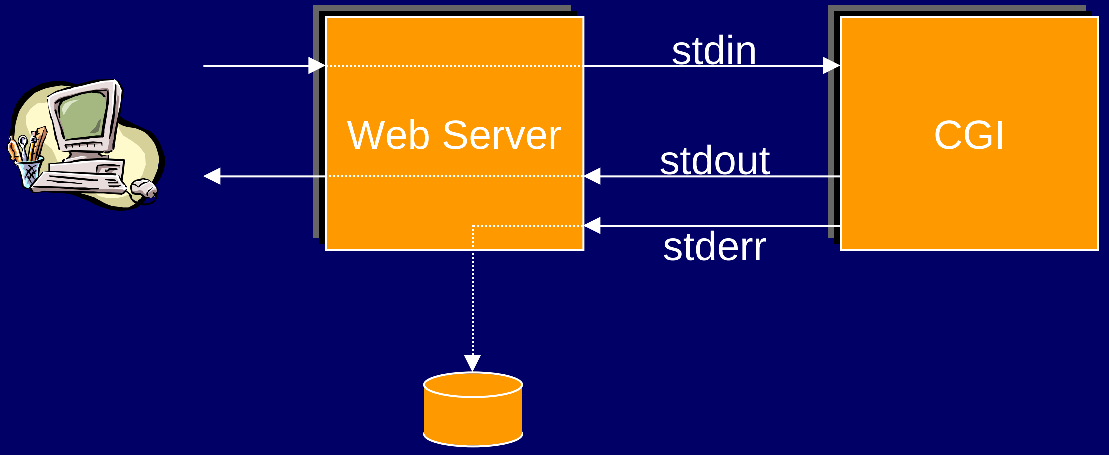
FastCGI

Rob Saccoccio
robs@InfiniteTechnology.com

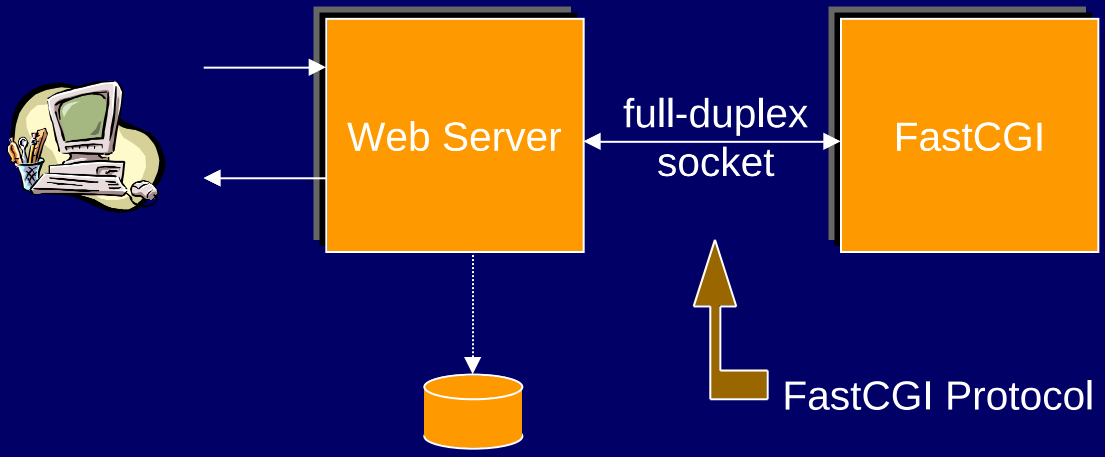
FastCGI is..

- A language and server independent, scalable, open extension to CGI that provides high performance and persistence
- A protocol for data interchange between a web server and a FastCGI application
- The set of libraries that implement the protocol

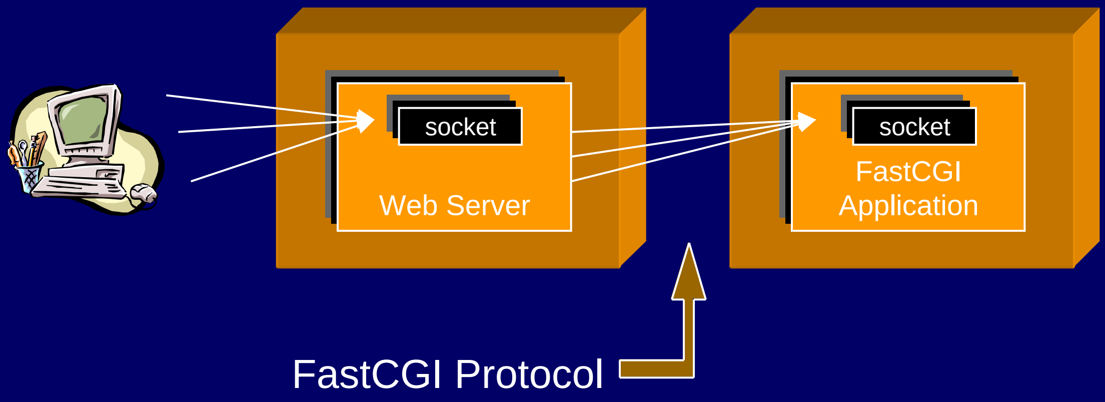
Web Server - CGI Relationship



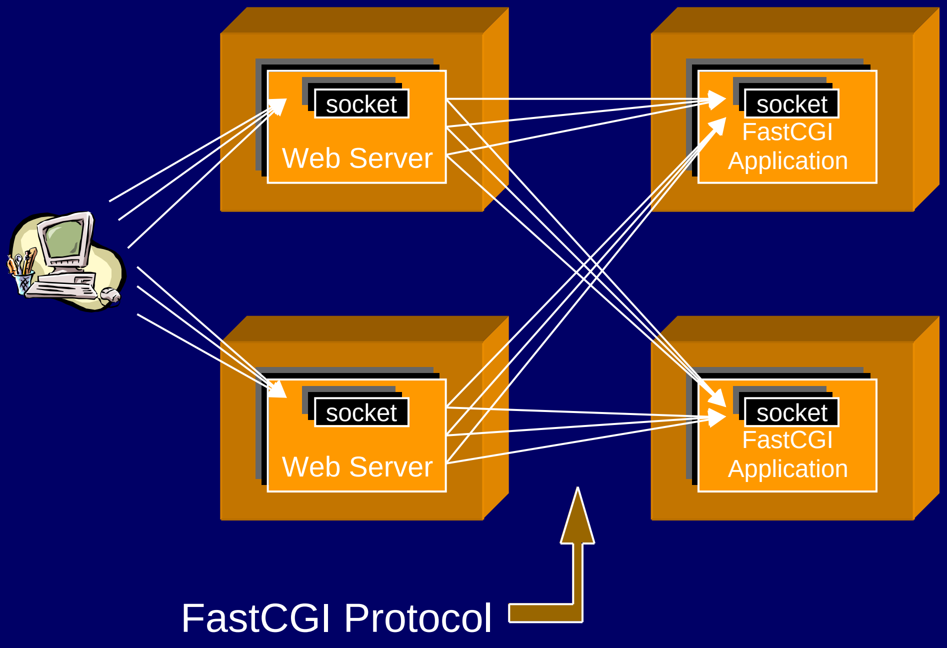
Web Server - FastCGI Relationship



Backend FastCGI Server



Piles of Servers



FastCGI Application Organization

```
Initialization

Response Loop {
    Response Handler
}
```

Perl FCGI API

- accept() – Accepts a new FastCGI connection request, implicitly finishes the current request
- flush() – Flushes output buffers
- finish() – Finishes the current request

Hello World - Perl

```
use FCGI;

$count = 0;

while (FCGI::accept() == 0) {
    print "Content-type: text/html\r\n“,
        "\r\n“,
        "<h1>Hello World</h1>\n“,
        "Request “, ++$count,
        “ from server “, $ENV{'SERVER_NAME'};
}
```

Hello World – Perl, CGI.pm

```
use CGI::Fast qw(:standard);

$count = 0;

while (new CGI::Fast) {
    print header,
        h1(“Hello World"),
        "Request “, ++$count,
        “ from server “, $ENV{'SERVER_NAME'};
}
```

How Fast?

- What’s the fork/exec mean to performance?
- Compare a tiny fcgi_stdio application in both CGI and FastCGI mode.

```
#include "fcgi_stdio.h“
main(void) {
    while (FCGI_Accept() == 0) {
        printf("Content-type: text/html\r\n“
               "\r\n“
               "<h1>FastCGI Hello</h1>");
    }
}
```

Pretty Useless Statistics (PUS)

- Using Apache’s ab (Apache benchmark) program with 20 simultaneous requests, 100/1000 total requests
- Data represents the number of requests/sec
- tiny1 sleeps for 1 sec in the middle of handling
- 1tiny sleeps for one second at initialization

	tiny.c	tiny1.c	1tiny.c
Static File		(968)	
CGI	184	16	13
FastCGI	516	1	409

Considerations

- How long will a typical request take? Are there any external dependencies?
- What type of load will the application have?
- How much data is sent to/from the client?
- Run as many processes/threads as you need
- Don’t send content if a redirect will do (e.g. banner ads)

Reasons for Using FastCGI

- No learning curve, everyone knows CGI
- Same binary works with Netscape, IIS, OpenMarket, and Apache
- Same code works across SPARC and Intel
- Allows mix-n-match OS/servers
- Choice of standard programming languages
- Flexible deployment
- No vendor tie or proprietary languages

Which Solution?

- Whether to use FastCGI, an embedded interpreter, a server API, or an application server depends on the project, personal experience and preferences.
- They’re all fast and persistent.

Summary

- FastCGI is a protocol and the set of libraries that implements it
- FastCGI provides a fairly low level toolkit for developing fast, persistent, and portable solutions

FastCGI Servers

- Apache - mod_fastcgi (free)
  http://www.fastcgi.com/
- Zeus - http://www.zeustech.net/
- Microsoft & Netscape – FastServ plug-in
  http://www.fastengines.com/

Links

- The FastCGI Home Page
  - http://www.fastcgi.com
- These slides
  - http://www.fastcgi.com/docs/OpenSource99