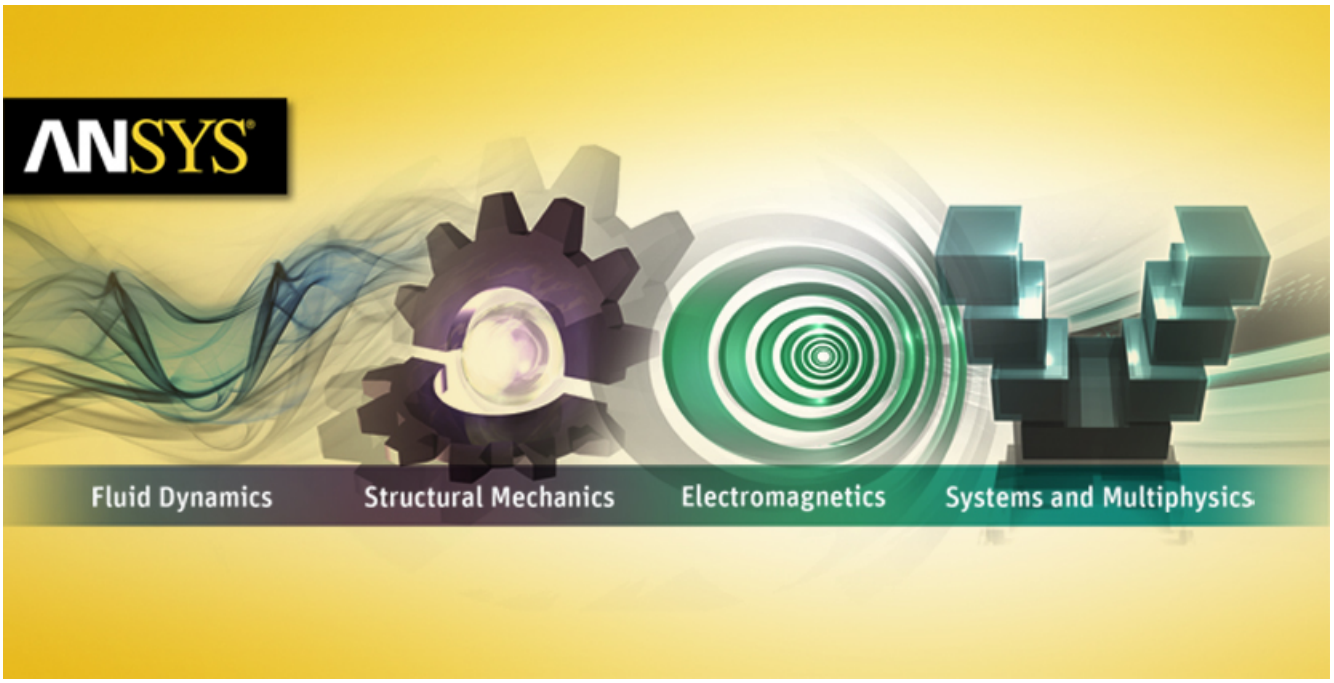


Running ANSYS FLUENT Under LSF

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 14.0
November 2011

ANSYS, Inc. is certified to ISO 9001:2008.

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

About This Document

This document provides general information about running ANSYS FLUENT under LSF. Examples have also been included, where available.

Information in this document is presented in the following chapters:

This document is made available via the ANSYS, Inc. website for your convenience. Please contact Platform Computing Inc. (`http://www.platform.com/`) directly for support of their product.

Chapter 1: Introduction

Platform Computing's LSF software is a distributed computing resource management tool that you can use with either the serial or the parallel version of ANSYS FLUENT. This document provides general information about running ANSYS FLUENT under LSF, and is made available via the ANSYS, Inc. website for your convenience. Please contact Platform directly for support of their product.

Using LSF, ANSYS FLUENT simulations can take full advantage of LSF checkpointing (i.e., saving ANSYS FLUENT `.cas` and `.dat` files) and migration features. LSF is also integrated with various MPI communication libraries for distributed MPI processing, increasing the efficiency of the software and data processing.

Important

Running ANSYS FLUENT under LSF is not supported on Windows.

Platform's Standard Edition is the foundation for all LSF products, it offers users load sharing and batch scheduling across distributed Linux and Windows computing environments. Platform's LSF Standard Edition provides the following functionality:

- comprehensive distributed resource management
  - provides dynamic load sharing services
  - allows for batch scheduling and resource management policies
- flexible queues and sharing control
  - prioritizes jobs
  - schedules jobs with load conditions
  - processes jobs with time windows
  - provides limits on the number of running jobs and job resource consumption
- fair-share scheduling of limited resources
  - manages shares for users and user groups
  - ensures fair sharing of limited computing resources
- maximum fault tolerance
  - provides batch service as long as one computer is active
  - ensures that no job is lost when the entire network goes down
  - restarts jobs on other compute nodes when a computer goes down

For more information, please see the following section:

1.1. Overview of ANSYS FLUENT and LSF Integration

1.1. Overview of ANSYS FLUENT and LSF Integration

For more information, please see the following sections:

1.1.1. Requirements

- LSF 6.0–7.0, available from Platform Computing at `http://www.platform.com/`
- FLUENT 6.0–ANSYS FLUENT 14.0

1.1.2. Optional Requirements

- The `echkpnt.fluent` and `erestart.fluent` binary files

  These files are available from Platform Computing or ANSYS, Inc., and permit ANSYS FLUENT checkpointing and restarting from within LSF.
- (For Linux) Hardware vendor-supplied MPI environment for network computing

1.1.3. Integration Components

The LSF components used by the ANSYS FLUENT integration are included in all versions of LSF packages that are able to be downloaded from the Platform ftp site (`ftp.platform.com`). If you are a current LSF customers, contact Platform support personnel for downloading instructions at `support@platform.com`. New LSF customers should contact the Platform sales department.

Chapter 2: Checkpointing and Restarting

LSF provides utilities to save (i.e., checkpoint), and restart an application. The ANSYS FLUENT and LSF integration allows ANSYS FLUENT to take advantage of the checkpoint and restart features of LSF. At the end of each iteration, ANSYS FLUENT looks for the existence of a checkpoint or checkpoint-exit file. If ANSYS FLUENT detects the checkpoint file, it writes a case and data file, removes the checkpoint file, and continues iterating. If ANSYS FLUENT detects the checkpoint-exit file, it writes a case file and data file, then exits. LSF's `bchkpnt` utility can be used to create the checkpoint and checkpoint-exit files, thereby forcing ANSYS FLUENT to checkpoint itself, or checkpoint and terminate itself. In addition to writing a case file and data file, ANSYS FLUENT also creates a simple journal file with instructions to read the checkpointed case file and data file, and continues iterating. ANSYS FLUENT uses that journal file when restarted with LSF's `brestart` utility. For more details on checkpointing features and options within ANSYS FLUENT, see the ANSYS FLUENT User's Guide.

The greatest benefit of the checkpoint facilities occurs when it is used on an automatic basis. By starting jobs with a periodic checkpoint, LSF automatically restarts any jobs that are lost due to host failure from the last checkpoint. This facility can dramatically reduce lost compute time, and also avoids the task of manually restarting failed jobs.

For more information, please see the following sections:

2.1. ANSYS FLUENT Checkpoint Files

In order to allow you to checkpoint ANSYS FLUENT jobs using LSF, LSF supplies special versions of `echkpnt` and `erestart`. These ANSYS FLUENT checkpoint files are called `echkpnt.fluent` and `erestart.fluent`.

2.2. Checkpoint Directories

When you submit a checkpointing job, you specify a checkpoint directory. Before the job starts running, LSF sets the environment variable `LSB_CHKPNT_DIR`. The value of `LSB_CHKPNT_DIR` is a subdirectory of the checkpoint directory specified in the command line. This subdirectory is identified by the job ID and only contains files related to the submitted job.

2.3. Checkpoint Trigger Files

When you checkpoint an ANSYS FLUENT job, LSF creates a checkpoint trigger file (`check`) in the job subdirectory, which causes ANSYS FLUENT to checkpoint and continue running. A special option is used to create a different trigger file (`exit`), to cause ANSYS FLUENT to checkpoint and exit the job. ANSYS FLUENT uses the `LSB_CHKPNT_DIR` environment variable to determine the location of checkpoint trigger files. It checks the job subdirectory periodically while running the job. ANSYS FLUENT does not perform any checkpointing unless it finds the LSF trigger file in the job subdirectory. ANSYS FLUENT removes the trigger file after checkpointing the job.

2.4. Restart Jobs

If a job is restarted, LSF attempts to restart the job with the `-restart` option appended to the original `fluent` command. ANSYS FLUENT uses the checkpointed data and case files to restart the process from that checkpoint point, rather than repeating the entire process.

Each time a job is restarted, it is assigned a new job ID, and a new job subdirectory is created in the checkpoint directory. Files in the checkpoint directory are never deleted by LSF, but you may choose to remove old files once the ANSYS FLUENT job is finished and the job history is no longer required.

Chapter 3: Configuring LSF for ANSYS FLUENT

LSF provides special versions of `echkpnt` and `erestart` called `echkpnt.fluent` and `erestart.fluent` to allow checkpointing with ANSYS FLUENT. You must make sure LSF uses these files instead of the standard versions.

To configure LSF 6.0+ for ANSYS FLUENT:

- Copy the `echkpnt.fluent` and `erestart.fluent` files to the `$LSF_SERVERDIR` for each architecture that is desired.
- When submitting the job from the command line, include the `-a fluent` parameter when specifying the checkpoint information (see *Submitting an ANSYS FLUENT Job from the Command Line* (p. 7) for details).

Important

Note that LSF includes an email notification utility that sends email notices to users when an LSF job has been completed. If a user submits a batch job to LSF and the email notification utility is enabled, LSF will distribute an email containing the output for the particular LSF job. When an ANSYS FLUENT job is run under LSF with the `-g` option, the email will also contain information from the ANSYS FLUENT console.

Chapter 4: Working with ANSYS FLUENT Jobs

Information in this chapter is provided in the following sections:

4.1. Submitting an ANSYS FLUENT Job from the Command Line

When submitting an ANSYS FLUENT job from the Linux line command using LSF, you can include certain LSF checkpointing parameters in the standard call to ANSYS FLUENT.

Submitting a batch job requires the `bsub` command. The syntax for the `bsub` command to submit an ANSYS FLUENT job is

`bsub [-R` *option*`]` [*checkpoint_info*] [*bsub_options*] `fluent` *solver_version* [*FLUENT_options*] `-lsf`

where

- `-R` *option* provides you with the following options:
  - If *option* is set to `fluent`, you are specifying that the ANSYS FLUENT application is only installed on certain hosts in the cluster and is a shared resource. When this is the case, an LSF administrator must specify "fluent" with the hosts on which ANSYS FLUENT can be executed (in `lsf.cluster.`*cluster_name*).
  - If *option* is set to `"select[`*solver_license_feature*`] rusage[`*parallel_license_feature*`]"` (where *solver_license_feature* represents the settings that provide the serial **ANSYS** solver license feature names, and *parallel_license_feature* represents the settings that provide the **ANSYS** parallel/HPC license feature names), your LSF job will be coupled with **ANSYS** licensing so that the job will wait in the queue until licenses are available. See *Coupling LSF Job Submissions and **ANSYS** Licensing* (p. 11) for all of the settings necessary for such coupling.
- *checkpoint_info* provides the parameters needed for checkpointing, and has the following syntax:

  `-a fluent -k "`*checkpoint_directory*`[`*duration*`]"`

  where
  - `-a fluent` ensures that LSF uses the correct versions of `echkpnt` and `erestart`.
  - `-k` is the LSF option that makes the job able to be checkpointed.
  - `checkpoint_directory` specifies the name of the checkpoint directory.
  - `duration` specifies the number of minutes between automatic checkpoints.
- *bsub_options* are options for the job submission. See the Platform LSF Reference guide for a complete list with descriptions.

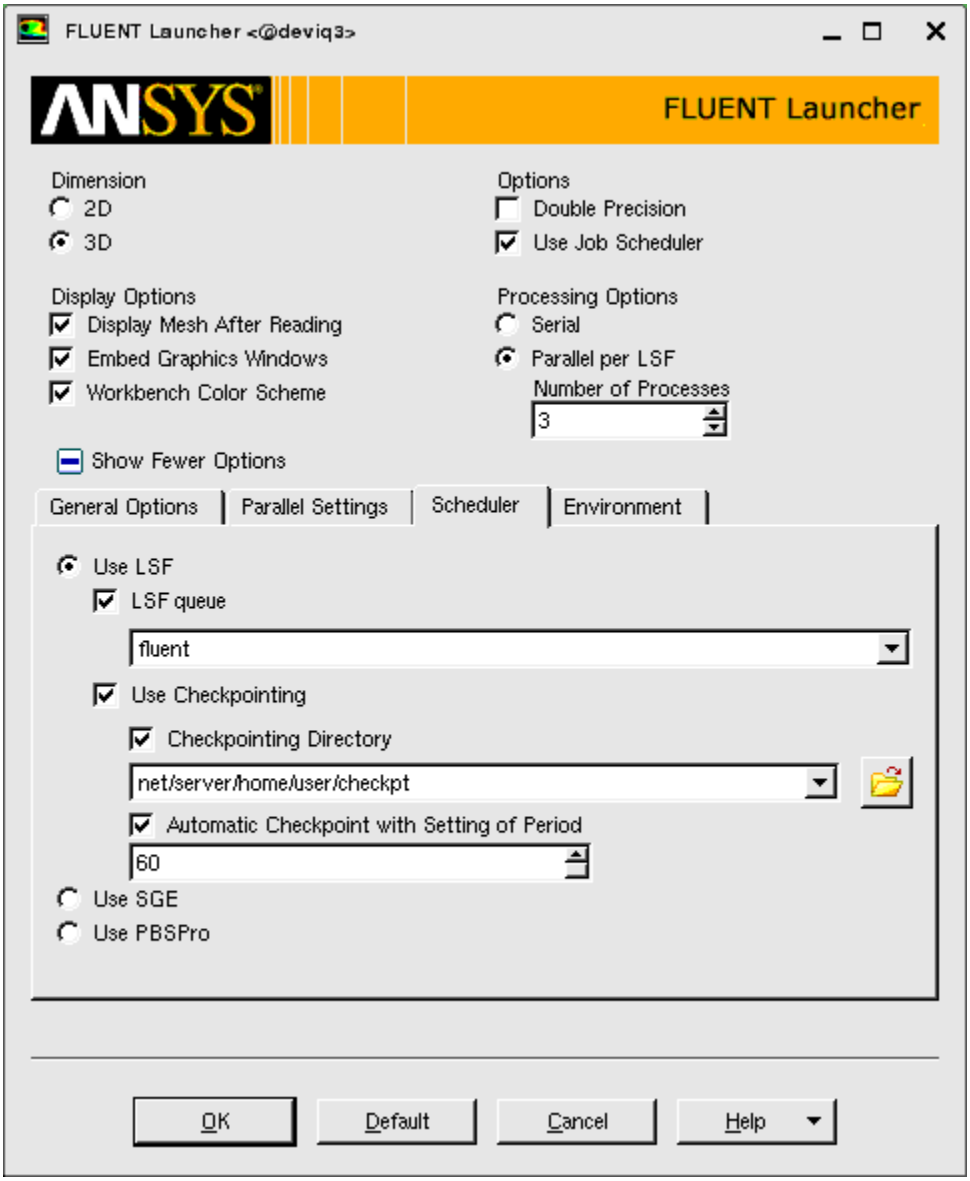
- `fluent` is the command that launches ANSYS FLUENT.
- *solver_version* specifies the dimensionality of the problem and the precision of the ANSYS FLUENT calculation (e.g., `3d`, `2ddp`).
- *FLUENT_options* can be added to specify the startup option(s) for ANSYS FLUENT, including the options for running ANSYS FLUENT in parallel. For more information, see the ANSYS FLUENT User's Guide.
- `-lsf` is added to the ANSYS FLUENT command to specify that you are running under LSF. This option causes ANSYS FLUENT to check for trigger files in the checkpoint directory if the environment variable `LSB_CHKPNT_DIR` is set.

4.2. Submitting an ANSYS FLUENT Job Using FLUENT Launcher

FLUENT Launcher has graphical user input options that allow you to submit an ANSYS FLUENT job using LSF. Perform the following steps:

1. Open FLUENT Launcher (*Figure 4.1* (p. 9)) by entering `fluent` without any arguments in the Linux command line.

Figure 4.1 The Scheduler Tab of FLUENT Launcher (Linux Version)



2. Enable **Use Job Scheduler** under **Options**.
3. Click the **Show More Options** button to expand FLUENT Launcher.
4. Click the **Scheduler** tab.
   a. Make sure that **Use LSF** is selected.
   b. To specify a job queue, enable the **LSF queue** option and enter the queue name in the text box.
   c. To utilize LSF checkpointing, enable the **Use Checkpointing** option. By default, the checkpointing directory will be the current working directory; you have the option of enabling **Checkpointing Directory** and specifying a different directory, either by entering the name in the text box or by browsing to it.

      You can specify that the checkpointing is done automatically at a set time interval by enabling the **Automatic Checkpoint with Setting of Period** option and entering the period (in minutes) in the text box; otherwise, checkpointing will not occur unless you call the `bchkpnt` command.

5. Set up the other aspects of your ANSYS FLUENT simulation using the FLUENT Launcher GUI items. For more information, see the ANSYS FLUENT User's Guide.

4.3. Manually Checkpointing ANSYS FLUENT Jobs

You can checkpoint a batch job manually by using the `bchkpnt` command. Note that you can set up automatic checkpointing using FLUENT Launcher as described previously. The syntax for the `bchkpnt` command is

`bchkpnt [`*bchkpnt_options*`] [-k]` *job_ID*

where

- *bchkpnt_options* are options for the job checkpointing. See the Platform LSF Reference guide for a complete list with descriptions.
- `-k` is the regular option to the `bchkpnt` command, and specifies checkpoint and exit. The job will be killed immediately after being checkpointed. When the job is restarted, it does not have to repeat any operations.
- *job_ID* is the job ID of the ANSYS FLUENT job, which is used to specify which job to checkpoint.

4.4. Restarting ANSYS FLUENT Jobs

Restarting a batch job requires the `brestart` command. The syntax for the `brestart` command is

`brestart [`*bsub_options*`]`*checkpoint_directory job_ID*

where

- *bsub_options* are options for the job restart. See the Platform LSF Reference guide for a complete list with descriptions.
- `checkpoint_directory` specifies the checkpoint directory where the job subdirectory is located.
- *job_ID* is the job ID of the ANSYS FLUENT job, and specifies which job to restart. At this point, the restarted job is assigned a new job ID, and the new job ID is used for checkpointing. The job ID changes each time the job is restarted.

4.5. Migrating ANSYS FLUENT Jobs

Migrating an ANSYS FLUENT job requires the `bmig` command. The syntax for the `bmig` command is

`bmig [`*bmig_options*`]` *job_ID*

where

- *bmig_options* are options for the job migration. See the Platform LSF Reference guide for a complete list with descriptions.
- *job_ID* is the job ID of the ANSYS FLUENT job, and specifies which job to checkpoint and restart on the migration target. At this point, the restarted job is assigned a new job ID, and the new job ID is used for checkpointing. The job ID changes each time the job is restarted.

4.6. Coupling LSF Job Submissions and ANSYS Licensing

You can couple your LSF job submission with **ANSYS** licensing by revising the LSF configuration files. Such a coupling will ensure that submitted jobs do not fail because licenses are not available. The jobs will be held in the queue until licenses are available. To set up such coupling, perform the following steps:

1. Copy the `elim` script from your **ANSYS** installation area to `$LSF_SERVERDIR`. The `elim` script is located in the following directory:

   path`/ansys_inc/v140/fluent/fluent14.0.0/multiport/mpi_wrapper/bin/`

   where *path* is the directory in which you have installed ANSYS FLUENT (e.g., `/opt/apps/`).

2. Edit the copy of the `elim` script to add your license server and license feature details. The following is an example where `acfd_fluent`, `acfd_par_proc`, and `anshpc` are **ANSYS** solver and parallel license features:

   ```
   $ENV{'ANSYSLMD_LICENSE_FILE'} = "1055\@deva12"
   my @features = qw(acfd_fluent acfd_par_proc anshpc);
   ```

3. Set the permissions to `755` and set `root` as the owner for the `elim` script.

4. Add all of your **ANSYS** solver license feature names and **ANSYS** parallel/HPC license feature names under `Resource section` in the file `lsf.shared`, which is located in `$LSF_ENVDIR` in your LSF installation area. The following is an example in which `acfd`, `acfd_fluent`, `acfd_solver`, and `acfd_fluent_solver` are the **ANSYS** solver license feature names and `anshpc`, `anshpc_pack`, and `acfd_par_proc` are the **ANSYS** parallel/HPC license feature names.

   ```
   acfd Numeric 20 N (available ANSYS Fluent Solver licenses)
   acfd_fluent Numeric 20 N (available ANSYS Fluent Solver licenses)
   acfd_solver Numeric 20 N (available ANSYS Fluent Solver licenses)
   acfd_fluent_solver Numeric 20 N (available ANSYS Fluent Solver licenses)
   anshpc Numeric 20 N (available ANSYS Fluent Parallel licenses)
   anshpc_pack Numeric 20 N (available ANSYS Fluent Parallel licenses)
   acfd_par_proc Numeric 20 N (available ANSYS Fluent Parallel licenses)
   ```

5. Add all of your **ANSYS** solver license feature names and **ANSYS** parallel/HPC license feature names in the file `lsf.cluster.`*cluster_name* (where *cluster_name* is the name of the cluster), which is located in `$LSF_ENVDIR` in your LSF installation area. The following is an example in which `acfd`, `acfd_fluent`, `acfd_solver`, and `acfd_fluent_solver` are the **ANSYS** solver license feature names and `anshpc`, `anshpc_pack`, and `acfd_par_proc` are the **ANSYS** parallel/HPC license feature names.

   ```
   # For LSF-ANSYS Licensing Coupling
   Begin ResourceMap
   RESOURCENAME LOCATION
   acfd ([all])
   acfd_fluent ([all])
   acfd_solver ([all])
   acfd_fluent_solver ([all])
   anshpc ([all])
   anshpc_pack ([all])
   acfd_par_proc ([all])
   End ResourceMap
   ```

6. Reconfigure the LSF daemons using the following commands, to specify that they reread their configuration. Note that you need administrator privileges to implement these changes.

   ```
   lsadmin reconfig
   ```

```
badmin reconfig
```

7. Submit the ANSYS FLUENT job using the "`select[`*solver_license_feature*`] rusage[`*parallel_license_feature*`]`" option, as described in *Submitting an ANSYS FLUENT Job from the Command Line* (p. 7). In this option, *solver_license_feature* has the following form:

   serial_lic_name>0

   where *serial_lic_name* is the name of the serial **ANSYS** solver license feature name. Similarly, *parallel_license_feature* has the following form:

   parallel_lic_name=N

   where *parallel_lic_name* is the name of the **ANSYS** parallel/HPC license feature name, and *N* is the number of processes to use.

   The previous descriptions are applicable when you have a single serial and/or parallel license feature. If you have multiple serial and/or parallel license features, you must add additional *solver_license_feature* and/or *parallel_license_feature* entries, separating them with `||`; additionally, you must enclose all of the *solver_license_feature* entries in a single pair of parentheses. The following is an example of submitting an ANSYS FLUENT job in which `acfd` and `acfd_fluent` are the **ANSYS** solver license feature names, `anshpc` and `acfd_par_proc` are the **ANSYS** parallel/HPC license feature names, and the number of processes to use is 4:

```
bsub -R "select[(acfd>0 || acfd_fluent>0)] rusage[anshpc=4 ||
acfd_par_proc=4]" -n 4 fluent 3d -t4 -peth -lsf
```

Chapter 5: ANSYS FLUENT and LSF Examples

This chapter provides various examples of running ANSYS FLUENT and LSF.

5.1. Examples Without Checkpointing

- Serial 3D ANSYS FLUENT interactive job under LSF

```
bsub -I fluent 3d -lsf
```

- Serial 3D ANSYS FLUENT batch job under LSF, which reads the journal file called `journal_file`

```
bsub fluent 3d -g -i journal_file -lsf
```

- Parallel 3D ANSYS FLUENT interactive job under LSF, on 4 CPUs

```
bsub -I -n 4 fluent 3d -t4 -lsf
```

> **Important**
>
> PAM is an extension of LSF that manages parallel processes by choosing the appropriate compute nodes and launching child processes. When using ANSYS FLUENT on Linux, PAM is not used to launch ANSYS FLUENT (so the JOB_STARTER argument of the LSF queue should not be set). Instead, LSF will set an environment variable that contains a list of N hosts, and ANSYS FLUENT will use this list to launch itself.

- Parallel 3D ANSYS FLUENT batch job under LSF, which uses 5 processes and reads the journal file called `journal_file`

```
bsub -n 5 fluent 3d -t5 -g -i journal_file -lsf
```

5.2. Examples with Checkpointing

The examples that follow apply to both interactive and batch submissions. For brevity, only batch submissions are described. Usage of the LSF checkpoint and restart capabilities, requiring `echkpnt` and `erestart`, are described as follows:

- Serial 3D ANSYS FLUENT batch job under LSF with checkpoint/restart

```
bsub -a fluent -k " /home/username 60" fluent 3d -g -i journal_file
-lsf
```

  - In this example, the LSF `-a fluent` specification identifies which `echkpnt`/`erestart` combination to use, `/home/username` is the checkpoint directory, and the duration between automatic checkpoints is 60 minutes.

The following commands can then be used:

– `bjobs -l` *job_ID*

→ This command returns the job information about *job_ID* in the LSF system.

– `bchkpnt` *job_ID*

→ This command forces ANSYS FLUENT to write a case file, a data file, and a restart journal file at the end of its current iteration.

→ The files are saved in a directory named *checkpoint_directory/job_ID*. The *checkpoint_directory* is defined in the original `bsub` command.

→ ANSYS FLUENT then continues to iterate.

– `bchkpnt -k` *job_ID*

→ This command forces ANSYS FLUENT to write a case file, a data file, and a restart journal file at the end of its current iteration.

→ The files are saved in a directory named *checkpoint_directory/job_ID* and then ANSYS FLUENT exits. The *checkpoint_directory* is defined in the original `bsub` command.

– `brestart` *checkpoint_directory job_ID*

→ This command starts an ANSYS FLUENT job using the latest case and data files in the *checkpoint_directory/job_ID* directory.

→ The restart journal file *checkpoint_directory/job_id/*`#restart.inp` is used to instruct ANSYS FLUENT to read the latest case and data files in that directory and continue iterating.

- Parallel 3D ANSYS FLUENT batch job under LSF with checkpoint/restart, which specifies `/home/username` as the checkpoint directory, uses 4 processes, and reads a journal file called `journal_file`

```
bsub -a fluent -k " /home/username" -n 4 fluent 3d -t4 -g -ijournal_file -lsf
```

The following commands can then be used:

– `bjobs -l` *job_ID*

→ This command returns the job information about *job_ID* in the LSF system.

– `bchkpnt` *job_ID*

→ This command forces parallel ANSYS FLUENT to write a case file, a data file, and a restart journal file at the end of its current iteration.

→ The files are saved in a directory named *checkpoint_directory/job_ID*. The *checkpoint_directory* is defined in the original `bsub` command.

→ Parallel ANSYS FLUENT then continues to iterate.

– `bchkpnt -k` *job_ID*

→ This command forces parallel ANSYS FLUENT to write a case file, a data file, and a restart journal file at the end of its current iteration.

→ The files are saved in a directory named *checkpoint_directory/job_ID*. The *checkpoint_directory* is defined in the original `bsub` command.

→ Parallel ANSYS FLUENT then exits.

– `brestart` *checkpoint_directory job_ID*

  - → This command starts an ANSYS FLUENT network parallel job using the latest case and data files in the *checkpoint_directory/job_ID* directory.
  - → The restart journal file *checkpoint_directory/job_ID/#restart.inp* is used to instruct ANSYS FLUENT to read the latest case and data files in that directory and continue iterating.
  - → The parallel job will be restarted using same number of processes as that used for the original `bsub` submission (4 in the previous example).

- `bmig -m` *host* `0`
  - → This command checkpoints all jobs (indicated by `0` job ID) for the current user and moves them to host *host*.

Index