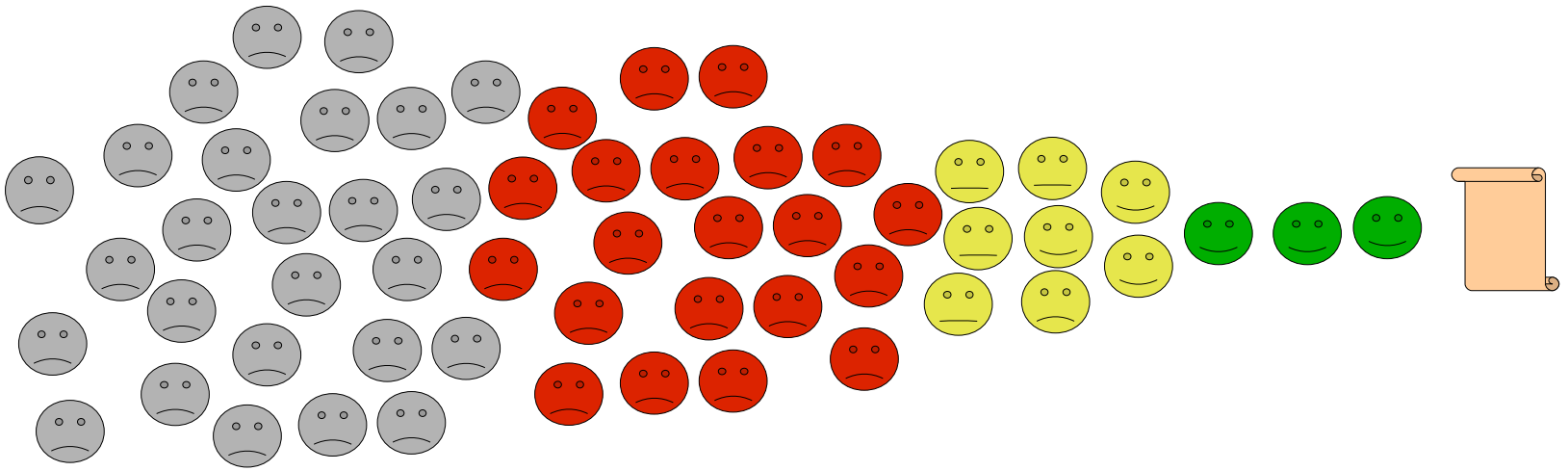
Serving Large Sites from Memory with Perl, Nginx and memcached

Tim Goddard – Catalyst IT

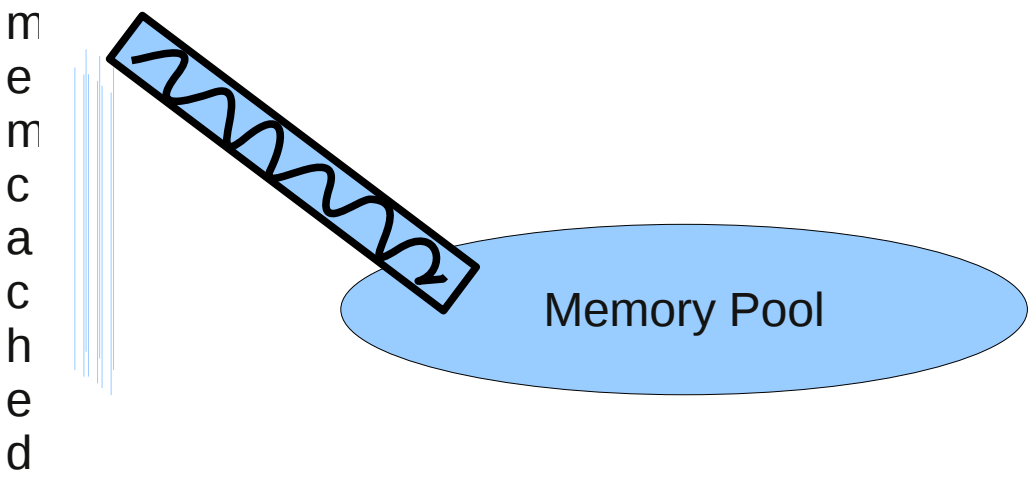
Why care?

- Creating web pages with Perl is slow
- High Load Sites => Minimise Cost per User
- Several possible solutions
- Caching to reduce cost/user



memcached

- Very fast in-memory cache
- A cache only – few guarantees
- Web hosts w/free memory
- Can cluster

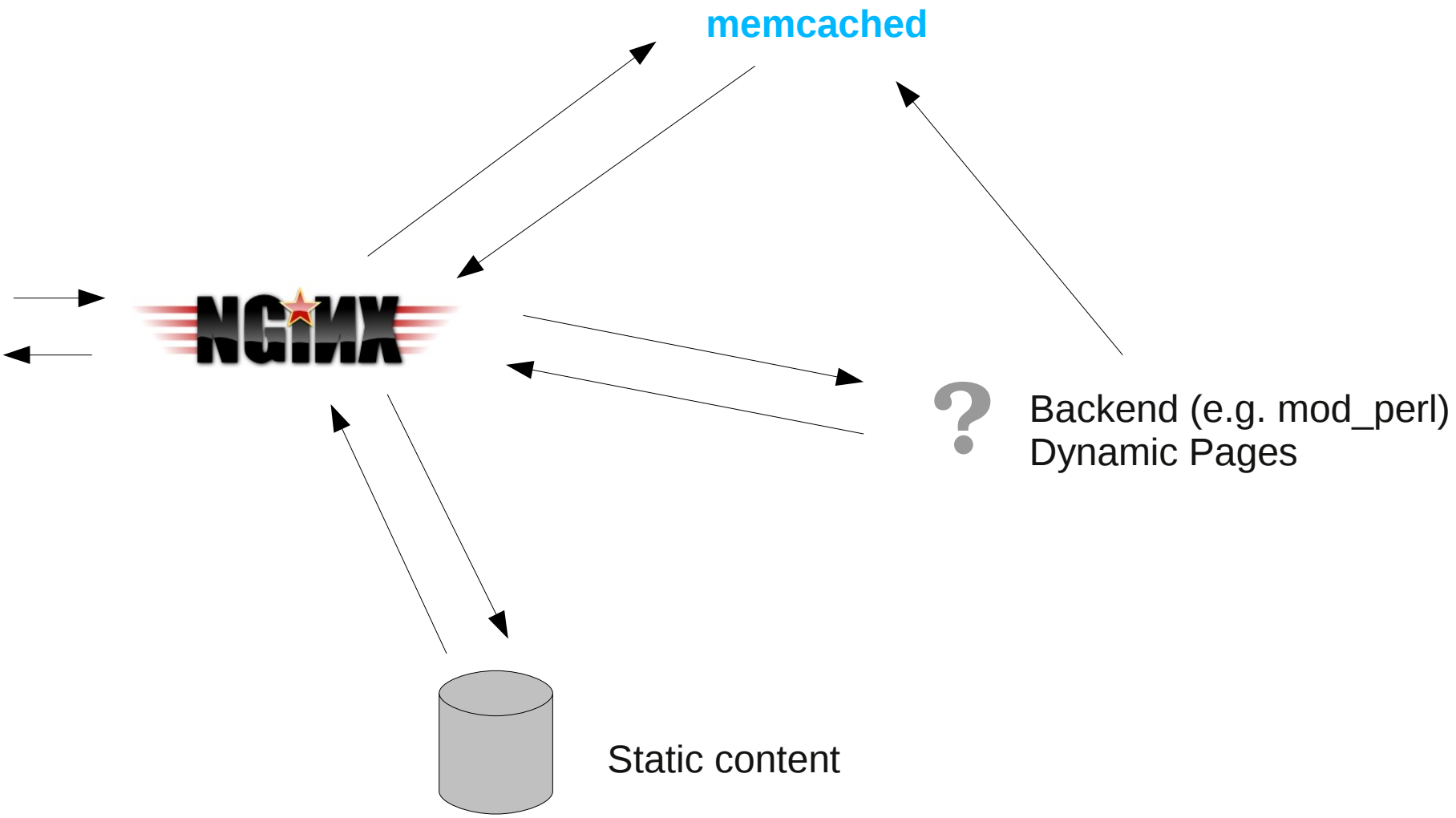


- State-of-the-art web server from Russia.
- Serves static files, dynamic backends.
- No CGI, mod_* - always proxies.
- Can handle thousands of simultaneous connections.
- Can extend with Perl – out of scope today.
- Insanely fast.

Serving Perl

- Perl can be served a bunch of ways.
- They're all annoying and slow.
- Not directly from nginx.
- Behind nginx - ProxyPass

Putting it together



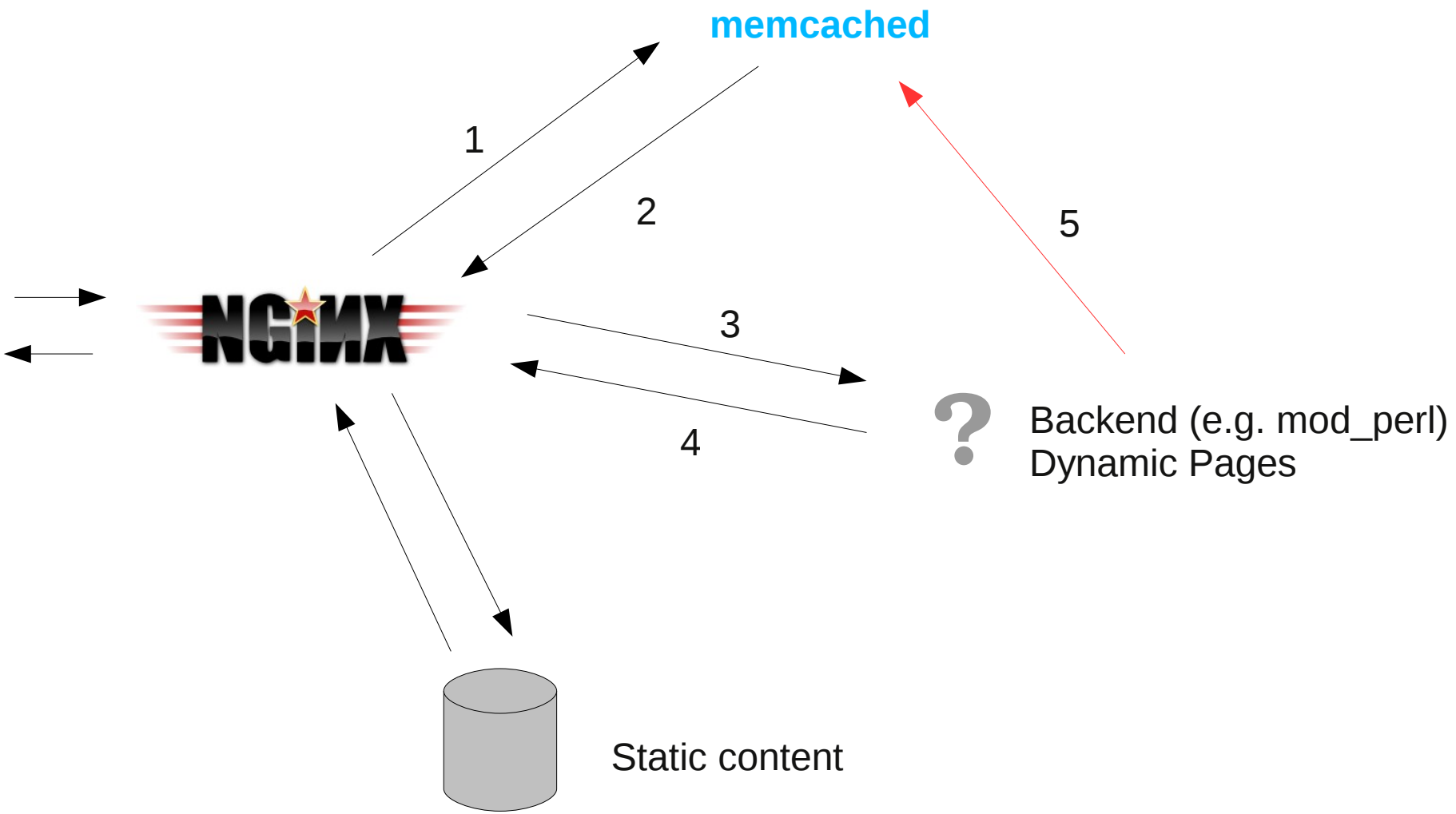
Nginx likes to proxy

```
server {
    server_name my.site.org;

    listen 80;

    location / {
        proxy_pass http://localhost:81;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /static {
        alias /usr/share/my-files/;
    }
}
```

Add a memcached pass

```
server {
    server_name my.site.org;

    listen 80;

    location / {
        set $memcached_key $uri;
        memcached_pass     localhost:11211;
        default_type       text/html;
        error_page         404 = /fallback;
    }

    location /fallback {
        proxy_pass http://localhost:81;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /static {
        alias /usr/share/my-files/;
    }
}
```

Where do they come from?



Advantages Abound

Set cache time in application at last minute

Can build the key however you like – could include a cookie.

Uncacheable content

But, but...

User sessions

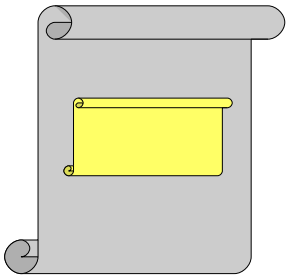
Expiry gap

Cluster/Ketama

Others?

Neat Things I Haven't Tried

SSI



Nginx/Perl