Tuning for Very Large Archive Searches

From 30 Seconds to Three Hundredth of a Second in Ten Minutes

19 June 2018

Overview

The LISTSERV HPO search engine has been enhanced in version 16.5 to better handle lists with very large archives, and in particular searches involving words that occur very frequently in the archives. Although baseline performance has been improved across the board, manual tuning is required to take full advantage of this new feature. This should only takes about ten minutes, and can lead to performance improvements as high as a factor of 1,000 – as observed on a large "real world" list discussing a religious topic, where words that occur in almost every posting are frequently used in searches.

If you are in a hurry, jump to *"Updating LISTSERV's Configuration"* and try the suggested settings; they will work in most cases, provided you have sufficient hardware resources.

Pre-Requisites

Before you begin, you will need the following:

- A 64-bit operating system with the necessary system and process quotas to enable the main LISTSERV process to open at least 128 files[1] and allocate the amount of virtual storage that you plan to allocate to search acceleration.
- A 64-bit LISTSERV HPO build, either version 16.5 or an Early Release build dated 23 July 2012 or later.

If necessary, you can also deploy these optimizations on a 32-bit operating system, although it would be prudent not to exceed a 1GB cache size. The 32-bit version of LISTSERV 16.5 does include the new search engine.

Choosing a Data Cache Size

The first step is to decide how much memory you are comfortable setting aside for search acceleration. One method is to look at how much memory lies unused in your server – in most cases, there will be over 1GB of unused RAM, and this could be a good starting point. Another method is to look at the size

[1] Depending on the operating system, SMTP worker resources may be charged to the LISTSERV process, or may be counted separately. The numbers given are for the LISTSERV process alone.

of your archives, including indexes. If searches occur very frequently, you should aim to allocate a big enough cache to allow the entire archives to remain in memory, adding RAM to the server as necessary. If searches only occur from time to time, a 50% ratio could be sufficient. Most sites have one or two very large public lists, which should be 100% cached if possible, while other lists are only searched from time to time by local staff and need not be considered.

An "overkill allocation" would be counter-productive as it would cause LISTSERV to cache every file ever accessed since it was started. There is a certain overhead for managing the cache, so if you only have 2GB of active list archives, you should not allocate 20GB just because there happens to be that much unused memory in the server. It is better to let the web interface processes and the operating system have that memory.

You can observe data cache usage with the `SHOW FIOCache` command.

If in doubt, just start with 1GB – it is sufficient in most cases.

Choosing an Index Cache Size

The search engine maintains a separate cache for database indexes. Because every search accesses the index, but only a subset of the data files, it is more important for the index than for the data files to remain cached. Index sizes vary widely based on the keyword density of your particular lists, but a rule of thumb is to set the index cache size to 25% of the data cache size.

An "overkill allocation" is acceptable as LISTSERV will not use index cache memory unless it needs it. Cache management overhead is minimal, so feel free to go for a large value if you have spare memory.

If in doubt, start with 256MB.

Choosing a Concurrent Open File Limit

One of the biggest challenges of the search engine is that list archives are typically spread over hundreds of separate monthly or weekly archive files. Searches on a particular topic could retrieve postings spanning years or even decades, requiring LISTSERV to open dozens of disk files to retrieve context information, or perform resource-intensive verifications such as the NEAR operator and certain forms of complex searches. The HPO search engine addresses this by maintaining a number of concurrently open files, which it closes based on usage statistics when a new file needs to be opened. This limit is necessary to avoid exceeding process quotas and, perhaps more importantly, because the search engine reads archive files sequentially from start to end in order to take advantage of the comparatively high sequential transfer rate of mechanical drives, as compared to their access time (seek time + rotational latency). Thus, every open archive file is not just one more file descriptor for the operating system, but the entire file contents pinned into the data cache.

In 16.0, there was no way to change the maximum number of concurrently open files, which was set to a conservative value of 12 in order to avoid allocating too much memory. This is insufficient for lists with very large archives, and results in counter-productive open/close/re-open sequences.

This number can now be increased to 100, which should be sufficient and could already pin over 1GB of data in the cache for the duration of the search. Since the HPO search engine will normally not return more than 100 matches, the maximum value of 100 is always sufficient for simple searches, such as "contains all of these words." In an environment with intensive use of complex searches, a higher setting could theoretically be helpful, but the tests we have conducted with 200 showed only diminutive returns.

If in doubt, start with 50, or just go for 100 if you have at least 1GB of spare RAM.

Updating LISTSERV's Configuration

The last thing to do is to update LISTSERV's configuration. In this example, we will allocate 1,000,000 kilobytes (approximately 1GB) to the Data Cache; 250,000 kilobytes to the index cache; and 50 concurrent open files.

```
FIOC_TARGET=1000000
FIOC_TRIM=1050000
FIOC_WARNING=1100000
DBRINDEX_CACHE=250000
MAX_OPEN_ARCHIVE_FILES=50
```

Please refer to the product documentation for an in-depth description of the various keywords.

The first search after restarting LISTSERV will still have to load up the index file as well as all the relevant data files. Eventually, all the data files will have made their way into the data cache, and, assuming a sufficient value for MAX_OPEN_ARCHIVE_FILES, you should see a massive performance improvement.

Known issues and restrictions

The following known issues and restrictions exist:

- **This document is for 16.5 only.** While some of the settings mentioned in this document exist in earlier versions, they do not have the same impact as with 16.5, and may be limited in other ways. For instance, the Data Cache is limited to 2GB in earlier versions.
- **This document is for LISTSERV HPO only.** While some of the settings mentioned in this document exist in LISTSERV Classic, they do not have the same impact at all because the Classic search engine does not reference them. Of the settings mentioned in this document, the only one that is effective with LISTSERV Classic is DBRINDEX_CACHE.