IBM® Platform LSF®

Best practices

Using MPI under IBM Platform LSF

Zhaohui Ding
LSF Development
Systems & Technology Group

Haibin Wang
LSF Development
Systems & Technology Group

Jin Ma
LSF Development
Systems & Technology Group

Chong Chen
LSF Development
Systems & Technology Group

Issued: October 2013

Executive Summary

Much of the workload in High Performance Computing environment is generated by MPI applications. There are various implementations of MPI standards, such as IBM Parallel Environment Runtime Edition, Platform MPI, Intel MPI, Open MPI, MPICH2 and MVAPICH2.

IBM Platform LSF (LSF) is a powerful workload scheduling and management tool for HPC environments. LSF provides a flexible, scalable and extensible distributed application integration framework called *blaunch*, with which LSF integrates most popular MPI implementations. With LSF, users can allocate resources, schedule, launch, control MPI jobs and collect MPI job resource usage.

This document presents guidelines for running MPI jobs under LSF and applies to LSF 9.1.1 with service pack 9.1.1.1 or later releases.

Introduction

This document serves as a best practice guide for how to use various MPI implementations under LSF. This document describes the following:

- How to run various MPI applications under LSF
  - IBM Parallel Environment Runtime Edition (IBM PE)
  - IBM Platform MPI
  - MPICH2
  - MVAPICH2
  - Intel MPI
  - Open MPI
- How to run Open MP jobs under LSF
- LSF tuning parameters for parallel jobs
- Test conditions of LSF MPI integrations

How to run MPI jobs under LSF

IBM Parallel Edition Runtime Edition

In version 9.1.1, LSF integrates with the IBM Parallel Environment Runtime Edition (IBM PE) product (version 1.3 or later) to run IBM PE jobs through the IBM Parallel Operating Environment (IBM POE). The integration enables network-aware scheduling, allowing an LSF job to specify network resource requirements, collect network information, and schedule the job according to the requested network resources.

IBM PE jobs can be submitted through `bsub`, and monitored and controlled through LSF commands. To enable the LSF integration with IBM PE, the LSF administrator must define the parameter LSF_PE_NETWORK_NUM=*num_network* in `lsf.conf`, and restart LSF. *num_network* is an integer that represents the number of InfiniBand (IB) networks in the cluster.

Use `bhosts -l` to check IBM PE network information collected by LSF. In the following example, the section (PE NETWORK INFORMATION) highlighted in red represents network information.

```
$ bhosts -l ibmx02
HOST  ibmx02
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
DISPATCH WINDOW
ok              60.00     -     40     15     15      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem
slots
 Total           0.0   0.1   0.8   23%   0.0    47    7     0   62G    4G  116G
25
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M
-

 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 PE NETWORK INFORMATION:
                NetworkID            Status     rsv_windows/total_windows
                    33077                ok                         0/256
                    33088                ok                         0/256
```

Network requirements can be specified at job submission with the `bsub -network` option, and configured at the queue (`lsb.queues`) or application level (`lsb.applications`) with the NETWORK_REQ parameter.

The `-network` option and NETWORK_REQ parameter has the following syntax:

```
[type=sn_all | sn_single]
[:protocol=protocol_name[(protocol_number)][,protocol_name[(protocol_number)]]
[:mode=US | IP] [:usage=shared | dedicated] [:instance=positive_integer]
```

The following is an IBM PE job submission with `bsub`:

```
$ bsub -n 12 -network "type=sn_all:protocol=mpi:usage=shared:mode=us:instance=4" -
R "span[ptile=6]" -o /scratch/project/%J.out -e /scratch/project/%J.err poe
mypoe_prog
```

This IBM PE job requests 12 slots in total and 6 slots for each host. For network requirements, the job needs to reserve network windows from all networks, using MPI as the communication protocol in `shared` and `user space` mode. The job reserves 4 windows per network per task.

You can also create a submission script (for example, `poejob.lsf`), and use `bsub` to submit the script:

```
$ cat poejob.lsf
#!/bin/sh
#BSUB -n 12
#BSUB -network "type=sn_all:protocol=mpi:usage=shared:mode=us:instance=4"
#BSUB -R "span[ptile=6]"
#BSUB -o /scratch/project/%J.out
#BSUB -e /scratch/project/%J.err

poe mypoe_prog

$ bsub < poejob.lsf
```

The two submission methods are equivalent.

Use `bjobs` to check job status and resource usage information. For example:

```
$ bjobs -l 5

Job <5>, User <user1>, Project <default>, Status <RUN>, Queue <normal>, Comma
                     nd <poe mypoe_prog>
Thu Jan 10 03:23:29: Submitted from host <ibmx01>, CWD </home/user1>, 12
                     Processors Requested, Requested Network
                     <type=sn_all:protocol=mpi:mode=US:usage=shared:instance=4>;
Thu Jan 10 03:28:18: Started on 12 Hosts/Processors <6*ibmx01> <6*ibmx02>, Exec
                     ution Home </home/user1>, Execution CWD </home/user1>, PE
                     Network ID <33077> <33088> used <4> window(s) per network per
                     task;
Thu Jan 10 03:28:42: Resource usage collected.
                     MEM: 3 Mbytes;  SWAP: 0 Mbytes;  NTHREAD: 4


 MEMORY USAGE:
 MAX MEM: 3 Mbytes;  AVG MEM: 2 Mbytes

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == local] order[r15s:pg]
 Effective: select[type == local] order[r15s:pg]
```

From the output, you can see that LSF allocated 6 slots on host `ibmx01` and `ibmx02`, and allocated 4 windows on the following networks: `33077` and `33088`.

Use `bhosts -l` to check the network reservation on hosts:

```
$ bhosts -l ibmx02
HOST  ibmx02
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
ok               60.00    -     40     15     15      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           0.0   0.1   0.8   23%   0.0    47    7     0   62G    4G  116G     25
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -

 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 PE NETWORK INFORMATION:
                NetworkID            Status     rsv_windows/total_windows
                    33077                ok                        24/256
                    33088                ok                        24/256
```

From the output, you can see 24 (6*4) windows are reserved by LSF for each network.

NOTE: LSF and the IBM PE V1.3 integration is only supported on Linux (both X86 Linux and Power Linux).

NOTE: After installation, the LSF `hostsetup` script creates a symbol link under `/usr/lib64/libpermapi.so` to `$LSF_LIBDIR/permapi.so` on each compute node. After running `hostsetup`, you should make sure that the symbol link is created successfully.

The LSF integration with IBM PE also allows you to specify task geometry for flexibility in how tasks are grouped and launched for execution on system nodes. The LSF task geometry is specified via environment variable LSB_TASK_GEMOETRY.

For example, create a submission script (for example, `poejob2.lsf`), and use `bsub` to submit the script:

```
$ cat poejob2.lsf
#!/bin/sh
#BSUB -n 12
#BSUB -network "type=sn_all"
#BSUB -R "span[ptile=3]"
#BSUB -o /scratch/project/%J.out
#BSUB -e /scratch/project/%J.err
export LSB_TASK_GEOMETRY="{0,5,7}{2,6}{1,3}{4}"
poe mypoe_prog

$ bsub < poejob2.lsf
```

This job to will spawn 8 tasks and span 4 nodes, tasks 0, 5, and 7 run on one node, tasks 2 and 6 run on another node, tasks 1 and 3 run on a third node, and task 4 runs on one node alone. These 8 tasks are launched in task ID order.

Platform MPI

Platform MPI is integrated with LSF through the `blaunch` API. The recommended version is 9.1.2 or later. To use Platform MPI under LSF, create the IBM LSF job and include the `-lsf` flag with the `mpirun` command.

With the `-lsf` flag specified, Platform MPI reads the $LSB_MCPU_HOSTS environment variable set by IBM LSF and uses this information to start an equal number of ranks as allocated slots. The IBM LSF `blaunch` command starts the remote execution of ranks and administrative processes instead of `ssh`.

Use `bsub` to submit a Platform MPI job:

```
$ export PATH=/opt/mpi/pmpi/bin:$PATH
$ bsub -n 1024 -e pmpi_%J.err -o pmpi_%J.out -R "span[ptile=16]" mpirun -lsf
mypmpi_prog
```

The following is the equivalent job script submission:

```
$ cat pmpijob.lsf
#!/bin/sh
#BSUB -n 1024
#BSUB -e pmpi_%J.err
#BSUB -o pmpi %J.out
#BSUB -R "span[ptile=16]"
export PATH=/opt/mpi/pmpi/bin:$PATH
mpirun -lsf pmpi_prog

$ bsub < pmpijob.lsf
```

This job script asks for 1024 slots in total, 16 slots per host, and redirects `stdout` and `stderr` to files.

Intel MPI

Intel MPI is also derived from MPICH, and also integrates with LSF via the Hydra process manager.

Based on version 4.1.1.036, Intel MPI provides a Hydra patch that uses the `blaunch -z` option. It only starts one `blaunch` process on the head node, no matter how many execution hosts are allocated. The integration in this patch is more stable and scalable than multiple `blaunch` running in the background. So IntelMPI 4.1.1.036 with this patch is recommended if you are planning to run large scale jobs.

TIP: You can download the latest Intel MPI from http://software.intel.com/en-us/intel-mpi-library.Download the installation manual from http://software.intel.com/en-us/articles/intel-mpi-library-documentation

To submit an Intel MPI job, create a submission script (for example, `inteljob.lsf`):

```
#!/bin/sh
#BSUB -n 1024
#BSUB -e intelmpi %J.err
#BSUB -o intelmpi_%J.out
#BSUB -R "span[ptile=16]"
export INTELMPI_TOP=/opt/mpi/intelmpi/impi/4.1.1.036
export PATH=$INTELMPI_TOP/bin:$PATH
export I_MPI_HYDRA_BOOTSTRAP=lsf
export I_MPI_HYDRA_BRANCH_COUNT=64 #64 is number of hosts, i.e., 1024/16
export I_MPI_LSF_USE_COLLECTIVE_LAUNCH=1
mpiexec.hydra intelmpi_program
```

The following Intel MPI environment variables are recommended to make large parallel jobs run successfully:

- **I_MPI_HYDRA_BOOTSTRAP** sets the Intel MPI bootstrap server. `lsf` means to use LSF `blaunch`.
- **I_MPI_HYDRA_BRANCH_COUNT** sets the Intel MPI hierarchical branch count. You should set to the number of hosts to disable hierarchical launching.
- **I_MPI_LSF_USE_COLLECTIVE_LAUNCH** environment variable introduced in the 4.1.1.036 patch. Set to 1 to let Intel MPI use `blaunch -z` to launch tasks.

This job script requests 1024 slots in total, 16 slots per host, redirects `stdout` and `stderr` to files, and exports the necessary environment variables used by Intel MPI.

Use `bsub` to submit the job:

```
$ bsub < inteljob.lsf
```

Open MPI

Open MPI is integrated with LSF through the `blaunch` API. . The recommended version is 1.6.x or later.

TIP: You can download Open MPI 1.6.x from http://www.open-mpi.org/software/ompi/v1.6/. Get the installation manual from http://www.open-mpi.org/doc/current/.

NOTE: When building OpenMPI, pass the `-with-lsf` option to the *`configure`* script.

To submit an Open MPI job, create a submission script (for example, `openmpijob.lsf`):

```
#! /bin/sh
#BSUB -n 128
#BSUB -e openmpi_%J.err
#BSUB -o openmpi_%J.out
#BSUB -R "span[ptile=8]"
export PATH="/opt/mpi/openmpi/bin:$PATH"
export LD_LIBRARY_PATH="/opt/mpi/openmpi/lib:$LD_LIBRARY_PATH"
mpirun openmpi_prog
```

This job script asks for 128 slots in total, 8 slots per host, redirects `stdout` and `stderr` to files, and exports the necessary environment variables used by Open MPI.

Use `bsub` to submit the job:

```
$ bsub < openmpijob.lsf
```

MPICH2

MPICH2 integrates with LSF via the Hydra process manager. Hydra process manager starts up `blaunch` for each allocated execution host to launch *hydra_pmi_proxy*. If *N* execution hosts are allocated to the job, hydra process manager will run in background and start *N-1* `blaunch` processes on the head node.

TIP: You can download MPICH2 from http://www.mcs.anl.gov/research/projects/mpich2staging/goodell/downloads/index.php?s=downloads. Download the installation manual from http://www.mcs.anl.gov/research/projects/mpich2staging/goodell/documentation/index.php?s=docs

```
#!/bin/sh
#BSUB -n 16
#BSUB -e mpich2_%J.err
#BSUB -o mpich2_%J.out
#BSUB -R "span[ptile=8]"

export PATH=/usr/local/mpich2-install/bin:$PATH
mpiexec.hydra mpich2_program
```

This job script asks for 16 slots in total, 8 slots per host, redirects stdout and `stderr` to files, and exports the necessary environment variables used by MPICH2.

Use `bsub` to submit the job:

```
$ bsub < mpich2job.lsf
```

MVAPICH2

MVAPICH2 is an MPI implementation for InfiniBand networks, it is derived from MPICH. It is integrated with LSF via the Hydra process manager.

TIP: You can download the MVAPICH2 package and install manual from http://mvapich.cse.ohio-state.edu/download/mvapich2/download.php.

To submit an MVAPICH2 job, create a submission script (for example, `mvapich2job.lsf`):

```
#!/bin/sh
#BSUB -n 16
#BSUB -e mpich2_%J.err
#BSUB -o mpich2 %J.out
#BSUB -R "span[ptile=8]"
export PATH=/usr/local/mvapich2-install/bin:$PATH
mpiexec.hydra mvapich2_program
```

This job script asks for 16 slots in total, 8 slots per host, redirects `stdout` and `stderr` to files, and exports the necessary environment variables used by MPICH2.

Use `bsub` to submit the job:

```
$ bsub < mvapich2job.lsf
```

How to run OpenMP jobs

Platform LSF provides the ability to start parallel jobs that use OpenMP to communicate between processes on shared-memory machines and MPI to communicate across networked and non-shared memory machines.

You can set the environment variable OMP_NUM_THREADS manually, or let LSF set OMP_NUM_THREADS automatically with an affinity scheduling resource requirement.

In this section, three LSF job script examples are shown. Examples 1 and 2 are for pure openMP jobs. Example 3 is for an MPI and OpenMP hybrid job.

Example 1

```
#! /bin/sh
#BSUB -n 4
#BSUB -R "span[hosts=1]"
export OMP_NUM_THREADS=4
openMP_prog
```

This job script requests 4 slots on one host and exports OMP_NUM_THREADS manually.

Example 2

```
#! /bin/sh
#BSUB -R "affinity[core(4)]"
openMP_prog
```

This job script requests 1 slot and binds to 4 cores on same host. Because an affinity scheduling resource requirement is specified, LSF sets OMP_NUM_THREADS to (*num_processor_unit* * *num_subtask*). In this example, *num_processor_unit* is 4, and *num_subtask* is 1.

Example 3

```
#! /bin/sh
#BSUB -n 4
#BSUB -R "affinity[core(2)]"
#BSUB -R "span[ptile=2]"

poe mypoe_prog
```

This job script requests 2 hosts to start 4 IBM POE tasks in total. Each host starts 2 tasks, and binds each task to 2 cores. OMP_NUM_THREADS is set to (*num_processor_unit* * *num_subtask*). In this example, *num_processor_unit* is 2, and *num_subtask* is 1.

Tuning parameters for LSF parallel jobs

A parallel job will run across multiple hosts. The LSF `blaunch` mechanism provides a framework to cover the full life cycle of parallel workload, including starting, monitoring, collecting resource usage and cleaning up left over processes in normal and abnormal situations. The network and host conditions in the cluster may impact how LSF starts and monitors parallel jobs. LSF default settings are selected to run normal production parallel jobs. Very large scale parallel jobs may require some special tuning. The following parameters can be adjusted to make sure large-scale parallel jobs can run successfully and achieve better performance.

- **LSF_DJOB_TASK_REG_WAIT_TIME**

This parameter defines the time period for LSF to wait for all parallel job task starting registration messages. If the parameter is not specified, the time out is calculated based on dynamic conditions, typically around 1 to 1.5 minutes.

- **LSB_DJOB_RU_INTERVAL**

This parameter defines the time interval that LSF reports parallel job resource usage on each execution host. The default value is calculated with the following formula. The result is in seconds.

`MAX(60, number_of_execution_hosts * 0.3)`

You can also disable parallel job resource usage updates by setting the parameter to 0.

- **LSB_DJOB_HB_INTERVAL**

This parameter defines the time interval that the remote execution tasks send heartbeat messages to the head node. The default value is calculated with the following formula. The result is in seconds. LSF uses heartbeat to determine whether execution hosts of the parallel job is available and takes clean up action if the execution node has been identified as unavailable.

`MAX(60, number_of_execution_hosts * 0.12).`

Heartbeat message sending cannot be disabled.

- **LSB_FANOUT_TIMEOUT_PER_LAYER**

This parameter defines the communication timeout when LSF sets up the execution environment on allocated hosts for the job. The default value is 20 seconds. It should be increased if network latency is high or for a large parallel job.

You can export these parameters as environment variables, then submit the job. For example:

```
$ export LSB_FANOUT_TIMEOUT_PER_LAYER=300
$ export LSF_DJOB_TASK_REG_WAIT_TIME=600
$ export LSB_DJOB_RU_INTERVAL=0
$ export LSB_DJOB_HB_INTERVAL=300
$ bsub < inteljob.lsf
```

NOTE: These environment variables must be exported before running `bsub`. They do not take effect if you export them in job script.

LSF administrators can also configure these parameters in `lsf.conf` to enable the parameters for all jobs. Restart `sbatchd` and RES on all compute hosts to make `lsf.conf` changes take effect.

```
LSB_FANOUT_TIMEOUT_PER_LAYER=300
LSF_DJOB_TASK_REG_WAIT_TIME=600
LSB_DJOB_RU_INTERVAL=0
LSB_DJOB_HB_INTERVAL=300
```

Test conditions

Various LSF MPI integrations have been tested under different job scale conditions. IBM PE under LSF has been tested up to an 80000-way job (5000 hosts * 16 tasks). Intel MPI under LSF has been tested up to a 16000-way job (1000 hosts * 16 tasks). Open MPI under LSF has been tested up to an 8192-way job (512 hosts * 16 tasks).

NOTE: Test results for reference only. Results in your environment may differ, and depend on your MPI application behavior and network conditions.

Best practices

This document has described best practices for the following:

- Using `bsub -network` and `poe` to submit IBM PE jobs.
- Using `bsub` and `mpirun -lsf` to submit PMPI jobs.
- Using `bsub` and Intel MPI script to submit Intel MPI jobs.
- Using `bsub` and Open MPI script to submit Open MPI jobs.
- Using `bsub` and `mpiexec.hydra` to submit MPICH2 jobs.
- Using `bsub` and `mpiexec.hydra` to submit MVAPICH2 jobs.
- Running Open MP jobs under LSF.
- Tuning parallel jobs using LSF parameters.

Conclusion

This document describes how to run MPI workload under IBM Platform LSF, including IBM PE, Platform MPI, MPICH2, MVAPICH2, Intel MPI, and Open MPI. It also introduces common LSF tuning parameters for parallel jobs, and gives the scalability test conditions for MPI workload.

Further reading

- *Administering Platform LSF* Version 9 Release 1.1
  http://publibfp.dhe.ibm.com/epubs/pdf/c2753021.pdf
  - *Running Parallel Jobs*
- *IBM Parallel Environment Runtime Edition for Linux: Operation and Use*
  http://publib.boulder.ibm.com/epubs/pdf/c2367818.pdf

Contributors

Zhaohui Ding
LSF Architect

Haibin Wang
LSF Developer

Jin Ma
LSF Developer

Chong Chen
LSF Principal Architect

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Without limiting the above disclaimers, IBM provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any recommendations or techniques herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Anyone attempting to adapt these techniques to their own environment does so at their own risk.

This document and the information contained herein may be used solely in connection with the IBM products discussed in this document.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE: © Copyright IBM Corporation 2013. All Rights Reserved.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Contacting IBM

To provide feedback about this paper, contact dingzhh@cn.ibm.com.

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide.

To learn more about IBM Information Management products, go to http://www.ibm.com/software/data/.