Scalla: Extended Features Supplement
Clustering
Configuration Reference
olbd & xrootd

Andrew Hanushevsky
Stanford Linear Accelerator Center
18-July-2007
1 Introduction............................................................................. 5
  1.1 Directives and Components................................................... 8
    1.1.1 Clusters with 64 or fewer data servers......................... 9
    1.1.2 Clusters with 65 or more data servers...................... 11
    1.1.3 Frequently asked questions........................................ 14
  1.2 Created Files......................................................................... 16
  1.3 Starting the olbd Process..................................................... 17
    1.3.1 Running multiple olbd’s on the same host.................... 20
2 Mandatory Configuration Directives............................... 21
  2.1 manager................................................................................ 21
  2.2 role ....................................................................................... 23
    2.2.1 Role Summary Table..................................................... 25
3 Common olb Configuration Directives........................... 27
  3.1 allow .................................................................................. 27
  3.2 cache ................................................................................... 30
  3.3 defaults ............................................................................... 33
  3.4 export ................................................................................. 35
  3.5 localroot............................................................................. 37
  3.6 perf ...................................................................................... 38
  3.7 prep ...................................................................................... 39
  3.8 sched ................................................................................... 43
  3.9 space ................................................................................... 45
4 Esoteric olb Configuration Directives ............................. 47
  4.1 adminpath............................................................................ 27
  4.2 delay .................................................................................... 47
  4.3 fxhold.................................................................................. 50
  4.4 fsxeq ................................................................................... 51
  4.5 namelib............................................................................... 53
  4.6 nowait .................................................................................. 54
  4.7 pidpath............................................................................... 55
  4.8 ping ...................................................................................... 57
  4.9 remoteroot.......................................................................... 59
  4.10 trace .................................................................................... 60
  4.11 xmilib............................................................................... 61
5 Deprecated olb Configuration Directives....................... 63
  5.1 path ...................................................................................... 63
  5.2 port...................................................................................... 64
  5.3 subscribe ............................................................................. 65
6 Esoteric odc Configuration Directives ......................... 67
  6.1 conwait ........................................................................ 67
  6.2 msgkeep ...................................................................... 68
  6.3 request ....................................................................... 69
  6.4 trace .......................................................................... 70
7 Deprecated odc Configuration Directives .................. 71
  7.1 olbapath ..................................................................... 71
8 Document Change History ........................................... 72
1 Introduction

This document describes Open Distributed Clustering (odc) and Open Load Balancing (olb) configuration directives. These two components are meant to be used together to provide dynamic load balancing of files and name-space consolidation of distributed data regardless of location. The odc and olb components are meant to be used with xrootd’s Open File System (ofs) component. Refer to the “xrootd ofs & oss Configuration Guide” for detailed information.

Directives for odc and olb come from a configuration file. The characters “odc” or “olb” must prefix each directive in the configuration file. Directives that apply to multiple components must be preceded by the characters “all”. This makes odc and olb directives compatible with the xrootd’s other configurable components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc</td>
<td>Access control (i.e., authorization)</td>
</tr>
<tr>
<td>odc</td>
<td>Open Distributed Clustering</td>
</tr>
<tr>
<td>ofs</td>
<td>Open file system coordinating acc, odc, &amp; oss components</td>
</tr>
<tr>
<td>olb</td>
<td>Open Load Balancing</td>
</tr>
<tr>
<td>oss</td>
<td>Open storage system¹ (i.e., file system implementation)</td>
</tr>
<tr>
<td>sec</td>
<td>Security authentication</td>
</tr>
<tr>
<td>xrd</td>
<td>Extended request daemon.</td>
</tr>
<tr>
<td>xrootd</td>
<td>xrootd protocol.</td>
</tr>
<tr>
<td>all</td>
<td>Applies the directive to all of the above components.</td>
</tr>
</tbody>
</table>

Records that do not start with a recognized identifier are ignored. This includes blank record and comment lines (i.e., lines starting with a pound sign, #). This guide documents the all, odc, and olb configuration directives (i.e., the un-shaded rows). Other directives are documented in supplemental guide specific to the component they deal with.

The location of the configuration file is specified on the command line². Because each component has a unique prefix, a common configuration file can be used for the whole system.

Refer to the manual “Configuration File Syntax” on how to specify and use conditional directives and set variables. These features are indispensable for complex configuration files usually encountered in large installations.

¹ Certain oss directives are recognized and honored by olbd.
² The olbd command line for olb directives and the xrootd command line for odc directives.
A configuration file is mandatory, even if nothing is specified in it. Because load balancing is sensitive to other configuration options, the olbd does recognize certain specific directives that start with the oss prefix. This allows the use of a single configuration file without the need to duplicate directives. These directives may also be prefixed by olb, to direct them specifically to olbd and no other component. Directives that apply to the olb and odc components must be prefixed by the characters “all”.

Open load balancing is performed by a set of cooperating servers. One or more olbd servers run in manager mode and can be used by one or more xrootd’s to determine where to redirect a client’s file open request. The request can only be redirected to a machine that is running an olbd in server mode. There can be up to 64 olbd servers. Each machine can run one or more xrootd’s. The following figure illustrates a simple minimal system.

In the diagram, there are three hosts: x, y, and z. Host y serves as the load balancer. Hosts x and z are the hosts that can be used to serve data to clients and load is to be distributed between the two. Consequently, host y runs a manager olbd while hosts x and z run server olbd’s.
The servers connect to the manager and provide load information. The xrootd running on host y connects to the manager as well, using the odc component. However, the xrootd on host y uses the manager to determine which server to direct client requests. It does not serve any actual data files.

The typical open request is handled in four steps:
1. The client directs the open request to the xrootd that runs on the manager’s host.
2. The xrootd odc component asks the olbd manager which machine is the best to use to process the file. The manager determines the best machine using a variety of configurable parameters.
3. The xrootd on host y tells the client, in this example, that host z is the best host to use for the file.
4. The client then redirects the request to the xrootd running on host z.

In order to make the system as flexible as possible, the manager olbd does not know how many or which hosts will acts as servers. For security purposes, you can restrict hosts based on host name as well as by NIS netgroup. Thus, servers essentially subscribe to the manager claiming that they have file resources. During the subscription process, each server indicates the file paths to which it is willing to provide data access. Periodically, the manager olbd requests load information from each server. Each server reports CPU, network I/O, queue, memory, paging load as well as free space. This information is used to select the best available server for an open request.

The decision is tempered whether or not the server already has the file on disk or whether the file must be staged to disk from a Mass Storage System. The manager may decide that all available servers are too loaded and force a file to be replicated on a less loaded server. This provides additional data paths to the file. Replicated load balancing is only compatible with read-only files. The manager can direct client’s to a writable version of a file but only on servers that have indicated that they offer write access on the associated path. In general, only one such server may exist for each particular path.

In order to provide a fully redundant service, all servers may be replicated and cross-connected. The following figure shows a full crossbar configuration. Each server olbd subscribes to two manager olbd’s. Each xrootd that can redirect clients subscribes to two managers. Thus, the loss of any single manager xrootd does not affect load balancing. More complex arrangements are possible since each server may have any number of managers and each xrootd can subscribe to any number of managers.
In order to ease migration, any peripheral (i.e., data server) `xrootd` can always be directly used\(^3\). This means that load balancing only occurs when a client contacts a redirecting `xrootd`. For systems that are being configured this way for the very first time, you should always use the “`xrd.port any`” directive for data server `xrootd`’s. This allows the `xrootd` to use an arbitrary port number. In this mode it is very difficult for any client to directly use a data server `xrootd` without first contacting the manager `xrootd` first.

### 1.1 Directives and Components

Open Load Balancing consists of four distinct components:

1. The manager `olbd` process (typically in a separate machine).
2. Supervisor `olbd` processes (\textit{only} for clusters of more than 64 servers).
3. Server `olbd` processes, and
4. The `odc` component running in the `xrootd` process, which can be a manager, supervisor, or server.

A manager `olbd` always communicates with supervisor and server `olbd`’s as well as the `odc` component of a manager `xrootd`. Server `odc` components only communicate only with their server `olbd` counterpart. Two distinct directives are used to identify the participants:

1. `all.role` which tells each component whether it is to function as a manager, supervisor, or server; and
2. `all.manager` that tells each component the DNS name of the manager.

\(^3\) You make effectively prevent this by using the `–a xrootd` command line option.
1.1.1 Clusters with 64 or fewer data servers

Use the following general steps to successfully configure a dynamic load balancing system that has 64 or fewer data servers:

- Determine which servers will be used for data serving (i.e., run a data server `olbd`) and which for redirection (i.e., run a manager `olbd`). A manager is not capable of also serving data. Use the “all.role” directive to differentiate servers and managers.
- Use the `allow` directive to restrict the set of valid data servers.
- Determine the scheduling policy using the `olb.sched` and `olb.space` directives.
- Determine which port number will be used for inter-server communication and specify it for the manager `olbd` using the `mandatory all.manager` directive (see below).
- Remember to specify the `xrootd` port number for the associated manager `xrootd` using the `xrd.port` directive.
- Determine the set of data servers. Unless direct access is important, you should configure data servers with “`xrd.port any`”.
- For each data server, determine which file paths it will handle. In general, a number of servers should serve the same path.
- Use the `export` directive on each data server to restrict it to its set of paths.
- If a data server will be also creating files, use the `cache` directive to indicate which file systems may be used for file creation.
- Use the `all.manager` directive to tell each `xrootd` and `olbd` the location of its set of managers.
- Start a data server `olbd` server on each `xrootd` data server machine. For each `xrootd` that will be redirecting, use the `all.role manager` directive to enable redirection mode.\(^4\)
- Start the manager, server, and `xrootd`. See the `olbd` command described in the next section. The start-up order is not important.

The following diagram and corresponding configuration file illustrates how to cluster 30 data servers with two managers.

\(^4\) Refer to the “`xrootd ofs & oss` Configuration Guide” for more information.
# Specify the data server port number. This is only relevant to managers, so we qualify the specific port number using the “if”.  
# xrd.port any  
xrd.port 1094 if man01.u.org man02.u.org

# Using an olbd requires the enhanced file system. We must specify the location of the shared library that implements the file system.  
# xrootd.fslib /opt/xrootd/lib/libXrdOfs.so

# Specify which paths are to be exported (default is r/w)  
# all.export /data

# Tell everyone the role it will have. Use a default of server but then qualify it depending on hostname using the “if”.  
# all.role server  
all.role manager if man01.u.org man02.u.org

# Tell everyone the location of each manager.  
# all.manager man01.u.org:1213  
all.manager man02.u.org:1213

# Tell the olbd which machines are allowed to connect  
# olb.allow host man*.u.org  
olb.allow host data*.u.org

**Configuration “myconfig.cf” for a 30 Data Server Cluster**

There are additional directives to further tune the system and are described on the following pages. You should also review the `odc` directives since these are used to tell a redirecting `xrootd` how to handle load balancing.
1.1.2 Clusters with 65 or more data servers

Configuring a cluster of more than 64 data servers is just slightly more complicated than configuring a smaller cluster. The complication arises from the fact that some additional management servers need to be started. The configuration file, however, is no more complicated. Below are the steps you should take to successfully configure large clusters.

- Choose the port numbers you wish to use for the manager *xrootd* and *olbd* servers. Typically, *xrootd* uses port 1094 and specified with the `xrd.port` directive. For *olbd* port 1213 is used and specified with the `all.manager` directive. Other *xrootd* servers should specify “`xrd.port any`”.

- Choose the number of manager nodes you wish to run. You must configure at least one manager node. The manager is the first point of contact for a client and is also the cluster leader. A manager should run on a dedicated machine of modest power (e.g., 512MB RAM, 800MHz clock speed, 100Mb ethernet).

A manager node consists of
  a) an *xrootd* configured with the “all.role manager” directive.
  b) an *olbd* configured with “all.role manager” directive.

You may configure more than one manager and run them in either fail-over mode (the default) or in load balancing mode where each manager shares part of the client load (see the `all.manager` directive). Each manager *xrootd-olbd* pair must run on a separate machine.

- Compute the number of supervisor nodes you need. A supervisor node acts as a local manager for a group of 64 other nodes. These nodes may be data servers or supervisors. A supervisor node consists of
  a) an *xrootd* configured with the “all.role supervisor” directive.
     Additionally, specify the “`xrd.port any`” directive.
  b) an *olbd* configured with the “all.role supervisor” directive.

You only need to configure supervisor nodes if you are running more than 64 data servers. The number of supervisor nodes is based on the number of available manager plus supervisor slots. A recursive formula is needed to calculate the minimum number. Since you normally wish to start more than the minimum number of supervisors, a simplified formula can be used.

Conservatively, you will need one supervisor node for each group of 64 data servers. For instance, if you plan to run 500 data servers you will need the upper limit of 500/64 supervisors (i.e., 8).

Each supervisor node can run on a data server node. If you wish to share resources in this way, choose data server nodes that will be as lightly loaded as possible. The performance requirements for a supervisor node are the same as a manager node.
• Configure the data server nodes. A data server node delivers actual data to clients. It consists of
  a) an *xrootd* configured with the “*all.role server*” directive. Additionally, specify the “*xrd.port any*” directive.
  b) an *olbd* configured with the “*all.role server*” directive.

Configure as many data server nodes as you need. Keep in mind that at least one additional supervisor node is needed for each group of 64 data servers.

The performance requirements are determined by the performance needs of clients. The server should have enough disk space, adequate network bandwidth (e.g., Gb ethernet), and significant cpu and i/o resources. If you wish to use memory mapped files, then the node should have a commensurate amount of real memory.

For example, assume you wish to cluster 99 data servers in the way shown below.

![Diagram of 99 Data Server Cluster]

Here we wish to have only one manager. We will need at least one supervisor. While the simplistic formula indicates two supervisors are needed; in practice, the cluster could self-organize by affiliating 63 data servers and one supervisor (a total of 64) with the manager and affiliating the remaining data servers (36) with the supervisor.

With two supervisors, the cluster would affiliate 62 data servers and two supervisors with the manager, and split the remaining data servers across the two supervisors. So, either configuration would work. Fortunately, the cluster attempts to automatically find the best organization given the resources at hand. Configuration files for small and large clusters will differ only slightly from each other. Notable differences involve *allow* and *role* directives. Configuration file simplicity relies on the use of regular names for various hosts.
# Specify the data server port number. This is only relevant to
# managers, so we qualify the actual port number using the “if”.
#
# xrd.port any
# xrd.port 1094 if man01.u.org
#
# Using an olbd requires the enhanced file system. We must specify the
# location of the shared library that implements the file system.
#
# xrootd.fslib /opt/xrootd/lib/libXrdOfs.so
#
# Specify which paths are to be exported (default is r/w)
#
# all.export /data
#
# Tell the olbd which machines are allowed to connect
#
# olb.allow host man01.u.org
# olb.allow host sup01.u.org
# olb.allow host data*.u.org
#
# Indicate the role this server will have based on host name (the
# default role is that of server)
#
# all.role server
# all.role supervisor if sup01.u.org
# all.role manager if man01.u.org
#
# Tell everyone the location of the manager.
#
# all.manager man01.u.org:1213
1.1.3 Frequently asked questions

**Does start-up order matter?**

Generally, it does not matter in which order nodes are started. For the efficiency minded, starting supervisor nodes ahead of data server nodes allows the system to converge on a stable configuration faster.

**How long will it take for the system to converge?**

This depends on how many servers are in the configuration. Generally, it takes approximately 1 to 13 seconds for a server to find its correct place in the cluster. However, the process is run in parallel across all of the servers. So, the system should converge in less than 30 seconds for a configuration of about 1,000 nodes. By default, the system delays full availability for 90 seconds, this should be sufficient time for convergence of even extremely large installations.

**What happens if I have too few supervisors?**

If there are not enough supervisors relative to the number of data servers, one or more data servers will be orphaned and unavailable. If you suspect this, check the manager’s log. It will contain warnings about orphaned data servers.

**What happens if I have more supervisor nodes than I need?**

Since the system tries to evenly distribute data servers across all available supervisors, excess supervisors are used to further reduce the load on supervisor nodes. The excess supervisors are also used as “hot spares” in the event one of the supervisors becomes unavailable. You should configure as many “extra” supervisors as you feel are necessary to provide a suitable level of fault tolerance.

**Can I run all the supervisors on a single node?**

Yes, but you will need to assign each **olbd** a unique instance name using the `-n` option. Additionally, the same `-n` option value must be specified for the **xrootd** that is paired with a particular **olbd**. Use the “if” directive, keyed off the instance name, to maintain a single configuration file. Finally, each **xrootd**, other than the one tied to the manager **olbd**, must be started with the “port any” directive to allow for arbitrary port selection. You should realize that running all of the supervisors on a single node creates a large single point of failure.

**How do I run a data server and a supervisor on the same node?**

Use the provided **StartOLB** and **StartXRD** scripts. For a supervisor **olbd** and **xrootd**, specify the “all.role supervisor”. For a data server **olbd** and **xrootd** specify the “all.role server” directive. You should make sure that “xrd.port any” is specified for supervisor and data server **xrootd’s** to prevent any port conflicts.
What does the “--port any” xrootd command line option actually do?
The “--port any” option allows xrootd to choose any port that is available. The selected port number is then forwarded to the olbd. This allows the olbd to redirect clients to the proper port even though it’s not known ahead of time. This only works if the olbd is not started with the -i option (the default) and the xrootd is started with the “all.role server” (for data servers) or all.role supervisor directive (for supervisors). This does not eliminate the need for starting the manager olbd and its xrootd counterpart with well-known ports.

Does that mean I can use --port any to run multiple data servers on a single node?
Yes. See the answer to “Can I run all the supervisors on a single node?”.

Can I use the --port any option to prohibit clients to bypass the olbd?
Yes. This is actually recommended. Since arbitrary port numbers are chosen, a client cannot directly connect to a data server without using the manager xrootd. However, while significant programming effort is required to capture port numbers at run-time; any “management by obscurity” method can be defeated.
1.2 Created Files

The following files are created by the `olbd`:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Modified by</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/tmp/[name/].olb/olbd</code></td>
<td>TCP Socket</td>
<td><code>adminpath</code></td>
<td>Local xrootd – server <code>olbd</code> communications</td>
</tr>
<tr>
<td><code>/tmp/[name/].olb/olbd.super</code></td>
<td>TCP Socket</td>
<td><code>adminpath</code></td>
<td>Local xrootd - supervisor <code>olbd</code> communications</td>
</tr>
<tr>
<td><code>/tmp/[name/].olb/olbd.notes</code></td>
<td>UDP Socket</td>
<td><code>adminpath</code> and <code>-n</code> option</td>
<td>Local <code>olbd</code> server event notifications</td>
</tr>
<tr>
<td><code>/tmp/[name/].olb/olbd.seton</code></td>
<td>UDP Socket</td>
<td><code>adminpath</code></td>
<td>Local <code>olbd</code> supervisor event notifications</td>
</tr>
<tr>
<td><code>/tmp/[name/].olbd.pid</code></td>
<td>File</td>
<td><code>pidpath</code> and <code>-n</code> option</td>
<td>Holds the process id and the local path prefix (i.e., <code>localroot</code>) for a server <code>olbd</code>.</td>
</tr>
<tr>
<td><code>/tmp/[name/].olbd.mangr.pid</code></td>
<td>File</td>
<td><code>pidpath</code> and <code>-n</code> option</td>
<td>Holds the process id and the local path prefix (i.e., <code>localroot</code>) for a manager <code>olbd</code>.</td>
</tr>
<tr>
<td><code>/tmp/[name/].olbd.superpid</code></td>
<td>File</td>
<td><code>pidpath</code> and <code>-n</code> option</td>
<td>Holds the process id and the local path prefix (i.e., <code>localroot</code>) for a supervisor <code>olbd</code>.</td>
</tr>
<tr>
<td><code>/var/adm/olbd/core/[name/].core</code></td>
<td>File</td>
<td><code>-n</code> option</td>
<td>Core file via default in <code>StartOLB.cf</code></td>
</tr>
<tr>
<td><code>/var/adm/olbd/logs/[name/].olbd.log</code></td>
<td>File</td>
<td><code>-l</code> option and <code>-n</code> option</td>
<td>Log file via default in <code>StartOLB.cf</code></td>
</tr>
</tbody>
</table>

The `adminpath` directive specifies the directory where the remaining files are written. The `pidpath` directive specifies the directory where the pid file is written. The `-n` option specifies the `olbd` instance name. If specified, the instance name is automatically suffixed to the `adminpath` and `pidpath`, as shown by `“[name/].”`. A directory is also create in the current working directory for core files and the log file destination is modified by inserting `“[name/].”` in the destination specified by the `-l` option. If necessary, the directory is created.
1.3 Starting the olbd Process

Use the following command to start a manager or server `olbd` process.

```
olbd [ options ] -c cfn
```

**options:**  
`[-l fn][-n name] [esoteric]`

**esoteric:**  
`[-d] [-i] [-k {num | sz{k|m|g}}]`

**Function**  
Start the Open Load Balancing Daemon.

**Parameters**  
`-c cfn` is the name of the configuration file. You must specify the name of a configuration file even if it is empty.

**Options**  
`-l fn` directs error messages and any trace output to the indicated file, `fn`. By default, messages are directed to standard error.

`-n name` sets the instance name of the `olbd`. There is no default. See the notes for more information on this option.

**Esoteric Options**  
`-d` turns on debugging.

`-i` allows the `olbd` subscribes to a manager `olbd` whether or not the local primary data server contacts the `olbd`. Also, see the `olb.nowait` directive.

`-k num | sz{k|m|g}` keeps no more than `num` old log files. If `sz` is specified, the number of log files kept (excluding the current log file) is determined by how much space they use. Hence, kept log files will not exceed `sz` bytes. The `sz` must be suffixed by `k`, `m`, or `g` to indicate kilobytes, megabyte, or gigabytes, respectively.

**Defaults**  
`olbd` starts in manager mode.
Notes

1) A configuration file is not optional.
2) The same configuration file may be used for manager and server olbd’s. Directives not relevant to a particular mode of operation are ignored.
3) The odc and olbd related directives may be placed in the xrootd configuration file as well. Thus only one configuration file needs to be maintained per machine.
4) If a log file is specified, the file is closed at midnight, renamed to have a date suffix (i.e., `fn.yyyymmdd`) and possible sequence number (i.e. `fn.yyyymmdd.n`), and a new log file is opened. This allows you to write an external time-driven script that rotates olbd log files.
5) The order in which servers are started is unimportant.
6) You must start at least one olbd in manager mode. The number of supervisor olbd’s is approximately determined by dividing the number of server mode olbd’s by 64 less one.
7) In a supervisor role, the olbd acts as both manager and server. Supervisor olbd’s are used to cluster groups of 64 server olbd. Since a supervisor olbd can subscribe to a manager or supervisor olbd, it is possible to cluster together a virtually unlimited number of data servers.
8) During start-up, an olbd writes its process id as well as its local path prefix and administrative path into a file. The location of the file is determined by the pidpath directive, as modified by the –n option. The name of the file is determined by the olbd’s role (see the role directive). The following table summarizes the various forms of the process id file name using the default location of `/tmp`.

<table>
<thead>
<tr>
<th>Process ID File Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/tmp/\[name/\].olbd.mangr.pid</code></td>
<td>Manager</td>
</tr>
<tr>
<td><code>/tmp/\[name/\].olbd.super.pid</code></td>
<td>Supervisor</td>
</tr>
<tr>
<td><code>/tmp/\[name/\].olbd.pid</code></td>
<td>Server</td>
</tr>
</tbody>
</table>

The instance name is substituted for “\[name/\]”. If there is no instance name, then “\[name/\]” is not present in the file’s name.

Notes on Esoterics

1) The –i option provides for a loose coupling between servers running on the same host. The olbd executes asynchronously from the host’s data server and can subscribe to a manager before the data server is available on the host.
2) Without –i, a host is not available for selection until the host’s data server is ready.
3) Once the xrootd contacts the olbd, the host automatically becomes ineligible for selection whenever the data server becomes unready,

4) The –i option is meant for to be used with data servers that are unable to communicate with the local olbd. You should not specify this option for the xrootd server.

5) **Warning:** the default olbd mode (i.e., wait for data server) must be used in conjunction with xrootd’s configured for clustering; otherwise the host will never be selected by the manager olbd.

6) **Warning:** The –i option disables port remapping. With port remapping, a client is redirected to the port actually being used by the data server that is the target of the redirection. This allows arbitrary or hidden ports to be used, none of which need be the same. When port remapping is disabled, clients are always redirected to the port they initially used to contact the redirector.

7) The –i option is automatically implied if you include the nowait directive in the configuration file.

8) **Warning:** Command line options over-ride configuration file directives.

**Example**

    olbd -c /opt/olb/olbd.cf
1.3.1 Running multiple olbd’s on the same host

The olbd supports running as many olbd’s as you would like on the same host (i.e., machine). This is accomplished by the –n command line option. This option assigns an instance name to the olbd. The olbd uses instance name to maintain a separate disk name space for files that it needs to create.

There is no default instance name; however, the system uses the word anon to refer to unnamed olbd’s.

In order to run more than one olbd on the same host, each olbd must be given a unique instance name that is also valid as a directory name. Server and supervisor olbd’s pose no port contention problems since they always use whatever port happens to be free. Manager olbd’s are assign specific port numbers (see the port directive). Therefore, if you wish to run more than one olbd manager on a host, each manager must also be assigned a unique port number.

The olbd’s always work in pairs with xrootd’s. The pairing only works within the same instance. That is, if an olbd with an instance name of foo is to be used with a particular xrootd; then that xrootd must be given an instance name of foo as well.

Failure to follow these directions will prevent proper communications from being established between xrootd’s and olbd’s.

Notes

1) You may run one instance of a supervisor olbd on the same machine as you run a server mode olbd. You may not run manager and supervisor olbd’s on the same machine unless you designate a new instance name (see the –n option) for the manager olbd.

2) By design, there can only be one logical instance combination of a manager, supervisor, and server running on the same machine. The -n option allows you to create new logical instances by assigning each instance a different name. This allows you to run multiple instances of the olbd on the same machine.

3) Since the instance name is used to pair xrootd’s with their respective olbd’s, specify instance names consistently for xrootd’s and olbd’s using the -n option on the xrootd command line.
2 Mandatory Configuration Directives

This section describes directives that are must be specified to configure the Open Load Balancing component as well as the Open Distributed Cluster component of xrootd.

2.1 manager

all.manager [ peer | proxy ] [ all | any ]

host[+]{:port | port} [ if conds ]

Function
Specify the manager olbd location.

Parameters
peer identifies the olbd managers that olbd servers with peer roles (i.e., “peer manager” or “peer proxy manager”) should subscribe to.

proxy identifies the olbd managers that xrootd servers with proxy roles (i.e., “proxy” or “proxy server”) should subscribe to.

all uses a load distribution algorithm to select an appropriate manager. Refer to the usage notes for more information.

any uses a fail-over algorithm to select an appropriate manager. Refer to the usage notes for more information.

host is the DNS name or IP address of the olbd manager. If host ends with a plus sign (+), then the all hosts addresses associated with host are considered to be available managers.

port is the TCP port number or service name at which the manager will accept connections. The port may be specified with an adjacent colon or space separation.

conds are the conditions that must exist for this directive to apply. Refer to the description of the if directive on how to specify conds.

Defaults
None; see the Notes for requirements. If you do not specify all or any, then any is assumed.
Notes
1) You must specify the “manager” directive for each xrootd given a manager role and for every olbd given a server or supervisor role.
2) You must specify the “manager peer” directive for every olbd given a peer or peer manager role.
3) You must specify “manager proxy” directive for each xrootd given a proxy or proxy server role.
4) This is a global directive and must be qualified by the “all” prefix.
5) All non-peer manager olbd’s use the manager directive to establish a communications channel with each indicated manager.
6) You may specify up to 16 different managers.
7) For xrootd requests, when more than one manager is present, requests can either be distributed among the managers (all mode) or sent to a primary manager; with alternates used as backup servers (any mode). In either case, should a manager fail, the next available manager is selected.
8) If the manager host name ends with a plus, then all the IP addresses associated with host are treated as managers and every non-manager olbd and xrootd subscribes to each one. This allows you to easily construct fault-tolerant configurations using the IP address alias feature in DNS.
9) The host specifies the machine that is running olbd in a manager role.

Example
   all.manager beastmanager.slac.Stanford.edu olbport
2.2 role

```
all.role rolename [ if conds ]
rolename: [peer] [proxy] manager | peer | proxy |
            [proxy] server | [proxy] supervisor
```

Function
Designate the role the server is to have.

Parameters
rolename
specifies the server’s role in the configuration. See the usage notes and the following section for an explanation of roles.

conds are the conditions that must exist for this directive to apply. Refer to the description of the if directive on how to specify conds.

Defaults
all.role manager

Notes
1) This is a global directive and must be qualified by the “all” prefix.
2) A role of manager indicates that the olbd is at the top-most level of the server hierarchy and is used to locate files.
3) A role of server indicates that the olbd is at the bottom-most level of the server hierarchy and is used by pure data servers to serve data files.
4) A role of supervisor indicates that the olbd is at an intermediate-level of the server hierarchy and is used to bridge the top-most level and the bottom-most level.
5) A role of peer indicates that the olbd is not only to act as a manager but also is to subscribe to other managers and act as a peer.
6) A role of proxy indicates that the xrootd is at the top- and bottom-most level of the server hierarchy. When contacted, the xrootd acts like a manager to locate the target file. However, unlike a true redirector, the xrootd actually performs the requested operation as if it were a server acting in behalf of the client making the request.
7) Any xrootd’s designated as proxies may only communicate with olbd’s that have also been designated as proxies. However, peer proxy components may communicate with any type of manager olbd.
8) The following table describes the effect each role has on an \texttt{xrootd}
server and its corresponding \texttt{olbd} server.

Example
\begin{verbatim}
all.role supervisor if sup*.slac.stanford.edu
\end{verbatim}
### 2.2.1 Role Summary Table

<table>
<thead>
<tr>
<th>Role</th>
<th>olbd</th>
<th>xrootd</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manager</strong></td>
<td>Provides a search service across one or more “server” or “supervisor” olbd’s.</td>
<td>Logs into one or more olbd’s, identified by the “manager” directive, and provides a redirection service</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Subscribes to a “manager” olbd, identified by the “manager” directive, in order to form a cluster <em>and</em> accepts logins from a local xrootd.</td>
<td>Logs into a local “server” olbd and provides data from a locally accessible file system.</td>
</tr>
<tr>
<td><strong>Supervisor</strong></td>
<td>Same as “server” <em>plus</em> provides a search service across one or more server or supervisor olbd’s.</td>
<td>Logs into a local “supervisor” olbd and provides a redirection service.</td>
</tr>
<tr>
<td><strong>Proxy</strong></td>
<td><em>Not supported.</em> This role is only meant to designate stand-alone xrootd proxy servers.</td>
<td>Logs into one or more olbd’s, identified by the “manager proxy” directive, <em>and</em> provides data from servers identified by the manager olbd.</td>
</tr>
<tr>
<td><strong>Proxy manager</strong></td>
<td>Same as “manager” <em>but</em> only accepts olbd’s and xrootd’s that have a “proxy” role (i.e., can only manage proxies).</td>
<td>Same as manager role <em>except</em> that the manager olbd’s must also have a proxy role.</td>
</tr>
<tr>
<td><strong>Proxy server</strong></td>
<td>Same as “server” <em>except</em> that managers and the local xrootd must also have a proxy role.</td>
<td>Same as “proxy” <em>and</em> logs into a local “proxy server” olbd to be part of a cluster.</td>
</tr>
<tr>
<td><strong>Proxy supervisor</strong></td>
<td>Same as “supervisor” <em>but</em> only allows proxy olbd subscriptions (i.e., can only manage proxies).</td>
<td>Logs into a local “proxy supervisor” olbd and provides a redirection service.</td>
</tr>
<tr>
<td>Role</td>
<td>olbd</td>
<td>xrootd</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Peer</td>
<td>Same as peer manager <em>but</em> neither needs nor allows any olbd subscribers.</td>
<td>Same as manager role.</td>
</tr>
<tr>
<td>Peer manager</td>
<td>Same as manager role <em>plus</em> subscribes to other manager olbd’s, identified by the “manager peer” directive, as an alternate.</td>
<td>Same as manager role.</td>
</tr>
<tr>
<td>Peer proxy manager</td>
<td>Same as “peer manager” <em>but</em> only allows proxy olbd subscriptions (i.e., can only manage proxies).</td>
<td>Same as manager role <em>except</em> that the manager olbd’s must also have a proxy role.</td>
</tr>
<tr>
<td>Peer</td>
<td>Same as peer manager <em>but</em> neither needs nor allows any olbd subscribers.</td>
<td>Same as manager role.</td>
</tr>
</tbody>
</table>
3 Common olb Configuration Directives

This section describes directives that are commonly specified to configure the Open Load Balancing component.

3.1 adminpath

```
all.adminpath path [ group ]
```

Function
Designate the path used for administrative communications.

Parameters

path is the path to used for Unix named sockets. Administrative commands can be directed to this socket.

group allows group access to path.

Defaults (see warning in the notes)
The default administrative path is “/tmp/olb”.

Notes

1) This directive is only used by manager and server-mode olbd’s.
2) Warning: if idle /tmp directories and socket files are automatically deleted by the system, you should neither accept the default path nor allow the adminpath path to reside in /tmp.
3) Unless group is specified, only the user running olbd can write the named socket. This is done to ensure that non-privileged users cannot send olbd administrative requests.
4) The server and manager olbds respectively create stream sockets named “olbd.admin” and “olbd.nimda” in this directory. These sockets are used for administrative communications.
5) The server and manager olbds respectively create datagram socket named “olbd.notes” and “olbd.seton” in this directory. These sockets are used for external notifications.
6) Supervisor olbd’s create a stream socket name “olbd.super”.
7) The adminpath is modified by the –n command line option.
8) If your system periodically cleans out /tmp, you may wish to relocate the named sockets. Typically, “/var/run/olbd” is the normally accepted alternative location.
Example

\texttt{all.adminpath \textasciitilde/var/run/olbd}
3.2 allow

```
olb.allow { host | netgroup } name
```

**Function**

Restrict the hosts that can subscribe to the manager *olbd*.

**Parameters**

**host name**

specifies the DNS host name allowed to subscribe to the *olbd*. Substitute for *name* a host name. The host name may contain a single asterisk anywhere in the name. This lets you allow a range of hosts should the names follow a regular pattern.

**netgroup name**

specifies the NIS netgroup allowed to subscribe to the *olbd*. Substitute for *name* a valid NIS netgroup. Only hosts that are members of the specified netgroup are allowed to subscribe to the *olbd*.

**Defaults**

None. If *allow* is not specified, any host is allowed to subscribe.

**Notes**

1) This directive is only used by manager-mode *olbd*’s.
2) You may specify any number of hosts and netgroups. Any host matching a specified name or is a member of a specified netgroup is allowed to subscribe to the *olbd*.
3) **Warning!** Using hostname based security relies on the security of the DNS server and the inability of other hosts spoofing and successfully using the “allowed” IP addresses. The two security assumptions have severe limitations.

**Example**

```
olb.allow host kandata*.slac.stanford.edu
```
3.3 cache

```plaintext
olb.cache group { path | ppfx* }
```

or identically

```plaintext
oss.cache group { path | ppfx* }
```

### Function

Specify the location of a disk cache partition.

### Parameters

- **group**
  
  is the arbitrary logical group name for the disk partition. Specify a 1- to 16-character name. While the cache group is required, the **olbd** does not use it for any purpose.

- **path**
  
  is the absolute path at which the disk cache partition is mounted.

- **ppfx***
  
  all directory entries that start with `ppfx` in the containing directory are to be used as disk cache partitions.

### Defaults

None. Staging is disallowed if no cache directories are specified.

### Notes

1) This directive is identical to the **oss.cache** directive. In fact, the **olbd** honors the **oss** version of this directive. This allows you to keep a single configuration file.

2) In order to redirect staging operations and file creations, the manager **olbd** must know how much space is available on each server.

3) If the **xrootd** server is running a caching file system (i.e., files are allocated via symbolic links to one of many possible file systems) then specify each file system cache area.

4) The path may end in an asterisk, indicating that all entries in the parent directory that start with the specified prefix are part of the cache area. This is useful when cache areas have regular names (e.g., `/data/cache01`, `/data/cache02`, etc.).
5) If the **olb** does not find any **cache** directives, it infers the file systems to be used using the **path** directive. If no path directive exists, the **olb** uses the root file system.

**Example**

```
olb.cache public /xrootd/cache01
```

*or identically*

```
oss.cache public /xrootd/cache01
```
3.4 defaults

```markdown
oss.defaults options


{forcero | readonly | r/o | r/w | [not]writable}

{inplace | outplace} {local | global | globalro}

{[no]mig | [not]migratable} [no]mkeep


[no]ssdec [no]stage
```

Function

Specify default file processing options.

Parameters (Non-grayed options apply to the olbd as well as the oss component)

<table>
<thead>
<tr>
<th>Option</th>
<th>Disabled/Enabled Function</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>forcero</td>
<td>Convert all file open requests to read-only access.</td>
<td>writable</td>
</tr>
<tr>
<td>local</td>
<td>Do not export this path via the cluster manager (i.e., redirector).</td>
<td>global</td>
</tr>
<tr>
<td>global</td>
<td>Export this path via the cluster manager (i.e., redirector)</td>
<td>global</td>
</tr>
<tr>
<td>globalro</td>
<td>Export this path via the cluster manager (i.e., redirector) as a read-only path.</td>
<td>global</td>
</tr>
<tr>
<td>readonly</td>
<td>Files may only be opened for read access.</td>
<td>writable</td>
</tr>
<tr>
<td>r/o</td>
<td>Path is writable.</td>
<td>writable</td>
</tr>
<tr>
<td>r/w</td>
<td>Path is writable.</td>
<td>writable</td>
</tr>
<tr>
<td>[no]stage</td>
<td>[Do not] stage a file from a Mass Storage System should it not exist in the local file system at open time.</td>
<td>stage when the stagecmd has been specified; otherwise, nostage.</td>
</tr>
<tr>
<td>[not]writable</td>
<td>Path is [not] writable.</td>
<td>writable</td>
</tr>
</tbody>
</table>
Notes

1) Directive options may be applied to selected paths using the export directive. This allows you to selectively over-ride the default.
2) The defaults directive should be specified prior to any export directives.
3) The oss uses the oss.defaults directive to establish path defaults.
4) Mass Storage System interfaces are configured using the stagecmd.

Notes on forcero

1) The forcero option forces all read/write file open requests to be converted to read/only opens, thus preventing modifications to all files. No errors are returned unless an actual write operation is attempted.
2) When the forcero option is specified, files may not be created, deleted, or renamed. Any attempt to do so will cause a “read only filesystem” error to be reflected to the client program.
3) The mlock, mkeep, and mmap options cause a path to have the forcero attribute.

Notes on readonly

1) The readonly option forces an immediate error to occur should a file be opened for read/write access. Files may also not be created, deleted, or renamed. Any attempt to do so will cause a “read only filesystem” error to be reflected to the client program.

Notes on [no]stage

1) When stage is in effect, files are dynamically staged into the local file space when opened, if the file is not already on disk.
2) The nostage is useful in those instances where installation policy mandates that files be manually pre-staged prior to use.
3) The nostage and stage directives may be applied to selected paths using the path directive.
4) Mass Storage System interfaces are configured using the oss.stagecmd and the oss.msscmd directives.
5) If you explicitly specify stage, you must also specify the oss.stagecmd.

Example

oss.defaults stage forcero
3.5 export

```
all.export  path [ options ]
```

_options_: [no]check [no]compchk [no]dread

{forcero | readonly | r/o | r/w | [not]writable}

{inplace | outplace} {local | global | globalro}

{[no]mig | [not]migratable} [no]mkeep


[no]ssdec [no]stage

**Function**

Specify processing options for any entry matching the specified path prefix.

**Parameters**

*path* is the path prefix to which the specified options apply. If no options are specified, the current defaults are used.

*options*

are the options to apply to any path whose prefix matches *path*. The options are identical to those allows with the *oss.defaults* directive. See that directive for an explanation of each option.

**Defaults**

All paths are processed according to the default options in effect at the time the path directive is encountered. Defaults can be set for all of the options (see the *defaults* directive).

**Notes**

1) Any number of *export* directives may be specified. They are cumulative and are checked in decreasing length order (i.e., most-specific to least specific).
2) Refer to the corresponding *defaults* directive for complete details on the option’s effect.
3) The *export* directive is used by the *xrootd* protocol to determine which paths are valid for incoming client requests.
4) The **export** directive is used by **oss** component to enforce desired processing attributes.

**Example**

```bash
oss.export /xrd/files/staged mig nodread rcreate
```
3.6 localroot

```plaintext
olb.localroot path
or equivalently
oss.localroot path
```

**Function**
Specifies where the local file system name space is actually rooted.

**Parameters**
`path` is the path to be pre-pended to any local path specified by a client request.

**Defaults**
None. Paths are used locally as specified.

**Notes**
1) The `localroot` parameter allows you to keep the external namespace consistent even when you move the associated file system from one location to another. Say, that a file system is mounted at `/xrd`. This means that all file paths start with `/xrd`. If now you needed to mount the file system at `/usr/xrd` then by specifying

```plaintext
olb.localroot /usr
or equivalently
oss.localroot /usr
```

the external view of the file system would remain the same since `oss` will automatically prefix all paths with `/usr` and use the new mount point.

2) The `olbd` honors the `localroot` directive (also present for the same purpose in the `oss` layer) even when prefixed by `oss`. This allows you to use a single configuration file.

3) The `localroot` option is confusing to system administrators, especially when it is used in combination with the `remoteroott` option. The use of this option is highly discouraged unless absolutely necessary.

**Example**
```plaintext
olb.localroot /usr
```
3.7 perf

```bash
tab
olb.perf parms

parms:  [ int time ] [ pgm prog ]
```

**Function**
Specify how load is computed and reported.

**Parameters**

**int time**
specifies the estimated time between load reports as computed by `prog`. The time may be suffixed by `s` (the default), `m`, or `h` to indicate seconds, minutes, and hours, respectively.

**pgm prog**
specifies the program that will be used to compute the machine load and write the information to standard out. The `pgm` parameter must be the last parameter on the line.

**Defaults**

`olb.perf int 3m`.

**Notes**

1) This directive is only used by server-mode `olbd`s.
2) There is no default value for the program and load information cannot be collected and reported unless a load collector exists. A sample program, `olb_MonPerf`, is supplied for this purpose. This program uses the `rperf` command, among others, to calculate the cpu, i/o, and various other load levels.
3) The specified program is started by the server-mode `olbd` at startup time. It is automatically restarted after two failures to report a load within the specified interval.
4) The specified program must write 5 white-space separated numbers to standard out. The last number must be terminated by a new-line character (`\n`). Each number must be normalized to 100, with 0 indicating no load and 100 indicating saturation. The numbers are, in order, 1) system load, 2) cpu utilization, 3) memory utilization, 4) paging load, and 5) network utilization.

**Example**

```bash
olb.perf int 5m pgm /usr/etc/ooss/olb_MonPerf 300
```
3.8  **prep**

```
olb.prep parms

parms:  [ echo ] [ reset cnt ] [ scrub time ]
        [ ifpgm ifprog ]
```

**Function**

Specify how offline file preparation is done.

**Parameters**

- **echo**  writes to the log all of the files found in the external in-preparation queue whenever a reset occurs.

- **reset cnt**
  specifies the maximum number of scrubs of the in-preparation queue that can be done before the contents of the queue are recomputed. The default is three (3).

- **scrub time**
  specifies the time between scrubs of the in-preparation queue. The time may be suffixed by s (the default), m, or h to indicate seconds, minutes, and hours, respectively. The default is 20 minutes.

- **ifpgm ifprog**
  specifies the interface program that must be called to add, remove, and list preparation queue files. The usage notes detailed the input, output, and calling conventions that *ifprog* must have. The **ifpgm** parameter must be the last parameter on the line. Any parameters after *ifprog* are passed to the program via the argument list. Quoted values must be avoided as they are not correctly passed.

**Defaults**

**None.** Preparation queue handling is normally disabled.
Notes

1) This directive is only used by server- and manager-mode olbd’s.
2) Each olbd that can stage files is also capable of preparing files to be online prior to their active use. This is done through the prepare protocol. The mechanism that is actually used to bring files to local disk is the responsibility of the external infrastructure. The prep directive describes the interface to that infrastructure.
3) The ifpog is used to add, remove, and list reparation queue files. It is started at initialization time and is expected to run continuously, and is automatically restarted should it fail. Parameters are sent via standard in, one request for each new line terminated record. Except for the “list” (i.e., ?) request, the program should not write any output to standard out. Output to standard error will be included in the olbd log file.
4) When the olbd needs to know the exact contents of the preparation queue (e.g., files waiting to be brought to local disk) it sends a single question. The ifpog must produce on standard out the list of absolute file names, each separated by a new-line character (\n’) that are being prepared.
5) When the olbd needs to add one or more files to the external preparation queue, it sends the following line via standard in:

```
+ requestid npath priority mode path [path [. . .]]
```

requestid is the request identifier that can be used to group this request into a unique set of requests. The requestid is globally unique.

npath is the notification path to be used to indicate how the request complete. This field may contain:

- no notification is to be sent.

mailto://user send e-mail to user
tcp://rhost:port/msg send msg via tcp to rhost:port
udp://rhost:port/msg send msg via udp to rhost:port

priority is the request’s priority as a number 0 through 9, inclusive. Zero is the lowest priority, while 9 the highest.

mode is the processing mode and may contain a combination of the following letters:

f send fail notice (not affected by q flag)
n send success notice
q suppress default failure notice (i.e., quiet)
r file is expected to be only read
w allow the file to be modified

path is the absolute path of the file to be prepared. If more than one path is specified, each path is separated by a blank.
6) When the **olbd** needs to remove files from the external preparation queue, it sends the following line via standard in:

```
- requestid
```

`requestid` is the request identifier associated with the request to be removed. All entries with this `requestid` should be removed.

7) The format for tcp and udp preparation notifications must adhere to the following format:

**Successful:**
```
  ready  requestid msg path
```

**Unsuccessful:**
```
  unprep requested msg path
```

`requestid` is the request identifier associated with the completed request.

`msg` is the text that followed the notification url (see above). This text is sent without inspection.

`path` is the absolute path of the file that successfully prepared or whose preparation failed.

8) When the **olbd** needs a list of all requests that are queued for action, it sends a single question mark (?). The program should respond with a list of `paths` that are queued for reading from the **MSS**.

9) Because file preparation is done on a best-effort philosophy, the preparation program is free to honor (or not) the requests in any way. Currently, the **olbd** does not check the return status of the program nor expects any error output (e.g., messages).

**Example**
```
  olb.prep batch 9 scrub 10m ifpgm /opt/xrd/bin/prep_mngr
```
3.9 sched

\[ \text{olb.sched parms} \]

\begin{verbatim}
parms:  [ cpu pctcpu ]  [ io pctio ]  [ mem pctmem ]
   [ pag pctpag ]  [ runq pctrunq ]  [ fuzz fnum ]
   [ maxload mload ]  [ refreset sec]
\end{verbatim}

Function
Specify the parameters for the load balancing scheduling algorithm.

Parameters
cpu pctcpu
specifies the percentage of cpu load to be used to compute the overall load of a server.

fuzz fnum
specifies the percentage difference two overall load values must have before they are considered different. A value of 100 suppresses the use of load in any scheduling decisions.

io pctio
specifies the percentage of io load to be used to compute the overall load of a server.

maxload mload
specifies the maximum overall load a server may have. Servers whose overall load is greater than \text{mload} are not scheduled.

mem pctmem
specifies the percentage of memory load to be used to compute the overall load of a server.

pag pctpag
specifies the percentage of paging load to be used to compute the overall load of a server.

refreset sec
specifies the number of seconds between server reference count resets.
**runq pctrunq**

specifies the percentage of runq load to be used to compute the overall load of a server.

**Defaults**

```
olb.sched cpu 0 io 0 mem 0 pag 0 runq 0 fuzz 20 refreset 3600
```

**Notes**

1) This directive is only used by manager-mode olbd’s.
2) The load-balancing algorithm chooses from all available servers the server whose computed overall load is smallest. When two servers have the same load, as determined by fuzz, the least selected server is chosen.
3) Other factors apply in selecting a server. For instance, whether or not the server has the requested file on disk, whether the server is allowed to dynamically stage a file, whether the server has sufficient disk space, etc.
4) The sum of pctcpu, pctoi, pctmem, pctpag, and pctrunq should be equal to 100.
5) If the sum of pctcpu, pctoi, pctmem, pctpag, and pctrunq is equal to zero, or if fuzz is 100, servers are selected strictly round-robin.
6) Round-robin scheduling is also forced when performance monitoring is disabled (see the ping usage directive).
7) Server selection, with or without load information, is accomplished by using an internal reference counter in order to equalize the selection process. Since this counter may drift due to external anomalies encountered during scheduling, it is periodically reset. The refreset parameter controls the minimal reset frequency. However, the counter is only reset if sufficient selection activity occurred.

**Example**

```
olb.sched cpu 50 io 50
```
3.10 space

olb.space [linger num] [recalc sec]

[[min] min[k | m | g] [hwm[k | m | g]]]

Function
Specify how servers are selected for file creation.

Parameters

linger num
specifies the number of times a server may be reselected without an intervening server being selected for allocation. The default is zero (0).

recalc sec
specifies the number of seconds between free space recalculations. The default is 15 seconds.

min
specifies the minimum amount of free space a server must have in order for it to be selected. You may suffix the byte quantity by k, m, or g to indicate kilobyte, megabytes, or gigabytes, respectively.

hwm
specifies the minimum amount of free space a server must have in order for it to be selected after free space has fallen below min. You may suffix the byte quantity by k, m, or g to indicate kilobyte, megabytes, or gigabytes, respectively.

Defaults
olb.space linger 0 recalc 15 min 10g 11g

Notes
1) This directive is only used by manager-and server mode olbd’s.
2) The space values are used during server selection when either a file is opened in create mode or when a file must be dynamically staged.

Example
olb.space min 2g 10g
4 Esoteric olb Configuration Directives

This section describes directives that are normally not specified. You may wish to review these directives to be familiar with additional configuration options that are available.

4.1 delay

```
olb.delay parms

parms:  [discard num] [drop sec] [full {sec | *}] [lookup sec] [overload {sec | *}] [peer sec]
[servers num[%]] [service sec] [startup sec] [suspend sec]
```

Function
Manage processing latency.

Parameters

discard num
is the maximum number of times a message can be forwarded before it gets discarded.

drop sec
the number of seconds a malfunctioning server is allowed to stay in the configuration before it gets dropped. The delay allows time for a server recover before clients are sent to other functioning servers. Clients are delayed during the recovery window.

full sec
is the number of seconds to delay a client when no eligible servers have sufficient space to place a file. By default, delays due to insufficient disk space are not allowed and when the condition occurs, the client is given an ENOSPC error condition. You may decide that this is a recoverable condition and are willing to let clients wait until disk space becomes available. Specifying an asterisk uses a dynamically computed optimal value (see the notes).

lookup sec
is the number of seconds to delay a client when trying to determine which servers have a wanted file on disk.
**overload** *sec*

is the number of seconds to delay a client when all available servers are overloaded. Specifying an asterisk uses a dynamically computed optimal value (see the notes).

**peer** *sec*

is the maximum number of seconds to delay a client when resources are not available in the immediate cluster in the presence of a peer manager. Should any individual wait time exceed *sec*, the client is redirected to an appropriate peer manager.

**servers** *num[\%]*

is the minimum number of servers that must be subscribed for load balancing to be effective. The number may be suffixed with a percent sign. When specified this way, the number of available servers must be no less that the specified percentage of the maximum number of servers ever subscribed to the **olbd** manager since startup. This option effectively determines the server quorum necessary for the **olbd** to redirect clients.

**service** *sec*

is the number of seconds to delay a client when fewer than *num* servers are subscribed.

**startup** *sec*

is the number of seconds to delay enabling manager service when initially started. This time period allows for servers to subscribe while client requests are delayed. Clients are delayed “**service**” seconds during this time.

**suspend** *sec*

is the number of seconds to delay a client when a selected server is in suspend state.

**Defaults**

```
olb.delay discard 7 drop 10m full 0 lookup 5 overload * peer 0
olb.delay servers 80% service 15 startup 90 suspend 30
```

**Notes**

1) This directive is only used by manager-mode **olbd**s.
2) Clients do not time out when a delay is imposed.
3) The **overload** delay is imposed when all eligible servers have a load greater than the one specified **maxload** on the **sched** directive.
4) The **full** and **load** options allow you to specify an asterisk to choose the optimal delay value. The optimal value is computed as 

$$\texttt{ping.ptime \ast ping.pcnt + 30}$$

The value is optimal because the load balancer will see no change in external conditions until this amount of time has gone by. See the **ping** directive for additional details.

5) **Warning**: The 80% default for **servers** works better as more servers join the configuration since more servers can fail before the system enters a holding pattern. For sites with less than 6 servers, you should specify a fixed number.

6) When the system enters a holding pattern, also known as safe-mode, clients are delayed until the conditions causing the situation are removed. For example, when the number of servers falls below the quorum established by the **servers** option, safe-mode is entered. The system remains in safe-mode until a quorum is re-established.

**Example**

```
olb.delay lookup 3 full *
```
4.2 fxhold

\texttt{olb.fxhold \textit{time}[m \mid h \mid s]}

**Function**

Set the time file existence information is to be cached in memory.

**Parameters**

\textit{time} is the number of seconds file existence information may be cached. The time may be suffixed by \texttt{m}, \texttt{h}, or \texttt{s} (the default) to indicate minutes, hours, or seconds, respectively.

**Defaults**

\texttt{olb.fxhold 8h}

**Notes**

1) This directive is only used by manager-mode \texttt{olbd}'s.
2) A manager \texttt{olbd} keeps track of where files are at each server-mode site. To prevent information from getting very stale, it is discarded after the time specified by the \texttt{fxhold} directive.
3) Setting the cache time too low will substantially increase overhead.

**Example**

\texttt{olb.fxhold 3h}
4.3 fsxeq

\[
\text{olb.fsxeq \{ func \} xpath}
\]

\[
\text{func: chmod mkdir mv rm rmdir}
\]

**Function**
Designate the program to handle file meta-data operations.

**Parameters**

- **func** one or more of the indicated functions (i.e., `chmod`, `mkdir`, `mv`, `rm`, and `rmdir`) that are to be mapped to `xpath`.

- **xpath** is the absolute path to an executable file. The file will be invoked whenever the `olbd` is asked to execute one of the functions. Parameters specified after `xpath` are passed to the program via the argument list. Quoted parameters should not be specified as these are not correctly passed.

**Defaults**

- **None.** The `olbd` will use the native operating system call to perform the functions.

**Notes**

1) This directive is only used by server-mode `olbd`'s.
2) Any number of `fsxeq` directives may be specified in order to map different programs to different functions.
3) The `fsxeq` directive is meant to be used in those situations where additional processing needs to occur when one of the indicated functions is executed (e.g., a file needs to be deleted from online disk as well as a Mass Storage System).
4) The `olbd` is asked to execute functions only if the `ofs.forward` directive has been specified for the redirecting file server (e.g., `xrootd`). Refer to the `ofs` configuration manual for more information.
5) Each function invokes `xpath` as follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>Command Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmod</td>
<td><code>xpath mode path</code></td>
</tr>
<tr>
<td>mkdir</td>
<td><code>xpath mode path</code></td>
</tr>
<tr>
<td>mv</td>
<td><code>xpath oldpath_newpath</code></td>
</tr>
<tr>
<td>rm</td>
<td><code>xpath path</code></td>
</tr>
<tr>
<td>rmdir</td>
<td><code>xpath path</code></td>
</tr>
</tbody>
</table>

6) The executable function must return a status code of zero upon success. Upon failure, the status code should map to the appropriate `<errno.h>` code that describes the failure.

Example

```
olb.fsxeq mv rm /usr/local/bin/fs_olbd -c /opt/fs/fs.cf
```
4.4 namelib

```
olb.namelib path [parms]
or equivalently
oss.namelib path [parms]
```

**Function**
Specify the location of the file name mapping layer.

**Parameters**
- `path` is the absolute path to the shared library that contains an implementation of the Name2Name interface that `olbd` is to use to make logical file names to physical name for file system specific operations (e.g., open, close, read, write, rename, etc).
- `parms` are optional parameters to be passed to the Name2Name object creation function.

**Defaults**
A built-in minimal implementation driven via the `localroot` and `remoteroot` directives is used.

**Notes**
1) The Name2Name interface is defined in XrdOucName2Name.hh include file. Refer to this file on how to create a custom file name mapping algorithm.
2) The Name2Name interface is also used by the oss component of xrootd.

**Example**
```
olb.namelib /opt/xrootd/lib/libN2N.so
```
4.5 nowait

| nowait
| olb.nowait |

**Function**
Specify that the olbd should not wait for the data server.

**Defaults**
None, you must specify the nowait directive or start the olbd with –i to not wait for a data server.

**Notes**
1) The nowait directive provides for a loose coupling between servers running on the same host. The olbd executes asynchronously from the host’s data server and can subscribe to a manager before the data server is available on the host.
2) Without nowait, a host is not available for selection until the host’s data server is ready.
3) Once the xrootd contacts the olbd, the host automatically becomes ineligible for selection whenever the data server becomes unready.
4) The nowait option is meant for to be used with data servers that are unable to communicate with the local olbd. You should not specify this option for the xrootd server.
5) **Warning**: the default olbd mode (i.e., wait for data server) must be used in conjunction with xrootd’s –t option; otherwise the host will never be selected by the manager olbd.
6) **Warning**: The nowait directive disables port remapping. With port remapping, a client is redirected to the port actually being used by the data server that is the target of the redirection. This allows arbitrary or hidden ports to be used, none of which need be the same. When port remapping is disabled, clients are always redirected to the port they initially used to contact the redirector.
7) The nowait directive is automatically implied if you start the olbd with the –i option.

**Example**

olb.nowait
4.6 pidpath

```bash
olb.pidpath path
```

**Function**
Specify the location of the pid file.

**Parameters**
`path` is the path to be used to create the file where the daemon’s process id and local prefix are stored.

**Defaults**
The process id file is written into `/tmp`.

**Notes**
1) The name of the pid file is determined by the `olbd`’s role and the `-n` option or `name` directive.
2) If the `olbd` cannot create the pid file because either one already exists but is not owned by the `olbd`, or the directory permissions prohibit the `olbd` from creating new file; initialization fails and the `olbd` exits.

**Example**
```
olb.pidpath /var/run/olbd
```
4.7 ping

```
olb.ping ptime [ log ucnt ] [ usage pcnt ]
```

Function
Control the keep-alive and load reporting frequency.

Parameters

- **ptime** specifies the time between keep-alive requests sent to each server `olbd`. The time may be suffixed by `s` (the default), `m`, or `h` to indicate seconds, minutes, and hours, respectively.

- **log ucnt** specifies the number of usage requests that must be made before the reported usage is logged. A value of 0 suppresses any logging of usage information.

- **usage pcnt** specifies the number of pings that must occur before usage is requested from a server `olbd`. A value of 0 suppresses usage requests.

Defaults

```
olb.ping 60 log 10 usage 10
```

Notes

1) This directive is only used by manager-mode `olbd`'s.
2) Unspecified values in subsequent ping directives default to the last known value.
3) Smaller `ptime` values will discover a failing `olbd` is a smaller time window at increasing overhead.
4) Smaller `pcnt` values will ask for usage information averaged across a smaller time-window.
5) Usage information will be requested every `pcnt*ptime` seconds, assuming `ptime` is in seconds. Select a `pcnt/ptime` value that averages usage across a reasonable time window for your load (e.g., 5 to 10 minutes).
6) Usage information for each `olbd` server will be logged every `ucnt*pcnt*ptime` seconds, assuming `ptime` is in seconds. Choose any value appropriate to your logging needs. For instance, 1 logs usage every time it is requested while 0, the default, does not log usage.
7) When \textit{pcnt} or \textit{ptime} is set to zero, usage based load balancing is disabled. This means that requests are scheduled round-robin.

8) In the subsequent example, keep-alive pings occur every 30 seconds. Usage is requested every five minutes and never logged.

Example

\begin{verbatim}
olb.ping 30 log 0 usage 10
\end{verbatim}
4.8 remoteroot

\begin{verbatim}
olb.remoteroot path
or equivalently
oss.remoteroot path
\end{verbatim}

Function
Specifies where the local file system name space is actually rooted in the remote Mass Storage System.

Parameters
\textit{path} is the path to be pre-pended to any path sent to the Mass Storage System for processing.

Defaults
None. Paths are sent to the Mass Storage System as specified.

Notes
1) The \texttt{remoteroot} parameter allows you to place the online file namespace in a different location within the Mass Storage System. Say, that the online file system is mounted at \texttt{/xrd}. This means that all file paths start \texttt{/xrd}. If you specified
\begin{verbatim}
olb.remoteroot /usr
or equivalently
oss.remoteroot /usr
\end{verbatim}
then the file namespace would be rooted at \texttt{/usr/xrd} within the Mass Storage system because all paths would be prefixed by \texttt{/usr} before being sent to the Mass Storage System for processing.

2) The \texttt{remotelroot} option is confusing to system administrators, especially when it is used in combination with the \texttt{localroot} option. The use of this option is highly discouraged unless absolutely necessary.

Example
\begin{verbatim}
olb.remoteroot /usr
\end{verbatim}
4.9 **trace**

```bash
olb.trace [-]toption [ [-]toption ] [• • •]
```

`toption: all | debug | defer | stage`

**Function**

Enable tracing at the **olb** level.

**Parameters**

`toption`

specifies the tracing level. One or more options may be specified. The specifications are cumulative and processed left to right. Each option may be optionally prefixed by a minus sign to turn off the setting. Valid options are:

- `all`: selects all possible trace levels except `debug`
- `debug`: traces internal functions
- `defer`: traces imposed wait responses
- `stage`: traces binding of locate requests to servers to have promised to stage in files.

**Defaults**

`olb.trace -all`.

**Notes**

3) The **olbd** `-d` command line option is equivalent to **olb.trace all**.

**Example**

`olb.trace debug`
4.10 xmilib

```
olb.xmilib path [parms]
```

**Function**
Specify the location of the external manager interface layer.

**Parameters**

*path* is the absolute path to the shared library that contains an implementation of the External Manager Interface that *olbd* is to use to locate filee and perform meta-data operations on file names (e.g., mv, rm, etc.).

*parms* are optional parameters to be passed to the XMI object creation function.

**Defaults**
A built-in implementation based on distributed co-operating *olbd*s is used.

**Notes**
1) The XMI interface is defined in XrdOlbXmi.hh include file. Refer to this file on how to create a custom external manager.

**Example**
```
olb.xmilib /opt/xrootd/lib/libXMIDPM.so
```
5 Deprecated olb Configuration Directives

This section describes directives that are maintained only for backward compatibility purposes. The directives should not be specified because support will be terminated in the near future.

5.1 path

```
olb.path { r | w }[ s ] path
```

**Function**
Designate paths handled by a server.

**Parameters**
- **r**: the path can be handled only if it is opened in read-mode.
- **w**: the path can be handled regardless of open mode.
- **s**: files can be dynamically staged on this path.

**path** is the path that can be handled by the server. Only open requests on this path can be sent to the server.

**Defaults**
```
olb.path r /
```

**Notes**
1) This directive is only used by server-mode olbd’s.
2) Use the path directive to parcel out certain parts of the name space to discreet servers for load balancing purposes.
3) Client will be sent to servers that allow dynamic staging on a path only as a last resort. That is, clients are preferentially sent to servers that already have a requested file on disk. Only when all such servers are overloaded is the client sent to a server that must stage the file to disk.
4) If you are using local path prefixing (i.e., olb.localroot or oss.localroot directive); do not include the local prefix with the path directive.
5) Use the cache directive to specify which file systems can be used to create files.

**Example**
```
olb.path r /xrootd/files
```
5.2 port

Function
Designate the port numbers to use for incoming requests.

Parameters

tcpnum
specifies the TCP port number that the server olbd should use for incoming requests.

Defaults
There is no defaults for olbd’s started in pure manager mode; the directive must be specified.

Notes
1) The port directive is mandatory for olbd’s given a manager role (i.e., configured with an all.role of manager, peer manager, or peer proxy manager).
2) The port directive ignored by olbd’s given server or supervisor roles.
3) Servers and supervisors always choose an arbitrary port number. Since they ignore the port directive, you can still have a common configuration file for all olbd modes.
4) The port directive also establishes the port number for the manager.
5) The port number being used by an olbd is reported in the log.

Example
olb.port 3121
5.3 subscribe

```
olb.subscribe host[+] { port } [ if conds ]
```

**Function**
Specify the manager `olbd` location to a server or supervisor `olbd`.

**Parameters**
- `host` is the DNS name or IP address of the `olbd` manager. If host ends with a plus sign (+), then the `olbd` subscribes to all hosts addresses associated with host.
- `port` is the TCP port number at which the manager will accept connections. The default comes from the previous `port` directive.
- `conds` are the conditions that must exist for this directive to apply. Refer to the description of the `if` directive on how to specify `conds`.

**Defaults**
None, you must specify the subscribe directive in order to allow a server to be load-balanced.

**Notes**
1) This directive is only used by server and supervisor mode `olbd`’s. It has been superseded by the `all.role` directive. Use this directive only under special circumstances to over-ride the default behavior of the `all.role` directive.
2) Generally, a manager is the top-most server providing the initial point of contact for service.
3) Servers and supervisors may subscribe to any number of different managers.
4) If the manager host name ends with a plus, then all the IP addresses associated with host are treated as a manager and the `olbd` subscribes to each one. This allows you to easily construct fault-tolerant configurations using the IP address alias feature in DNS.
5) The `port` directive must be specified prior to any `subscribe` directives if the port is not specified.

**Example**
```
olb.subscribe objymanager.slac.Stanford.edu+
```
6  Esoteric odc Configuration Directives

The odc directives listed here are intended for the xrootd process with an enabled odc component. Therefore, these directives should be placed together with the ofs and oss directives in the same configuration file. It is rare that any odc directives are specified. You may wish to review these directives to be familiar with additional configuration options that are available.

6.1  conwait

```
  odc.conwait  sec
```

**Function**
Set the number of second to delay a client in the absence of a manager olbd.

**Parameters**

- `sec` is the number of seconds that a client is delayed when there is no connection to a manager olbd.

**Defaults**

```
  odc.conwait 10
```

**Notes**

1) When a client attempts to locate a file and no connection exists to a manager olbd process, xrootd defers the client for conwait seconds. After the time period expires, the client automatically retries the request.

2) The time period chosen for conwait should be sufficiently long to establish a connection to an olbd.

**Example**

```
  odc.conwait 6
```
6.2 msgkeep

```
    odc.msgkeep  num
```

**Function**
Set the number of message blocks to keep for re-use.

**Parameters**

```
num   the number of message blocks to keep for re-use.
```

**Defaults**

```
    odc.msgkeep  32
```

**Notes**

1) Message block are used when communicating with the olbd. The
number chosen should be the close to the peek number of outstanding
messages that are expected. Generally, the default is sufficient.

**Example**

```
    odc.msgkeep  44
```
6.3 request

```
odc.request [delay secd] [noresp num] [repwait secr]
```

Function
Specify request timing parameters.

Parameters
- **secd**: the number of seconds to delay a client when the **olbd** has not responded in **secr** seconds to a request to locate the file the client wishes to access.
- **num**: is the number of consecutive **secr olbd** response timeouts that may be tolerated before the **odc** attempts to find another working **olbd** manager.
- **secr**: the number of seconds to wait for an **olbd** response.

Defaults
```
odc.request delay 5 noresp 4 repwait 3
```

Notes
1) When a client attempts to locate a file a request is sent to the **olbd** to locate the best possible copy of the file. Should the **olbd** not respond in **secr** seconds, **xrootd** defers the client for **secd** seconds. After the time period expires, the client automatically retries the request.

Example
```
odc.request delay 3 repwait 1
```
6.4 trace

\texttt{odc.trace [-]toption [ [-]toption ] [• • •]}

\textit{toption}: \texttt{all | debug | forward | redirect}

\textbf{Function}
Enable tracing at the \texttt{odc} level.

\textbf{Parameters}
\textit{toption}

specifies the tracing level. One or more options may be specified. The specifications are cumulative and processed left to right. Each option may be optionally prefixed by a minus sign to turn off the setting. Valid options are:

\begin{itemize}
  \item \texttt{all} selects all possible trace levels except \texttt{debug}
  \item \texttt{debug} traces internal functions
  \item \texttt{forward} traces forwarded functions
  \item \texttt{redirect} traces request redirection
\end{itemize}

\textbf{Defaults}
\texttt{odc.trace -all}.

\textbf{Notes}
1) Trace output is currently routed to standard error.

\textbf{Example}
\texttt{odc.trace redirect}
7  Deprecated odc Configuration Directives

This section describes directives that are maintained only for backward compatibility purposes. The directives should not be specified because support will be terminated in the near future.

7.1  olbapath

```
odc.olbapath  path [ group ]
or equivalently
olb.adminpath  path [ group ]
```

Function
Designate the path used for olbd administrative communications.

Parameters
path is the path used for Unix named sockets. Administrative commands can be directed to these sockets.

group allows group access to path.

Defaults
The default administrative path is “/tmp/.olb”, as modified by the –n command line option.

Notes
1) This directive is only used by the odc component of xrootd to communicate with the local olbd.
2) The specified path must correspond to the one specified with the old.adminpath directive.
3) In order to simplify the configuration, the odc uses the olb.adminpath specification if it does not find the odc.olbapath directive. If you are using a single configuration file for the olbd and xrootd then only specify the olbd.adminpath directive.

Example
```
olb.olbapath  /var/run/olbd
```
8 Document Change History

28 March 2005
- Remove documentation on local redirection mode.
- Remove documentation on portbal and portsel directives.
- Document olbd supervisor mode (i.e., -m and -s are specified)
- Document the odc.olbapath directive.
- Simplify the thread directive.

29 March 2005
- Better explain “-port any” and “xrd.port any” usage.
- Remove “-a” explanation.

26 April 2005
- Deprecate the –m and –s command line options
- Document the role directive.
- Document how path directive locations are used when the cache directive is not specified.

10 May 2005
- Add the -n option to specify the instance name.
- Explain the pidpath directive.
- Describe temporary files that are created.

1 June 2005
- Add the generalized if facility explanation.
- The if is also accepted on olb.subscribe and odc.manager.
- Describe if selection by instance name.
- Explain how to run multiple olbd’s on the same host.

12 Jan 2006
- Add the namelib directive.

22 March 2006
- Add exec condition to if/else/fi.

12 Jun 2006
- Add the xmilib directive.
- Documented simplified cluster configuration involving role and manager directives.
10 Jan 2007
- Expand the role and manager directives by adding peer and proxy roles and subcomponents.
- Deprecate the subscribe and threads directives.

2 Apr 2007
- Deprecate the olb.path, olb.port, and odc.olbapath directives.
- Move explanation of conditionals to a separate manual.
- Introduce the oss.defaults and all.export directives.
- General cleanup of various sections.