Using Platform **LSF®** Make

November 2003
Platform Computing
Comments to: doc@platform.com

LSF Make is a load-sharing, parallel version of GNU Make. It uses the same makefiles as GNU Make and behaves similarly, except that additional command line options control parallel execution.

LSF Make allows you to use your Platform LSF cluster to run parts of your make in parallel. Tasks are started on multiple hosts simultaneously to reduce the execution time.

Platform LSF Make is only supported on UNIX.

Platform LSF is a prerequisite for LSF Make. The Platform LSF Make product is sold, licensed, distributed, and installed separately. For more information, contact Platform Computing.

The LSF Make executable, `lsmake`, is covered by the Free Software Foundation General Public License. Read the file `LSF_MISC/lsmake/COPYING` in the Platform LSF software distribution for details.

Contents

Platform™

About Platform LSF Make

Tasks often consist of many subtasks, with some dependencies between the subtasks. For example, to compile a software package, you compile each file in the package, then link all the compiled files together.

In many cases, most of the subtasks do not depend on each other. For a software package, the individual files in the package can be compiled at the same time; only the linking step needs to wait for all the other tasks to complete.

LSF Make allows you to use your Platform LSF cluster to run parts of your make in parallel. Tasks are started on multiple hosts simultaneously to reduce the execution time.

LSF Make is a load-sharing, parallel version of GNU Make. It uses the same makefiles as GNU Make and behaves similarly, except that additional command line options control parallel execution.

GNU Make compatibility

LSF Make is based on GNU Make and supports all GNU Make features. GNU Make is upwardly compatible with the make programs supplied by most UNIX vendors. LSF Make is compatible with makefiles for most versions of GNU Make.

LSF Make is fully compatible with GNU Make version 3.77. There are some incompatibilities between GNU Make and some other versions of make; these are beyond the scope of this document.

How Platform LSF Make Works

LSF Make is invoked using the `lsmake` command.

For command syntax and complete information about command line options that control load sharing, see the `lsmake(1)` man page.

lsmake command

The following examples show how to build your software in parallel and control the execution hosts used, the number of processors used, and the number of tasks run simultaneously on one processor.

```
% lsmake -f mymakefile
```

`lsmake` uses one processor on the submission host, and runs one task at a time (one task per processor). This is the default behavior.

```
% lsmake -R "swp > 50 && mem > 100" -f mymakefile
```

`lsmake` uses one processor, on the best available host that satisfies the specified resource requirements, and runs one task at a time. If there are no eligible hosts, the job fails.

By default, LSF Make selects the same host type as the submitting host. This is necessary for most compilation jobs; all components must be compiled on the same host type and operating system version to run correctly. If your make task requires other resources, override the default resource requirements with `-R`.

```
% lsmake -V -j 3 -f mymakefile
[hostA] [hostD] [hostK]
<< Execute on local host >>
cc -O -c arg.c -o arg.o
<< Execute on remote host hostA >>
cc -O -c dev.c -o dev.o
<< Execute on remote host hostK >>
cc -O -c main.c -o main.o
<< Execute on remote host hostD >>
cc -O arg.o dev.o main.o
```

`lsmake` uses 3 processors, on hosts that are the same host type as the submission host. Use `-V` to return output as shown, including the names of the execution hosts. Use `-j` to specify a maximum number of processors.

If 5 processors are eligible, LSF Make automatically selects the best 3.

If only 2 processors are eligible, LSF Make uses only 2 processors. At least one processor is always eligible because the submission host always meets the default requirement.

```
% lsmake -R "swp > 50 && mem > 100" -j 3 -c 2 -f mymakefile
```

`lsmake` uses up to 3 processors, on the best available hosts that satisfy the specified resource requirements, and starts 2 tasks on each processor. If there are no eligible hosts, the job fails.

Use `-c` to take advantage of parallelism between the CPU and I/O on a powerful host and specify the number of concurrent jobs for each processor.

```
% lsmake -m "hostA hostA hostB" -f mymakefile
```

`lsmake` uses 2 processors on hostA and one processor on hostB, and runs one task per processor. Use `-m` to specify exactly which hosts to use.

Using GNU make options

LSF Make supports all the GNU Make command line options. See the `gmake(1)` man page.

Resetting environment variables

By default, LSF Make sets the environment variables on the execution hosts once, when you run `lsmake`. If your tasks overwrite files or environment variables during execution, use `-E` to automatically reset the environment variables for every task that executes on a remote host.

Running interactive tasks

When LSF Make is running processes on more than one host, it does not send standard input to the remote processes. Most makefiles do not require any user interaction through standard I/O. If you have makefile steps that require user interaction, put the commands that require interaction into your local task list. Commands in the local task list always run on the local host, where they can read from standard input and write to standard output.

Running lsmake under LSF

Make jobs often require a lot of resources, but no user interaction. Such jobs can be submitted to LSF so that they are processed when the needed resources are available. The command `lsmake` includes extensions to run as a parallel batch job under LSF:

```
% bsub -n 10 lsmake
```

This command queues an LSF Make job that needs 10 hosts. When all 10 hosts are available, LSF starts LSF Make on the first host, and passes the names of all hosts in an environment variable. LSF Make gets the host names from the environment variable and uses RES to run tasks.

You can also specify a minimum and maximum number of processors to dedicate to your make job:

```
% bsub -n 6,18 lsmake
```

Because LSF Make passes the suspend signal (SIGTSTP) to all its remote processes, the entire parallel make job can be suspended and resumed by the user or by LSF.

Performance Issues

Ways to improve the performance of LSF Make:

- Tune your makefile and increase parallelism
- Process subdirectories in parallel
- Adjust the number of tasks run depending on the file server load
- Ensure tasks always run on the best processors available at the time

Reorganizing your makefile

You do not need to modify your makefile to use LSF Make, but reorganizing the contents of the makefile to increase the parallelism might reduce the running time.

The smallest unit that LSF Make runs in parallel is a single make rule. If your makefile has rules that include many steps, or rules that contain shell loops to build sub-parts of your project, LSF Make runs the steps serially.

Increase the parallelism in your makefile by breaking up complex rules into groups of simpler rules. Steps that must run in sequence can use make dependencies to enforce the order. LSF Make can then find more subtasks to run in parallel.

Building recursive makes

LSF Make includes control over parallelism for recursive makes, which are often used for source code trees that are organized into subdirectories.

If your make job is divided into subdirectories, `-M` allows you to process the subdirectories in parallel. The total number of parallel tasks is shared over all the subdirectories. Without `-M` , LSF Make processes subdirectories sequentially, although tasks within each subdirectory can be run in parallel.

To process subdirectories in parallel they must be built as separate targets in your makefile. You must specify the `make` command for each subdirectory with the built-in `$(MAKE)` macro so that LSF Make can substitute the correct `lsmake` command for the subdirectory.

Some makefiles may work correctly when run on a single machine, but may not work correctly when run in parallel through LSF Make.

Below is a makefile rule that uses a shell loop to process subdirectories.

```
DIRS = lib misc main
prog:
    for subdir in $(DIRS) ; do \
        cd $${subdir} ; $(MAKE) ; cd .. ; done
```

When this makefile is run on a single machine, the directories are processed sequentially; in other words, `lib` is built before `misc` and `main`. However, when run using `lsmake -M`, all directories can be built in parallel. Therefore, it is possible for the `misc` and `main` directories to be built before `lib`, which is not correct.

Below is a set of makefile rules to perform the same tasks and allows the subdirectories to be built in parallel in the correct order. An extra rule is added so that the `lib` and `misc` subdirectories are built before the main directory:

```
DIRS = lib misc main
prog:  $(DIRS)
$(DIRS):
       cd $@ ; $(MAKE)
```

Dynamic parallelism

LSF Make can significantly reduce the response time of your make; however, it may also overload your file server or network if the tasks you are running are I/O intensive.

Parallelism can be controlled by the load on the NFS file server, so that parallel makes do not overload the server and slow everyone else down.

You can specify a threshold load so that parallelism is automatically reduced, when the file server load is above a threshold, and expanded, when the file server load is below the threshold.

```
% lsmake -j 10 -F "r15s < 5 && pg < 20"
```

`lsmake` uses up to 10 processors, and reduces the parallelism if the file server CPU load `r15s` goes beyond 5, or if the file server paging rate goes beyond 20 pages per second.

LSF Make automatically determines the file server for the current working directory.

Processor reselection

LSF Make selects the best available hosts to run tasks. Over time, the values of dynamic resources change, so the original best host does not neccessarily remain the best host for the entire duration of a long-running task.

To ensure that your tasks always run on the best available hosts, use `-P` to automatically reselect the execution hosts.

```
%lsmake -j 3 -P 90 -f mymakefile
```

`lsmake` uses 3 processors and then evaluates eligible hosts at regular 90-minute intervals, until the make is finished. If a processor currently in use can be replaced with a better one, LSF Make stops using the original processor and starts using the better processor.

Technical Support

Contacting Platform

Contact Platform Computing or your LSF Make vendor for technical support. Use one of the following to contact Platform support:

Email support@platform.com

World Wide Web www.platform.com

Phone

- North America: +1 905 948 4297
- Europe: +44 1256 370 530
- Asia: +86 10 6238 1125

Toll-free phone 1-877-444-4LSF (+1 877 444 4573)

Mail

Platform Support
Platform Computing
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

When contacting Platform Computing, please include the full name of your company.

We'd like to hear from you

If you find an error in any Platform documentation, or you have a suggestion for improving it, please let us know:

Email doc@platform.com

Mail

Information Development
Platform Computing
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

Be sure to tell us:

- The title of the manual you are commenting on
- The version of the product you are using
- The format of the manual (HTML or PDF)

Copyright

Last update November 13 2003

Latest version www.platform.com/services/support/docs_home.asp