iMatix Corporation sprl

13/15 rue des Ateliers, 1080 Bruxelles, Belgium
Tel.+32-75-235-984, fax.+32-2-414-8404
http://www.imatix.com
info@imatix.com

Technical White Paper

Constructing Fast Internet Services

We discuss the importance of speed when designing Internet services. We then look at several alternatives approaches, and finally explain OpenKernel, our platform for constructing fast, portable Internet services.

This document is aimed at a technical audience with a knowledge of software development issues, and an interest in understanding the mechanisms involved in building rapid, portable Internet servers.

Copyright

Version Information

Written: 1 December 1999
Revised: 23 January 2000

Disclaimer

The information contained in this document is distributed on an “as-is” basis without any warranty either expressed or implied. The customer is responsible for the use of this information and/or implementation of any techniques mentioned. iMatix Corporation has reviewed the information for accuracy, but there is no guarantee that a customer using the these techniques and/or information will obtain the same or similar results in its own operating environment.

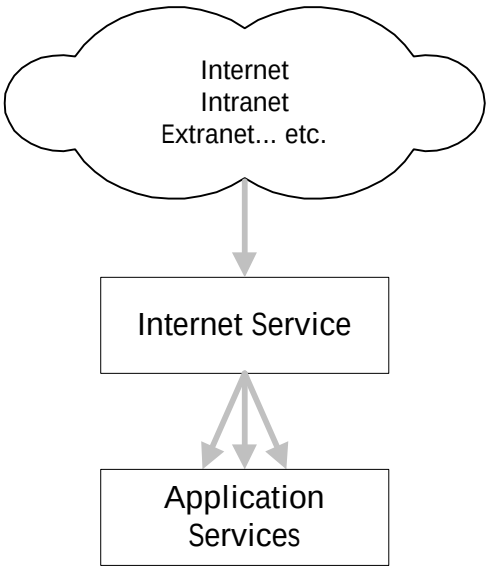
It is possible that this material may contain references to, or information about, iMatix Corporation products or services that have not been announced. Such references or information must not be construed to mean that iMatix intends to announce such products or services.

iMatix Corporation retains the title to the copyright in this paper, as well as title to the copyright in all underlying works. iMatix Corporation retains the right to make derivative works and to republish and distribute this paper to whoever it chooses to.

What is an Internet Service?

Defining an Internet Service

An *Internet Service* is simply a TCP/IP server connected to the Internet and handling some protocol:



A number of standard services are well-known and provided with most operating systems:

- Web and file transfer services;
- E-mail fetching and sending;

Some less visible but equally important services are:

- Domain-name lookups;
- Certificate authentication.

Finally, organisations can create their own services by writing or customising TCP/IP servers to handle specific protocols. Such services can often be business-oriented in a way that standard services cannot be.

Businesses can exploit the Internet by developing their own, specialised services.

Why Internet Services are Special

In an enclosed intranet, the maximum traffic is known and the necessary capacity can be designed into the system from the start.

A successful Internet service may end-up handling millions of requests per day.

Success brings traffic.

An Internet service can never be fast enough.

An Internet service can never be robust enough.

The Importance of Internet Services

The world is moving towards more and more back-end use of the Internet to drive supply chains and order chains as well as the current front-end trend towards using the Internet for front-end purchasing.

Such moves to exploit the Internet require fast, reliable Internet services.

Internet services form the backbone of the Internet.

Control over the Internet requires control over the functionality and quality of Internet services.

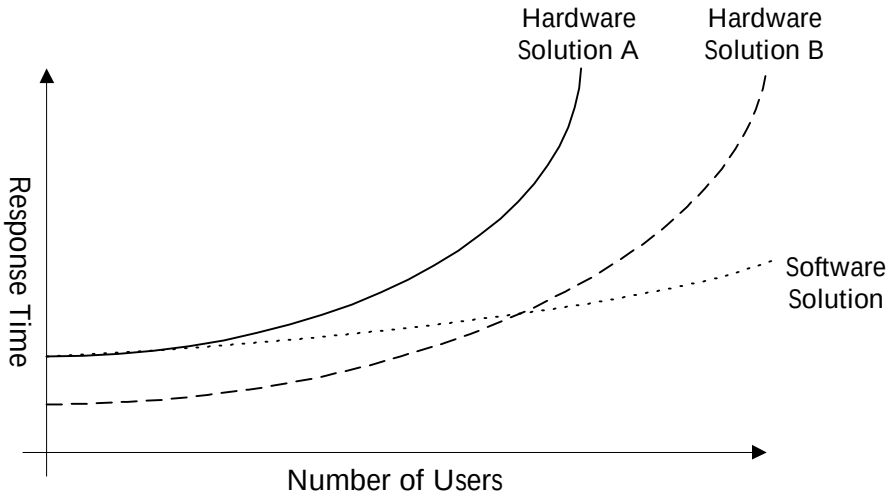
Designing for Speed

There are many ways to get more speed out of a software application:

- Use a faster system, with a faster CPU or multiple CPUs, more memory, faster disks, and a faster network connection.
- Use a smarter software design.

Any software application follows a curve of performance against usage. The more users there are, the slower the system reacts. Because of inefficiencies, this is not a straight line but a curve that tends towards infinity as the number of users grows. Hardware gains provide strictly linear improvements: a system that is twice as fast will support twice as many users. Software gains can provide exponential improvements: a system that is tuned to be twice as fast may support four times the number of users. This is because software tuning directly eliminates or reduces inefficiencies that themselves work exponentially to lower performance, while hardware tuning does not change these inefficiencies.

The following diagram shows this type of curve:



We see a hardware solution A reaching a *thrashing point* at which no useful work gets done any longer. By installing a hardware solution B that is twice as fast, we get an improvement in performance, but we still reach a thrashing point fairly quickly because we have not resolved the basic inefficiencies of the software. With a software solution we can flatten the curve and prevent thrashing even with high numbers of users.

Software is cheaper than hardware.

It helps to understand what kind of 'inefficiencies' we're talking about. An Internet service consists of a server program that responds to requests on some TCP/IP port, and talks to a client program using some agreed protocol.

Most protocols use many short connections, so the time and cost taken to create a connection is vital to performance. Older Unix servers handle new connections by

duplicating the entire server process. This is fast, but not fast enough. Newer servers pre-allocate a number of server processes. This is faster, but consumes large amounts of memory, and still leaves a limit. Systems like Windows NT handle multiple connections as threads in one process. This is very fast. But it's not portable. That means you can't move such a program from a (relatively slow) PC to a much faster RISC server.

Existing approaches to constructing servers are not satisfactory.

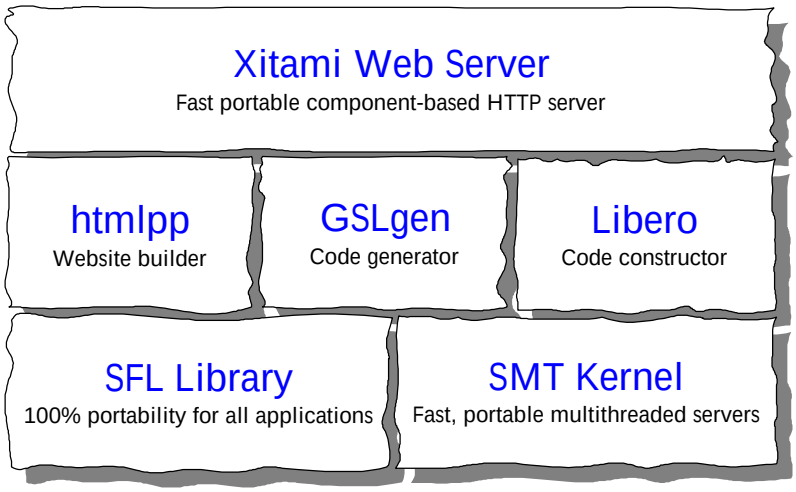
The ideal software approach has these features:

1. It eliminates inefficiency so that connections are served at the strict minimum cost. This reduces the server reaction time and keeps the performance curve close to a flat line.
2. It is scalable. Since the server performance curve is close to flat, the number of users can be increased almost indefinitely.
3. It is portable. A server can be moved to any computer system that is newer and faster. With non-portable solutions, the scope for hardware upgrades is limited and often runs into dead-ends.

The OpenKernel Solution

OpenKernel is unusual. There are many excellent Internet servers. There are many portable Internet servers. There are many fast Internet servers. There are even a few excellent, portable and fast Internet servers. But there are no toolkits that let anyone construct their own excellent, portable, fast Internet server.

The overall architecture of OpenKernel is:



OpenKernel consists of a set of tools that can work independently and together. These tools have a long history of development and use, and have been available separately from the www.imatix.com web site since 1995. They are all Open Source tools.

- SFL is a library that encapsulates the operating system functions and thus lets developers construct portable applications that use the operating systems socket, process control, file access functions. SFL also provides a rich set of functions to handle compression, conversion, etc.
- SMT is the core of the OpenKernel system. It provides an event-driven multithreading architecture that is simpler and faster than POSIX threads and platform-specific multithreading functions. SMT threads execute according to a finite-state machine model which eliminates concurrency and critical section issues.
- Libero is the iMatix code construction tool. We use a finite-state machine model to design the logic of components and programs. This model converts directly to an executable code framework in about fifteen programming languages. The Libero code generation process is easily customisable to handle specific requirements – for instance, we use a highly-customised 'schema' for SMT components.
- GSLgen implements the iMatix 'Generator Script Language', a powerful tool for constructing code generators, XML transformation tools, reporting tools, and more. GSLgen can be embedded in OpenKernel applications, giving them the full power of GSL.

- Xitami (see xitami.com) is a full-featured web server that is built using OpenKernel. It also provides valuable HTTP and FTP components for OpenKernel applications. For example: it is quite easy to construct a web-driven administration interface for applications which is handled directly by an embedded Xitami web server running on a private HTTP port.
- Htmlpp (see htmlpp.org) is our tool for web site creation. It is a sophisticated factory that lets you generate entire web sites from scripts. Htmlpp will create links, tables of contents, indexes, split long documents into pages, etc.

OpenKernel has these principal features:

- Fully portable to Windows, OS/2, Linux, Unix, OpenVMS.
- Completely Open Source. The OpenKernel licensing agreement does not place restrictions on the way you distribute, sell, or use products built using it.
- Tried and tested since 1995 in a raft of products and applications.
- Support for HTTP/1.1, FTP, XML, and other Internet standards.