PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification :
Not classified

A2

(11) International Publication Number: WO 00/41455

(43) International Publication Date: 20 July 2000 (20.07.00)

(21) International Application Number: PCT/CA00/00027

(22) International Filing Date: 12 January 2000 (12.01.00)

(30) Priority Data:
60/115,635 12 January 1999 (12.01.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 09/481,628 (CON)
Filed on 12 January 2000 (12.01.00)

(71) Applicant *(for all designated States except US)*: DIGITAL VIDEO BROADCASTING INC. [CA/CA]; 608–1200 Pender St. West, Vancouver, British Columbia V6M 2W2 (CA).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: SOKOL, John, L. [US/US]; 136 Maplewood Ave., Clinton, NJ 07013 (US).

(74) Agent: SCHISLER, Mark; Bereskin & Parr, 40 King St. West, 40th Floor, Toronto, Ontario M5H 3Y2 (CA).

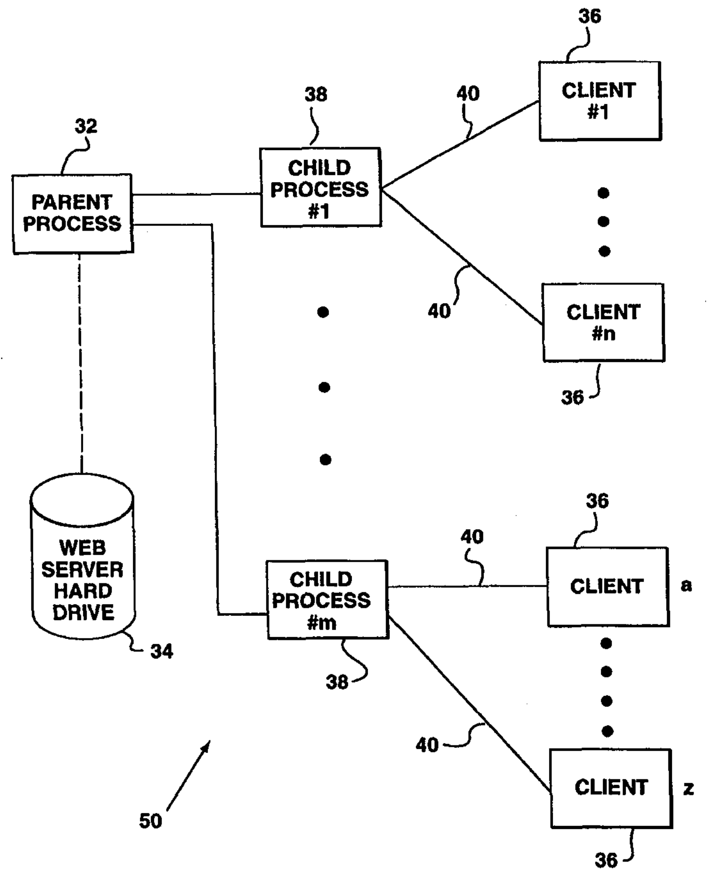
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: HIGH PERFORMANCE WEB SERVER

(57) Abstract

The invention is a high performance web server. A single server process on the web server serves data to a plurality of requesters. Should the number of requesters become too great for a single process, further processes will be spawned as required. By utilizing as few processes as possible to serve a plurality of requesters, system overhead to manage processes is greatly reduced over existing web servers. In addition, the invention acts as a cache server by serving data to the requesters from RAM.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

Title: HIGH PERFORMANCE WEB SERVER

FIELD OF THE INVENTION

The present invention relates to the field of computers connected to the internet that act as servers of data. In particular, the present invention relates to a high performance web server or cache server designed to provide rapid and reliable access to large amounts of data.

BACKGROUND OF THE INVENTION

Most people know about the Internet. Few people, however, understand how it works and the details of how computers connected long distances via a common set of protocols and on a shared telecommunication resource works. The present invention is a revolutionary Internet server, or information provider as another way of looking at it, that can simultaneously serve more visitors to a given website than other servers (which are based on more traditional techniques). An explanation of how it does this first requires a more detailed understanding of how the Internet works and its rapid improvement in technology over the past 30 years.

In the Early 1960's in the RAND corporation , America's foremost Cold War think-tank was faced with a problem. How to make a communications network that could withstand a nuclear attack? Any traditional network at that time was like the phone company with central command and control location, should a problem occur with the control center all systems would stop. In 1964 Paul Baran made public a plan for a network that would have no central authority and was designed to operate with major parts of the network destroyed. The principles are simple. The network is assumed to be unreliable at all times, and it is designed to work around sections that are not working. Each machine (node) on the network has equal authority with every other node. The messages passed between nodes are broken into smaller pieces called packets. Each packet contains the address of where it is going and a checksum to verify that it is correct. The nodes on the network pass each packet along in the direction of its final destination until it arrives. The path that these packets take may be different for each and it is possible that a packet may get lost, copied, or arrive in a different order than it was sent. It is the responsibility of the destination node to collect and organize these packets into the original message and ask for missing packets if any do not arrive after a certain period of time.

This concept was funded in 1969 by the Pentagon's Advanced Research Projects Agency (ARPA) to connect several supercomputers around the country, it was called the ARPANET. Initially the NCP or Network Control Protocol was used. A protocol describes the way the computers on the network communicate with one another, make decisions and resolve problems. NCP was then replaced with an improved protocol called TCP/IP or Transmission Control Protocol/Internet Protocol. One of the major things about TCP/IP is that it makes no assumption about the type of connection that it is being used. TCP/IP allows the internet to go everywhere, it is one of the few protocols that will work for both a LAN (Local Area Network) and a WAN (Wide Area Network).

Over the years the number of computers joining the ARPANET grew rapidly. One of the most popular uses of the ARPANET became e-mail, but this was limited to only those who could afford the expense of special leased lines to directly connect to the ARPANET. As a solution for those unable to receive e-mail a new method was devised in 1976. UUCP (Unix to Unix Copy Program) was developed to pass e-mail along from one UNIX machine to another UNIX machine over standard phone lines using modems. UUCP uses a store and forward method and as such is slow and complicated, but it did extend the reach of e-mail. This network became known as the USENET and although it is not a part of the Internet it has played a vital role in the development of the Internet. Machines called gateways have been established that allow mail, news and even file transfers to pass between the Internet and other networks like the USENET.

In 1982 - 1983 the ARPANET officially switched to TCP/IP from NCP. The TCP/IP protocols are in the public domain and with the released of BSD UNIX version 4.2 from Berkeley, a cheap, reliable, operating system became available with full TCP/IP support. BSD UNIX came with full source code allowing people to port this operating system and networking tools to almost every type of computer made at the time. All the networking software was free and it was almost impossible to stop people from connecting to the Internet.

Universities around the country were linking together and installing Ethernet across their campuses. Ethernet is a LAN networking standard that originally used a 1/2 inch round bright orange cable run throughout ceilings. It was very simple to plug in this cable and obtain instant high speed access to almost every computer in every university around the world. The demand for this was enormous. After a point, most college and universities offered free shell accounts to students and faculty. A shell account is an account on a UNIX machine that allows you run a TCP/IP program on that machine and see the results over a terminal. Soon people were able to call from home using modems and check their e-mail, transfer large files using FTP (File Transfer Protocol), read thousands of news groups or operate a computer on the other side of the country just as if they were sitting directly in front of it. With access to such a powerful system, students started developing more software for the Internet, most of it in the public domain. As these students left school the demand to use these services from home and business brought about many private companies that did nothing else but offer university like shell accounts for UNIX machines. As the price of modems dropped and the speeds became higher new things became possible. One of these was SLIP and later PPP, which allowed TCP/IP to travel over the temporary modem connections allowing home users to be directly connected to the Internet for the first time. FTP servers became popular as a way to exchange free software and documents and the demand on some of these servers became high.

Even so, navigating the Internet was a complicated procedure. On-line users were forced to memorize long, esoteric UNIX commands or deal with awkward programs. Many attempts were made to bring these services to computers running Microsoft Windows and Macintosh with mild success. In 1989 Tim Berners-Lee, a European scientist, developed the World Wide Web and released the first web server CERN-HTTPD. The World Wide Web was text based and had only a small impact until around late 1993 when the National Center for Supercomputing Applications (NCSA) released Mosaic, the first true web browser into the public domain. They also released NCSA-HTTPD their implementation of a WEB server. Mosaic added a windows interface to the Internet. Several other important innovations also came with Mosaic including URL (Uniform Resource Locators) and MIME (Multipurpose Internet Mail Extensions) types that allow servers to tell what type of data is in a file being sent. Mosaic supported most Internet applications, but the World Wide Web with a hypertext interface allowing images and text to be displayed at the same time soon came to dominate.

The World Wide Web made the Internet extremely easy to navigate and allowed an explosion of new content to take place with video, sound, text, and pictures on virtually any subject. With the content came an exponential flood of new Internet users. Suddenly the Internet became a serious business and drove the demand for new applications such as: e-commerce, advertising, and live event broadcasting. Performance reliability and security have suddenly become a multi billion dollar industry.

Through out all of this the basic server architecture has changed little. Most servers are based on the early example set by FTP. The FTP architecture is very simplistic and thus not very efficient and was never intended to support the type of loads of the current Internet.

This rather lengthy discussion leads to web servers and how the phenomenon of the Internet has fueled growth for this niche market. Ultimately, Internet growth can only grow as fast as new technology is developed to host, translate, and transport information faster and more efficiently. The present invention will make a significant impact in this competitive market.

The following is a description of how a user would connect to an ISP (Internet Service Provider) and obtain web pages and graphics from a web server. First, the user connects from their "client" PC to an ISP via a communication link and requests a file (for example an image). The ISP receives the transmitted request and forwards it to the web server that controls access to the file. The web server then processes the request and sends the requested file back to the ISP. The ISP then transfers the requested file to the client PC. It is important to note that the transmission of the request for the file and the transmission of the file may occur over communication links such as telephone lines, coaxial cable or wireless transmissions, or any other form of communication allowing the various computers to exchange data. Further, the request and transmission may pass through multiple machines in an indirect path as is the nature of the internet. In this scenario the web server is only processing one request from one ISP, but typically, a given web server is connected to a plurality of other machines on the Internet and if popular enough could receive thousands of hits (i.e. requests) per second.

Current web servers are largely based on research and development completed decades ago. This may not seem very old but Internet hardware and software solutions have been maturing so rapidly that what worked ten years ago, simply cannot suffice for the demands placed on servers today. Chief among the problems current web servers face is poor socket programming (i.e. communication connections) which leads to limitations in both response times and the maximum number of connections that a web server can provide in a given moment.

During the early phases of development of the Internet, researchers used FTP servers to transfer documents across to other researchers. As the number of nodes on the Internet at the time was not great, these servers were programmed to use a new thread process for each request made to the server. A user accessing the FTP server is given permission (read, write, modify) based on their username and password. Anonymous users typically get access to only read (this includes download) the files from the server.

Web servers and the HTTP (Hyper Text Transfer Protocol) have been the major influence for the Internet's rapid growth in the past five years. Overall HTTP is much simpler than FTP, but also can place a greater demand on the server than FTP did for the same level of downloads. In addition, the more appealing automatic display of graphical data increases the user demand of these servers.

Current Web servers are based on the original CERN and NCSA server that is based on the original twenty year old FTP server source code. With HTTP, clients still access files, but the FTP "download session" is eliminated, connections are used for a single file transfer and then are closed. The default is that no passwords or usernames are required and some information about the file contents is now transmitted in the form of MIME headers. The elimination of a constant connection forces a web browser to establish many short duration connections to fetch each file. In order to increase speed many browsers will open up several connections in parallel. The server has more TCP/IP connections because of this, and these connections are constantly being opened and closed much more than FTP would have had. HTTP was clearly not designed with server efficiency in mind.

Within the foreseeable near future, end user connections to the Internet will rapidly improve and Internet backbone connection speeds will have increased substantially. Existing server technology will be unable to cope with the traffic, i.e. the number of hits and data that must be transmitted.

One method of improving the performance for delivering data on the internet is through the use of cache servers. A basic understanding of how computers work is essential to understanding the behaviour of cache based servers. In computers, there are two basic types of memory; the hard drive and RAM. The hard drive is the long-term memory that stores data even when the computer is turned off. RAM, on the other hand, holds the most frequently accessed data because it operates faster than a hard drive. The concept of caching is to keep a copy of the most recently accessed data in RAM after reading it from the hard disk. This is so the CPU will not need to access the hard disk again for the same data. This is much faster than having to re-read the same files from the disk repeatedly.

A "cache server" is a class of server that acts as a web server but serves as a bridge between the client's request for a file and a slower web server that offers the content. If content is requested that is not stored at the cache server, that request is passed further on to the web server. A cache server stores the most frequently requested files into RAM and/or high speed disk arrays in order to speed up content delivery to outside requests for the files. Nonetheless, cache servers often suffer the same limitations as web servers, and most emphasize using the disk storage in a faster and more efficient manner then the traditional operating system of the web server they try to enhance. They are best used to somewhat improve the performance of hard to reach web sites and to reduce the cost of bandwidth used by ISP's by storing local copies of frequently requested files instead of consuming precious bandwidth to the external internet. In practice cache servers often degrade access speed rather then improve it but they can reduce an ISP's bandwidth costs.

The present invention resolves the problem of inefficient file serving by eliminating the traditional FTP based architecture. It is also similar to some cache servers in that it makes extensive use of RAM, but has a number of very significant differences.

The present invention can make a significant impact on web servers that experience high hit volumes such as search engine sites, media intensive sites that serve still and motion images and video, and other large sites. Currently these sites are held on extremely large, powerful and expensive machines, all of which is unnecessary with the use of the present invention. ISP's, enterprises, and others would benefit from the use of the present invention because of the reduced bandwidth requirements and the related benefits; lower costs, faster access, and room for more users.

BRIEF SUMMARY OF THE INVENTION

In accordance with the present invention there is provided a method for implementing a web server, said method comprising the steps of:

a) initializing a memory resident web server parent process;

b) establishing a communication connection with each requesting client;

c) serving data to said each requesting client; and

d) continually repeating steps b) to c), wherein the method includes communicating with a plurality of requesting clients simultaneously through the web server parent process.

In accordance with the present invention there is further provided a web server comprising:

a) a computing environment;

b) a single web server parent process within said computing environment, said single process being resident in RAM for serving a plurality of clients;

c) communication means for establishing a plurality of communication links, each of said links connecting said single process to one of said plurality of clients; and

d) data serving means for serving data from said single process to each of said plurality of clients via the communication links connected to said clients.

In accordance with the present invention there is further provided a web server connected to the Internet, said web server being one of a plurality of computers connected to the Internet, said web server having a single serving process, said single process being resident in RAM and serving data to a plurality of clients, said data being stored in RAM and indexed by a lookup table, each of said clients being served said data via individual Internet connections by said single serving process, said single serving process replicating when a predetermined limit of clients has been reached.

In accordance with the present invention there is further provided a computer program embodied on a computer-readable medium for serving data to a plurality of clients connected to a web server containing said data, said program being a single process capable of establishing connections with a plurality of clients and serving data to said plurality of clients, and said program including means for replicating said single process only when a predetermined limit of clients has been reached.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a traditional web server process;

Figure 2 is a block diagram of the web server process of the present invention;

Figure 3 is a flow chart illustrating the logical flow of the web server process of the present invention;

Figure 4 is a chart illustrating the performance of the present invention in delivering 512 kilobyte files; and

Figure 5 is a chart illustrating the performance of the present invention in a Psuedo-SPEC benchmark.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is designed to run on any common operating system, such as Linux, Unix, Solaris or Windows NT. For illustration purposes, we refer to the FreeBSD operating system and a

Pentium class personal computer. FreeBSD is a UNIX-derived freely distributable operating system and is widely supported by volunteer programmers worldwide. FreeBSD was the platform for the implementation of the present invention due to its chosen performance, reliability, scalability, and ease of use and installation.

Currently, many web servers use a separate process for each socket (i.e. communication) request made to the web server. This traditional socket programming has been passed on from the design legacy of FTP servers.

In the traditional architecture a parent process creates a new child process or thread to serve each request. These threads can remain through multiple connections but there is still the requirement of having one per incoming connection. Each child process receives the clients request and opens the requested file and copies it out over the network.

The present invention is a small light and very fast web server. It achieves this by serving multiple connections per process instead of one per process. A reduction of processes, requires fewer memory and CPU resources. In addition all files (objects) are pre-loaded into memory, in UNIX this is virtual memory so it is possible to exceed the amount of physical RAM installed in the server.

The architecture of the original design of the present invention used a single process to support up to 255 connections. The current version has added a parent process. This is for the purpose of recovery in the event of the main process crashing for some unknown reason. It allows for a graceful restart and to continue serving with almost no noticeable interruption. This is achieved by retaining important data structures such as the out bound port in a TCP/IP connection, and current memory allocation. This architecture also allows the number of connections to exceed any system limit by creating more child processes. In BSD this limit is 250 child process.

The original version of the present invention used the UNIX function malloc to allocate enough memory for all the files to be served. At startup the files were then loaded and never touched again. Any changes made to the disk would not be picked up until the server was stopped and re-started. The preferred embodiment uses the BSD MMAP function.

Then end result of MMAP is almost identical to a malloc of memory space and then loading the file into it. There are several benefits. Files are not loaded into memory until needed, but the present invention accesses all RAM after the call to MMAP to force the files to be loaded to RAM. The biggest advantage is if the installed RAM is exceeded, instead of loading the files and having the operating system swap them to the swap space on the disk, meaning writing them to a special part of the disk and re-reading them later, the memory is reused and when needed again, the operating system reloads just the part of the file needed to handle the memory access as if it were still in RAM.

Referring to Figure 1 illustrating a block diagram of a traditional web server process shown generally as 30, it can be seen that the parent process 32 controlling the serving of data from server hard drive 34 to clients 36 has spawned a plurality of child processes 38 one for each client connection 40.

Traditional web servers and FTP servers as shown in Figure 1 work on the following basic principles.

a) a file request comes into the server through the operating system's communication areas;

b) the server sets up a dedicated parent process in memory (RAM or virtual/swapfile) to handle the TCP/IP networking processes;

c) sending a request for the file to the operating system and waiting for the operating system to reply;

d) transferring the requested file to the user;

e) closing the connection when finished and deciding what to do with a crashed connection; and

f) writing to the log files if necessary

For each concurrent request for a file the server must duplicate the entire process above. Traditional server software design requires another copy (child process 38) of the parent process 32 to be placed in memory and data is retrieved from the hard drive 34 by each new child process 38. This design results in:

a) high memory usage resulting in memory overflow into virtual memory (disk space reserved in addition to high-speed RAM) thus reducing memory access time;

b) using more RAM for child processes 38 or threads leaves less RAM for caching files and increases hardware requirements; and

c) more processor cycles required to manage the new child processes 38.

In this model there may be hundreds of new child processes 38 running and the computer must swap back and forth according to where attention is required. Much of this swapping ends up requesting information from the hard drive 34.

Such Web servers are at the mercy of the operating systems communications layers and hardware speed. For example, the web server has to open a file handle, open the file, read the file from disk 34 or the CPU system cache, then close the file handle.

The present invention works on an entirely different concept. The original design philosophy of the present invention is that the web server is copied into memory once, no matter how many concurrent users there are. Additionally, the web server tries to keep as much RAM free as possible and whatever RAM is not used by the server is reserved for the storage of the data being served to clients. Wherever possible, the present invention avoids using the hard drive of the hosting computer. Compared to the hard disk, RAM is many orders of magnitude faster. Once files are stored in RAM, the present invention does not need to query the operating system as far as finding files that are stored in RAM. Instead, it uses a table database lookup to store memory addresses within RAM and keeps this table in the RAM as well. Thus the only data stored in RAM is:

a) the server process (one copy only);
b) the table database that matches memory addresses with files; and
c) the remainder of the RAM (the majority of RAM), for the files to be served.

Referring now to Figure 2, a block diagram of the web server process of the present invention, shown generally as 50, has a single parent process 32 which accesses data to be served from hard drive 34 upon initialization of parent process 32. Child processes 38 are created as required to meet system load requirements. Each child process 38 services a plurality of clients 36 via client connections 40.

Referring now to Figure 3, a flow chart illustrating the logical flow of the web server process of the present invention is shown generally as 60. Initiate server process step 62 initiates the server process and loads the necessary images and files into RAM (less approximately 256K for the server process). At this time a lookup table is also created for the files that are stored in RAM. The present invention utilizes MMAP, a program used by many web servers to manage the files stored in RAM. With MMAP, files are placed into RAM only when they are requested by the user for the first time. Subsequently, the file is served directly out of RAM. This functionality permits the present invention to only put in RAM the files which are perceived to be more popular (based on that file being requested at least one time previously). This is especially useful when there is more data to be served (from the hard disk), than there is available memory in RAM. If enough RAM is not present, files will be taken out of RAM and the lookup table, and newly requested files will be served out of RAM.

At step 64 new TCP/IP socket connections are initialized by 'listening' for requests from the operating system. The present invention never says 'no' to the operating system. It always accepts and if it reaches the limitation within this loop, that request will be served in the next loop or by a child process loop. A non-blocking flag is set when listening for new connections to accept, which means that all connections are accepted until the queue is clear. At that point the loop sets a non-fatal error flag, which the present invention catches. This is a small change from the usual method of accepting connections one at a time, but in practice it has large performance benefits.

At step 66 the requested data is served to the client. A check is made to determine if the required data is in RAM. If not, the data is obtained from disk. Data is served only to existing connections and only served to clients who are ready to take the data. As mentioned earlier, since file locations are stored as memory addresses in the lookup table, knowing what file to serve the client next is determined by knowing what memory address that client was looking at previously and jumping to the next space in memory where that file is located. A non-blocking I/O flag is set when sending data files to the client machine. This allows the server to send the maximum size chunk of data in each operation. In practice, the server attempts to send the entire file each time, and the operating system accepts as much as it is ready for and tells the server how much that was. This is an improvement over the usual practice of sending fixed-size chunks, because it avoids some system calls.

Proceeding to step 68 all the finished connections are located and closed. Normally, when a socket is closed on the server side, there is a concept called a timewait that occurs. It is a 30 to 60 second delay in closing the socket and thus consumes resources. The present invention does not close connections immediately. Instead it waits a few milliseconds before closing. The result is that the client's side ends up closing the socket instead and the present invention does not waste valuable resources.

At step 70 data is written to the log files. With NCSA, Zeus, Apache and most other servers, logs are in ASCII and grow in size with great speed. An entry in a log file consists of:

a) the network address that the client connection can from (4 bytes);
b) the time (4 bytes); and
c) an index into a log index file (4 bytes).

The log index file is in ASCII form and contains a list of filenames and file lengths. Storing the log files of the present invention in binary format results in log files approximately 1/20th the size of traditional web server log files. This not only ensures increased capacity, but also enables the present invention to spend less time writing log files and more time serving data. In addition, log files are updated on the hard drive only after 1,000 hits have been recorded in the RAM. This reduces hard drive usage and significantly improves disk speed.

Proceeding to step 72 a test is made to determine if a new child process copy of the server process is necessary. If so, the loop returns to step 62 to create a new server process otherwise it returns to step 64 to continue the loop.

The steps 64 to 72 loop continuously. To do everything in a single process, a UNIX function called SELECT is used. This function determines which sockets need more information. With native UNIX and FreeBSD, SELECT caps out at 250 connections. To overcome this limitation, a second copy of the server process is initiated for every multiple of 250 connections. Also, the process server keeps a cache of the file-descriptor-set data structures used with the SELECT system call. These data structures are somewhat expensive to compute, so the process server avoids recomputing them if it can use the cached version. This is different from the usual practice of recomputing them on every trip through the main loop. For example, if 1,000 concurrent users want information from the server, the present invention goes through a single "FOR" loop and begins serving data. After 250 connections have been filled, a second, third and fourth copy of the server process is initiated for every pass through the "FOR" loop. A key comparison to make at this point is that traditional web servers start a new child process for each and every socket connection. The present invention only starts a child process after the 250 concurrent sockets limitation has been reached.

When running in multiple-worker-process mode with more than 250 connections, the workers use a round-robin token-passing protocol to negotiate which worker listens for new connections. The worker that owns the token listens for new connections and also processes its existing connections, while all the other workers only process their existing connections. When the worker with the token becomes loaded to at least half its capacity, it passes the token to the next worker in line.

There are no known limitations to how many child processes the present invention can make. Other traditional, physical limitations of memory size, network card speed, and processing power will 'max out' before the system of the present invention ever does.

The present invention also includes a feature known as reftab. This feature allows the user to restrict certain files stored on the web server, so they may only be served when the proper referer is supplied. Referers are sent by the clients web browser and indicate which page is referencing the file to be served. If the administrator of the web server of the present invention wishes to restrict other sites from copying content to their own site, most typically by using images from the present invention in their own .html pages, the administrator may set up a reftab to prevent this. The reftab file contains lines of the format "filename referer", for example:

images/anvil.jpg http://www.acme.com/anvil.html

This entry would ensure that the images/anvil.jpg could only be fetched or served via the page http://www.acme.com/anvil.html. Attempt to fetch it through any other page would result in a "403 Forbidden" error. Both the filenames and the referers may be wild card patterns. Wildcard patterns may use "*" meaning any string, or "?" meaning any character. They may also contain " | " which separates multiple patterns. This lets the system administrator set up some very powerful referer restrictions with a single line, for example:

.gif|.jpg http://www.acme.com/*|http://www.otheracme.com/*

This entry indicates that all GIF and JPEG files must have a referer, and the referer may be at any URL on either www.acme.com or www.otheracme.com. Thus, the images may be referenced from anywhere on those two sites, and nowhere else. If no reftab file is present, then there are no referer restrictions, and any page anywhere on the web is allowed to link to the files on the web server. The resolution of filenames that make use of wildcard patterns is done at start up so that there are no expensive wildcard matches required at run time.

Benchmark Results

The present invention is a high efficiency server solution that can be used as a stand-alone server for static web pages or as a capacity enhancement appliance used with existing servers to increase capacity or to protect a server system against overload caused by traffic peaks. As a stand-alone server or as a co-server, the present invention can handle over 2,875 hits per second and over 4,000 concurrent users. These performance results are even more impressive given they were achieved on an inexpensive hardware platform. Performance can be improved even further if more efficient Ethernet drivers are used given that on the test FreeBSD system the standard drivers used 70% of the available power of the 400 Mhz Pentium II Processor.

In a recent test run on a hardware platform (consisting of a 400 Mhz Pentium Pro processor and motherboard, 1 gigabyte of RAM, a 1 gigabyte Ethernet card, and 1 VGA card) the present invention handled over 3,200 hits per second.

In addition, the present invention handled 1,240 hits per second in a test similar to the SpecWeb96 test. Each of these results are significantly better than performance of Apache or Zeus-based software running on the same hardware platform.

However, the most compelling statistic is that the present invention is able to handle over 4,000 concurrent users. This is significantly higher than similar servers that use Apache (90 users), httpd (250), or Zeus (1,000 users) software. In fact, the present invention may be able to handle significantly more users because its absolute limit was never reached due to the capacity limits of the testing infrastructure. Handling multiple concurrent users is particularly valuable for sites with high peak demand.

Figure 4 illustrates the efficiency of the present invention over existing servers. The vertical axis is operations per second x 10^3 and the horizontal axis is the number of simultaneous users. The results for the present invention are charted at line 80. As can be seen, the present invention has less per connection overhead than the Zeus server (line 82), a small custom server thttpd (line 84), and Apache (line 86).

Referring now to Figure 5, in the Pseudo-SPEC test using the same four servers, once again the efficiency of the present invention as illustrated at line 90 is greater than that of the Zeus server (line 92), a small custom server thttpd (line 94), and Apache (line 96). The Pseudo-SPEC bench mark is a mix of files ranging in size from a 102 bytes to 900 kilobytes.

Having the present invention working independently or in conjunction with existing servers increases capacity, reduces response time and prevents servers from crashing during peak demands.

As will be understood by those skilled in the art, the various constant values and selected options mentioned within this disclosure such as:

a) the size, format and frequency of writing of log file entries;
b) the number of files the web server may serve;
c) the size of the web server process;

d) the use of FreeBSD as the operating system;

e) the number of simultaneous client connections; and

f) the use TCP/IP.

are all design choices which do not preclude the use of other choices, while still practicing the invention as disclosed.

I CLAIM:

1. A method for implementing a web server, said method comprising the steps of:

a) initializing a memory resident web server parent process;

b) establishing a communication connection with each requesting client;

c) serving data to said each requesting client; and

d) continually repeating steps b) to c), wherein the method includes communicating with a plurality of requesting clients simultaneously through the web server parent process.

2. The method of claim 1 wherein a child process is initiated to execute said method if said parent process has established a maximum number of communication connections and wherein a new child process is initiated whenever each previously initiated child process has established a maximum number of communication connections.

3. The method of claim 2 wherein said data is stored in RAM and indexed by a lookup table.

4. The method of claim 3 wherein said communication connection is a TCP/IP socket.

5. The method of claim 4 further comprising the step of within the loop b) to c), waiting for said each requesting client to close said communication connection.

6. The method of claim 5 further comprising the step of, before step c), writing information to a log file, said information being in binary format.

7. The method of claim 6 wherein all processes take a turn in being responsible for establishing new communication connections with requesting clients.

8. The method of claim 7 further comprising the step of, prior to step a), creating a reftab, said reftab containing a plurality of restricted data entries, each restricted data entry comprising the name of an object on said web server and a referer name, wherein said referer name may contain any of the tokens "*", "?", or " | ", for enabling restricted access to the object referenced by said restricted data entries.

9. The method of claim 8, wherein each of said referer names containing said tokens is resolved upon initiation of said parent process.

10. A web server comprising:

a) a computing environment;

b) a single web server parent process within said computing environment, said single process being resident in RAM for serving a plurality of clients;

c) communication means for establishing a plurality of communication links, each of said links connecting said single process to one of said plurality of clients; and

d) data serving means for serving data from said single process to each of said plurality of clients via the communication links connected to said clients.

11. The server of claim 10 further comprising means for creating at least one subsequent RAM resident web server child process to serve said clients if said parent process is unable to adequately serve said plurality of clients.

12. The server of claim 11 including means for storing said data in RAM and indexing said data by a lookup table.

13. The server of claim 12 wherein said communication means includes means for establishing a TCP/IP socket connection with each of said plurality of clients.

14. The server of claim 13 further comprising a log file, said log file containing information in binary format.

15. The server of claim 14 including sharing means for causing said processes to share in the establishing of said communication links.

16. The server of claim 15 further comprising a reftab, said reftab containing a plurality of restricted data entries, each restricted data entry comprising the name of an object on said web server and a referer name, wherein said referer name may contain any of the tokens "*", "?", or "|", for enabling restricted access to the object referenced by said restricted data entries.

17. The server of claim 16, wherein each of said referer names containing said tokens is resolved upon initiation of said parent process.

18. The server of claim 17 wherein said processes wait for said clients to terminate said communication links.

19. A web server connected to the Internet, said web server being one of a plurality of computers connected to the Internet, said web server having a single serving process, said single process being resident in

RAM and serving data to a plurality of clients, said data being stored in RAM and indexed by a lookup table, each of said clients being served said data via individual Internet connections by said single serving process, said single serving process replicating when a predetermined limit of clients has been reached.

20. The web server of claim 19 wherein said Internet connections are TCP/IP sockets.

21. The web server of claim 20 further comprising a log file, said log file containing information on data served by said server, said information being in binary format.

22. The web server of claim 21 including sharing means for causing all processes to take a turn in being responsible for establishing said Internet connections.

23. The server of claim 22 further comprising a reftab, said reftab containing a plurality of restricted data entries, each restricted data entry comprising the name of an object on said web server and a referer name, wherein said referer name may contain any of the tokens "*", "?", or "|", for enabling restricted access to the object referenced by said restricted data entries.

24. The web server of claim 23 wherein each of said referer names containing said tokens is resolved upon initiation of said single process.

25. A computer program embodied on a computer-readable medium for serving data to a plurality of clients connected to a web server containing said data, said program being a single process capable of establishing connections with a plurality of clients and serving data to said

plurality of clients, and said program including means for replicating said single process only when a predetermined limit of clients has been reached.

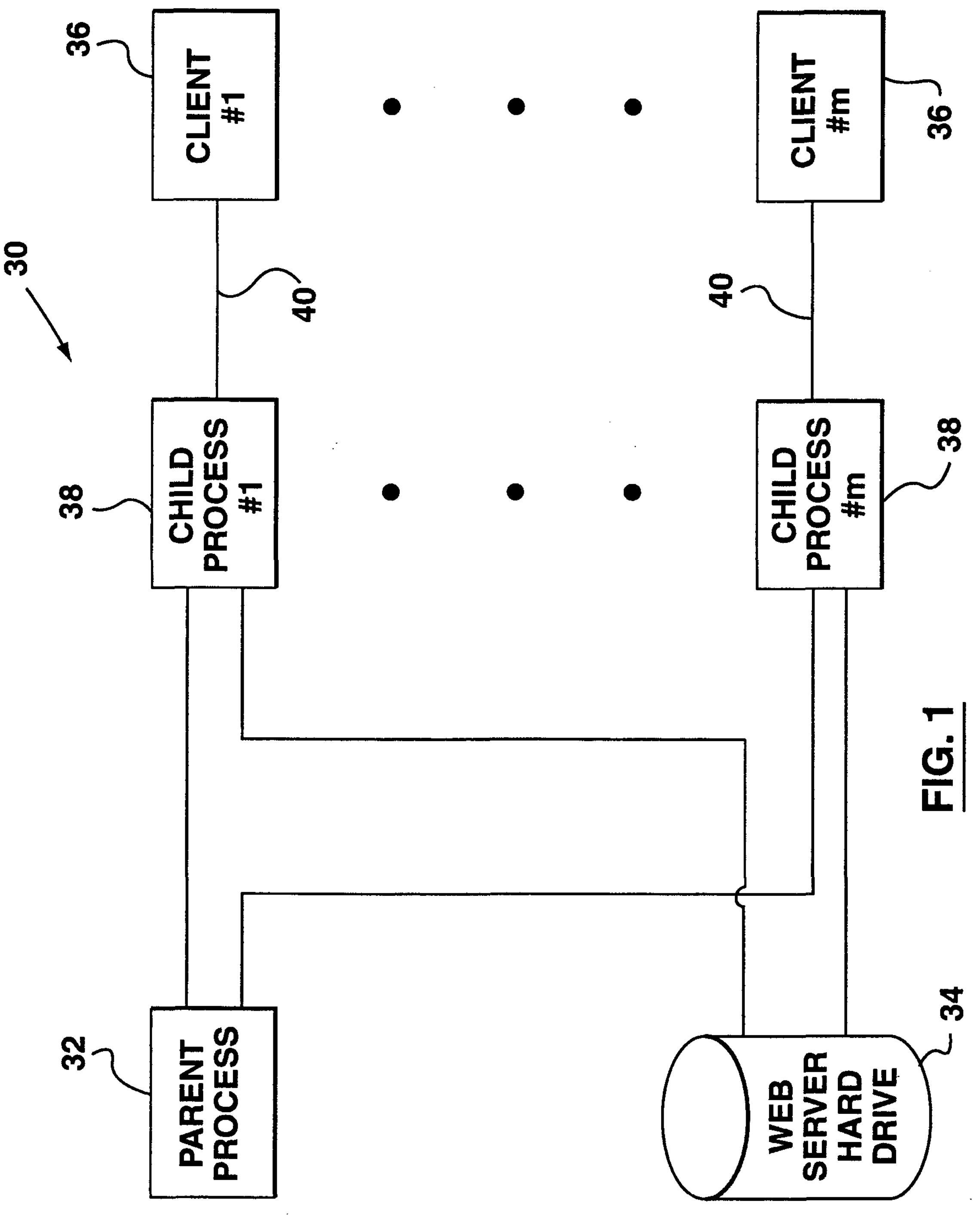

30
CLIENT #1
36
CLIENT #m
36
40
40
CHILD PROCESS #1
38
CHILD PROCESS #m
38
PARENT PROCESS
32
WEB SERVER HARD DRIVE
34

FIG. 1

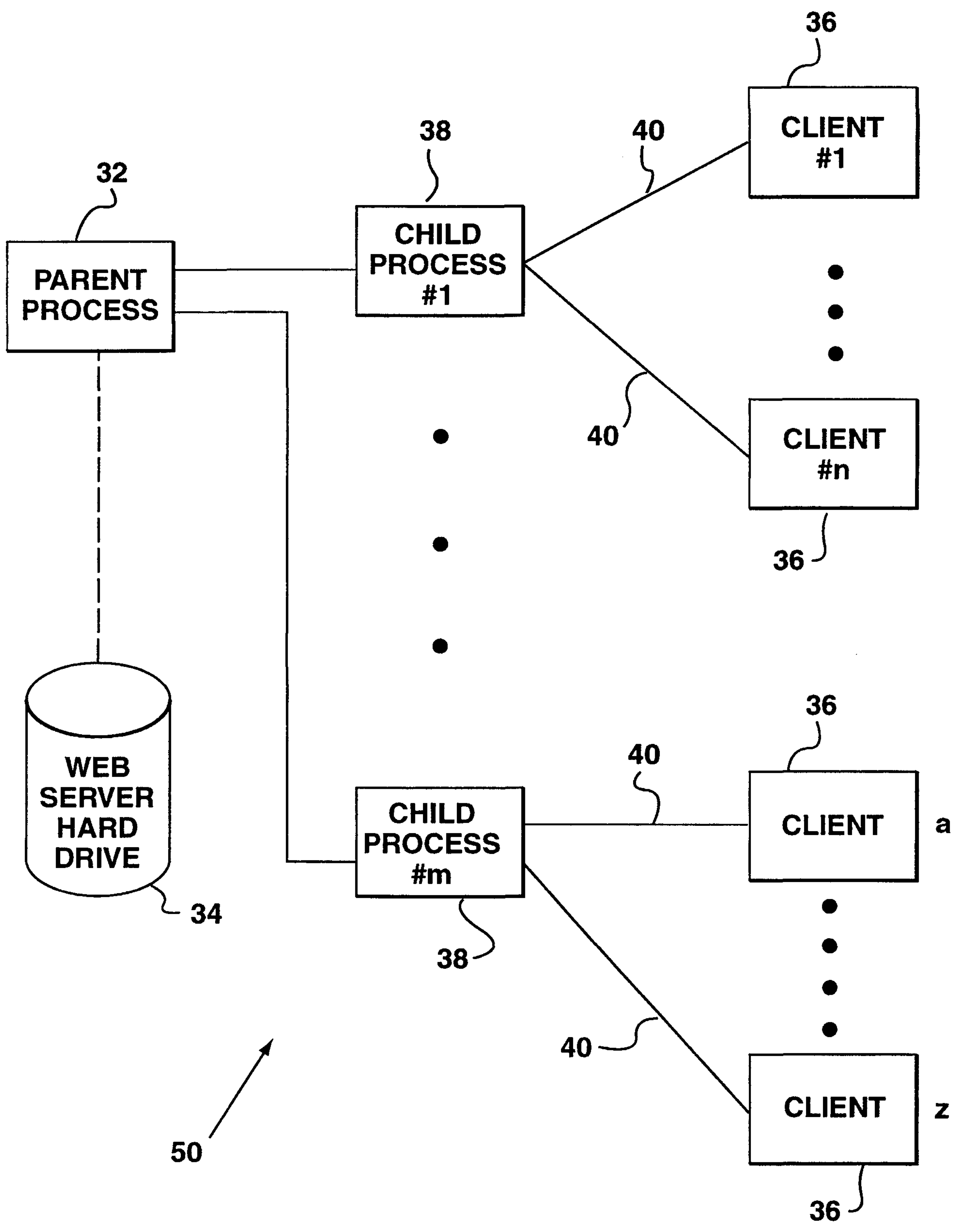

36
CLIENT #1
38
40
32
CHILD PROCESS #1
PARENT PROCESS
40
CLIENT #n
36
36
WEB SERVER HARD DRIVE
40
CHILD PROCESS #m
CLIENT
a
34
38
40
CLIENT
z
50
36

FIG. 2

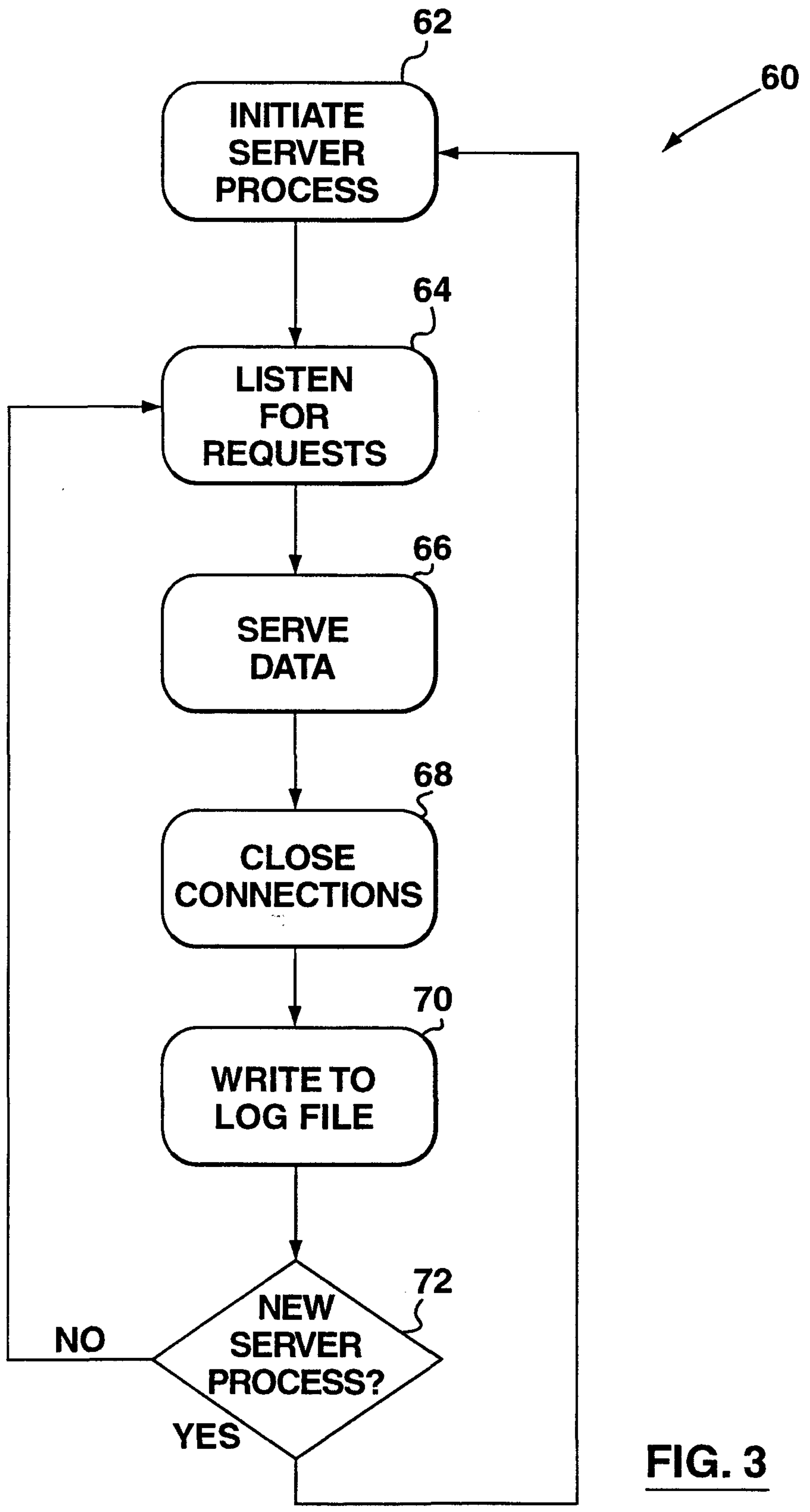

62
60
INITIATE SERVER PROCESS
64
LISTEN FOR REQUESTS
66
SERVE DATA
68
CLOSE CONNECTIONS
70
WRITE TO LOG FILE
72
NEW SERVER PROCESS?
NO
YES

FIG. 3

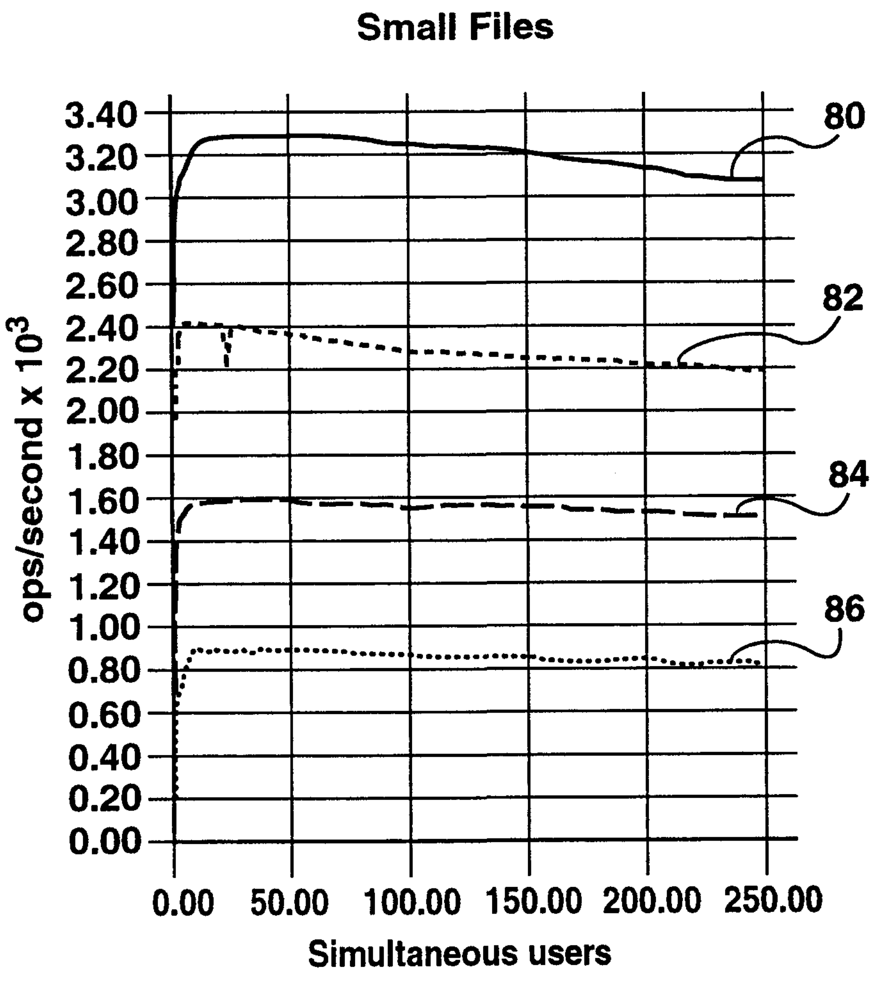

Small Files
3.40
3.20
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
ops/second x 10^3
0.00
50.00
100.00
150.00
200.00
250.00
Simultaneous users
80
82
84
86

FIG. 4

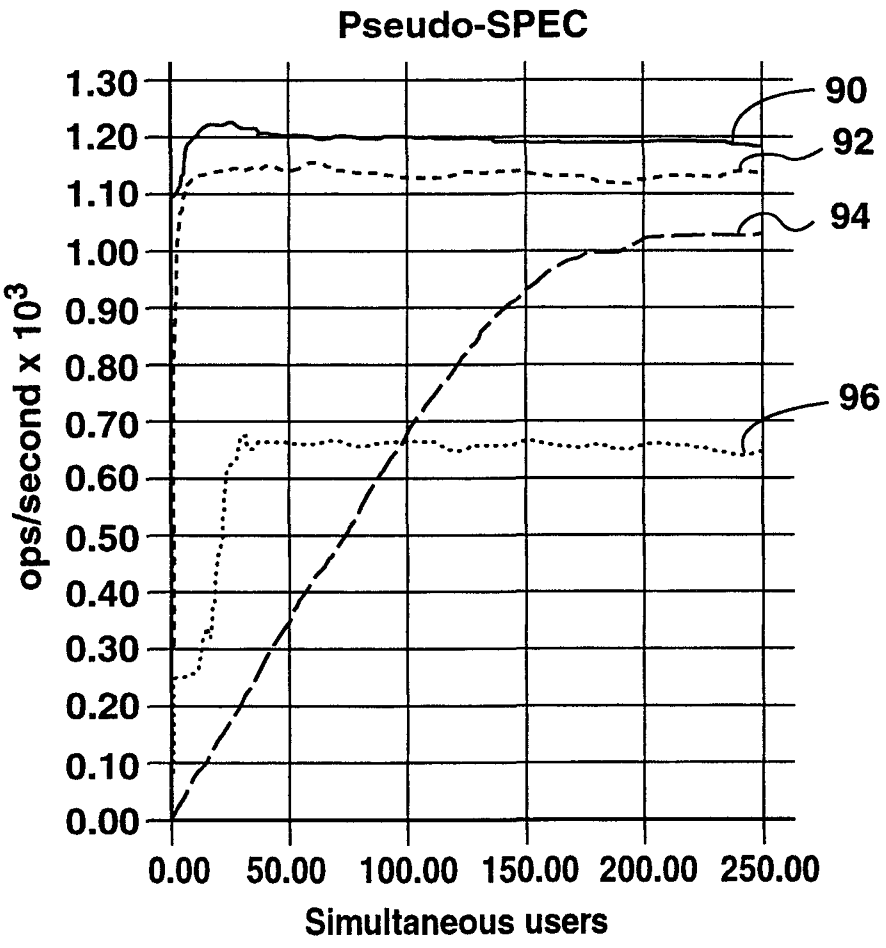

Pseudo-SPEC
ops/second x 10^3
1.30
1.20
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
50.00
100.00
150.00
200.00
250.00
Simultaneous users
90
92
94
96

FIG. 5