Multiserver extensions to HTTP

draft-ford-http-multi-server-00

Mark Handley, UCL

Alan Ford, Roke Manor Research

General Idea

- Have one HTTP client pull different parts of the same document from multiple mirror servers simultaneously.

Mirroring

- Common solution to spreading load across more than one server/site.
  - Manual choice of mirror doesn’t work well; primary site takes most of the load.
  - Auto mirroring works fairly well at balancing server load.
    - DNS load balancing (a bit of a hack).
    - Front-end load balancing (if all your servers are at one site).
- No mirroring solution balances network traffic well.

BitTorrent

- Very effective solution for serving content from many unreliable peers.
  - Divide content into blocks.
  - Peer with many peers.
  - Pull blocks from the fastest peers.
    - (also because it’s P2P, upload to peers)
- Very robust to server overload or failure.
- Makes extremely good use of spare network capacity and avoids using congested paths.

Multiserver HTTP

- Robustness of BitTorrent for managed web servers.
  - Resilience to server or net outages.
  - Allows geographic distribution of servers.

- Avoids congested paths, and automatically load balances the network.
  - Works for multihoming, as well as geographic distribution of mirrors.

Basic Idea

- Client advertises multi-server capability in HTTP request.
- Server advertises set of mirrors in response.
  - Uses chunked encoding.
  - Starts sending first chunk of requested data.
- Client can request additional chunks from other mirrors in parallel.
  - Sends more requests to mirrors that respond fastest.
- Result: most data from fastest mirrors.

Increased net performance

- No need to pick a mirror and stick with it.
  - Try four in parallel.
  - Download from the fastest three.
  - Use the fourth TCP connection to experiment with new mirrors.

- Very little data is sent along congested paths.

Increased server performance

- Use one port for initial request, a second port for subsequent mirror requests.
  - Prioritise initial requests.
    - Failed initial requests are noticeable by user.
  - Serve mirror chunk requests as capacity allows.
    - Each chunk served eases work on some other mirror.

- Result: a busy set of mirrors offload work to each other, to maximise overall performance.

- Possible extension: the initial server could send no data, or very little data, when overloaded, and then move all load to mirrors.

HTTP Request Extensions

`X-Multiserver-Version:`

- in requests, declares capability and version

`X-If-Checksum-Match:`

- conditional on mirror chunk request. Only return the chunk if the checksum given matches that of the data.

`Range:`

- regular HTTP range request used to say which chunk is required.

HTTP Response Extensions

`X-Multiserver-Version:`

- Declares server’s multi-server HTTP version

`X-Checksum:`

- Used in response to initial request.
- Data checksum, used to ensure all chunks are from the same version of the content

`X-Mirrors:`

- Used in response to initial request.
- List of URLs that mirror the content.

`Content-Range:`

- regular HTTP content range used to indicate the chunk being sent.

Example: Initial Request

```
GET /wibble/download.zip HTTP/1.1
Host: www.example.com
X-Multiserver-Version: 0.1
```

Example: Initial Response

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 10240
Content-Type: application/zip
Content-Range: bytes 1-10240/2025121
X-Multiserver-Version: 0.1
X-Checksum: MD5 "d6862c992a3d6736ad678cc865dee67f"
X-Mirrors: /wibble/download.zip 3600 \
  http://www.example2.com/wibble/download.zip \
  http://www.example3.com/wibble/download.zip
```

First chunk of data….

Subsequent chunk requests

```
GET /wibble/download.zip HTTP/1.1
Host: www.example.com
X-Multiserver-Version: 0.1
Range: 10241-20480
```

```
GET /wibble/download.zip HTTP/1.1
Host: www.example2.com
X-Multiserver-Version: 0.1
X-If-Checksum-Match: MD5 "d6862c992a3d6736…
Range: 20481-30720
```

A chunk response

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 10240
Content-Type: application/zip
Content-Range: bytes 10241-20480/2025121
X-Multiserver-Version: 0.1
```

…this chunk of data...

Details

- How to handle checksum failure?
- How to handle mirror failure during chunk download?
- How best to manage connection pool with using too many parallel connections?
- How to ensure a request pipeline from a client stays full to each active server?
- How to allow overloaded servers to express policy and move load efficiently?

- We have an implementation that works well
  - Credit: Javier Vela Diago
  - Lots of possibility here for good client heuristics.

Uses

- Very good for very large file download.
  - Music or video download (eg. iTunes, BBC’s iPlayer)
  - Software download.
  - Streaming video over HTTP

- May be good for many small images if wildcarding could be used appropriately.
  - Fetch most images from fastest mirror.

Uses

- Mirror URLs can even be different interfaces on the same server.
  - More traffic transferred over the less congested link.
  - Dynamic load-balancing of a multi-homed site.

Summary

- Very simple extension to HTTP
  - Could significantly improve net and server pool behavior.

- Very few changes required on server.
- Most work happens on client.
  - Even a fairly dumb implementation gets very good performance.

The Bigger Picture

- This is part of a larger effort to improve the robustness of the Internet, improve its ability to self-balance traffic, and better match costs to revenues.
- Other complentary work here:
  - Multi-path TCP
  - Re-ECN
  - LEDBAT congestion control for BitTorrent
  - Network Neutrality talks at Thurs Plenary.