

StarCCM+ : Execution Guidelines for running applications in aggregated environment using ScaleMP's vSMP Foundation

Overview

StarCCM+ is a multi-process application that uses MPI for inter-process communication. HP-MPI has been set as the default MPI for the StarCD application. In addition, StarCD supports several other MPIs, which enable to use MPICH2 indirectly.

Running StarCCM+ with HP-MPI on the aggregation platform would result in good performance. For achieving even better performance (especially for smaller jobs), using MPICH2 tuned for vSMP Foundation, may yield a performance improvement of ~ 10%.

Running StarCCM+ 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

You should set the following environment variables prior to running 'starccm+' to yield the optimal performance:

```
# For 32 CPU system
export MPI_BIND_MAP=7,6,5,4,3,2,1,0
export MPIRUN_OPTIONS="-cpu_bind=map_cpu,v"
export MPI_SHMEMCNTL=16,24000000,4000000
export HPMP_FRAGSIZE=102400
export MPI_ROOT=/opt/hpmpi
export STARFLAGS="-mpi=hp"
export PATH=/opt/cd-adapco/starccm+4.02.011/star/bin:$MPI_ROOT/bin:$PATH
```

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).

Handling Excessive System-Calls

HP-MPI performs excessive system calls to sched_yield, which are unnecessary when running with vSMP Foundation; hence running with a pre-load library which eliminates such calls would result in better performance.

The pre-load library (**libvsmpclib.so**) is installed with vSMPPP (vSMP Productivity Pack) under: **=/opt/ScaleMP/libvsmpclib/0.1/lib64**

In the run-script for StarCCM+ please add the pre-load library by adding:

```
export LD_PRELOAD=/opt/ScaleMP/libvsmpclib/0.1/lib64/libvsmpclib.so:$LD_PRELOAD
```

Running StarCCM+ with Ram-Drive

StarCCM+ may perform better if you copy your input files to a directory in a RAMFS mount, and 'cd' there before running starccm+. This is especially useful for large-scale jobs.

In order to run with Ram-Drive, perform the following:

```
mount | grep ramfs >/dev/null
if [ "$?" = "1" ] ; then
    sudo mkdir -p /ramfs
    sudo mount -t ramfs ramfs /ramfs
    sudo chmod 777 /ramfs
fi

mkdir -p /ramfs/starccm
cp -p civil_trim_20m.sim run* benchmark* star* /ramfs/starccm
cd /ramfs/starccm

export TMPDIR=/ramfs/tmp
mkdir -p $TMPDIR
```

After the run terminates, you will need to copy the output files back to a disk-backed file system; **files in RAMFS do not survive system reboots.**

Sample script for StarCCM+ with HP-MPI and Ram-Drive

```
#!/bin/bash
ulimit -s unlimited
BASE=`pwd`

# For 32 CPU system
export MPI_BIND_MAP=31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0

export MPIRUN_OPTIONS="-cpu_bind=map_cpu,v"
export MPI_SHMEMCNTL=16,24000000,4000000
export HPMP_FRAGSIZE=102400
export MPI_ROOT=/opt/hpmi
export STARFLAGS="-mpi=hp"
export PATH=/opt/cd-adapco/starccm+4.02.011/star/bin:$MPI_ROOT/bin:$PATH

export MALLOC_TOP_PAD_=20000000

mount | grep ramfs >/dev/null
if [ "$?" = "1" ] ; then
    sudo mkdir -p /ramfs
    sudo mount -t ramfs ramfs /ramfs
    sudo chmod 777 /ramfs
fi

mkdir -p /ramfs/starccm
cp -p civil_trim_20m.sim run* benchmark* star* /ramfs/starccm
cd /ramfs/starccm

export TMPDIR=/ramfs/tmp
mkdir -p $TMPDIR

cpus=32
TAG=`date +%j-%H%M`.M2
OUTFILE=$BASE/civil.$cpus.$TAG.out

date >$OUTFILE
time starccm+ -power -np $cpus civil_trim_20m.sim -batch benchmark.java >>$OUTFILE
date >>$OUTFILE
```

Running StarCCM+ with MPICH2 tuned for vSMP Foundation

StarCCM+ is provided with support for Cray rapid-array (as well as other MPI implementations). In order to configure StarCCM+ to work with MPICH2 tuned for vSMP Foundation, we are taking advantage of the fact that Star-CCM+ supports certain external MPI implementations, and in this case, we are instructing Star-CCM+ to use Cray rapid-array, while the underlying implementation in use is MPICH2.

The following should be adjusted in the run-script:

```
-----  
export MPI_ROOT=/opt/ScaleMP/mpich2/1.0.8  
cp -p $MPI_ROOT/lib/shared/libmpich.so.1.0 $BASE/librapl.so  
export RA_HOME=$BASE  
export RAMPI_HOME=$MPI_ROOT  
export VSMP_PLACEMENT=PACKED  
export VSMP_MEM_PIN=YES  
export VSMP_VERBOSE=YES  
export USE_MPI_WRAPPER=MPICH1  
  
time starccm+ -power -np $RANKS civil_trim_20m.sim -batch benchmark.java -mpidriver rapidarray -rsh ssh  
>>$OUTFILE  
vsmputil -unpinall  
-----
```

Sample script for MPICH2 tuned for vSMP Foundation

```
#!/bin/bash

ulimit -s unlimited
BASE=`pwd`

export MALLOC_TOP_PAD_=20000000

mount | grep ramfs >/dev/null
if [ "$?" = "1" ] ; then
    sudo mkdir -p /ramfs
    sudo mount -t ramfs ramfs /ramfs
    sudo chmod 777 /ramfs
fi

mkdir -p /ramfs/starccm
cp -p civil_trim_20m.sim run* benchmark* star* /ramfs/starccm
cd /ramfs/starccm

export TMPDIR=/ramfs/tmp
mkdir -p $TMPDIR
export MPI_ROOT=/opt/ScaleMP/mpich2/1.0.8
export PATH=/opt/cd-adapco/starccm+4.02.011/star/bin:i$MPI_ROOT/bin:$PATH

# Setup a mockup for rapidarray so that we can use our MPICH1

cp -p $MPI_ROOT/lib/shared/libmpich.so.1.0 $BASE/librapl.so
export RA_HOME=$BASE
export RAMPI_HOME=$MPI_ROOT
export VSMP_PLACEMENT=PACKED
export VSMP_MEM_PIN=YES
export VSMP_VERBOSE=YES
export USE_MPI_WRAPPER=MPICH1

for cpus in 64 48 32 16 8 ; do
    cpus=32
    TAG=`date +%j-%H%M`.M2
    OUTFILE=$BASE/civil.$cpus.$TAG.out
    STATFILE=$BASE/civil.$cpus.$TAG.stat

    date >$OUTFILE
    time starccm+ -power -np $cpus civil_trim_20m.sim -batch benchmark.java -mpidriver rapidarray -rsh ssh
    >>$OUTFILE
    date >>$OUTFILE
done

vsmputil --unpinall
```
