Managing Memory

When a PBS job is running, the Linux operating system uses part of the physical memory of the compute nodes to store data that is either read from disk or written to disk. The memory used in this way is called *page cache*. In some cases, the amount or distribution of the physical memory used by page cache can affect job performance.

For more information, see the Wikipedia entry on Page cache.

Checking Page Cache Usage

To find out how much physical memory is used by page cache in a node your PBS job is running on, connect to the node via SSH and then run one of the following commands:

- cat /proc/meminfo
- top -b -n1 | head

Examples

The following example shows output of the cat /proc/meminfo command on a node of a running job. The output reports that 60,023,544 KB (~60 GB) of the node's memory is used by page cache:

```
% cat /proc/meminfo
MemTotal:      132007564 kB
MemFree:         605680 kB
Buffers:              0 kB
Cached:        60023544 kB
SwapCached:           0 kB
```

The next example shows output from the top -b -n1 | head command, run on the same node as in the previous example. The output reports that the node has 128,913 MB total memory, that 128,314 MB of the total memory is used, and that 59,220 MB (~59 GB) of the used memory is occupied by page cache:

```
% top -b -n1 | head
top - 15:04:00 up 3 days, 16:22,  2 users,  load average: 1.29, 1.23, 1.38
Tasks: 546 total,   2 running, 543 sleeping,   0 stopped,   1 zombie
Cpu(s):  0.5%us,  2.7%sy,  0.0%ni, 96.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   128913M total,   128314M used,      598M free,        0M buffers
Swap:        0M total,        0M used,        0M free,    59220M cached

PID    USER  PR  NI  VIRT  RES  SHR S   %CPU %MEM    TIME+ COMMAND
```

Managing Page Cache

The amount of physical memory used as page cache can be significant if your code reads a large amount of data from disk or writes a large amount of data to disk. For this reason, you may want to manage page cache using one of the methods described in this section.

Assigning Rank 0 to a Single Node

In the case of an MPI application where I/O is done solely by the rank 0 process, it is possible for the page cache to fill up the physical memory of a node, leaving no memory for the other MPI ranks. To avoid this situation, you can assign the rank 0 process to a node by itself. For example, the following PBS resource request allows for 49 MPI ranks with rank 0 to run on one node, and assigns 28 MPI ranks to each of the other two nodes:

```
#PBS -l select=1:mpiprocs=1:model=bro_ele+2:mpiprocs=28:model=bro_ele
```

Limiting Page Cache Usage

If there is enough physical memory on the head node to accommodate both the page cache and the memory used by other ranks, keep in mind that the total amount of memory in a node is split between two sockets. For example, the 128 GB of memory in an Electra Broadwell node is evenly divided between the node's two sockets, as shown in the Broadwell configuration diagram. Therefore, if other MPI processes are running on the same socket as the rank 0 process, you should check whether the amount of memory on one socket is enough for both the rank 0 process and the other ranks. If there is not enough memory, the other ranks may be forced to allocate and use memory on the other socket, causing a non-local memory access that may result in degraded performance.

In cases like this, you can set a limit for page cache usage to ensure there will be enough memory to accommodate both the I/O from the rank 0 process and the amount of memory used by other ranks on the same socket. You can accomplish this by running pcachem, a page cache management tool. For example, to set the limit at 8 GB (2^{33} bytes = 8589934592 bytes) run:

```
module load pagecache-management/0.5
setenv PAGECACHE_MAX_BYTES   8589934592
```

```
mpiexec -n XX pcachem -- a.out
```

where *XX* represents the total number of MPI processes. The exact value that you specify for PAGECACHE_MAX_BYTES is not important, however, be sure you specify a value that leaves enough physical memory on the socket (after subtracting the amount used by page cache) to accommodate the other MPI ranks on the socket.

Note: You must also load an MPI module for running MPI applications.

If you are using OVERFLOW, consider the option shown in the following example, which demonstrates the use of both page cache management (with pcachem) and process binding (with mbind.x) for better performance:

```
module load pagecache-management/0.5
setenv PAGECACHE_MAX_BYTES   8589934592
setenv MBIND /u/scicon/tools/bin/mbind.x

$HOME/bin/overrunmpi -v -n XX -aux "pcachem -- $MBIND -v -gm -cs" YY
```

where *YY* refers to the input file for OVERFLOW. The mbind.x option -gm prints memory usage for each process, and the -cs option distributes the MPI processes among the hardware cores.

For more information about mbind.x, see Using the mbind Tool for Pinning.

You can use the gm.x tool to check the memory usage of jobs running on NAS x86_64 or aarch64 systems. This tool reports the memory usage from each process at the end of a run. Optionally, gm.x can also report memory usage at a fixed interval during the run.

Note: The NAS mbind tool provides the options -gm, -gmc, -gmr for reporting memory usage information similar to gm.x.

The gm.x and mbind.x tools are available in the /u/scicon/tools/bin directory fo the NAS x86_64 systems and in the /u/.scicon/tools/arm/bin directory for the NAS aarch64 system.

On either the NAS x86_64 or aarch systems, loading the module scicon/cli_tools provides acess to gm.x, mbind.x, and a few other tools for the appropriate architecture.

Use the -h option to find out what types of memory usage can be reported:

```
% gm.x -h
Usage: gm.x [-options] [-h|-he] a.out [args]

Options -
  -hwm     ; high water mark (VmHWM)
  -rss     ; resident memory size (VmRSS)
  -wrss    ; weighted memory size (WRSS)
  -c<s:n>  ; print memory usage every <s> seconds for <n> times
  -r<list> ; print memory usage for selected ranks in the list
  -g<n:m>  ; monitor GPU usage (mem, power, energy)
  -time    ; print timing stats (wtime, utime, stime)
  -memv    ; use variable memory unit
  -v       ; verbose flag

Default: type is set by environment variable GM_TYPE (def=VmHWM)
       memory usage is printed at the end of the run by all ranks
```

Additional information about each option can be found with gm.x -he. Here are brief descriptions for some of the options:

-hwm (default)
: Reports the peak resident set size (high water mark) that the kernel captures for the process. Specifically, the status is taken from the value shown in the VmHWM field of the /proc//status file.

-wrss
: Calls the system function get_weighted_memory_size. (See **man get_weighted_memory_size** for more information about this function.)

-c
: Reports the memory usage during the run; useful in the event that a job fails to complete.

-memv
: Provided for backward compatibility; allows memory usage to be reported in a variable unit. Without this option, memory usage is reported in MB.

The gm.x tool can be used for either OpenMP or MPI applications (linked with HPE's MPT library). You do not need to recompile your application to use it.

To use gm.x for an MPI code, add gm.x just before the executable in the command line. For example:

```
mpiexec -np 4 gm.x ./a.out
Memory usage for (r1i1n0,pid=9767): 1.458 MB (rank=0)
Memory usage for (r1i1n0,pid=9768): 1.413 MB (rank=1)
Memory usage for (r1i1n0,pid=9770): 1.413 MB (rank=3)
Memory usage for (r1i1n0,pid=9769): 1.417 MB (rank=2)
```

You can feed the standard output into the gm_post.x script to compute the total memory used and the average memory used per process. For example:

```
mpiexec -np 4 gm.x ./a.out | gm_post.x -v
Memory usage for (r1i1n0,pid=9767): 1.458 MB (rank=0)
Memory usage for (r1i1n0,pid=9768): 1.413 MB (rank=1)
Memory usage for (r1i1n0,pid=9770): 1.413 MB (rank=3)
Memory usage for (r1i1n0,pid=9769): 1.417 MB (rank=2)

Number of nodes     = 1
Number of processes = 4
Processes per node  = 4
Total memory        = 5.701 MB

Memory per node     = 5.701 MB
Minimum node memory = 5.701 MB
Maximum node memory = 5.701 MB

Memory per process  = 1.425 MB
Minimum proc memory = 1.413 MB
Maximum proc memory = 1.458 MB
```

If you use dplace to pin the process, add gm.x after dplace in the command line:

```
mpiexec -np NN dplace -s1 gm.x ./a.out
```

The gm.x tool was developed by NAS staff member Henry Jin.

You can use the qsh.pl tool to check the memory usage of jobs running on Aitken or Electra. This tool is a Perl script that uses Secure Shell (SSH) protocol to access all of the nodes used by a PBS job in order to run a command that you supply.

For Aitken and Electra jobs, you can run qsh.pl on the PFEs.

The qsh.pl tool is available in the /u/scicon/tools/bin directory.

TIP: You can either run module load scicon/cli/tools or add the directory to your shell startup script so that you can invoke qsh.pl without typing the full path each time. For example:
set path = ($path /u/scicon/tools/bin)

Syntax:

pfe21% qsh.pl *job_id command*

You can use this script to check the amount of free memory in the nodes of your PBS job.

Example:

```
pfe21% qsh.pl 30329 "cat /proc/meminfo"

running "cat /proc/meminfo" on:  r56i2n14 r56i2n15
r56i2n14 :
  MemTotal:      8079728 kB
  MemFree:        857936 kB
  Buffers:             0 kB
  Cached:        3775472 kB
...
r56i2n15 :
  MemTotal:      8079728 kB
  MemFree:       5840920 kB
  Buffers:             0 kB
  Cached:         784280 kB
...
```

The qsh.pl script was provided by NAS staff member Bob Hood.

You can use the qtop.pl tool to check the memory usage of jobs running on Aitken or Electra. This tool is a Perl script that uses Secure Shell (SSH) to access the nodes of a PBS job in order to perform the top command. The output provides memory usage information for the whole node and for each process.

For Aitken and Electra jobs, you can run qtop.pl on the PFEs.

The qtop.pl tool is available after loading the module scicon/cli_tools or by adding to your path in the /u/scicon/tools/bin directory.

For example:
```
set path = ( $path /u/scicon/tools/bin )
```

Syntax:

pfe21% qtop.pl [-b] [-p n] [-P s] [-h n] [-H s] [-t s] [-N s] *job_id*

-b
: (for running in background or batch) don't run 'resize' command

-p n
: show at most n processes per host

-P s
: show only procs in s, a comma-separated list of ranges
e.g. -P 1,8-9

-h
: don't show the column header line

-H s
: show only header lines in s, comma-separated ranges
e.g. -H 1-2,7
e.g. -H 0 (don't show any lines)

-t s
: pass string s (must be one argument) to top command

-n s
: show output only from nodes in s, comma-separated ranges
e.g. -n 0,2-3 (relative node numbers)

-N s
: show output only from nodes in s, a comma-separated list
e.g. -N r1i1n14,r1i1n15 (absolute node numbers)

Example: To skip the header and list 8 procs per host:

```
pfe21% qtop.pl -H 0 -p 8 996093
all nodes in job 996093:  r184i2n12
r184i2n12   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
        20027 zsmith    25   0 23.8g 148m 5320 R  101  0.6   5172:37 a.out
        20028 zsmith    25   0 23.8g 140m 5140 R  101  0.6   5173:35 a.out
        20029 zsmith    25   0 23.9g 286m 6640 R  101  1.2   5172:23 a.out
        20030 zsmith    25   0 23.9g 245m 5040 R  101  1.0   5171:18 a.out
        20031 zsmith    25   0 23.9g 265m 6040 R  101  1.1   5171:46 a.out
        20032 zsmith    25   0 23.9g 246m 5300 R  101  1.0   5171:00 a.out
        20033 zsmith    25   0 23.8g 158m 5476 R  101  0.7   5172:41 a.out
        20034 zsmith    25   0 23.8g 148m 5280 R  101  0.6   5173:02 a.out
```

The qtop.pl script was provided by NAS staff member Bob Hood.

You can use the qps tool to check the memory usage of jobs running on Aitken or Electra. This tool is a Perl script that uses Secure Shell (SSH) protocol to access each node of a running job in order to get process status (ps) information on each node.

For Aitken and Electra jobs, you can run qps on the PFEs.

The qps tool is available in the /u/scicon/tools/bin directory.

TIP: You can run the command module load scicon/cli_tools or add the directory to your shell startup script so that you can invoke qps without typing the full path each time. For example:
set path = ($path /u/scicon/tools/bin)

Syntax:

pfe21% qps *job_id*

Example:

pfe21% qps 26130

*** Job 26130, User abc, Procs 1
NODE TIME %MEM %CPU STAT TASK
r1i0n14 10:17:13 2.8 99.9 RL ./a.out
r1i0n14 10:17:12 2.9 99.9 RL ./a.out
r1i0n14 10:17:18 2.9 99.9 RL ./a.out
r1i0n14 10:16:34 2.9 99.8 RL ./a.out
r1i0n14 10:17:11 2.9 99.9 RL ./a.out
r1i0n14 10:17:13 2.9 99.9 RL ./a.out
r1i0n14 10:17:12 2.9 99.9 RL ./a.out
r1i0n14 10:17:15 2.9 99.9 RL ./a.out

Note: The percentage of memory usage by a process reported by this script is a percentage of the memory in the entire node.

Several factors can affect the amount of memory available for your job and the use of memory by your applications. We provide a number of tools you can use to monitor and manage memory usage.

Physical Memory Available on a Node When a Job Starts

The total physical memory of a NAS compute node varies from 128 GB (compute model bro_ele) to 192 GB (compute models sky_ele, cas_ait) for Intel Xeon processors, or 512 GB (compute models rom_ait, mil_ait) for the AMD Rome and Milan processors. However, the amount of memory on a node that is available for a PBS job is less than the total physical memory, because the system kernel can use up to four GB of memory in each node.

When a job starts running, the PBS prologue tries to clean up the memory on the nodes used by the previous job that ran on the nodes, by flushing the job's data from memory to disk. If there is a delay in this cleanup attempt—for example, due to Lustre issues—less memory will be available for the job. After the prologue has the opportunity to flush the memory, if the available memory is less than 85% of the total physical memory of the node, the prologue will terminate the job and return it to the queue. The node will be taken offline.

WARNING: For nodes with a large amount of total physical memory, the use of the 85% rule in the PBS prologue to start a job can lead to a significant imbalance of available memory for nodes assigned to your job. For example, in the case of Aitken's Rome and Milan nodes, the PBS prologue will start your job if each of the assigned nodes has more than 85% of its 512 GB physical memory available. This could lead to some nodes assigned to your job having only 435 GB while other nodes assigned to your job having more than 500 GB—and the possibility of running out of memory on nodes that had lower available memory when the job started.

Memory Used By and For Your Application

Segments and Processes

When you run an application, each of the following segments takes up space in the virtual memory:

text
: Executable instructions.

data
: Pre-defined data—global and static variables initialized by the programmer.

bss
: Uninitialized data—global and static variables that do not have explicit initialization in source code and/or are initialized to zero by the kernel.

stack
: Used for storing local variables and function parameters of a stack frame, which is created when a function is called and removed when the function returns.

heap
: Used for dynamic memory allocation during runtime.

shared libraries
: For example, math and MPI libraries.

You can run the Linux `size` command to list the bytes required (but not necessarily used) by the text, data, and bss memory segments.

Note: Some user applications have built-in reports of the memory requirement. These reports might not take into account the memory used by the shared libraries.

During runtime, the memory needed by each process in the application is paged (in 4 KB units) into the physical memory. The total amount of physical memory used by each process is called resident set size (RSS), which is reported in the RSS column of the `ps` command output or in the RES column of the `top` command output. The total physical memory used by application processes in a node can be estimated by the sum of the processes' RSS; keep in mind that the sum may be an overestimate, as the usage by the shared libraries might be counted more than once among processes.

Memory resource limits set by the system administrator for the Linux shells (such as `stacksize`, `memoryuse`, `vmemoryuse`, and so on) can affect your application. To see the settings, run the `limit` command (for csh) or `ulimit -a` (for bash) on a compute node. You might need to adjust one or more of the settings for your application to run properly.

Page Cache for Application I/O

Page cache, also known as buffer cache, is used to speed up the disk I/O of your application. Page cache usage is reported in the `Cache Mem` entry in the system usage section of the `top` command output.

If your job performs a large amount of I/O, there will be less memory available for your running processes. For some compute-intensive jobs, page cache management (limiting the amount of memory used by page cache) may improve performance. For more information, see Checking and Managing Page Cache Usage.

The /tmp Filesystem

The `/tmp` filesystem on a compute node is a fast, memory-based temporary local filesystem. You can access it directly via the `/tmp`

path or by the $TMPDIR environment variable created by PBS for your job under /tmp/pbs.jobid. pbs_servername.nas.nasa.gov. Usage by /tmp is included in the Cache Mem entry of the top command output.

Note: /tmp usage is limited to 50% of the total physical memory on the node. Using more than 50% results in the error No space left on device.

Tools for Checking Memory Usage

To help you assess the memory usage of your job, we provide the following NAS-developed tools:

qtop.pl
: Invokes the top command on the compute nodes of a job, and provides a snapshot of the amount of used and free memory of the whole node and the amount used by each running process.

qps
: Invokes the ps command on the compute nodes of a job, and provides a snapshot of the %mem used by its running processes.

qsh.pl
: Can be used to invoke the cat /proc/meminfo command on the compute nodes to provide a snapshot of the total and free memory in each node.

gm.x
: Provides the memory high-water mark for each process of your job when the job finishes.

Notes:

- The qtop.pl, qpe, and qps, tools can be used only for jobs running on Aitken and Electra. The gm.x tool can be used for jobs running on Aitken, Electra, and Endeavour.
- For Aitken or Electra jobs, you can run the tools on the PFEs.
- If your Aitken or Electra job uses more than one node, be aware that the memory usage reported in the PBS output file is not the total memory usage for your job; rather it is *the memory used in the first node* of your job.

If Your Job Runs Out of Memory

If your Aitken or Electra job runs out of memory it may be killed by the kernel. If the job needs more memory, see How to Get More Memory for Your Job for possible approaches.

WARNING: The out-of-memory event is not always recorded in the system log files and when this happens, no email is sent to the owner of the job. If you suspect that your job may run out of memory, we recommend getting into the habit of monitoring memory usage using the tools listed above. If you need help, contact us at support@nas.nasa.gov.

For jobs running on Endeavour, the Linux kernel cgroup is used to enforce the resources allocated to your job, such as number of cores and amount of memory. When your job reaches the memory enforced by cgroup, your job will be terminated. If this happens, increase the memory request in your PBS script, and resubmit the job.

For additional information, see the recording and slides for the "Memory Usage on NAS Xeon-Based Systems" webinar, which are available in the HECC webinars archive.

If your job terminates because the nodes it is running on do not have enough available memory, try using one or more of the methods described below.

Note: The memory listed for each type of node in this article refers to the total physical memory in the node. The amount of memory that is actually available to a PBS job is slightly less than the total physical memory of the nodes it runs on, because the operating system can use up to 4 GB of memory in each node. At the beginning of a job, the PBS prologue checks the nodes for free memory and guarantees that 85% of the total physical memory is available for the job.

Use A Processor With More Memory

The following list shows the amount of total physical memory per node for the various NAS x86_64 processor types:

Broadwell
: 128 GB

Skylake
: 192 GB

Cascade Lake
: 192 GB

Rome
: 512 GB

Milan
: 512 GB

If your job runs out of memory on a Broadwell node, you can try using one of the other node types instead. For example, change:

#PBS -lselect=1:ncpus=**28**:model=**bro_ele**

to one of the following:

#PBS -lselect=1:ncpus=**40**:model=**sky_ele**

#PBS -lselect=1:ncpus=**40**:model=**cas_ait**

#PBS -lselect=1:ncpus=**128**:model=**rom_ait**

#PBS -lselect=1:ncpus=**128**:model=**mil_ait**

Run Fewer Processes per Node

If all processes use approximately the same amount of memory, and you cannot fit 28 processes per node (Broadwell), 40 processes per node (Skylake/Cascade Lake), or 128 processors per node (Rome/Milan), then reduce the number of processes per node and request more nodes for your job.

For example, change:

#PBS -l**select=1:ncpus=128:mpiprocs=128**:model=mil_ait

to the following:

#PBS -l**select=2:ncpus=64:mpiprocs=64**:model=mil_ait

Recommendation: Add the command mbind.x -gm -cs before your executable in order to (1) monitor the memory usage of your processes and (2) balance the workload by spreading the processes between the two sockets of each node. For example:

mpiexec -np 60 /u/scicon/tools/bin/mbind.x -gm -cs ./a.out/

If you are running a typical MPI job, where the rank 0 process performs the I/O and uses a large amount of buffer cache, assign rank 0 to one node by itself. For example, if rank 0 requires up to 80 GB of memory by itself, change:

#PBS -l**select=1:ncpus=28:mpiprocs=28**:model=bro_ele

to the following:

#PBS -l**select=1:ncpus=1:mpiprocs=1**:model=bro_ele**+1:ncpus=27:mpiprocs=27**:model=bro_ele

Use the Endeavour System

If any process or thread in a multi-process or multi-thread job needs more than about 500 GB, the job won't run on Electra or Aitken nodes. Instead, run it on Endeavour, which is a shared-memory system.

Report Bad Nodes

If you suspect that certain nodes have less total physical memory than normal, report them to the NAS Control Room staff at (800) 331-8737, (650) 604-4444, or by email at support@nas.nasa.gov. The nodes can be taken offline while we resolve any issues,

preventing them from being used before they are fixed.