CONVEX INTERNAL USE ONLY

FREQUENTLY ASKED QUESTIONS ABOUT GLOBAL SHARED MEMORY (SPP-UX OS VERSION 3)

General Comments

When having problems with running on GSM, always identify what you are running and where you are running it. In particular,

* Always run "scm -c" to identify what subcomplex configurations are on the machine you are using. Make yourself familiar with scm by consulting the man pages for it.
* Always run your a.out with "mpa -v a.out" to make sure that your program attributes are what you think they are. Make yourself familiar with mpa by consulting the man pages for it.
* In general, most of the problems encountered when moving to GSM are actually mpa and/or scm issues. Make yourself familiar with these tools (via man pages) and you will be much more productive.

Marketing Machine Specific information

In the answers below, Marketing Machine Specific information will be annotated with "MMS NOTE". If you have a question which doesn't appear here, email mops and/or isom with your question and include where (scm -c, sysinfo -a) and what (mpa -v) you are running.

FAQs

1. Q: My executable runs fine on single-node OS, but when I run on GSM it immediately aborts with signal 9 or "killed" message. What's wrong?

   A: This is typically due to no global memory being allocated on the subcomplex that you are trying to run on. This can be easily checked by executing "scm -c" (man scm) and examining the GMEM parameter in the output. If this parameter is 0, your ESOM executable cannot allocate any near shared or far shared memory and hence cannot execute.

2. Q: How do I create a subcomplex on multiple nodes?

   A: See the man page for scm for details. Typically this will/should require superuser privileges.

   MMS NOTE: On marketing machines, during standalone sessions, you will have "op scm" capability. Be sure to allocate enough global memory to run your ESOM/parallel executables.

3. Q: How do I execute on a particular subcomplex?

   A: By default, your executable will run on the "System" subcomplex. Use "mpa -sc subcomplex-name a.out" to run your a.out on the subcomplex specified by subcomplex-name. For example if you create a subcomplex with the name "SubcomplexC", then to run your a.out there, submit with: "mpa -sc SubcomplexC ./a.out". See mpa man page for more details.

4. Q: My executable runs fine on single node but when I move to GSM I

get segmentation violations or signal 11. What is wrong?

A: The default stack size, data size, etc. are not the same on all SPP machines whether they are GSM or single node OS. Try running with "mpa -v -STACK -DATA a.out" to see if the default limits are causing your program to get a segmentation violation. See mpa man page for more information.

5. Q: My program is getting wrong answers when I run on GSM. What's wrong?

   A: Could be many things, however try to use statically linked executables if at all possible. This is accomplished by linking with -Wl,-aarchive with fc or cc. Different versions of shared cpslib may actually be the culprit in many cases.

6. Q: Using scm I tried to change the network cache size by modifying the CTICACHE parameter. It doesn't give me an error, but the size doesn't change, what's wrong?

   A: Currently scm cannot change the amount of network cache. This is a hardware limitation on SPP. The amount of network cache must be changed using one of the configuration tools on the test station (ccmu or xconfig) and can only be done at boot time. Note that the value for CTICACHE provided by "scm -c" is bogus. To identify the current amount of network cache, use "sysinfo -a".

   MMS NOTE: On marketing machines this must be done using "op reconfig". See man page for reconfig for more detail.

7. Q: How do I change the network cache size?

   A: See A6 for more detailed information.

8. Q: When trying to change the amount of global memory (GMEM parm) with scm, it fails and gives the message "scm: Global memory has already been configured". Why isn't this working?

   A: The allocation of global memory for a subcomplex can typically only be done once. Global memory must reside in physically contiguous blocks of 16MB. In order to guarantee that global memory can be allocated to a specific subcomplex, you must configure the subcomplexes before user programs are run on the system. Otherwise, physical memory will become fragmented, and it will become impossible to allocate sufficient physical memory for use as global memory.

   MMS NOTE: On Convex marketing machines, you will need to reconfigure using "op reconfig" and the desired subcomplex configuration file (see man pages for scm and reconfig).

9. Q: I have a SOM executable that runs fine on the System subcomplex, but when I run it on a subcomplex that spans nodes, the cnx_sc_fork system call fails. Why?

   A: If the program did an mmap(MAP_ANON|MAP_SHARED) prior to the fork, it obtained a node private memory region, since mmap uses the same memory type as the bss, which for SOM is node private. When the program tries to fork on a remote node, the mmap'ed node-private region can't be shared, so the fork fails with an 'invalid argument' error. This is a bug and will be fixed in the next release.

10. Q: What's the difference between network cache and CTI cache?

A: Two names for the same thing.

11. Q: What is the optimal size for CTI cache?

    A: This will vary greatly depending on the applications being run and the overall hardware configuration. A good rule of thumb is to make CTI cache at least as large as (# nodes)*16MB. For example, with a 4 node SPP, the minimum network cache size should be 64MB (4*16MB). Generally speaking, try to allow enough network cache to hold all of the high re-use data in your benchmark.

12. Q: A customer has just installed their SPP and it boots fine, but it always boots with only processors on node 0 allocated and without global memory. Shouldn't the machine boot with all nodes and a default amount of global memory?

    A: One solution to this is to make /etc/rc run scm at boot time with the desired subcomplex. For example, simply include the line:

    ```
    scm -l /etc/customer.scmfile
    ```

    in /etc/rc (where /etc/customer.scmfile is the desired default

    subcomplex configuration in scm format).

13. Q: Where exactly does network cache reside? What about global memory (far shared and near shared)?

    A: CTI (aka network) cache is actually carved out of physical memory, just as global memory is. Roughly speaking, physical memory on SPP1 will be separated into three regions with near-rigid boundaries. These three regions are:

    1. Network Cache
    2. Global Memory (far shared and near shared)
    3. Local Memory (node private, SOM memory, etc.)

14. Q: How do I determine what *physical* node I am on? I understand that cps_node_id() returns the *logical* node number, but I need to identify what physical node that my process/thread is actually running on.

    A: Use the AIL call node_num(). This will return the node number that the calling thread is actually executing on. Note that on an N node machine the node numbers will not necessarily by numbered from 0 to N-1. If a two node machine is made from nodes numbered 5 and 7, then node_num() will only return the values 5 and 7. (Also note that current_num() is identical to node_num(). However, it's use is dicourraged: the function exists by it's old name for backwards compatibility with previous AIL releases only.)

15. Q: Which barrier routine should I use, the cpslib versions (wait_barrier, cps_barrier) or the low level routine in the AIL (barrier_sync)? Why?

    A: Always use the cpslib versions instead of the AIL where possible. You will save yourself a lot of headaches. For example, what follows is a summary of a real problem that a customer ran into when using the AIL barrier_sync routine incorrectly (which took days to track down).

    The code used the AIL barrier_sync, which is declared like this:

    ```
    unsigned int
    ```

```
barrier_sync(unsigned int          *sema,
             unsigned int           sema_init,
             volatile unsigned int *release,
             unsigned int           release_val);
```

The application in question defined a barrier structure like this:

```
struct __ail_barrier {
  unsigned int          sema;
  unsigned int          sema_init;
  volatile unsigned int release;
  unsigned int          release_val;
};
```

This structure was used in a barrier_sync() call like this:

```
barrier_sync(&(barr->sema), barr->sema_init,
             &(barr->release), barr->release_val));
```

The problem is that in the __ail_barrier structure, sema and sema_init are in the same network cache line, and the barrier_sync() call passes sema_init as an argument, causing sema_init (and sema) to be encached, which causes the barrier_sync to fail.

The requirement that memory-based semaphores not be encached is documented on the sema(3x) man page, but there are many subtle ways things can become encached when you don't expect it. This is especially true on the SPP1200 because it performs prefetching, so you don't even have to reference anything in the cache line.

You are encouraged to use CPS rather than using the AIL directly, because CPS provides allocation and initialization interfaces that prevent these problems.

16. Q: If I have an executable that uses far shared memory on a complex that consists of multiple hypernodes, where will the far shared memory be allocated?

    A: This will depend entirely on the configuration of the subcomplex you are running on. Global memory (far shared, near shared, block shared) is an attribute of a subcomplex and not the host.

    For example, consider the following subcomplex configuration:

```
#Subcomplex Id 1
SC=System:
        UID=0: GID=0: PERM=0555:
        POLICY=1:
        NODEID=0:
                GMEM=64:
                PROCID=0:
                PROCID=1:
                PROCID=2:
                PROCID=3:
#Subcomplex Id 2
SC=MCAE:
        UID=0: GID=0: PERM=0555:
        POLICY=1:
        NODEID=0:
                GMEM=64:
                PROCID=4:
                PROCID=5:
                PROCID=6:
                PROCID=7:
```

```
        NODEID=1:
                GMEM=64:
                PROCID=0:
                PROCID=1:
                PROCID=2:
                PROCID=3:
                PROCID=4:
                PROCID=5:
                PROCID=6:
                PROCID=7:
```

If you execute on the "System" subcomplex, then all far shared memory will be allocated on node 0. On the other hand, if you execute on the "MCAE" subcomplex then the far shared memory will be distributed across both node 0 and node 1 (because they are in the MCAE subcomplex). So, for any particular thread, every other page of far shared memory will be on a remote node.

As per the Exemplar Programming Guide, far shared memory is "distributed on a page basis across the memories of all the hypernodes in a subcomplex."

Note that far shared memory will be distributed across all hypernodes regardless of how many threads your process spawns or which node they execute on. However, also note that although the memory will be distributed, the current implementation does not guarantee node interleaving on a per-process level. Considering the extreme example of two processes allocating far shared memory on subcomplex MCAE. If each process allocated one page of memory sequentially, in turn, one would get all pages from node 0, and the other from node 1.

17. Q: What is the best way to run a throughput test/benchmark on the SPP?

    A: For starters, the SPP-UX load balancing algorithm does not always do a perfect job of evenly distributing processes across nodes. So, on a multiple hypernode configuration, you will want to create a single subcomplex on each hypernode, then submit 8 jobs (ideally with one script) on each subcomplex. For example, your master script might look like this for a 3 hypernode subcomplex:

```
#!/bin/csh -f
date
mpa -sc System ./run8.csh       &
mpa -sc Subcomplex1 ./run8.csh  &
mpa -sc Subcomplex2 ./run8.csh  &
wait
date
```

where System, Subcomplex1, and Subcomplex2 are subcomplexes that contain all CPUs on one (and only one) hypernode. The run8.csh script may look something like:

```
#!/bin/csh -f
cd target0; ./a.out &
cd target1; ./a.out &
cd target2; ./a.out &
cd target3; ./a.out &
cd target4; ./a.out &
cd target5; ./a.out &
cd target6; ./a.out &
```

```
cd target7; ./a.out &
```

It is critical to make the scripts as lean as possible to avoid any unnecessary interaction with the OS.

Also, if possible, try to time the individual jobs rather than report a "start to finish" time as with the above paradigm. For example, run with a master script like:

```
#!/bin/csh -f
mpa -sc System ./run8.csh       &
mpa -sc Subcomplex1 ./run8.csh  &
mpa -sc Subcomplex2 ./run8.csh  &
```

where run8.csh looks like:

```
#!/bin/csh -f
cd target0; /bin/time ./a.out >& time0 &
cd target1; /bin/time ./a.out >& time1 &
cd target2; /bin/time ./a.out >& time2 &
cd target3; /bin/time ./a.out >& time3 &
cd target4; /bin/time ./a.out >& time4 &
cd target5; /bin/time ./a.out >& time5 &
cd target6; /bin/time ./a.out >& time6 &
cd target7; /bin/time ./a.out >& time7 &
```

and report the maximum (or preferably average if allowed) of all jobs ran - on all subcomplexes.

The latter approach eliminates timing of all unnecessary shell script overhead.

Finally, for any benchmark doing even a marginal amount of I/O, it is very important to run executables on subcomplexes that are "local" to their file system. That is, make sure that if a.out is ran on hypernode 3 then all of its files are mounted on a file system that is physically mounted on hypernode 3.

Cross node I/O can be expensive in that system call latency and I/O bandwidth suffer when performed across hypernodes.

18. Q: After building my PVM application with SPP tools, I am getting very little speedup on multiple hypernode configurations. Is there something that can be done with my memory configuration to help things?

    A: Possibly. By default, executables produced with Convex loaders and compilers allocate memory with a far shared paradigm. That is, its memory is allocated in 4K pages on each hypernode in the subcomplex it is running on. As most PVM executables will be single threaded, all its memory should be allocated on the hypernode it is executing on. The memory allocation can be changed using the mpa utility as follows:

```
% mpa -m -private near_shared ./a.out
```

which will change the default behavior so that the process' memory is all on a local hypernode. See the man page for mpa(1) for more

details.

19. Q: When trying to execute on a subcomplex other than the "System" subcomplex, I keep getting errors from mpa:

```
% mpa -sc subcomplex2 /bin/csh
mpa: cnx_sc_fork: Permission denied
% mpa -node 3 /bin/tcsh
mpa: cnx_sc_fork: No more processes
% mpa -node 2 /bin/tcsh
mpa: cnx_sc_fork: Invalid argument
```

A: The first problem (Permission denied) indicates that you do not have execute permission on that subcomplex (subcomplex2). Check the permissions of the subcomplexes with scm and note that the PERM keyword is set much like file permissions. For example, 'PERM=0666' would indicate that the owner, group, and everyone have read and write permission (6), but not execute permission. The permission of the subcomplex 'subcomplex2' apparently is marked so that you do not have execute privileges. The permission on the subcomplex may be modified with scm.

The second and third messages indicate that nodes 2 and 3 do not exist, i.e. your host only has one or two hypernodes (0 and 1).

CONVEX INTERNAL USE ONLY

Isom Crawford, <isom@convex.com>