Server-Side Technologies

CGI, ASP and JSP

Requesting HTML Pages

URL specifies the address of a server machine as well as the location of a resource within that server

Web Server Host

Client

g2.fit.qut.edu.au

web server engine

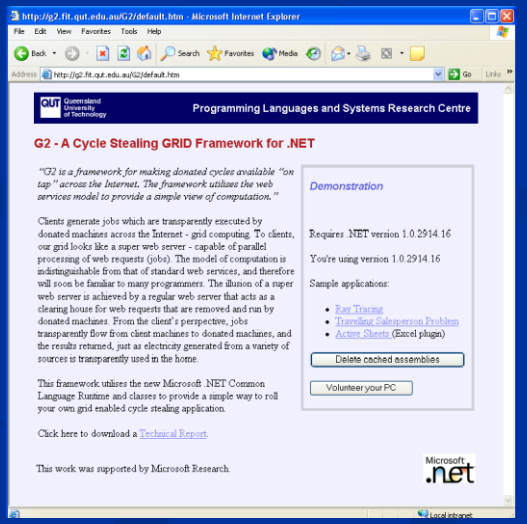
local file system

http://g2.fit.qut.edu.au/G2/default.htm

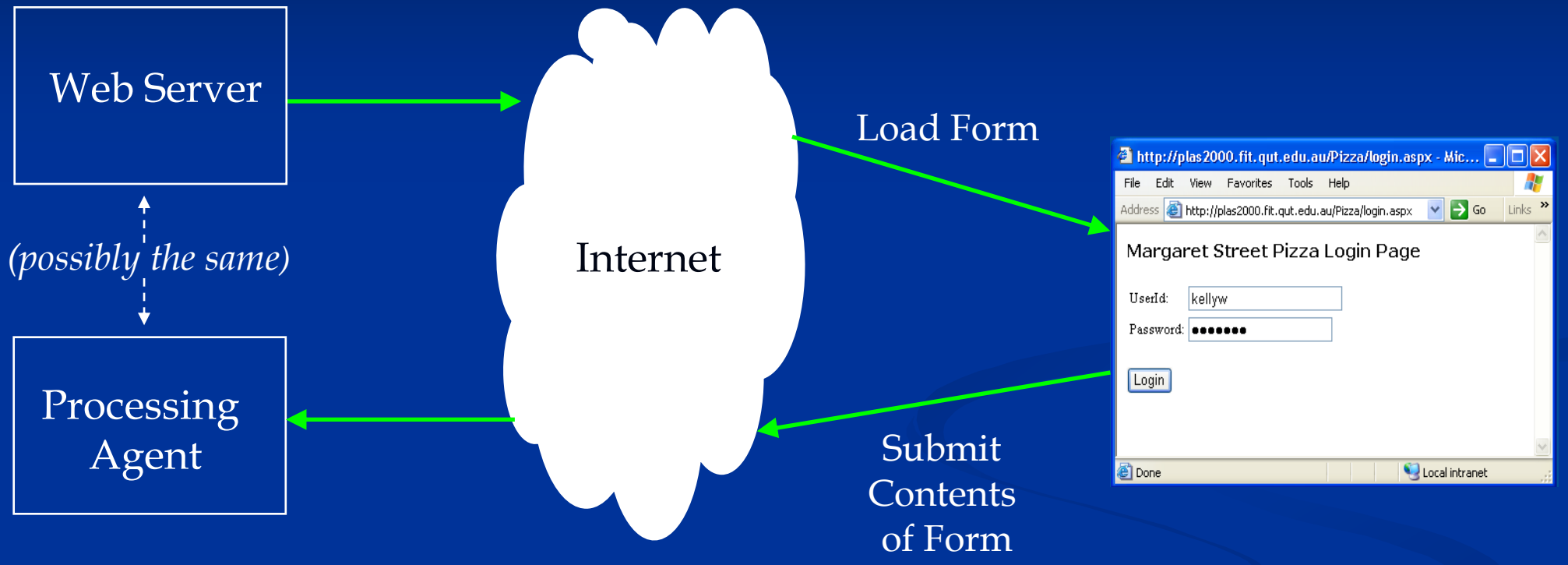
Request message containing URL

Internet

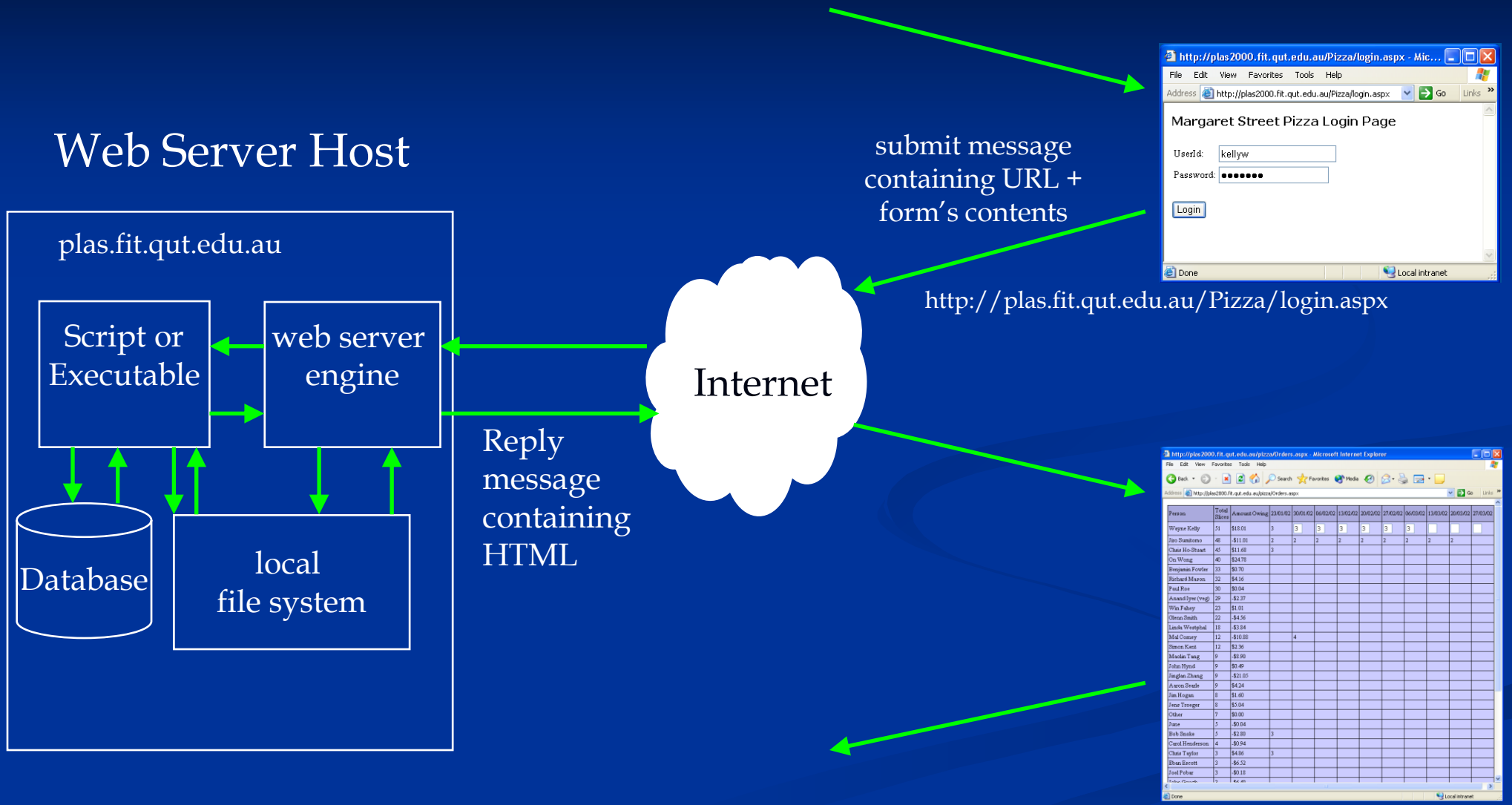
Reply message containing HTML



Loading and Submitting HTML Forms



Processing HTML Forms on Web Servers



Form Submission

- `<form name="formname" method="get|post" action="URI">`
- When the user presses the form's submit button, the "contents" of the form is sent to an agent at the specified URI.
- The contents of the form consists of a (name, value) pair for each "successful" control on the form.
- The manner in which this content is passed to the server depends on the form's `method` attribute.
- If `method="get"`, the form's content is appended to the requested URI, eg:

```
http://somesite.com/prog/adduser?comment=Hello+world&selection=option+4&sex=Male
```

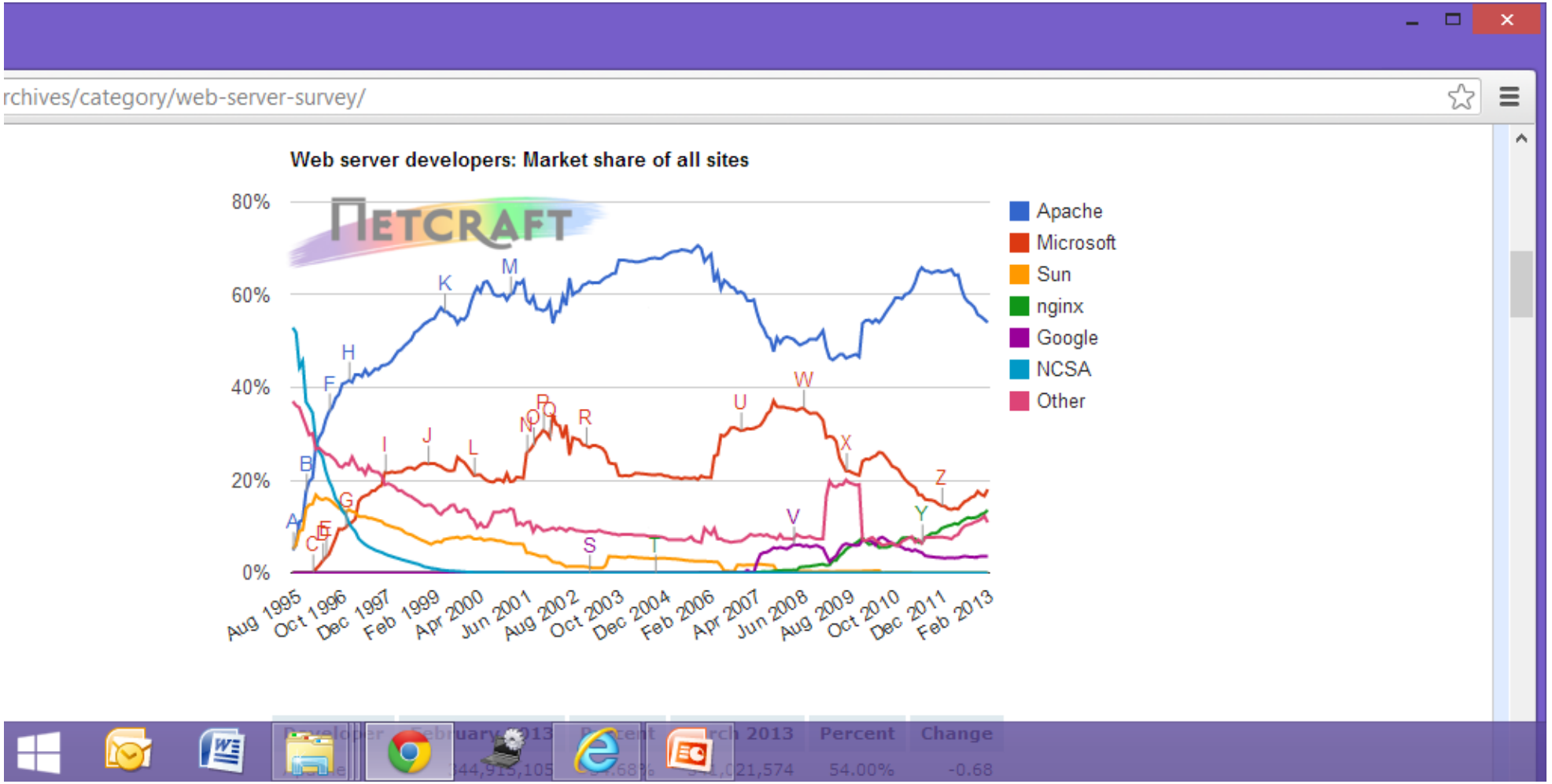
- If `method="post"`, the form's content is sent as the body of the URI request.

```
POST http://somesite.com/prog/adduser/prog/adduser HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 47

comment=Hello+world&selection=option+4&sex=Male
```

Web Server Popularity

Market Share for Top Servers Across All Domains August 1995 – February 2013



http://news.netcraft.com/archives/web_server_survey.html

Server-Side Technology Popularity

	Altavista	Google	AllTheWeb
html	909,000,000	4,800,000,000	284,552,892
htm	783,000,000	1,720,000,000	214,049,499
php	410,000,000	894,000,000	91,883,357
asp	309,000,000	771,000,000	96,172,365
cgi	116,000,000	221,000,000	35,579,537
pl	71,200,000	73,600,000	6,460,196
jsp	49,400,000	220,000,000	14,128,619
aspx	36,600,000	355,000,000	14,831,782
cfm	31,887,383	346,000,000	25,703,913

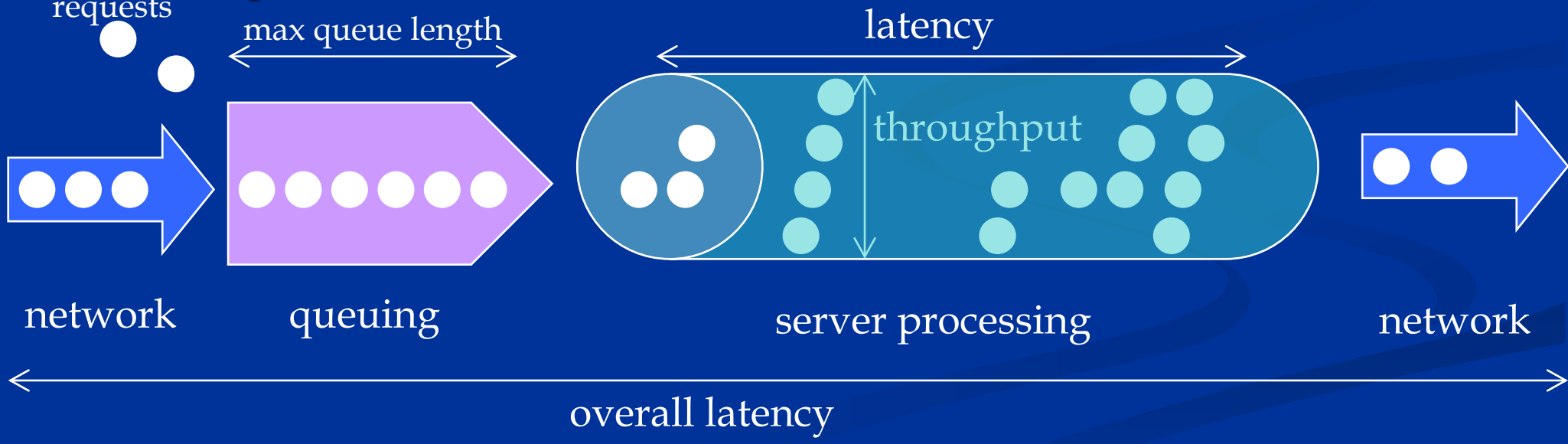
Unscientific survey: `url:????`

Issues

- Performance and Scalability
- Ease of Development and Maintenance.
- Reliability.
- Portability.

Performance

- Latency (response time)
- Throughput (maximum requests per second)
- Load (requests per second)
  - current
  - average
  - peak



Scalability

- Scalability vs Performance
  - If system A has a response time of 2 seconds under a given load and system B has a response time of 1 second under the same load, then we would say that system B exhibits better ***performance*** for that load.
  - However, if we double the load, and system A maintains a response time of 2 seconds, while system B doubles its response time to 2 seconds, then we would say that system A exhibits better ***scalability***.
- Scalability also considers whether adding additional hardware resources produces a proportional increase in throughput.
  - To **scale vertically** or **scale up**, means to add resources to a single node in a system, such as adding processors or memory to a computer.
  - To **scale horizontally** or **scale out**, means to add more nodes to a system, such as adding a new computer to a clustered software application.
- Scalability is largely about planning for the future with the expectation that the load will grow over time.

Factors affecting Performance and Scalability

- Performance:
  - Language: Interpreted, Compiled or Just-In-Time (JIT) Compiled
  - System architecture
  - Hadrware
- Scalability:
  - What is the process model?
    - in the same process as the web server? (threads)
    - in separate processes that are created for each request?
    - in a separate process that remains active?

  - How is state maintained?

Ease of Development

- What is the API for accessing the list of (name, value) pairs submitted from HTML Forms?
- How do we keep the HTML form consistent with the code that processes it's output?
  - Both will evolve over time;
  - We need to keep the list of names generated consistent with the list of names expected.
- A web application is a program that generates HTML.
  - When writing HTML manually it's important to use indentation to keep clear in our minds the hierarchical relationship between the elements.
  - This becomes difficult if not impossible when snippets of the HTML to be generated are distributed throughout a source program written in some other computer language.
- How do we separate UI from logic?

Reliability

- What happens if a web application crashes or is badly behaved?
  - Can it bring down or corrupt the web server itself or other web applications?
  - Depends on process model:
    - in the same process as the web server?
    - in separate process
      - in the same process as other web applications
      - in a process separate from other web applications
  - Are the web applications coded in a type safe language?
    - does the runtime environment guarantee safety?
  - Do processes get recycled?

History of Server-Side Web Processing

- 1993 Common Gateway Interface (CGI)
- 1996 Microsoft ISAPI and ASP 1.0 (Active Server Pages).
- 1997 PHP 3.0
- 1997 Sun's Java Servlets
- 1998 Sun's Java Server Pages (JSP)
- 2001 Microsoft's ASP.NET
- 2004 Ruby on Rails

CGI Architecture Diagram
The following simple program shows a simple architecture of CGI:

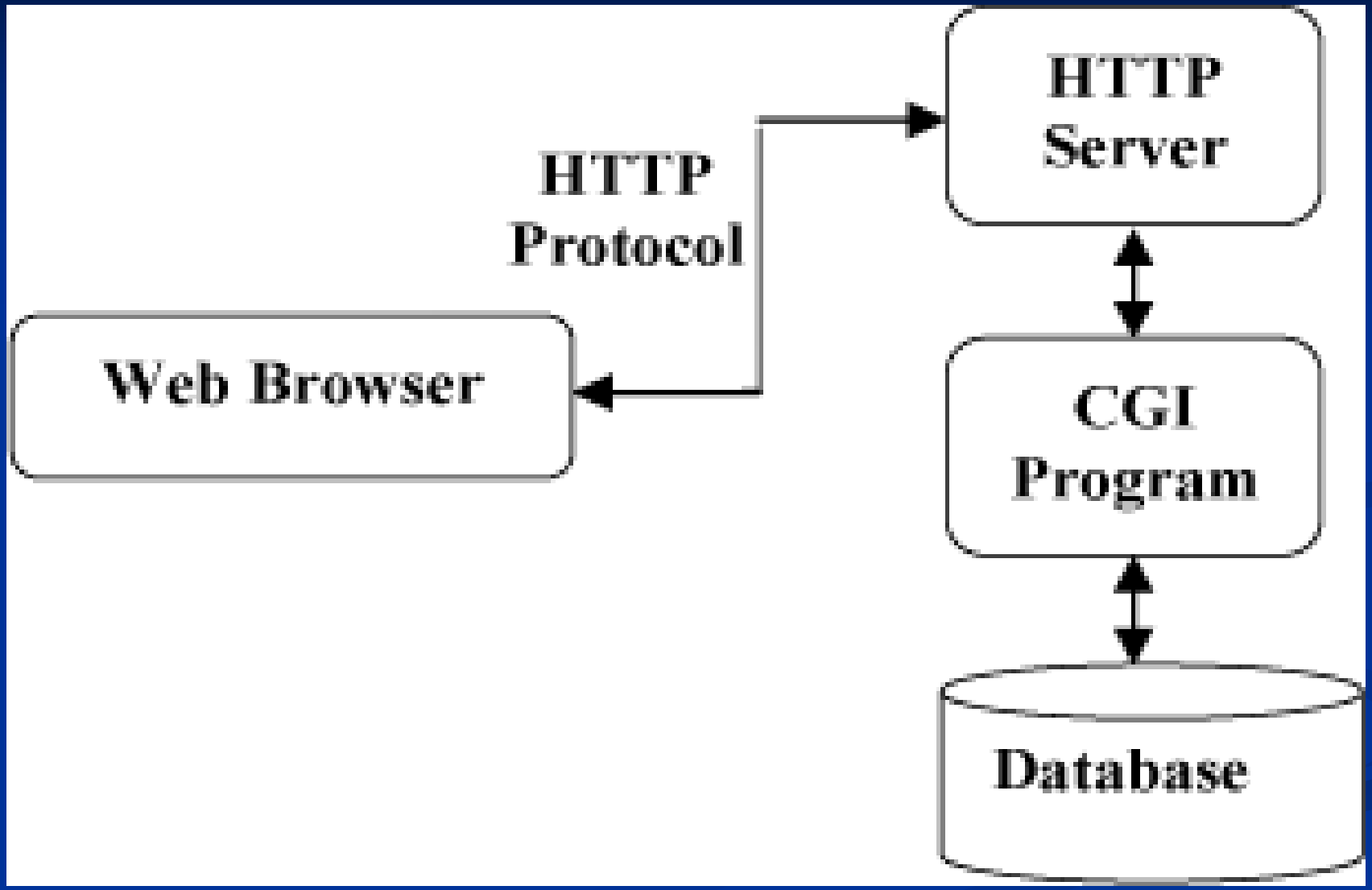
Common Gateway Interface (CGI)

- Rather than sending back the content of a static html text file, the web server executes a cgi application and sends its output.
- CGI Applications can be implemented in any language.
- The following languages are often used because of their strong string processing abilities and their simplicity:
  - Perl
  - VB
  - Python
- Many of these are interpreted scripting languages, but this needn't be the case.

CGI Architecture Diagram

The following simple program shows a simple architecture of CGI:

CGI Architecture



How CGI Works

- Rather than referring to a HTML file, the requested URI refers to a executable or script file.

  ```
  http://sky.fit.qut.edu.au/~kellyw/cgi-bin/test
  ```

  - The executable or script file must be placed in a specially configured cgi-bin directory.
    - otherwise, the web server will return its contents rather than executing it.
    - this is because cgi applications can open serious security holes.
  - A new process is created for each cgi request.
  - In the case of a script file, the web server needs to work out which interpreter to launch.
    - the path to the interpreter is specified in the first line of the script, eg:

      ```
      #!/bin/sh
      ```

CGI Input and Output

- The input mechanism depends on whether the submitted form used the "post" or "get" submit method.
  - In the case of the "get" method, the encoded contents of the form is made available via an environment variable called "`QUERY_STRING`".
  - In the case of a "post" action, the encoded contents of the form is made available via stdin and the length of that input is available via an environment variable called "`CONTENT_LENGTH`"
- The output of the cgi program must contain a content type followed by a blank line, followed by the body of the reply message. eg:

```
Content-type:text/html

<html><head><title>Test Page</title></head>
<body>
<h2>Hello, world!</h2>
</body></html>
```

Simple CGI Example (no input)

Perl:

```
#!/usr/bin/perl
print "Content-type:text/html\n\n";
print "<html><head><title>Test Page</title></head>\n";
print "<body>\n";
print "<h2>Hello, world!</h2>\n";
print "</body></html>\n";
```

C++:

```
#include <iostream>
using namespace std;

void main()
{
    cout << "Content-type:text/html" << endl << endl;
    cout << "<html><head><title>Test Page</title></head>" << endl;
    cout << "<body>" << endl;
    cout << "<h2>Hello, world!</h2>" << endl;
    cout << "</body></html>" << endl;
}
```

CGI Form Processing Example

- Form processing cgi programs need to parse their input and extract a list of names and values.

  eg: `fname=Thomas+John&lname=Weaver&email=tweaver@cs.umd.edu`

- Perl's string processing and associative array features make it ideal for this task:

```
!/usr/local/bin/perl
read (STDIN, $temp, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/,$temp);
foreach $item (@pairs)
  {
    ($key, $content) =split(/=/,$item,2);
    $content=~tr/+/ /;
    $content=~s/%(..)/pack("c",hex($1))/ge;
    $fields{$key}=$content;
  }
print "Content-type: text/html\n\n";
print "<HTML>\n";
print "<BODY>\n";
print "Thank you $fields{fname} $fields{lname} <BR>\n";
print "I will write to you at $fields{email}\n";
print "</BODY>\n";
print "</HTML>\n";
```

CGI Pros and Cons

- Pros:
  - can program in any language.
  - efficiency of native executables (not true for cgi scripts).
  - platform independent standard.

- Cons:
  - creates a separate process for each request.
    - doesn't scale.
  - need to manually decode form's content.
    - tedious
  - need to keep server-side code in sync with client-side form.
    - easy to introduce bugs as the form and code evolve.
  - need to write code to output all of the HTML (and embedded scripts)
    - not very easy to read.

CGI References

- Tutorials:

  `http://www.pageresource.com/cgirec/ctutors.htm`

  `http://hoohoo.ncsa.uiuc.edu/cgi/intro.htm`

- Specification:

  `http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`

Internet Server API (ISAPI)

- ISAPI was an API for directly interfacing with Microsoft's IIS.
  - IBM and Netscape had their own APIs (NSAPI and ICAPI)
- Used to create:
  - Extension applications (equivalent to cgi applications)
  - Filters (for redirecting, monitoring and fine-tuning web server behaviour)
- Pros: could dynamically link dlls directly into IIS.
  - better performance (both processor time and memory)
    - don't have to create new processes
    - can pool threads.
    - communication happens in-process.
- Cons: Even uglier than CGI.
  - Later developed MFC wrapper classes and Visual C++ Wizards.

ISAPI Example

```
#include "httpext.h"
#include <stdio.h>

BOOL WINAPI GetExtensionVersion (HSE_VERSION_INFO *pVersion)
{
    pVersion->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
    lstrcpyn(pVersion->lpszExtensionDesc, "My ISAPI Extension", HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
}

DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    ...
    sprintf(header, "Content-Type: text/plain\r\nContent-Length:%d\r\n\r\n", pECB->cbTotalBytes);
    pECB->ServerSupportFunction(pECB->ConnID,HSE_REQ_SEND_RESPONSE_HEADER,NULL,&headLng,header);

    ...
    pECB->WriteClient(pECB->ConnID, writeBuffer, &writeBufferLength, 0);

    ...
    pECB->ReadClient(pECB->ConnID, readBuffer, &readBufferLength);

    return HSE_STATUS_SUCCESS;
}
```

Java Servlets

- Java Servlets are the Java equivalent of ISAPI
  - but with a much cleaner/more high-level interface.
- Servlet containers exist for most web servers (Apache, IIS, iPlanet)
  - Tomcat is a popular open-source servlet container.
    - It is a complete Java based web server, but can also be used as an add-on to other web browsers.
    - IIS redirector implemented using ISAPI.
    - Netscape Enterprise Server redirector implemented using NSAPI.
    - Apache adapter implemented as an Apache module (mod_jserv)
- Typically execute within the same process as the web server.

Servlet Example

```
import javax.servlet.*;
import javax.servlet.http.*;

public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                  throws ServletException, IOException
{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    string title = "Reading Three Request Parameters";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY>\n" +
                "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                "<UL>\n" +
                "  <LI>param1: " + request.getParameter("param1") + "\n" +
                "  <LI>param2: " + request.getParameter("param2") + "\n" +
                "  <LI>param3: " + request.getParameter("param3") + "\n" +
                "</UL>\n" +
                "</BODY></HTML>");
  }
```

Servlet References

- Specification:

  `http://java.sun.com/products/servlet/download.html`

Tutorials:

`http://java.sun.com/docs/books/tutorial/servlets/`

`http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/`

Server Side Scripting

- Many dynamically built web pages are mostly static. CGI, ISAPI and Servlets make you generate the entire page via your program, even though most of it is always the same.
- Server-side scripting environments allow you to include server-side scripts in HTML documents (as well as client-side scripts).
  - Server-side scripts are interpreted by the web server and translated into HTML before being sent to the client, so the client's browser doesn't even see the server side scripts.
- Server-side scripts are like CGI programs in that they can access server-side resources such as databases, but they can't directly interact with the user.
  - they can't for example, directly pop up a message box.
- Like ISAPI and Servlets, server-side scripts are typically executed within the same process as the web server.
- A number of different server-side scripting environments exist:
  - Microsoft's Active Server Pages (ASP)
  - Sun's Java Server Pages (JSP)
  - Hypertext Preprocessor (PHP)
  - Microsoft's ASP.NET

Active Server Pages (ASP)

- Active Server Pages are identical to HTML documents, except they:
  - have an .asp extension rather than a .html extension
  - contain one or more (server-side) ASP scripts.
- ASP scripts are delineated by:

  `<% ... %>` or

  `<SCRIPT RUNAT=SERVER> ... </SCRIPT>`
- Normally only work with Microsoft's Internet Information Server (IIS)
  - but can purchase third party "emulators" for other servers (ChilliSoft).
  - use COM to communicate with server-side components.
- Like client-side scripts, ASP scripts can be coded in a variety of scripting languages, but the default is VBScript.
- Server-side script code is *interpreted* on the server.
- Provides standard request and response objects for accessing and manipulating data passed to and from the client. eg:

```
<%
If Request.QueryString("userstatus") = "new user" then
    Response.Write "This is your first visit to this Web site!"
End if
%>
```

Maintaining State

- State management is the process by which state and page information is maintained over multiple requests for the same or different pages.
  - as is true for any HTTP-based technology, ASP pages are stateless, which means the server cannot automatically tell whether a sequence of requests are all from the same client.
- Client-side options:
  - Cookies
    - stored on the client (may be refused), passed to the server along with the request.
  - Hidden fields
  - Query string
- Server-side options:
  - Session state
    - shared by a number of related pages accessed by a single client.
  - Application state
    - shared by a number of related pages across all client accesses.
  - Database

Other Standard ASP objects

- Server object (COM):

```
<%
    Set objConn = Server.CreateObject("ADODB.Connection")
    ...
%>
```

- Cookies:

```
<%
    Response.Cookies("firstname")="Alex"
    Response.Cookies("firstname").Expires="May 10,2002"
    ...
    fname=Request.Cookies("firstname")
    ...
%>
```

- Session and Application objects:

```
<H1> Welcome <%Response.Write(Session("username"))%> </H1>
You are visitor number <%=Application("hitcounter")++%>
```

ASP Example

```
<%
   Set rs = Server.CreateObject("ADODB.RecordSet")
   rs.Open "SELECT * FROM mytable", "DSN=example;"
%>
<html>
   <head> <title>ASP Example</title> </head>
   <body>
      <table>
         <tr> <td>Name</td> <td>Subject</td> <td>Date Posted</td> </tr>
         <%
            y = 0
            while NOT rs.EOF
         %>
            <tr>
               <td><%=rs("name")    %></td>
               <td><%=rs("subject")%></td>
               <td><%=rs("date")    %></td>
            </tr>
         <%
               rs.MoveNext
               y = y + 1
               if y = 10 then
                  while NOT rs.EOF
                     rs.MoveNext
                  wend
               end if
            wend
         %>
</table>
```

ASP References

- Tutorials:

  `http://www.w3schools.com/asp/`

- Specification:

  `http://msdn.microsoft.com/library/psdk/iisref/aspguide.htm`

Open Source LAMP software

- Linux http://www.linux.org
- Apache http://www.apache.org
- MySQL http://www.mysql.com
- PHP http://www.php.net

PHP (Hypertext Pre-Processor)

- Advantages:
  - Cross-platform support (PWS, IIS and Apache web servers)
  - Open Source, developed by Rasmus Lerdorf in 1994.
  - Language specifically designed for the web.
  - Typically runs in process.
  - Excellent string processing capabilities (like Perl)
  - Tight integration with MySQL (fast)
  - Zend optimizing compiler (available commercially)
- Disadvantages:
  - Quick and dirty ("stubborn function-over-form approach").
  - Poor error handling
  - "Tedious" objected-oriented programming support.
  - Normally interpreted

PHP Example

```
<html>
    <head>
        <title>Result of Database Query</title>
    </head>
    <body>
        <h1>Result of Database Query</h1>
        <?
            $dbcon=mysql_connect("clun.scit.wlv.ac.uk","demo");
            mysql_select_db("mydatabase");
            $sql="SELECT * FROM gazetteer WHERE feature = '" . $place ."'";
            $result = mysql_query($sql);
            $nrows = mysql_num_rows($result);
            if($nrows != 0)
            {
                print "<p>Data for " . $place;
                print "<table border=2><tr><th>Latitude<th>Longitude<th>Easting<th>Northing\n";
                for($j=0;$j<$nrows;$j++)
                {
                    $row = mysql_fetch_array($result);
                    print "<tr><td>" . $row["latitude"];
                    print "<td>" . $row["longitude"];
                    print "<td>" . $row["easting"];
                    print "<td>" . $row["northing"];
                    print "\n";
                }
                print "</table>\n";
            }
            else    print "<p>No Entry for " . $place;
            mysql_close($dbcon);
        ?>
        </p>
    </body>
```

Java Server Pages (JSP)

- Java Server Pages are HTML documents containing one or more JSP scripts.
- Like ASP, JSP scriptlets are delineated by:

  `<% ... %>` or

  `<SCRIPT RUNAT=SERVER>` tags.
- JSP scripts are implemented in full blown Java (not just JavaScript).
- JSP is 100% Java and therefore more platform independent than ASP.
- JSP pages are converted by the server into servlets.
- The entire file is converted into pure Java:
  - the <% and %> disappear
  - HTML and client-side JavaScript is converted to Java output statements.
  - is JIT compiled at request time – more efficient than interpreted VBScript.
- Can communicate with server-side JavaBean components.
- Like ASP, it provides predefined variables called `request`, `response`, `out`, `session` and `application`.

Advanced JSP

- Excessive server-side script code makes the static HTML portion of JSP pages difficult to read and maintain.
- Should try to minimize scriptlet code by using:
  - JavaBeans:

```
<jsp:useBean id="conv" class="Converter" />
…
<p> Temperature is <% =conv.ToCelcius(32.0) %>
```

  - or, Custom tags:

```
public class Greeting extends TagSupport {
    public int doStartTag() {
        JspWriter out = pageContext.GetOut();
        out.print("<em> Dear Sir/Madam, </em>");
        return SKIP_BODY;
    }
}

<%@ taglib url="letterlib.tld" prefix="Letter">
…
<Letter:Greeting />
```

JSP References

- Tutorials:

  `http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/`

  `http://www.coreservlets.com/`

- Specification:

  `http://java.sun.com/products/jsp/download.html`