Parallel Execution

Parallel Computing Environment

Several parallel machines are currently supported (with more being added all the time). The machines currently supported are ASCI Red (Janus & Janus-s), the Dec8400 Cluster and the SGI Origin 2000.

Before you can run a problem in parallel, you'll need to do some minimal setup. Each parallel machines differs. However, we have tried to make the interface to both parallel and serial machines as similar possible.

Several new concepts and files have been added for parallel. For a serial run, all that was necessary was a mesh file (basename.g) and an input file (basename.i). These two files are the starting point for a parallel analysis. Several additional files are needed to describe the decomposition of the mesh, the i/o structure, etc. Most of the details of all of the other files are not important to the user and are only described to the detail needed to perform an analysis on a parallel platform.

Sandia Labs ACCESS Setup

The first thing that a user must do to use the ACCESS system (either in serial or parallel) is define the ACCESS enviroment variable and path to access the ACCESS system. The following two lines need to added to the .cshrc file:

```
    setenv ACCESS PATH_FROM_TABLE_1
    set path = ($ACCESS/etc $ACCESS/bin $path)
```

Table 1 ACCESS environment variable for all platforms.

Machine	ACCESS variable
JAL LAN (Suns & Decs)	/usr/local/eng_sci/struct/current
Engineering Sciences LAN	/usr/local/
Janus (Janus-s)	/Net/projects/seacas/janus/current
Dec8400	/usr/community/Access/current
Cray J90's (sacj001 & sacj002)	/usr/community/Access/current
Atlantis & Discovery (Origin 2000)	/projects/seacas/atlantis/current
Atlantis & Discovery [64-bit version][a]	/projects/seacas/atlantis/current64
Tesla	/projects/seacas/current
Tesla [64-bit version]	/projects/seacas/current64
SRD Lan	/usr/local/eng_sci/struct
Edison	/projects/seacas

a.The 64-bit versions can be used for very large models. However, note that the exodus 1 databases created by the 64-bit versions are not compatible with the exodus 1 databases created by the normal 32-bit versions. The databases need to be converted to exodusII to be moved between the 32-bit and 64-bit versions.

The $path entry is whatever path you already have set up. Alternatively you can simply add the two ACCESS entries to your existing path definition. as follows:

```
setenv ACCESS PATH_FROM_TABLE_1
set path=($ACCESS/etc $ACCESS/bin ... )
```

ACCESS may also be installed on other systems. You will need to contact the system manager of the systems not mentioned here to find out where ACCESS is installed.

Distributed Computing

PRONTO3D and JAS3D use a message passing interface to take advantage of the distributed computing environment. A problem is typically decomposed into a series of smaller subproblems that are assigned to different processors and running in parallel. The message passing interface (MPI) is then used to pass information between processors.

Before we can run in this environment, the input mesh must be decomposed. We use a tool called nem_slice, which is based on Chaco, to partition the mesh into smaller parts. Once partitioned, the mesh must be broken up into individual input files that can be read by the analysis codes. We use the tool nem_spread to accomplish this task.

Once the mesh has been sliced and spread, a script file is used to launch the analysis codes on multiple processors. Each processor will read its individual sub mesh. One command input file will be read and broadcast to all processors. One text output file will be written. Multiple exodusII results files will be written. Before the analyst can visualize the results, these multiple (exodusII results) files have to be re-combined. Currently, either nem_join (the tool of choice) or concat (an old tool being phased out) can be used to recombine the mesh.

For most production calculations, the slicing and recombining of results files may need to be done on a different platform that the one used for performing the analysis. The details of when and where this is necessary are described below in the section about each machine.

The following steps are required to perform an analysis in parallel:

```
    1) Generate a basename.g (mesh) and basename.i (input) file.
    2) Decompose the mesh using Nem_Slice.
    3) Spread the mesh onto the file systems using Nem_Spread.
    4) Run the Analysis (includes restarts if necessarry).
    5) Combine the results into one file for post processing.
```

Step 1 will not be discussed here. The mesh file (basename.g) should be an ExodusII file. To test the type of mesh file you have issue:

```
    exotstv2 basename.g
```

If this command reports the file is not a netcdf file, convert it to an ExodusII file with the following commands:

```
    mv basename.g basename.g1
    ex1ex2v2 basename.g1 basename.g
```

Step 2 is a very time consuming and memory intensive operation (its also a serial operation regardless of platform). As a result, this step is often performed on a machine other than the machine on which the analysis will be perfomed (e.g., using Edison to slice a problem to be run on Janus-s).

Step 3 must be performed on the machine on which the analysis will be perfomed regardless of where step 2 was performed. This step writes an individual sub mesh file for each processor. This step does NOT need to be done on the same number of processors on which the analysis will be performed. In fact, it is order(s) of magnitude faster to perform the spread on only a few processors (an 8 million element mesh decomposed for 3200 processors was spread on Janus using 24 processors). The number of processors that are used for the spread does NOT change the number of processors on which the analysis will be performed (this is determined in step 2).

Step 4 varies significantly from machine to machine. On Janus, jobs can be run interactively (only small jobs) or from the queue system (used for production calculations). Due to time limitations in the queues on Janus restarts are usually necessary. On the Dec8400, everything is run interactively but a queue system will be added in the future. This step will be discussed in the machine specific sections below.

Step 5 can be performed on the machine where the analysis was run or on another machine. Currently, only small jobs are recombined on Janus. Instead, another machine such as a Sun workstation is used to recombine the results. On the Dec8400 and Atlantis, the results will probably be combined on the machine. The details are discussed below.

Figure 7 shown below illustrates the flow of data for a typical analysis.

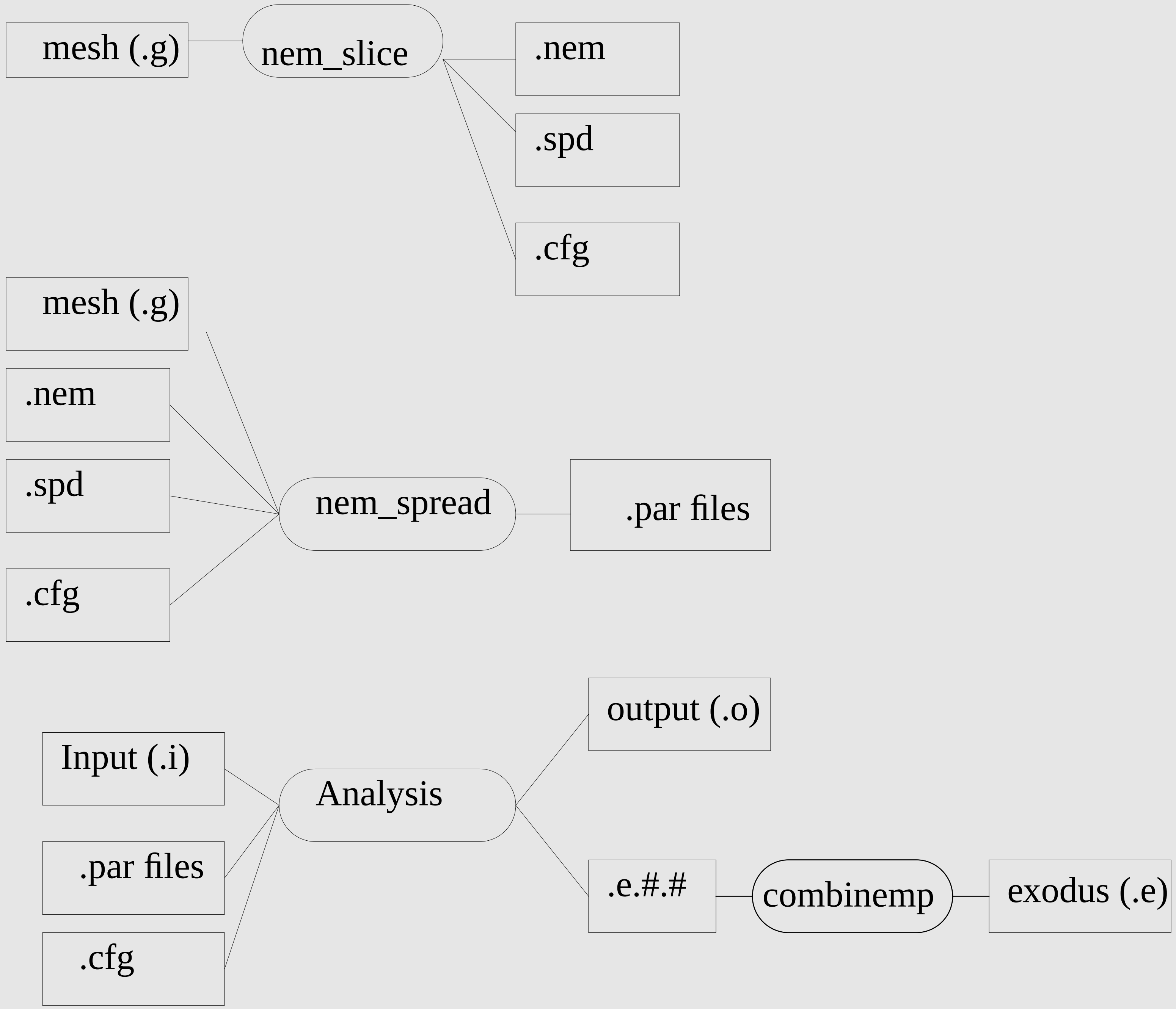

Figure 7 **Data flow for a parallel analysis.**

How to Run a Parallel Analysis (General Overview)

Introduction

The section gives an overview of all the steps required to run an analysis in parallel. It is recommended you read this section before jumping to the machine specific sections because it is assumed the reader is familiar with each of the steps.

Decomposing a Mesh (Slicing)

The procedure for slicing a mesh is independent of the platform. This is the step where the analyst must determine the number of processors that will be used for the analysis. If the number of processors is changed, this step must be redone. The command to slice is the mesh is (a method using a Make file will be discussed below)

```
loadbal -p #Processors basename [-m default/tflop]
  where #Processors is the number of processors on which the analysis
    will be performed
  -m default/tflop is an optional switch. “tflop” is for running on Janus
    only. “default” sets up the files to run on every other machine
    (Atlantis/Discovery, Dec8400, etc.).
```

This step will create the following files:

```
basename.cfg
basename.lbd.out (non-essential)
basename.lbd.err (non-essential)
basename.nem
```

```
    basename.spd
```

If the slice was performed on a machine other than the machine on which the analysis will be run, all the files must be tarred up and sent to the machine on which the analysis will be performed. To tar up the files issue the command

```
    tar -cvf basename.tar basename.*
```

This tar file can then be sent to another machine with

```
    scp basename.tar machine:PATH
      where PATH is the directory to put the tar file

    An example of sending impact.tar to /scratch/username on Janus is
    scp impact.tar janus:/scratch/username/.
```

If the analysis is run on another machine, log onto that machine, change to the directory where you want to run the analysis and untar the tar file with

```
    tar -xvf basename.tar
```

Spreading the Mesh

Spreading the mesh is performed using a C-Shell script called basename.spd. In general, the mesh can be spread by simply issuing

```
    chmod 755 basename.spd
    basename.spd
```

Unfortunately, there are exceptions to this. These exceptions only occur, if you slice a mesh on one machine and spread the mesh on another (this is required for large problems run on Janus or Janus-s). Bedrock2 (IRN) and Edison (ISN) are setup by default for running on Janus and Janus-s respectively. This means no changes are necessary. If you slice on a Sun workstation and run on Janus additional changes are required. This is not currently recommended and will be discussed in detail in the developers section.

Running an Analysis on a Parallel Platform

In this section, an overview of running an interactive job is presented. The machine specific details will be presented in the machine specific sections below. Any of the pronto3d/jas3d options can be added to the command line used to run the code. The ones listed here are the minimum required to run in parallel:

```
    pronto3d -parallel -mesh basename.par -- basename
```

NOTE: This is not an acceptable way to preform large analyses (>8 processors) or long running analyses (>30 Minutes) on Janus or Janus-s. Please see the Janus section for more details.

Recombining the Results

The final step is to recombine the individual results file into one ExodusII file for visualization. In most cases, this can be done on the machine where the analysis was performed (large jobs on Janus or Janus-s are the exception). The results can be recombined with the following command on all machines EXCEPT Janus:

```
    nem_join basename.pex
```

This will create a file called basename-out.e. This is an ExodusII results file and can be post-processed as any serial results file. In the case of running large jobs on Janus(-s), a script is used to ftp the individual processor files to a back end machine (Atlantis/

Discovery on the IRN and Edison on the ISN) and combine the files. The format for this script is

```
combinemp -v basename
```

More information on the script can be found in the tools section.

Using the Make System

For production calculations, most analysts are not using the Make system. This is a result of the numerous restarts, etc. which are required for running large jobs on Janus. Despite this fact, the make system still provides a fast and easy way to perform many of the steps required to perform a parallel analysis.

If you want to use the Makefile tool, you will need to create a “Makefile” in the directory that contains the mesh and input files. In order to generate a Makefile that has predefined problem rules, you must first create a file called Imakefile. This file only needs to contain a single entry:

```
NormalProblemRule(codename,basename)
```

In this case “basename” would be replaced with the name of the problem (i.e. the prefix of the .g and the .i files). The codename should be pronto3d or jas3d.

To generate the Makefile, the user types the following command at the UNIX prompt (the command name is derived from “access make makefile” and the -D MPI is used to turn on the parallel problem rules):

```
accmkmf -D MPI
```

This command will create a Makefile that has a predefined set of rules for how to build individual components of the analysis.

The advantage of the Makefile system is that the user can type a single command that will perform all of the operations necessary to produce output (this can not be used for large problems on Janus because the slice and recombine steps must be performed on another machine). These commands use the following format:

```
    make target
```

The targets that are relevant to running the problem are as follows:

```
    slice: performs the slice operation on the mesh file.
    spread: performs the spread operation by running the base.spd script.
    output: performs the analysis but does not recombine the output.
    base.e: performs the analysis and recombines the results files.
```

The advantage of these targets is that they perform the previous steps in an operation if they haven’t been completed. Thus the command:

```
    make base.e
```

would slice. spread, analyze, and then recombine the exodus files. Whereas if slicing had already been completed, the command would simply spread,analyze, and recombine the exodus files. Examples will be given in the example section for each machine below.

Running a Parallel Analysis on Janus

Introduction

Janus is the most powerful machine we support and also the most demanding to use. For large jobs (>50,000 elements) the mesh must be sliced and recombined on another machine (currently, Atlantis/Discovery on the IRN and Edison on the ISN). Production problems smaller than (50,000 elements) should be run on another platform. The procedure for running small problems is outlined in the end to help user’s learn how to run on Janus with a small mesh (learning the system using a large mesh is not recommended). The procedure is described for Janus and Atlantis/Discovery but Janus-s and Edison can be substituted for jobs on the ISN.

NOTE: You must set up your environment variables and paths before you begin. On Janus and Janus-s, it is recommended you set the following enviroment variable in you .cshrc file

```
    setenv YOD_OPTIONS “-stack 2M -comm 2M”
```

This increases the communication buffer and stack size (the reasons are not important to understand).

Procedure

The first step is to slice the mesh on Atlantis/Discovery. If you don’t already have a directory on the scratch disk create one (/scratch on Atlantis/Discovery and /32stripe on

Edison). Change to this directory and get your basename.g and basename.i files. Decompose the mesh with

```
loadbal -p # -m tflop basename
  where # is the number of processors
```

When this finishes, create a tar file using

```
tar -cvf basename.tar basename.*
```

Log onto Janus and create a directory for your problem on the scratch disk (/scratch).

From Atlantis/Discovery, put your tar file onto Janus with

```
scp basename.tar janus:/scratch/username/dir/.
```

On Janus, the next step is to spread the mesh. If the job is small enough (<200,000 elements), you can spread the mesh with

```
chmod 755
basename.spd
```

If the mesh is very large, you will need to use the queues to spread the mesh (this will be discussed in more detail in the queue section). After spreading the mesh, the next step is to run the analysis. For very small jobs, this can be done interactively with

```
pronto3d -parallel -mesh basename.par -- basename
```

Large jobs must be run using the queues. Once the job is finished running (either from the queues or interactively), the final step is to recombine the mesh. In your scratch directory on Atlantis/Discovery (where you sliced the mesh), issue

```
combinemp -v basename
```

You can get more details on this in the combinemp section. This will create a results file called basename.e.#processors.

Using the Queues

Their are multiple queues on Janus. Due to the evolving nature of the queue system, they will not be discussed here (refer to the FAQ’s on the application server SASN100). To submit a job to the queues, you must have a script which describes what commands you want executed. The minimum information necessary would be

```
janus% cat run_script
#!/bin/csh
cd directory
command
```

The general form for submitting a job to the queues is

```
qsub -q QUEUE -lT TIME -lP PROCESSOR -eo -ro -re SCRIPT
where
  QUEUE is the name of the queue (e.g. snl, snl.big, etc.)
  TIME is the amount of time requested (e.g. 2:00:00 )
  PROCESSORS is the number of processors (e.g. 512)
  SCRIPT is the name of the script (e.g. run_script)
```

An example will be given in the large example problem section.

Examples

A variety of examples are provided to aid the new user in running analyses on Janus. The first example is for very small problems. The second example is the full range of things a user would need to do to run a very large analysis including restarts, etc.

Very Small Problems

For problems in this category (<10,000 elements on <8 processors running <30 Minutes), everything can be done on Janus (slice, spread, analysis &recombining the results). The first method is to simply use the Make system to do everything. The second method is to run each step individually.

In this example, the problem basename is assumed to be “small” and the analysis code is pronto3d. The user (jdoe) has already created a directory on the scratch disk (/scratch/jdoe/small) where he/she has place the small.g and small.i files. The analysis is to be performed on 8 processors.

To use the make system, “jdoe” would do the following

```
cd /scratch/jdoe/small
echo “NormalProblemRule(pronto3d,small)” > Imakefile
accmkmf -D MPI
make NUM_PROCS=8 small.e
```

This would run all the pieces and create the results file. To run all the scripts individually, “jdoe” would do the following:

```
cd /scratch/jdoe/small
loadbal -p 8 small
small.spd
pronto3d -parallel -mesh small.par -- small
yod -sz 1 $ACCESS/bin/nem_join small.pex
```

Large Problems

As an example, consider a problem called impact that be run using 512 processors on Janus. On Atlantis/Discovery, create a directory on the scratch disk to slice the problem and get the input files into this directory

```
    Atlantis% mkdir /scr/username/impact
    Atlantis% cd /scr/username/impact
    Atlantis% cp path/impact.i .
    Atlantis% cp path/impact.g .
```

Slice the mesh

```
    Atlantis% loadbal -p 512 -m tflop impact
```

Tar up the files to be sent to JANUS

```
    Atlantis% tar -cvf impact.tar impact.*
```

Log onto janus and create a directory on the scratch disk for this problem

```
    janus% mkdir /scratch/username/impact
    janus% cd /scratch/username/impact
```

Copy the files from Atlantis/Discovery to janus

```
    Atlantis% scp impact.tar janus:/scratch/username/impact/.
```

Spread the mesh using either the queues or interactively (depends on the size)

```
    Interactively:
      janus% chmod 755 impact.spd
      janus% impact.spd

    Using the Queues:
      1) Change permissions on impact.spd
          janus% chmod 755 impact.spd
      2) Create a script called run_spread with the following information
          #!/bin/csh
          cd /scratch/username/impact
          impact.spd
      3) Change permissions on run_spread
          janus% chmod 755 run_spread
      4) Submit the job to the Queue. The spread is run using 12 processors (-
         lP 12) with a time limit of 2 hours (-lT 2:00:00) in the snl.day
        queue (-q snl.day) with the command:
          janus% qsub -q snl.day -lT 2:00:00 -lP 12 run_spread
```

To run the analysis using the queue, begin by creating a script called run_pronto with the following

```
#!/bin/csh
cd /scratch/username/impact
pronto3d -parallel -mesh impact.par -- impact
```

and change its permissions with

```
chmod 755 run_pronto
```

To submit this to snl.long queue use the following command:

```
janus% qsub -q snl.long -lT 24:00:00 -lP 512 run_pronto
```

Recombine the results on Atlantis/Discovery in the directory where you sliced the problem

```
Atlantis% cd /scr/username/impact
Atlantis% kinit -f
Atlantis% combinemp -v impact
```

Restart the analysis if required. NOTE: You results will be overwritten so make sure they have been combined back together on Atlantis/Discovery before proceeding with this step

```
janus% vi impact.i (set the restart time)
janus% movemp -v impact.rsout impact.rsin
janus% qsub -q snl.long -lT 24:00:00 -lP 512 run_pronto
```

Repeat the the recombine and restart steps until the analysis is completely finished. Note that the movemp command will overwrite the previous impact.rsin file. If you wish to keep it, use the movemp script to move it to a new name (e.g., impact.rsin_1)

Running an Analysis on the Dec8400 or Atlantis

Introduction

The procedure for every machine except Janus is the same. The mesh will be decomposed on the machine it will be run on regardless of the size. Machine specific details are contained in the table below.

Table 2 Machine Specific Information

Machine	Scratch Disks	.cshrc Specific Items
Dec8400	/scratch1 ... /scratch14	limit datasize 4096M limit vmemoryuse 4096M
Atlantis	/scratch	

Procedure

The first step is to create a directory to work from on the scratch disks. Put the mesh files and input files in this directory. The next step is to slice the mesh using the command:

```
    loadbal -p # basename
```

The mesh can be spread with the command

```
basename.spd
```

The analysis can be run with the command:

```
nohup pronto3d -parallel -mesh basename.par -- basename >& run.log &
```

The mesh can be recombined using the command

```
nem_join basename.pex
```

Useful (required) parallel tools:

copymp

Use this routine to copy files that have been spread to the parallel file system.

```
    copymp [-v] basename_a.extension_a basename_b.extension_b;
    The optional -v flag is for verbose.
    Neither extension_a nor extension_b can contain periods ('.').
     Copies files in directories (dir_a(n)) specified
      in the basename_a.cfg file to files in directories (dir_b(n))
      specified in the basename_b.cfg file.
     The number of files to copy is read from the basename_a.cfg  file,
       which must already exist.

       dir_a(n)/basename_a.extension_a.n --> dir_b(n)/
        basename_b.extension_b.n
               n=0,...,#processors-1

      basename_a.cfg     must already exist
      basename_b.cfg     may or may not already exist.  If nonexistent,
                         then basename_b.cfg will be created so that the
                         new files can be located, copied, moved, etc.

      The number of processors specified in  basename_a.cfg must equal
      the number of processors specified in  basename_b.cfg.
```

movemp

Use this command to move files that have been spread to the parallel file system.

```
    movemp [-v] basename_a.extension_a basename_b.extension_b; -v for verbose
    The optional -v flag is for verbose.
    Neither extension_a nor extension_b can contain periods ('.').
```

```
Moves (i.e. renames) files in directories (dir_a(n)) specified
 in the basename_a.cfg file to files in directories (dir_b(n))
 specified in the basename_b.cfg file.
The number of files to move is read from the basename_a.cfg  file,
  which must already exist.

  dir_a(n)/basename_a.extension_a.n --> dir_b(n)/
   basename_b.extension_b.n
          n=0,...,#processors-1

 basename_a.cfg     must already exist
 basename_b.cfg     may or may not already exist.  If nonexistent,
                    then basename_b.cfg will be created so that the
                    new files can be located, copied, moved, etc.

 The number of processors specified in  basename_a.cfg must equal
 the number of processors specified in  basename_b.cfg.
```

SEACAS Library
New Manuals
First Level 2 (Volume) Document
Other Information
Parallel Execution
< Go Back

combinemp

```
USAGE:  combinemp [-v] [-scp] [-ftp] [-nem_join] [-concat] basename
The optional -v flag is for verbose.
The optional -scp flag uses scp to get the files (default)
The optional -ftp flag uses ftp to get the files
The optional -nem_join flag use nem_join to combine the files (default)
The optional -concat flag use nem_join to combine the files
  Makes a basename.e.#proc file by scp'ing from remote machine the
     basename.e.#proc.n, n=0,...,#proc-1 files and concatenating them.
  This script requires the presence in the current directory
     of the basename.cfg file used to spread the files on the remote
    machine.
  You will be prompted for the name of the remote machine.
```