Помагало за компилиране, свързване и пускане на програми на суперкомпютър Blue Gene/P

1 Система Modules

Системата Modules позволява лесна промяна на променливите на обкръжението (environment variables) посредством така наречените модулни файлове.

Следва кратко описание на основните потребителски команди.

`module help`

Дава кратка помощ за употребата на Modules.

`module avail`

Показва модулите, които са на разположение на системата.

`module whatis име_на_модул`

Дава кратко описание на модула.

`module show име_на_модул`

Показва съдържанието на модулния файл.

`module list`

Показва заредените в момента модули.

`module load име_на_модул`

Зарежда модул.

`module unload име_на_модул`

Премахва модул.

`module purge`

Премахва всички модули.

2 Компилиране и свързване с библиотеките

2.1 ParMETIS

Нека имаме програма, написана на C++, във файл `test.cpp`. За да я компилираме и свържем с библиотеката ParMETIS, използваме следните команди:

```
module load ParMetis
mpixlcxx test.cpp -I$PARMETIS_INCLUDE -L$PARMETIS_LIB \
        -lparmetis -lmetis
```

2.2 Hypre

Нека имаме програма, написана на C++, във файл `test.cpp`. За да я компилираме и свържем с библиотеката Hypre, използваме следните команди:

```
module load hypre
module load essl
mpixlcxx test.cpp -I$HYPRE_INCLUDE -L$HYPRE_LIB \
        -L$ESSL_LIB -lHYPRE -lesslsmpbg
```

2.3 HDF5

Нека имаме програма, написана на C++, във файл `test.cpp`. За да я компилираме и свържем с библиотеката HDF5, използваме следните команди:

```
module load hdf5
module load zlib
mpixlcxx test.cpp -I$HDF5_INCLUDE -I$ZLIB_INCLUDE \
        -L$HDF5_LIB -L$ZLIB_LIB -lhdf5 -lz
```

2.4 Trilinos

Нека имаме програма, написана на C++, във файл `test.cpp`. За да я компилираме и свържем с библиотеката Trilinos, използваме следните команди:

```
module load trilinos
mpixlcxx test.cpp -I$TRILINOS_INCLUDE -L$TRILINOS_LIB \
        -lamesos -lepetra -laztecoo -lepetraext -lifpack \
        -lteuchos -lml -Wl,-z,muldefs
```

2.5 ParFE

Нека имаме програма, написана на C++, във файл `test.cpp`. За да я компилираме и свържем с библиотеката ParFE, използваме следните команди:

```
module load hypre ParMetis parfe zlib hdf5 trilinos essl lapack
mpixlcxx -c test.cpp -I$PARFE_INCLUDE -I$PARMETIS_INCLUDE \
        -I$ZLIB_INCLUDE -I$HDF5_INCLUDE -I$TRILINOS_INCLUDE
mpixlcxx test.o -Wl,-z,muldefs \
        -L$PARFE_LIB -L$PARMETIS_LIB -L$ZLIB_LIB -L$HDF5_LIB \
        -L$TRILINOS_LIB -L/opt/ibmcmp/xlf/bg/11.1/lib \
        -L$ESSL_LIB -L$LAPACK_LIB \
        -lparfe -lml -lamesos -lifpack -laztecoo -ltriutils \
        -lepetraext -lepetra -lteuchos -lhdf5 -lz -lzoltan \
        -llapack -lparmetis -lmetis -lesslbg -lxlf90_r -lrt \
        -lxlomp_ser -lxlfmath
```

2.6 Пример за компилиране и свързване с ParMETIS и Hypre

Фиг. 1 показва примерен makefile, който компилира програмата във файл demo.c. В този случай компилацията става с изпълнението на следните команди:

```
module load ParMetis hypre
make
```

```
CC=mpixlc_r
OMPFLAGS= -qsmp=omp
CFLAGS=-O3 -I$(HYPRE_INCLUDE) -I$(PARMETIS_INCLUDE) $(OMPFLAGS)
LDFLAGS= -L$(PARMETIS_LIB) -L$(HYPRE_LIB) -lparmetis -lmetis \
        -lHYPRE

demo : demo.o makefile
        $(CC) $(CFLAGS) -o $@ demo.o $(LDFLAGS)

demo.o : makefile
```

Фигура 1: Примерен makefile

3 Изпълнение на програмите

На суперкомпютъра Blue Gene/P програмите се изпълняват чрез системата за планиране на натоварването LoadLeveler. За да се пусне една програма трябва да се напише файл за задача. Фиг. 2 показва примерен файл на задача. Нека да поясним всеки един от редовете във файла със задачата. Полето `job_type = bluegene` трябва задължително да присъства за задачи, изпълнявани на Blue Gene/P. Полето `job_name` е свободно избираемо име на задачата. Полето `comment` е свободно избираемо пояснение към задачата (коментар). Полето `wall_clock_limit` е максималното време за изпълнение на задачата. Изпълнението на задачата се прекратява ако се превиши това време. Форматът е часове:минути:секунди. Полето `bg_size` е размер на дяла от компютъра (в брой възли), който ще се използва. Най-малкият размер е 128. Други възможности са 256, както и всички кратни на 512. Полето `bg_connection` е начина на свързване на възлите. Възможностите са `MESH`, `TORUS` или `PREFER_TORUS`. Тороидална топология е възможна само за дялове кратни на 512 възела. Полетата `output` и `error` задават имена на файловете, в които се записват съответно стандартният изход и стандартният изход за грешки на програмата. Полето `class` задава класа на задачата. Различните класове имат различни ограничения за време, както и за брой на възлите, които могат да се ползват. Информация за наличните класове може да се получи с командата `llclass`. В горният пример се изпълнява програмата `./test` с параметри „param1 param2“. Параметърът на `mpirun -mode` може да бъде SMP, DUAL или VN. Той определя по колко MPI процеса ще има на всеки възел. При SMP има по 1, при DUAL — по 2, при VN — по 4. Ако се използва OpenMP трябва да се зададе променливата на обкръжението `OMP_NUM_THREADS`

```
#!/bin/sh
# @ job_type = bluegene
# @ job_name = .-test.SMP.1.128param1param2.$(jobid)
# @ output = out..-test.SMP.1.128param1param2.$(jobid)
# @ error = elt..-test.SMP.1.128param1param2.$(jobid)
# @ comment = FEM simulation
# @ wall_clock_limit = 4:00:00
# @ restart = yes
# @ bg_size = 128
# @ bg_connection = MESH
# @ environment = COPY_ALL
# @ notification = error
# @ notify_user = user@domain.com
# @ class = n0128
# @ queue

mpirun -exe ./test -cwd $cwd -mode VN -np 128 \
        -env 'OMP_NUM_THREADS=1 BG_MAPPING=TXYZ' \
        -args "param1 param2"
```

Фигура 2: Примерен файл за изпълнение

с броя на нишките, които ще се използват от всеки процес. Максималния брой е 4, 2 или 1 съответно при режим SMP, DUAL или VN.

4 Основни команди на LoadLeveler

`llsubmit zadacha`

Зарежда задачата, чието описание се намира във файла „zadacha“, на опашката. При правилно направена задача се получава съобщение от вида:

```
llsubmit: The job "bgpfen.daits.government.bg.17993" has been submitted.
```

където частта в кавичките е идентификатора на задачата (като идентификатор може да се използва и само последната част — цифрите).

Със следната команда се получава информация за наредените задачи:

`llq`

Със следната команда се дава подробна информация за задачата с посоченият идентификатор:

`llq -l 17993`

Със следната команда се спира изпълнението на задача (или се премахва от списъка с чакащите задачи)

`llcancel 17993`