B336 Advanced Internet Computing

Web Server Performance Issues

Learning Objective

- Understand some issues dealing with making a web server perform at an optimal level.

Lecture Outline

- Server Resource Usage
- Process Control and Network Configurations
- Making use of logs
- Directory-level options

Hardware and Operating System Issues

- Your web server's performance is highly dependant on the underlying hardware and OS supporting it.
    - Web server administration goes hand-in-hand with general system administration.
- The more available and efficient hardware (RAM, CPU, disk, network, etc), the better performing the web server will be.
- The better configured the OS is, the better performing the web server will be.

Operating System and the Web Server

- Obviously the more resources your operating system allocates to the web server, the better the performance of the web server.
    - But generally there is no hard and fast rules that apply to all web servers - consult the web server documentation on any OS specific issues.
- The same web server ported to different operating systems have different performance issues.
    - Eg. we have mentioned in the last lecture that Apache's process creation model is different for its Win32 version compared to its original UNIX version.

Operating System and the Web Server

- Some examples from comparison for Apache between its UNIX and Win32 versions:
  - The performance of Apache's in UNIX depends on how well UNIX handles standalone processes or the inetd daemon, while in Win32 it depends on how well Windows handle their "services".
  - The `ThreadsPerChild` directive is not important in the UNIX version (since it depends more on multiple processes rather than threads in a single process), but is critical in the Win32 version (which depends solely on threads to serve multiple requests).

Monitoring System Resource Usage

- We can monitoring the resource usage of the web server by using whatever tools is available through our operating system
    - In this week's labs, we have a look at some simple operating system commands available in Linux to watch the performance of the Apache server.
- Apache also has a module which enables an administrator to watch the systems performance using a web interface.

Server Status using mod_status

- Apache has a module called `mod_status` which enables server status to be monitored through a web page.

- `mod_status` is compiled into the default installation.

Server Status using mod_status

- Eg. To enable anyone to view the server's status, put the following in the `httpd.conf` file:

```
<Location /server-status>
     SetHandler server-status
     Order Deny,Allow
     Deny from all
     Allow from all
</Location>
```

- Then access the page http://red.murdoch.edu.au:15678/server-status

Another example using mod_status

- Eg. To enable only Murdoch machines to access the server status:

```
<Location /server-status>
    SetHandler server-status
    Order Deny,Allow
    Deny from all
    Allow from .murdoch.edu.au
</Location>
```

- You can also add an extra directive anywhere in `httpd.conf` for more status display:

```
ExtendedStatus On
```

What the server-status page shows

- The standard page shows:
  - The number of children serving requests
  - The number of idle children
  - The time the server was started/restarted and the time it has been running for
- With `ExtendedStatus` on, you also get:
  - The status of each child, the number of requests that child has performed and the total number of bytes served by the child
  - A total number of accesses and byte count served
  - Averages giving the number of requests per second, the number of bytes served per second and the average number of bytes per request
  - The current percentage CPU used by each child and in total by Apache
  - The current hosts and requests being processed

Process Control

- We have gone through quite a few issues with process control over the last few lectures, since at the performance of a web server, really means the performance of its processes.
- Determining the correct settings for process start-up like `MinSpareServers`, `MaxSpareServers`, `StartServers`, etc, can be made by monitoring the server's resource usage.

Process Control

- The correct settings for process termination like the `MaxRequestsPerChild` directive is also important.
    - In Apache for UNIX, since `MaxRequestsPerChild` is usually set to `0`, which means there is not fixed period on when a child process will die. The only time it does is if the parent have created a lot to serve a heavy period, and then find too many sitting idle after the period.
    - This is OK for most versions of UNIX, but for some like the old SunOS, the number recommended to be about 10000 since the process suffers from possible memory leaks.

Process Control

- The correct configurations for process control should also be guided by what you know, and what you believe your web-site content will be, and what access behaviours your users will have.

- Eg.
  - You should know how to estimate expected sizes of files on your web site.
  - You should be able to determine how often the files will be accessed, by either monitoring your server, or by using a web-counter.

Network Configurations

- Similar to process control configurations, a web server's network-based behaviour also determines how well it performs.
- Eg. Directives like `KeepAlive`, `KeepAliveTimeOut`, `LimitRequestBody`, `RLimitCPU`, `RLimitMEM`, etc.
- Refer to manuals for more descriptions.

Logging

- Managing logs is an important task in web server administration. It allows you to :
  - detect security breaches, or possible security attacks
  - determine errors in your web server configurations, or web-site content (eg. a main link which has the wrong address will constantly generate an error when accessed)
  - Determine inefficiencies (eg. see example at `http://www.modperl.com/perl_conference/handout.html#Detecting_Robots` for how to use the logs to detect “impolite” web robots/spiders.

Logging

- In the last lecture, we had a look at what default logging is done in Apache, and an example log file.

- The example only demonstrate a small part of what you can choose to log.

Logging the Request Loop

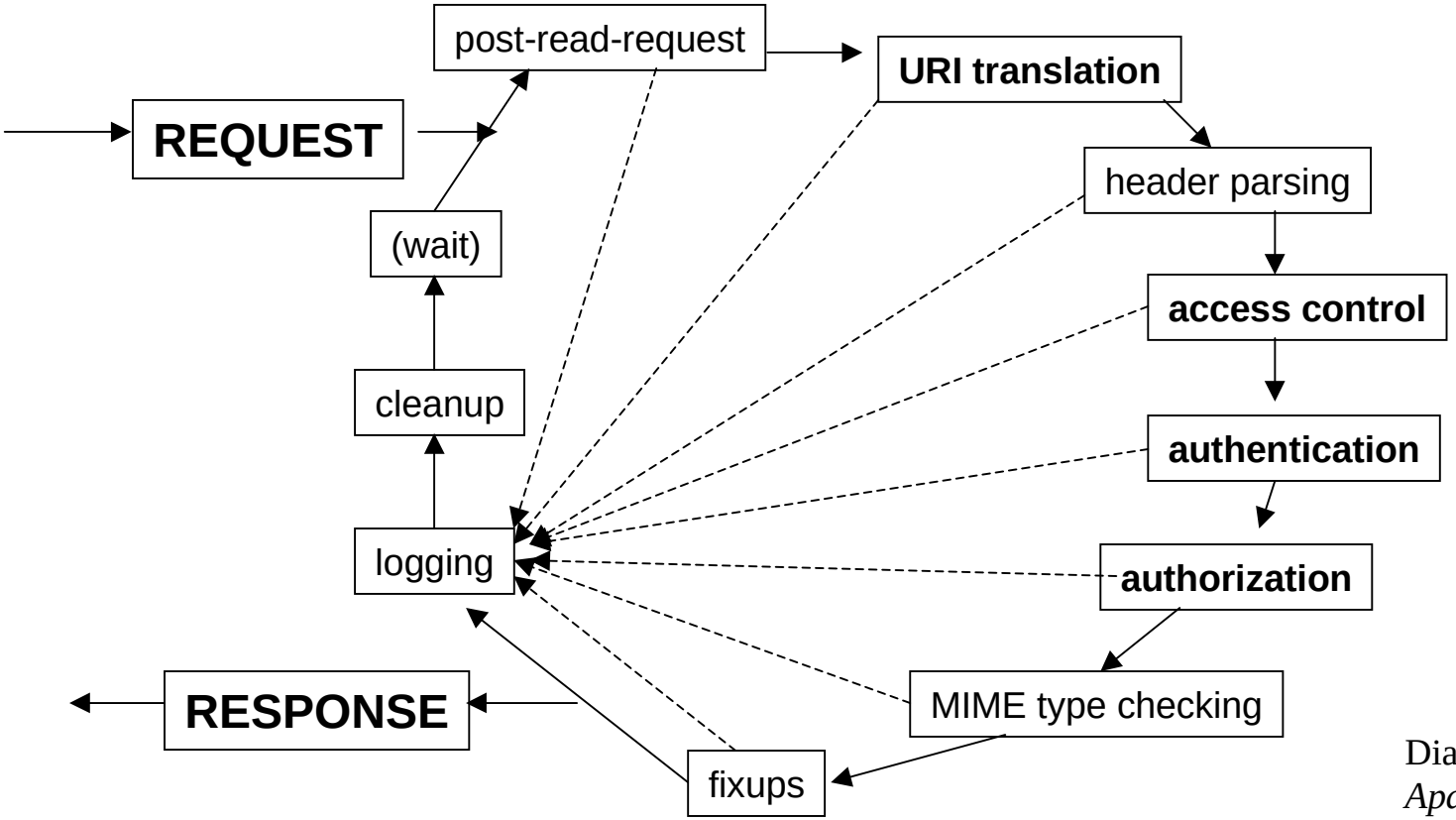

Diagram modified from *Writing Apache Modules with Perl and C*, Stein & MacEachern, 1999, page 60.

Directory Level Options

- We mentioned that the `httpd.conf` file (and possibly other .conf files if enabled) controls options and configurations for the web server.

- Context container tags in the configuration files like `<Directory>` or `<File>` controls the options specific to individual directories and files.

- Apache has a much more flexible way of controlling directory options: the `.htaccess` file.

The .htaccess files

- The name “.htaccess” is a remnant from the past where it only deals with access option. Nowadays, you can put any directory specific options in the .htaccess file.

- It is more flexible way of specifying directory options, since it allows the users to determine their own options without having the administrator modify the `httpd.conf` file.

The .htaccess files

- The syntax of a `.htaccess` file is the same as any configuration files (eg. `httpd.conf`).
  - It consists of a set of directives, and context tags.

- The enable `.htaccess` files to work, you must have the following lines in your `httpd.conf`:

```
AccessFileName .htaccess

AllowOverride All
```

Another example .htaccess setting

- To be more specific, you can do:

```
AccessFileName .htaccess
<Directory /home/~trusted/file>
      AllowOverride AuthConfig
</Directory>
```

- This means you are only granting the directory /home/~trusted/file the ability to change the default authorization configurations.

An example .htaccess file

- Here is an example `.htaccess` file I put in the B336 lecture-notes file directory:

```
AuthName "B336 2001"
AuthUserFile /home/staff/hiew/.htpasswd.b336
AuthType Basic
<Limit GET POST>
      order deny,allow
      require valid-user
</Limit>
```

An example .htaccess file

- For me to use the .htaccess file in the previous page, I have to create a .htpasswd.b336 file. To do that I use the htpasswd program that came with Apache. On the command line I type:

```
/usr/local/apache/bin/htpasswd -c .htpasswd.b336 b336
New password: netrulZ
Re-type new password: netrulZ
```

Security Implications

- By enabling `.htaccess` files, you are basically granting certain administration powers to users. You should be careful how much power you grant, and keep constant watch on potential problems.

- To disallow web clients from accessing the `.htaccess` files through the web, you should also include in `httpd.conf`:

```
<Files .htaccess>
        Order allow,deny
        Deny from all
</Files>
```

Web Server Administrator vs Web Content Managers

- Today, more and more, there is a distinction made between the person(s) who administers the web server, and the person(s) who administers the web-site contents.
  - The former deals with the installation, configuration and performance issues we have been dealing with in this course.
  - The other deals with the content of the web-site (HTML and JavaScript code, dynamic scripts, links to databases, etc)
- The term “webmaster” can mean either or both those responsibilities.

Reference Documentation

- As with previous lectures, this lectures is only be able to give you a basic introduction. Refer to the online documentation for more information.