The Distributed V Kernel and Its Performance for Diskless Workstations

by

David R. Cheriton and Willy Zwaanepoel

Department of Computer Science
Stanford University
Stanford, CA 94305

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

DTIC FILE COPY
The Distributed V Kernel
and Its Performance for Diskless Workstations

Robert C.口岸 and Willy Suenasee

Computer Systems Laboratory
Departments of Computer Science and Electrical Engineering
Stanford University

Abstract

The distributed V kernel is a modification to the
existing V kernel that allows for distributed file systems over
networks. It is primarily being used to maintain files
which are stored on diskless workstations that are attached to
a network. The distributed V kernel is a distributed version
of the traditional V kernel, with particular emphasis on the
use of network file systems. Our

1. Distributed version of V kernel. This version
of the V kernel allows files to be stored
on diskless workstations over a network.

2. The V kernel allows files to be stored
on diskless workstations over a
network. This version

It is

1. Introduction

The distributed V kernel is a modification to the
existing V kernel that allows for distributed file
systems over networks. It is primarily being used
to maintain files which are stored on diskless
workstations that are attached to a network. The
distributed V kernel is a distributed version of
the traditional V kernel, with particular emphasis on
the use of network file systems. Our

1. Distributed version of V kernel. This version
of the V kernel allows files to be stored
on diskless workstations over a network.

2. The V kernel allows files to be stored
on diskless workstations over a
network. This version

It is

1. Introduction

The distributed V kernel is a modification to the
existing V kernel that allows for distributed file
systems over networks. It is primarily being used
to maintain files which are stored on diskless
workstations that are attached to a network. The
distributed V kernel is a distributed version of
the traditional V kernel, with particular emphasis on
the use of network file systems. Our

1. Distributed version of V kernel. This version
of the V kernel allows files to be stored
on diskless workstations over a network.

2. The V kernel allows files to be stored
on diskless workstations over a
network. This version

It is

1. Introduction

The distributed V kernel is a modification to the
existing V kernel that allows for distributed file
systems over networks. It is primarily being used
to maintain files which are stored on diskless
workstations that are attached to a network. The
distributed V kernel is a distributed version of
the traditional V kernel, with particular emphasis on
the use of network file systems. Our

1. Distributed version of V kernel. This version
of the V kernel allows files to be stored
on diskless workstations over a network.

2. The V kernel allows files to be stored
on diskless workstations over a
network. This version

It is

1. Introduction

The distributed V kernel is a modification to the
existing V kernel that allows for distributed file
systems over networks. It is primarily being used
to maintain files which are stored on diskless
workstations that are attached to a network. The
distributed V kernel is a distributed version of
the traditional V kernel, with particular emphasis on
the use of network file systems. Our

1. Distributed version of V kernel. This version
of the V kernel allows files to be stored
on diskless workstations over a network.

2. The V kernel allows files to be stored
on diskless workstations over a
network. This version

It is
process. With the exception of GIF/TV, several operations with no process identifiers performed are implicitly tied to the virtual process. GIF/TV uses process identifiers to determine the mapping of a logical process identifier to each process identifier if the mapping is not known to the local host. Any local mapping between

3.2 Remote Message Implementation
When a process identifier is specified to GIF/TV with a logical host identifier different from that of the local machine, the local port

The GIF/TV router replies with a broadcast packet on the

In the event that both processes are initiated by the same

V's router is implemented using an 17/2 processor developed for

To read a page or block of a file, a client sends a message to the file server process specifying the file, block number, type count, and the address of the buffer into which

The virtual process identifier, if specified, is used by the

The network interface board is specified by the

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same

In the event that both processes are initiated by the same
Advantages of the approach include:

1. The advantages of the Turkish model with the occasional performance of the EDA, which is simplified.
2. Compatibility with different level structures. The balance of segments may be reached directly if in the same differential space or if the substitute process generates every database.
3. Use of Acquire/Release in segmentation and management to provide a substantially simplified process.

An additional objective in any process includes a simplified and flexible segmentation to provide a substantially simplified process.

We can now look at the dimension of the performance evaluation of the system. We must define the more accurate criteria on an appropriate level based on the size of segment combinations. Subsequently we discuss the efficiency of the benefits both in terms of coverage ranging and different.

4. Network Penalty

Consequences of the Turks on primary network with experimentation.

1. The use of some operations were the use of the network.
2. The use of some operations were the use of the network.

The above penalty is a fraction of the segments used. The approach, however, is limited and sometimes sacrifices the use of the segments. The approach is limited by comparing the data on segments a lighter than the data on the amount of data generated in the entire amount of data. This system and dividing the data into the data requirement by 3. The approach is limited by comparing the data on segments a lighter than the amount of data generated in the entire amount of data. This system and dividing the data into the data requirement by 3.
view interprocess communication as transparent across machines
when the speed ratio is no longer. However, no such
view interpretation is to recognize that the network operation with a
delay of less than 3 milliseconds, and that in many cases this
is negligible relative to the time necessary to process a request
in the server. Furthermore, the network is designed to keep
the server busy, as the network operation with a delay of
less than 3 milliseconds is significantly shorter than the
network in this database. Unfortunately, our measurements
of this network are very short-term in part 3.1. However,
presently the network is designed to have the capacity of
transmitting data at about 3000 messages per second.
Table 5-1: Kernel Performance: 3 Mbit Ethernet and 8 Mbit Processor (time in milliseconds)

<table>
<thead>
<tr>
<th>Kernel Operation</th>
<th>Local Speed</th>
<th>Remote Speed</th>
<th>Difference</th>
<th>Network Penalty</th>
<th>Client Penalty</th>
<th>Server Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTime</td>
<td>0.07</td>
<td></td>
<td></td>
<td></td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>Send-Receive-Reply</td>
<td>1.20</td>
<td>1.10</td>
<td>0.10</td>
<td></td>
<td>1.60</td>
<td>1.79</td>
</tr>
<tr>
<td>MoveFrom: 1024 bytes</td>
<td>1.26</td>
<td>0.63</td>
<td>0.63</td>
<td>7.77</td>
<td>6.15</td>
<td>3.76</td>
</tr>
<tr>
<td>MoveTo: 1024 bytes</td>
<td>1.26</td>
<td>0.63</td>
<td>0.63</td>
<td></td>
<td>7.77</td>
<td>6.15</td>
</tr>
</tbody>
</table>

Table 5-2: Kernel Performance: 3 Mbit Ethernet and 10 Mbit Processor (time in milliseconds)

<table>
<thead>
<tr>
<th>Kernel Operation</th>
<th>Local Speed</th>
<th>Remote Speed</th>
<th>Difference</th>
<th>Network Penalty</th>
<th>Client Penalty</th>
<th>Server Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTime</td>
<td>0.06</td>
<td></td>
<td></td>
<td></td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>Send-Receive-Reply</td>
<td>0.77</td>
<td>2.54</td>
<td>2.54</td>
<td>1.77</td>
<td>1.30</td>
<td>1.64</td>
</tr>
<tr>
<td>MoveFrom: 1024 bytes</td>
<td>0.95</td>
<td>0.63</td>
<td>0.32</td>
<td>7.05</td>
<td>6.77</td>
<td>3.22</td>
</tr>
<tr>
<td>MoveTo: 1024 bytes</td>
<td>0.95</td>
<td>0.63</td>
<td>0.32</td>
<td></td>
<td>7.05</td>
<td>6.77</td>
</tr>
</tbody>
</table>

A network team is a group of people working together. This is especially
critical in the current environment where there is a high
level of interaction and communication between different
teams. The expertise and experience of the team members
can significantly impact the overall performance of the
network. By properly configuring the network and ensuring
effective communication, the network team can help
improve the efficiency and effectiveness of the network.

A network team is a group of people working together. This is especially
critical in the current environment where there is a high
level of interaction and communication between different
teams. The expertise and experience of the team members
can significantly impact the overall performance of the
network. By properly configuring the network and ensuring
effective communication, the network team can help
improve the efficiency and effectiveness of the network.

A network team is a group of people working together. This is especially
critical in the current environment where there is a high
level of interaction and communication between different
teams. The expertise and experience of the team members
can significantly impact the overall performance of the
network. By properly configuring the network and ensuring
effective communication, the network team can help
improve the efficiency and effectiveness of the network.
serving the client process are measured and otherwise idle. A host system consists of two clients and one server on the file server.

We first describe the performance of random page-level file access.

6.1. Page-Level File Access

Table 6-1 lists the times for reading or writing a 512-byte block between two processes both local and remote using the 10 Mbps processor. The times do not include time to fetch the data from disk but do indicate expected performance when data is buffered in memory. A page read involves the sequence of basic operations: Send-Receive-Reply-Write-Segment. A page write is Send-Receive-Write-Segment-Reply.

**Random Page-Level Access**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Read Time</th>
<th>Write Time</th>
<th>Difference</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>page read</td>
<td>1.11</td>
<td>3.56</td>
<td>2.45</td>
<td>3.89</td>
<td>3.43</td>
</tr>
<tr>
<td>page write</td>
<td>1.11</td>
<td>3.69</td>
<td>2.58</td>
<td>3.89</td>
<td>3.52</td>
</tr>
</tbody>
</table>

Table 6-2: Page-Level File Access: 512-byte pages (times in milliseconds)

The columns are to be interpreted according to the explanation given for similarity-based caching (Section 5.3 and 5.4). Since the data to read or write a page using these primitives is approximately 1.5 megabytes more than the data size for these situations.

To use an example, assume that processor on the server for the PC runs a program that reads or writes a file that is larger than the buffer size. The file is read or written in 512-byte blocks. It is assumed that the buffer size is 512 bytes. The program is expected to read or write the entire file. The difference between the client process time for remote page reads and for local page access, namely 0.3 milliseconds. A process 5% of more than 1.3 milliseconds per request can be expected from the announcement made earlier using LOCUS figures.

These improvements indicate the performance when file reading and writing use explicit segment specification to the server and ReceiverWrite-Segment and ReplyWrite-Segment. However, a file write can also be performed in a more basic Thrift-like way using the Send-Reply-Write-PoP-Reply sequence. For a 512-byte file write, this costs 2.1 milliseconds; file reading is similar using Thrift-like. Thus, the segment mechanism saves 3.5 milliseconds on every page read and write operation, justifying this extension to the send primitive.

6.2. Sequential File Access

The traditional file access is the predominant pattern for file activity in many systems. Without explicit segment, file input/output results in a behavior to which users are accustomed. When this behavior is changed, the system writer to prohibit using the file pages (variable) for implementing chainable could pages (variable). The system writer used instead a chainable on-line paging file to maintain the effect of explicit memory on the system.

Table 6-3: Page-Level File Access: 512-byte pages (times in milliseconds)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Read Time</th>
<th>Write Time</th>
<th>Difference</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>file read</td>
<td>0.11</td>
<td>0.56</td>
<td>0.45</td>
<td>3.89</td>
<td>3.43</td>
</tr>
<tr>
<td>file write</td>
<td>0.11</td>
<td>0.69</td>
<td>0.58</td>
<td>3.89</td>
<td>3.52</td>
</tr>
</tbody>
</table>

Table 6-4: Page-Level File Access: 512-byte pages (times in milliseconds)

To ensure that the server process time is read or written in 512-byte blocks, the file is read or written in 512-byte blocks. The difference between the client process time for remote page reads and for local page access, namely 0.3 milliseconds. A process up to more than 1.3 milliseconds per request can be expected from the enhancement made earlier using LOCUS figures.

These improvements indicate the performance when file reading and writing use explicit segment specification to the server and ReceiverWrite-Segment and ReplyWrite-Segment. However, a file write can also be performed in a more basic Thrift-like way using the Send-Reply-Write-PoP-Reply sequence. For a 512-byte file write, this costs 2.1 milliseconds; file reading is similar using Thrift-like. Thus, the segment mechanism saves 3.5 milliseconds on every page read and write operation, justifying this extension to the send primitive.
many current multiprocessing systems do, such process loading could achieve the same performance given in the table, independent of disk speed. Thus, we expect that multiprocessor with large transfer units provide an efficient process loading mechanism that is as fast as can be achieved with the given hardware.

7. File Server Issues

File server performance is a critical issue for distributed workstations. Unfortunately, we do not yet have experience with a V kernel-based file server. Thus, this section describes what we believe are the key issues and estimates performance without providing conclusive data. In general, we view the latency as the key resource to consider in file server performance because, as argued earlier, the network bandwidth is plentiful and disk scheduling and buffering issues are identical to those encountered in conventional multi-user systems.

The number of workstations a file server can support can be estimated from previous research. If we assume pages read or written processing overhead on roughly 2.5 milliseconds for file system processing (from LOCOS) plus 3.5 milliseconds for kernel operation (from Table 6-1), a page request with about 7 milliseconds of processor time. Program loading appears to cost about 800 milliseconds for an 84-byte page request. Assuming that 20 percent of the file requests are page requests, the average request rate is 30 milliseconds. Thus, a file server based on the 35,000 workstations per house design request about 60 requests per second. For 20 percent of the page requests, the file server can serve about 12 simultaneous requests, but 20 percent are for simultaneous requests would only be available on server side. Thus, the file server's disk bandwidth cannot easily be saturated to provide more workstations scaling up to more workstations. The network would ultimately be limited by the time that file workstations would be reading over network.

For many problems, this estimation is on the conservative side. The current systems are running 3300 workstations, many of which contain several page requests. However, the estimated time for this load is still 12 simultaneous requests. Thus, we expect that the file server can support a larger number of workstations than the above estimation indicates.

8. Measurements with the 10 Mb Ethernet

Our limited access to a 10 Mb Ethernet has produced several measurements on this standard local network. However, more preliminary figures using the 10 Mb Ethernet indicate the effect of using a faster network and slightly faster network interfaces. First, the remote message exchange time is 2.71 milliseconds using an 8 Mb processor, roughly the time for the 10 Mb processor on the 3 Mb network and .5 milliseconds better than the 8 Mb processor on the 3 Mb network. Second, the page read time is 5.72 milliseconds. Finally, the program loading time is much improved, achieving 225 milliseconds for a 64 kilobyte load using 10 Kb transfer units. We have not identified to what degree the improvement is due to the faster network speed versus the differences in the network interface.

9. Related Work

There are a number of previous and concurrent efforts in providing communication mechanisms for distributed systems. For brevity, we compare our work with only a representative sample that demonstrates the search for, and evaluation of, better and more efficient communication mechanisms.

Swarup's recent paper [21] considered the feasibility of implementing remote task and remote operations on a local network. Neither's work, on remote procedures calls [10] interprets network communication for procedure-based systems whereas the V kernel provides for message-based systems. Both and Swarup's recent work [12] implement remote procedure calls with a number of features such as network buffering and connection management that are not provided by the V kernel. Finally, LOCOS decouples network communication into a UNIX-like distributed system and a network layer that operates on the underpinning layer.

Our design, as described above, offers several advantages over Swarup's and Previous's work. First, the V kernel provides an implementation of communication that places the network layer at the boundary of the system, including control and data structures for efficient communication control. In particular, the V kernel provides a clean separation between the network interface and the underlying network by providing a standard interface to the network. The V kernel is defined in a programming language that can be expressed in the V kernel's programming language. In addition, the V kernel provides a series of important features such as support for parallel processing of control in a distributed network, support for remote procedure calls, and support for the implementation of distributed systems by hiding the underlying network interface.

In summary, the V kernel provides a comprehensive framework for distributed systems that is both efficient and flexible. It provides a clean separation between the network interface and the underlying network, allowing the programmer to focus on the logic of the distributed system without being concerned with the details of network communication. This framework is designed to be easy to use and to provide a high level of performance, making it an excellent choice for distributed systems development.
versus message-based systems, although it is not clear these differences result in any significant difference in overall performance.

The V kernel performance is roughly comparable to that of the software implementations developed by Specter and Nelson, allowing for the non-trivial differences in operation semantics and host processors. We would have expected V kernel performance to be improved by a factor of 20 using microcode, similar to the improvement observed by Specter and Nelson for their primitives. Unfortunately, neither Specter nor Nelson provides results that affect a comparison with our file access results. In general, their work has concentrated on the speed of the basic mechanism and has not been extended to measure performance in a particular application setting.

In comparison to Acceptor, the V kernel provides a primitive form of message communication, and benefits accordingly in terms of speed, small code size and ability to run well on an inexpensive machine without disk or microcode support. For instance, Acceptor messages require an underlying transport protocol for reliable delivery because there is no client-level reply message associated with every Send as in the V kernel. We do not at this time have performance figures for Acceptor.

LOCUS does not attempt to provide applications with general network interprocess communication but exploits carefully tuned subnetwork-oriented protocols for efficient remote file access. It is difficult to compare the two systems from measurements available given the differences in network speeds, processor speeds and measurement techniques. However, with the specific comparisons with LOCUS presented earlier, we would expect overall file access performance for the V kernel to be comparable to LOCUS running on the same machines and network.

However, the memory requirements for the V kernel are about half that of LOCUS compiled for the PDP-11 and probably more like one fourth when LOCUS is compiled for a 32-bit processor like the 68000. Thus, for graphics workstations or process control applications, for instance, the V kernel would be more attractive because of its smaller size, real-time orientation and its provision of general interprocess communication. However, the V kernel does not provide all the functionality of the LOCUS kernel which includes that of the UNIX kernel and more. When equipped with V, these additional facilities must be provided by server processes emulating either on client workstations or network server machines.

10. Conclusions
We conclude that it is feasible to build a distributed system using file-based workstations connected by a high-speed local network to one or more file servers using the V kernel IPC. In particular, the performance study shows that V kernel IPC provides satisfactory performance despite its generality. Because the performance is so close to the lower bound given by the network penalty, there is relatively little room for improvement on the V IPC for the given hardware regardless of protocol and implementation used.

The efficiency of file access using the V IPC suggests that it can not only replace page-level file access protocols but also file transfer and remote terminal protocols, thereby reducing the number of protocols needed. We claim that V kernel IPC is adequate for a transport level for all our local network communication providing each machine runs the V kernel or at least handles the interkernel protocol. We do, however, see a place for these specific protocols in internetworking situations.

In addition to quantifying the elapsed time for various operations, our study points out the importance of considering processor requirements in the design of distributed systems. More experience and measurement of file server load and workstation file access behavior is required to decide whether file server processing is a significant problem in using diskless workstations.

The V kernel has been in use with the diskless SUN workstations, providing local and remote interprocess communication, since September 1982. It is currently 32 kilobytes including code, data and stack. The major use of the network interprocess communication is for accessing remote files. Our file servers are currently 6 VAX/UNIX systems running a kernel simulator and file server program which provides access to UNIX system services over the Ethernet using interkernel packets. A simple custom interpreter program allows programs to be loaded and run on the workstations using these UNIX servers. Our experience with this software to date supports the conclusions of the performance study that we can indeed build our next generation of computing facilities using diskless workstations and the V kernel.

Acknowledgements
We are indebted to all the members of the V research group at Stanford, which at this point includes two faculty members and roughly ten graduate students. In particular, we wish to thank Keith Lass for his patient comments on a seemingly endless sequence of drafts and Tim Graves for his many contributions to the design and the implementation of the kernel. We would also like to thank the referees whose comments and suggestions helped to enhance the clarity of the paper.

References


