

## LS-Dyna – Execution Guidelines for running applications in aggregated environment using ScaleMP’s vSMP Foundation

### Overview

LS-Dyna is a multi-process application that uses MPI for inter-process communication. LS-Dyna supports HP-MPI, Intel-MPI and also MPICH1. A separate executable is provided by LS-Dyna for each MPI. Running LS-Dyna with Intel-MPI on the aggregation platform would result in good performance. For achieving even better performance, MPICH2 tuned for vSMP Foundation can be used (it is binary compatible with Intel-MPI and can easily replace it) and may yield a performance improvement of 3% - 10%.

The following execution guidelines are relevant for LS-Dyna 971.

### Running LS-Dyna with MPICH2 tuned for vSMP Foundation

MPICH2 has a built-in mechanism for assigning MPI processes to specific CPUs. Process placement is controlled by an environment variable named **VSMP\_PLACEMENT**. When this variable is not set, process placement will not be performed. To run LS\_Dyna with MPICH2 optimally, it is recommended to also set the **VSMP\_MEM\_PIN** environment variable.

### Environment variables – MPICH2

You should set the following environment variables prior to running LS-Dyna with MPICH2 to yield the optimal performance:

```
export HERE=`pwd`
export MPI_DIR=/opt/ScaleMP/mpich2/1.0.8
export PATH=$MPI_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPI_DIR/lib:$HERE:$LD_LIBRARY_PATH
export VSMP_PLACEMENT=PACKED
export VSMP_MEM_PIN=yes
```

The value of VSMP\_PLACEMENT can specify one of the following placements:

PACKED means assign each MPI rank to one CPU from last downwards. Explicit placement format defines a list of rank:cpu pairs separated by commas.

For example: 0:8,1:9,2:10,3:11,4:12,5:13,6:14,7:15 assigns ranks 0-7 to CPUs 8 to 15 respectively.

### Using MPICH2 instead of Intel-MPI

The LS-Dyna executable that uses Intel-MPI requires certain MPI shared libraries. To make this executable work with MPICH2 instead, symbolic links are needed to point the names of the Intel-MPI shared libraries to the MPICH2 shared libraries. This symbolic links are created locally and the current directory should be added to the library path environment variable LD\_LIBRARY\_PATH (see example above).

```
rm libmpi.so.2.1 libmpiif.so.2.1
ln -s $MPI_DIR/lib/libmpich.so $HERE/libmpi.so.2.1
ln -s $MPI_DIR/lib/libfmpich.so $HERE/libmpiif.so.2.1
```

## ***Running LS-Dyna with Ram-Drive***

LS-Dyna may perform better if it uses a RAMFS mount for its file processing. In order to run with Ram-Drive, perform the following before running LS-Dyna

```
sudo mkdir -p /ramfs
sudo mount -t ramfs ramfs /ramfs -o noatime
sudo chmod 777 /ramfs

# This is just an example of a "pfile"...

cat > $HERE/data/ramfs.pfile <</EOF
general { nodump }
dir { global /ramfs local /ramfs }
/EOF

cd /ramfs
```

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 LS-Dyna with MPICH2 and Ram-Drive

```
# Setup Environment variables
export HERE=`pwd`
export DATA=$HERE/data
export MPI_DIR=/opt/ScaleMP/mpich2/1.0.8
export PATH=$MPI_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPI_DIR/lib:$HERE:$LD_LIBRARY_PATH
export VSMP_PLACEMENT=PACKED
export VSMP_MEM_PIN=yes

# Setup shared libraries to point to MPICH2
rm libmpi.so.2.1 libmpiif.so.2.1
ln -s $MPI_DIR/lib/libmpich.so $HERE/libmpi.so.2.1
ln -s $MPI_DIR/lib/libfmpich.so $HERE/libmpiif.so.2.1

# Setup RAM-Drive
sudo mkdir -p /ramfs
((`mount|grep ramfs|wc -l` == 0)) sudo mount -t ramfs ramfs /ramfs -o noatime
sudo chmod 777 /ramfs

# This is just an example of a "pfile"
cat > $DATA/ramfs.pfile <</EOF

general { nodump }

dir { global /ramfs local /ramfs }

/EOF

cd /ramfs
export OUTFILE=$HERE/Results
export EXE=$HERE/mpp971_s_7600.2.1224_Intel_linux86-64_IntelMPI

# Licensing (just an example)
export LSTC_LICENSE_SERVER={your license server}

export LSTC_FILE={port @ license server}

# The following is an example with paramaters that can be modified (32-way)

mpiexec -np 32 $EXE << /EOF &>$OUTFILE

P=$DATA/ramfs.pfile I=$DATA/3cars_shell12_150ms.k MEMORY=300000000

/EOF

cd $HERE
rm libmpi.so.2.1 libmpiif.so.2.1
-----
```