update ratio.
SECOND IMPLEMENTATION: ADD BLACK-BOX CACHING

Last-modification time is stored on every update and is used to mark message listings as cacheable.

- Takes advantage of high cache locality.
- Provides good scaleability results.
- Requires 3-tier architecture.
- Minimal application modification required.
Third Implementation: Application-Integrated Caching

Static pages are written to shared filesystem, and rewritten on update.

- Takes advantage of high cache locality.
- Efficient use of hardware.
- Good scalability.
Static pages are written locally, nodes use group communication tools to coordinate static page removal on updates.

- Ideal use of commodity hardware.
- Takes advantage of high cache locality.
- Excellent scaleability and avoidance of SPoFs.
IMPLEMENTATION

- MOD_REWRITE SETUP

  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteRule ^/forums/(.*)$ /admin/generator.php?forumid=$1
Implementation

- GENERATOR.PHP

```php
<?php

$forumid = $_GET['forumid'];
if(!$uri) {
    return_error();
}
ob_start();
if(generate_page($forumid)) {
    $content = ob_get_contents();
    $fp = fopen($SERVER['DOCUMENT_ROOT'].$uri, "w");
    fwrite($fp, $content);
    ob_flush();
}
ob_clean();
return_error();
?>
```
Implementation

- UPDATE PAGE:

```php
<?php

update_page($uri);
purge_cache($uri);

?>
```

- `purge_cache` can be something as simple as `unlink()` if we have a single machine or are using a shared mountpoint. Otherwise we can use something like `spread` to coordinate poisoning of all the caches.
Case Study II: Distributed Logging

- Need to consolidate logs across multiple webservers for auditing
- Need to do real-time analysis of logs
First (Traditional) Implementation

• Web logs written locally on every machine, periodically copied to central server and sorted/merged
  • Consolidation is slow
  • Real time log processing is not possible
Candidate Solutions

**Commercial Solutions**
- Expensive
- Lack Flexibility

**Existing Open Source Solutions**
- Syslog logging
  - Unreliable
  - Unicast
- Database logging
  - Reliable
  - Unicast
Custom Solution (mod_log_spread)

- Designed as Apache module for maintainability
- Reliable multicast transport for maximum flexibility
  - Aggregated log stream can be used to maintain/track user state and server health across multiple servers, asynchronously but in real-time
  - Multicast transport allows additional monitoring facilities to be added for ‘free’
Thanks!