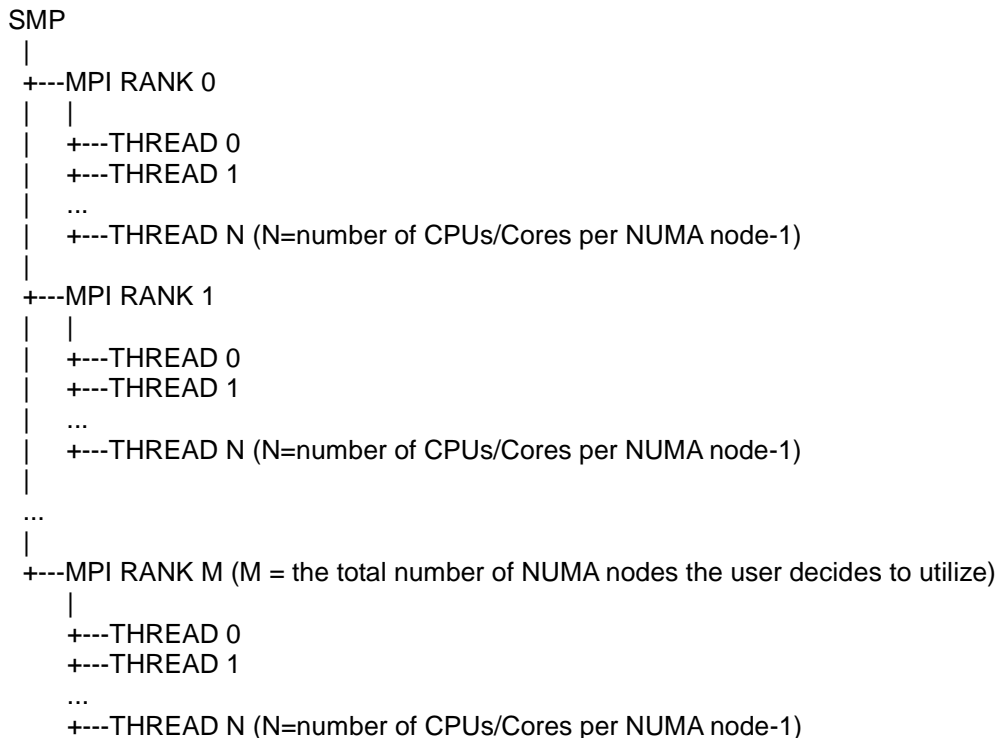


Abaqus Standard 6.8 – Execution Guidelines for running applications in aggregated environment using ScaleMP’s vSMP Foundation

Overview

Abaqus Standard uses a mix of threads and processes (MPI ranks) for parallel execution. For most solvers used by Abaqus, the most scalable arrangement of runtime tasks, on a large-scale shared-memory NUMA machine, is as follows:



where each MPI rank is bound to a different unique NUMA node (board, or blade) in the system. The MPI ranks are responsible for the cross-node communication, and the threads perform the actual solving of mathematical/physical simulations.

While it is possible to run Abaqus Standard on the aggregation platform with the HP-MPI implementation, using MPICH2 tuned for vSMP Foundation may yield a performance improvement of 5-15%.

Running Abaqus-Standard with HP-MPI

HP-MPI has a built-in mechanism for assigning MPI processes to specific CPUs. Process placement is controlled by environment variables named **MPI_BIND_MAP** and **MPIRUN_OPTIONS**. When these variables are not set, process placement will not be performed.

Environment variables – HP-MPI

If you are running with HPMPI, you should set the following environment variables prior to running Abaqus to yield the optimal performance:

```
export MPI_BIND_MAP=0,1,2,3,4,5,6,7 (For example)

export MPIRUN_OPTIONS="-cpu_bind=map_cpu,v"

export HPMP_FRAGSIZE=131072

export MPI_SHMEMCNTL=16,24000000,4000000
```

MPI_BIND_MAP specifies a list of CPUs to which MPI ranks will be bound. You should replace the list above with a list of integers, zero to #cpus-1.

For more information on HP-MPI CPU affinity settings, refer to the HP-MPI user's guide available from ["http://docs.hp.com/en/B6060-96022/B6060-96022.pdf"](http://docs.hp.com/en/B6060-96022/B6060-96022.pdf).

Running Abaqus- Standard with MPICH2 tuned for vSMP

To run Abaqus using MPICH2, the Intel MPI wrapper that came with Abaqus should be used.

Environment variables – MPICH2

```
export VSMP_PLACEMENT=NODES

export VSMP_MEM_PIN=YES
```

Create Symbolic link for libmpi:

```
cd /opt/ScaleMP/mpich2/1.0.8/lib
ln -s libmpich.so libmpi.so.3.1
```

Above variables are already set in "abaqus_v6.env" file below. Uncomment out mp_mpi_implementation & mp_mpi_implementation in "abaqus_v6.env".

Running Abaqus- Standard with Ram-Drive

Since Abaqus jobs perform a lot of scratch IO as they run out-of-core, running with a Ram-Drive may yield shorter runtimes. In order to run with Ram-Drive, perform the following:

Change 'memory' value in abaqus_v6.env so that 80% of total memory divided by # boards used.

```
export LM_LICENSE_FILE=<port>@<license server>

sudo mkdir /ramfs

sudo mount /dev/ram -t ramfs /ramfs -o noatime

sudo chmod 777 /ramfs
```

```

source <path_to_intel_compiler>/iccvars.sh

cp abaqus_v6.env s2a.inp /ramfs

cd /ramfs

export LD_LIBRARY_PATH=/opt/ScaleMP/mpich2/1.0.8/lib

abq681 job=stdx14 input=s2a.inp cpus=14 interactive

```

If cpus is equal or less than '# of cores on each board', then bind all threads to the first board as follows:

```
numactl -cpunodebind=0 abq681 job=stdx8 input=s2a.inp cpus=8 interactive
```

Sample script for HP-MPI and MPICH2 tuned for vSMP

```

----- abaqus_v6.env -----
memory='24 gb'

mp_host_list=("localhost",7), ("localhost",7), ("localhost",7), ("localhost",7)

mp_mpi_implementation = "intel"          # Comment this out for HPMPI
mp_mpirun_path={"intel": "/opt/ScaleMP/mpich2/1.0.8/bin/mpiexec"}
mp_rsh_command = 'ssh -n -l %U %H %C'

import os
os.environ['ABA_RESOURCE_MONITOR']='1'
os.environ['ABA_MPI_SKIP_BUNCH_NODES']='1'
os.environ["ABA_ALLOW_USER_MPI"]="1"
os.environ["ABA_MPI_NOPAM"] = "1"

os.environ['VSMP_PLACEMENT']='NODES'
os.environ['VSMP_MEM_PIN']='YES'
os.environ['VSMP_VERBOSE']='YES'

os.environ['HPMP_FRAGSIZE']="131072"
os.environ['MPI_SHMEMCNTL']="16,24000000,4000000"
os.environ['MPI_BIND_MAP']="0,1,2,3,4,5,6,7,8,9,10,11,12"
os.environ['MPIRUN_OPTIONS']="-cpu_bind=map_ldom,v"

def mp_mpiCommand():
    import os, uti

    print 'Preparing command for %s...' % str(mpi_implementation)
    newEnvironment = {}
    newCommand = None

    if mpi_implementation == 'intel':
        newCommand = mpirun_path
        newCommand = '%s %s -np %d %s %s' % (newCommand, mpirun_options, cpus, program,
arguments)
        print "newCommand = ", newCommand
    else:
        print "You are not using the SCALEMP MPI."
        newCommand = command
    return(newCommand,newEnvironment)
-----

```