VMM Tuning Tip: Protecting Computational Memory

By Barry J. Saad pSeries Advanced Technical Support

This VMM tuning tip is applicable pSeries servers running AIX 5.2 ML4+ or AIX 5.3 ML1+.

The goal of the new tuning approach is to prevent or protect computational memory (i.e. process memory (data, stack, and heap), kernel memory and shared memory (e.g., Oracle's System Global Area (SGA)) from being paged-out to paging space. Under the assumption that once paged-out, at some point in the future, the data will have to be paged-in from paging space, which would negatively impact system performance. Protecting computational memory is particularly important for applications that maintain their own data cache (e.g., DB2 and Oracle). The objective, protecting computational memory, is achieved by setting the VMM parameters as follows:

- ✓ `maxperm%=maxclient%=(A high value)`

  A high value prevents the lrud[1] from running unnecessarily, and if possible the value should be greater than numclient% (reported by `vmstat -v`). A typical setting is 90%.

- ✓ `minperm%=(A low value)`

  A low value ensures that the setting for `lru_file_repage` isn't overridden, and should be less than numperm % (report by vmstat –v). Typical settings, based on total system memory, are:

  - 32G of memory or less minperm%=5%.
  - Greater than 32G and less than 64G of memory minperm%=10%.
  - Greater than 64G of memory minperm%=20%.

- ✓ `strict_maxperm=0` (default)
- ✓ `strict_maxclient=1` (default)
- ✓ `lru_file_repage=0`

To understand why this combination of tuning parameters works you need to understand how the `lru_file_repage` parameter influences the VMM page stealing algorithm.

[1] lrud is the kernel process that is responsible for stealing memory when required.

Background: The VMM classifies memory into one of two buckets – either computational or non-computational. Computational memory includes working storage segments and application text segments[2]. Non-Computational, also referred to as File System Cache, includes file system data from JFS, JFS2, NFS or any filesystem type. The size of the file system cache is tracked by the kernel in a parameter called numperm, and the size of the client segment usage is tracked by in a kernel parameter called numclient. Now, the kernel does not track the size of JFS pages in memory and strictly speaking numclient is not a sub-set of numperm; however, for most purposes thinking of numclient as a sub-set of numperm will not cause any conceptual issues.

The process starts when the VMM needs memory because the number of free frames drops below `minfree` or a defined trigger point is reached (e.g., the number of client pages exceeds `maxclient%` and strict_maxclient=1). The lrud will make a determination to steal either memory type or limit the search to only file cache memory. This determination is made based on a number of parameters, but the key parameter is `lru_file_repage`. When `lru_file_repage` is set to 1, which is the default, the VMM will use the computational and non-computational re-page counts, in addition to other parameters, to determine whether to steal either memory type or just file memory. When the `lru_file_repage` is set to 0, the VMM will attempt to steal only file memory provided (1) numperm is greater than minperm and (2) the VMM is able to steal enough memory to satisfy demand. It's really that simple, setting `lru_file_repage=0` is a very strong hint to the VMM to steal file memory period.

[2] Reference the 'svmon' man page for more information about memory segment types.

Note: **What is a re-page?** A page fault is considered to be either a new page fault or a re-page fault. A re-page fault occurs when a page that is known to have been referenced recently is referenced again, and is not found in memory because the page has been replaced. In a sense, a re-page can be viewed as a failure in the page selection algorithm – in an ideal world you would not have any re-page faults. To classify a page fault as new or re-page, the VMM maintains a re-page history buffer and maintains two counters that estimate computational-memory repaging and file-memory repaging. The re-paging rates are multiplied by 90% each time the page-replacement algorithm runs, so that they reflect recent re-paging activity more strongly than historical repaging activity.

Until the VMM reaches the point when memory is required the system will fill memory with either file memory or computational memory without restriction – assuming with strict maxclient set to a hard limit the numclient% is less than `maxclient%`. Using this approach the system will not expend CPU resources unnecessarily. It is normal and expected behavior for the number of free frames to hover between the `minfree` and `maxfree` values. Remember the VMM is much like a person – let's put to off until tomorrow what doesn't have to be done today.

As a side note: The "legacy" tuning approach, shown below, was in wide spread use prior to the introduction of the `lru_file_repage` parameter. While this method achieves the same objective (i.e., protecting computational memory) it is no longer the preferred tuning method.

- ✓ `maxperm%=maxclient%=(typically a low percentage – 20 or 30)`
- ✓ `minperm%=5`
- ✓ `strict_maxperm=0` (default)
- ✓ `strict_maxclient=0`

It is possible, although not recommended, to use the "legacy" approach on systems with the `lru_file_repage` parameter. In that case you need to ensure that `lru_file_repage` is set to the default value of 1. The "new" recommendations are based on VMM developer experience and represent the best approach to tuning the system when the objective is to protect computational memory.

