CIFS/9000 Server Opportunistic Locking Usage Recommendations

Version 1.02 October, 2002

Eric Roseme
SNSL Advanced Technology Center

E0300

Printed in: U.S.A.

Legal Notices

The information in this document is subject to change without notice.
Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Company
19420 Homestead Road
Cupertino, California 95014 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright Notices
©copyright 1983-2001 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Contents

Chapter 1 Introduction

CIFS/9000 Server on HP-UX manages file access among Windows clients with Windows style file locking. It applies a very effective set of file locking features that are managed by the user-space client processes on the server, and provides excellent data security and integrity in a multi-user environment. CIFS/9000 Server also integrates some Windows locking protocols with the underlying HP-UX operating system locking protocols, and therefore provides some interoperability with UNIX and NFS style file locking. For a detailed discussion of CIFS/9000 Server file locking, read the "CIFS/9000 Server File Locking Interoperability" whitepaper at:

http://snsl.cup.hp.com/page.php?id=26

Further file locking enhancements are currently under development in the HP CIFS/9000 lab with the implementation of the CIFS/9000 Extended File System, which is a custom designed virtual file system (VFS) layer for HP-UX.

Opportunistic Locking is a unique Windows file locking feature. It is not really file locking, but is included in most discussions of Windows file locking, so is considered a defacto locking feature. Opportunistic Locking is actually part of the Windows client file caching mechanism. It is not a particularly robust or reliable feature when implemented on the variety of customized networks that exist in enterprise computing, but can be effective in providing modest perceived performance optimization.

Like Windows, CIFS/9000 Server implements Opportunistic Locking as a server-side component of the client caching mechanism. Because of the lightweight nature of the Windows feature design, effective configuration of Opportunistic Locking requires a good understanding of its limitations, and then applying that understanding when configuring data access for each particular customized network and client usage state.

Chapter 2 Opportunistic Locking Overview

OPPORTUNISTIC LOCKING (Oplocks) is invoked by the Windows file system (as opposed to an API) via registry entries (on the server AND client) for the purpose of enhancing network performance when accessing a file residing on a server. Performance is enhanced by caching the file locally on the client which allows:

Read-ahead:	The client reads the local copy of the file, eliminating network latency
Write caching:	The client writes to the local copy of the file, eliminating network latency
Lock caching:	The client caches application locks locally, eliminating network latency

The performance enhancement of oplocks is due to the opportunity of exclusive access to the file, even if it is opened with deny-none, because Windows monitors the file's status for concurrent access from other processes.

Windows defines 4 kinds of Oplocks:

Level1 Oplock - The redirector sees that the file was opened with deny none (allowing concurrent access), verifies that no other process is accessing the file, checks that oplocks are enabled, then grants deny-all read-write exclusive access to the file. The client now performs operations on the cached local file.

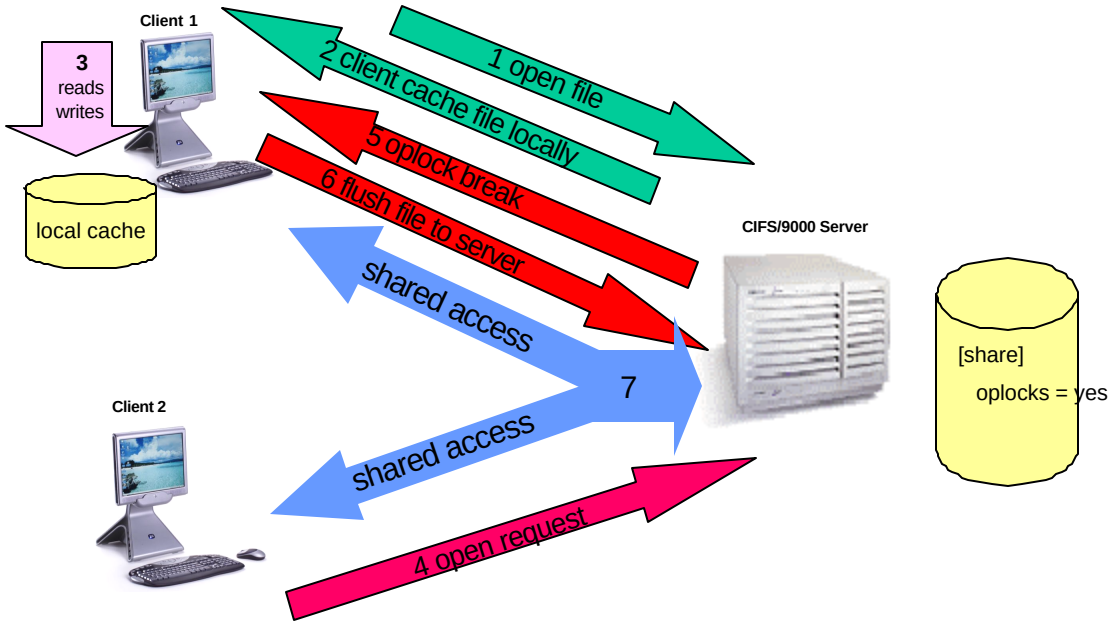
If a second process attempts to open the file, the open is deferred while the redirector "breaks" the original Oplock. The Oplock break signals the caching client to write the local file back to the server, flush the local locks, and discard read-ahead data. The break is then complete, the deferred open is granted, and the multiple processes can enjoy concurrent file access as dictated by mandatory or byte-range locking options. However, if the original opening process opened the file with a share mode other than deny-none, then the second process is granted limited or no access, despite the Oplock break.

Level2 Oplock – Performs like a level1 oplock, except caching is only operative for reads. All other operations are performed on the server disk copy of the file.

Filter Oplock – Does not allow write or delete file access.

Batch Oplock – Manipulates file openings and closings – allows caching of file attributes.

An important detail is that Oplocks are invoked by the file system, not an application API. Therefore, an application can close an oplocked file, but the file system does not relinquish the Oplock. When the Oplock break is issued, the file system then simply closes the file in preparation for the subsequent open by the second process.



1. Client 1 opens file on CIFS/9000 Server share. Client has exclusive access to file (no other processes are accessing the file). Exclusive access is not an access mode (like DENY_ALL) – it is an access state.
2. Client 1caches the file locally.
3. Client 1 performs reads, writes, and locks locally, thereby reducing network latency for these operations.
4. Client 2 sends an open request to the CIFS/9000 Server for the file that Client 1 is working on. The file open is blocked.
5. The CIFS/9000 Server sends an oplock break to client 1
6. Client 1 then sends the file with all of the applied changes back to the CIFS/9000 Server
7. After the server receives the file from client 1 and writes the changes to disk, it allows concurrent file access by Client 1 and Client 2, as dictated by the file open properties of each opening process. Now all operations are performed over the network.

Chapter 3 CIFS/9000 Oplock Configuration

OPPORTUNISTIC LOCKING **(*Oplocks*)** is implemented by the CIFS/9000 server on a per-share basis in the smb.conf file. CIFS/9000 Oplock functionality operates just like Windows. *Oplocks* are enabled by default for each share, which allows the Windows client to cache a local copy of a file for:

Read-ahead
Write-caching
Lock caching

CIFS/9000 disables *Oplocks* on a per-share basis in the smb.conf file:

```
[share_name]
    oplocks  = no
```

The default is "yes". The default oplock type is Level1.

CIFS/9000 enables Level2 Oplocks on a per-share basis in the smb.conf file:

```
[share_name]
    level2 oplocks = yes
```

The default is "no". Oplocks must also be set to "yes" for the Level2 oplock parameter to function.

Kernel oplocks is a Samba smb.conf parameter that notifies Samba if the UNIX kernel has the capability to send a Windows client an Oplock Break if a UNIX process is attempting to open the file that is cached. This parameter addresses sharing files between UNIX and Windows with Oplocks enabled on the a Samba server: the UNIX process can open the file that is Oplocked (cached) by the Windows client and the smbd process will not send an Oplock break, which exposes the file to the risk of data corruption. If the UNIX kernel has the ability to send an Oplock break, then the kernel oplocks parameter enables Samba to send the Oplock break. Kernel oplocks are enabled on a per-server basis in the smb.conf file. However, CIFS/9000 currently does not support kernel oplocks, so the parameter has no effect:

```
[global]
    kernel oplocks = yes
```

The default is "no". The planned enhancement for CIFS/9000 Oplocks will likely utilize this parameter.

Veto oplocks is a smb.conf parameter that identifies specific files for which Oplocks are disabled. When a Windows client opens a file that has been configured for veto oplocks, the client will not be granted the oplock, and all operations will be executed on the original file on disk instead of a client-cached file copy. By explicitly identifying files that are shared with UNIX processes, and disabling Oplocks for those files, the server-wide Oplock configuration can be enabled to allow Windows clients to utilize the performance benefit of

file caching without the risk of data corruption. Veto Oplocks can be enabled on a per-share basis, or globally for the entire server, in the smb.conf file:

```
[global]
    veto oplock files = /filename.htm/*.txt/

[share_name]
    veto oplock files = /*.exe/filename.ext/
```

oplock break wait time is a smb.conf parameter that adjusts the time interval for Samba to reply to an oplock break request. Samba recommends "DO NOT CHANGE THIS PARAMETER UNLESS YOU HAVE READ AND UNDERSTOOD THE SAMBA OPLOCK CODE." Oplock Break Wait Time can only be configured globally in the smb.conf file:

```
[global]
    oplock break wait time = 0 (default)
```

oplock break contention limit is a smb.conf parameter that limits the response of the Samba server to grant an oplock if the configured number of contending clients reaches the limit specified by the parameter. Samba recommends "DO NOT CHANGE THIS PARAMETER UNLESS YOU HAVE READ AND UNDERSTOOD THE SAMBA OPLOCK CODE." Oplock Break Contention Limit can be enable on a per-share basis, or globally for the entire server, in the smb.conf file.

```
[global]
    oplock break contention limit = 2 (default)

[share_name]
    oplock break contention limit = 2 (default)
```

Chapter 4 Opportunistic Locking Recommendations

Opportunistic locking is a desirable feature when it can enhance the perceived performance of a networked client. However, the opportunistic locking protocol is not robust, and therefore can encounter problems when invoked beyond a simplistic configuration, or on extended, slow, or faulty networks. In these cases, operating system management of opportunistic locking and/or recovering from repetitive errors can offset the perceived performance advantage that it is intended to provide.

"Opportunistic Locking" is actually an improper name for this feature. The true benefit of this feature is client-side data caching, and oplocks is merely a notification mechanism for writing data back to the networked storage disk. The limitation of opportunistic locking is the reliability of the mechanism to process an oplock break (notification) between the server and the caching client. If this exchange is faulty (usually due to timing out for any number or reasons) then the client-side caching benefit is negated.

The actual decision that a user or administrator should consider is whether it is sensible to share amongst multiple users data that will be cached locally on a client. In many cases the answer is no. Deciding when to cache or not cache data is the real question, and thus "opportunistic locking" should be treated as a toggle for client-side caching. Turn it "ON" when client-side caching is desirable and reliable. Turn it "OFF" when client-side caching is redundant, unreliable, or counter-productive.

Opportunistic locking is by default set to "on" by CIFS/9000 Server on all configured shares, so careful attention should be given to each case to determine if the potential benefit is worth the potential for delays. The following recommendations will help to characterize the environment where opportunistic locking may be effectively configured.

4.1 Exclusively Accessed Shares

Opportunistic locking is most effective when it is confined to shares that are exclusively accessed by a single user, or by only one user at a time. Because the true value of opportunistic locking is the local client caching of data, any operation that interrupts the caching mechanism will cause a delay.

Home directories are the most obvious examples of where the performance benefit of opportunistic locking can be safely realized.

4.2 Multiple-Accessed Shares or Files

As each additional user accesses a file in a share with opportunistic locking enabled, the potential for delays and resulting perceived poor performance increases. When multiple users are accessing a file on a share that has oplocks enabled, the management impact of sending and receiving oplock breaks, and the resulting latency while other clients wait for the caching client to flush data, offset the performance gains of the caching user.

As each additional client attempts to access a file with oplocks set, the potential performance improvement is negated and eventually results in a performance bottleneck.

4.3 Unix or NFS Client Accessed Files

Local HP-UX (Unix) and NFS clients access files without a mandatory file locking mechanism (see the whitepaper "CIFS/9000 File Locking Interoperation" at http://snslweb.cup.hp.com/getfile.php?id=58). Thus, these client platforms are incapable of initiating an oplock break request from the server to a Windows client that has a file cached. Local HP-UX or NFS file access can therefore write to a file that has been cached by a Windows client, which exposes the file to likely data corruption.

If files are shared between Windows clients, and either local HP-UX (Unix) or NFS users, then turn opportunistic locking off.

4.4 Slow and/or Unreliable Networks

The biggest potential performance improvement for opportunistic locking occurs when the client-side caching of reads and writes delivers the most differential over sending those reads and writes over the wire. This is most likely to occur when the network is extremely slow, congested, or distributed (as in a WAN). However, network latency also has a very high impact on the reliability of the oplock break mechanism, and thus increases the likelihood of encountering oplock problems that more than offset the potential perceived performance gain. Of course, if an oplock break never has to be sent, then this is the most advantageous scenario to utilize opportunistic locking.

If the network is slow, unreliable, or a WAN, then do not configure opportunistic lock if there is any chance of multiple users regularly opening the same file.

4.5 Multi-User Databases

Multi-user databases clearly pose a risk due to their very nature – they are typically heavily accessed by numerous users at random intervals. Placing a multi-user database on a share with opportunistic locking enabled will likely result in a locking management bottleneck on the CIFS/9000 Server. Whether the database application is developed in-house or a product such as Microsoft Access, ensure that the share has opportunistic locking disabled.

4.6 PDM Data Shares

Process Data Management (PDM) applications such as IMAN, Enovia, and Clearcase, are increasing in usage with Windows client platforms, and therefore SMB data stores. PDM applications manage multi-user environments for critical data security and access. The typical PDM environment is usually associated with sophisticated client design applications that will load data locally as demanded. In addition, the PDM application will usually monitor the data-state of each client. In this case, client-side data caching is best left to the local application and PDM server to negotiate and maintain. It is appropriate to eliminate the client OS from any caching tasks, and the server from any oplock management, by disabling opportunistic locking on the share.

4.7 Force User

CIFS/9000 Server includes an smb.conf parameter called "force user" that changes the user accessing a share from the incoming user to whatever user is defined by the smb.conf variable. If opportunistic locking is enabled on a share, the change in user access causes an oplock break to be sent to the client, even if the user has not explicitly loaded a file. In cases

where the network is slow or unreliable, an oplock break can become lost without the user even accessing a file. This can cause apparent performance degradation as the client continually reconnects to overcome the lost oplock break.

So avoid the following combination:

- "force user" in smb.conf share configuration, and
- Slow or unreliable networks, and
- Opportunistic locking

4.8 Advanced Samba Opportunistic Locking Parameters

Samba provides opportunistic locking parameters that allow the administrator to adjust various properties of the oplock mechanism to account for timing and usage levels. These parameters provide good versatility for implementing oplocks in environments where they would likely cause problems. The parameters are:

- oplock break wait time
- oplock contention limit

For most users, administrators, and environments, if these parameters are required, then the better option is to simply turn oplocks off. The samba SWAT help text for both parameters reads "DO NOT CHANGE THIS PARAMETER UNLESS YOU HAVE READ AND UNDERSTOOD THE SAMBA OPLOCK CODE." This is good advice.

4.9 Mission Critical High Availability

In mission critical high availability environments, data integrity is often a priority. Complex and expensive configurations are implemented to ensure that if a client loses connectivity with a file server, a failover replacement will be available immediately to provide continuous data availability.

Windows client failover behavior is more at risk of application interruption than other platforms because it is dependant upon an established TCP transport connection. If the connection is interrupted – as in a file server failover – a new session must be established. It is rare for Windows client applications to be coded to recover correctly from a transport connection loss, therefore most applications will experience some sort of interruption – at worst, abort and require restarting.

If a client session has been caching writes and reads locally due to opportunistic locking, it is likely that the data will be lost when the application restarts, or recovers from the TCP interrupt. When the TCP connection drops, the client state is lost. When the file server recovers, an oplock break is not sent to the client. In this case, the work from the prior session is lost. Observing this scenario with oplocks disabled, and the client was writing data to the file server real-time, then the failover will provide the data on disk as it existed at the time of the disconnect.

In mission critical high availability environments, careful attention should be given to opportunistic locking. Ideally, comprehensive testing should be done with all affected applications with oplocks enabled and disabled.

Chapter 5 Summary

Windows Opportunistic Locking is a lightweight performance-enhancing feature. It is not a robust and reliable protocol. Every implementation of Opportunistic Locking should be evaluated as a tradeoff between perceived performance and reliability. Reliability decreases as each successive rule above is not enforced. Consider a share with oplocks enabled, over a wide area network, to a client on a South Pacific atoll, on a high-availability server, serving a mission-critical multi-user corporate database, during a tropical storm. This configuration will likely encounter problems with oplocks.

Oplocks can be beneficial to perceived client performance when treated as a configuration toggle for client-side data caching. If the data caching is likely to be interrupted, then oplock usage should be reviewed. CIFS/9000 Server and Samba enable opportunistic locking by default on all shares. Careful attention should be given to the client usage of shared data on the server, the server network reliability, and the opportunistic locking configuration of each share.