Разработка и запуск параллельных программ на языках C-DVMH и Fortran-DVMH на вычислительном комплексе МВС-10П

DVM-система

17 декабря 2016 г.

Для компиляции и запуска параллельных DVMH-программ на вычислительном комплексе МВС-10П необходимо:

1. Инициализировать версию DVM-системы, установленную на вычислительном комплексе МВС-10П. Выполнить команду:

   ```
   /nethome/admdvm/MVS-10P/bin/dvminit
   ```

   В результате выполнения данной команды в текущую директорию пользователя будут скопированы файлы `dvm` и `usr.par`, необходимые для компиляции и запуска DVMH-программ, а также будет создана символическая ссылка `~dvm_current` на установленную версию DVM-системы. Демонстрационные DVMH-программы доступны в директории `~dvm_current/dvm_sys/demo` (файлы с расширениями `*.cdv` и `*.fdv`).

2. Компиляция DVMH-программ. Вычислительный узел МВС-10П состоит из двух 8-ми ядерных процессоров Intel Xeon E5-2690 и двух сопроцессоров Intel Xeon Phi 7110X. Исполняемый код для процессора и сопроцессора отличается.

   (a) Компиляция программы на C-DVMH для выполнения на процессоре Intel Xeon осуществляется при помощи команды:

   ```
   ./dvm c <program name>.<cdvm-ext>,
   ```

   где `<cdvm-ext> = cdv|c|h`
   Например:

   ```
   ./dvm c jac3d_perf.cdv
   ```

   В результате выполнения данной команды будет получена исполняемая параллельная программа в модели MPI/OpenMP с именем `jac3d_perf`.

   (b) Компиляция программы на Fortran-DVMH для выполнения на процессоре Intel Xeon осуществляется при помощи команды:

   ```
   ./dvm f <program name>.<fdvm-ext>,
   ```

   где `<fdvm-ext> = fdv|f|ftn|for|f90|f95|f03`
   Например:

   ```
   ./dvm f jac3d_perf.fdv
   ```

   В результате выполнения данной команды будет получена исполняемая параллельная программа в модели MPI/OpenMP с именем `jac3d_perf`.

   (c) Компиляция программы на C-DVMH для выполнения на сопроцессоре Intel Xeon Phi осуществляется при помощи команды:

   ```
   ./dvm c -mmic <program name>.<cdvm-ext>,
   ```

   где `<cdvm-ext> = cdv|c|h`
   Например:

   ```
   ./dvm c -mmic jac3d_perf.cdv
   ```

   В результате выполнения данной команды будет получена исполняемая параллельная программа для сопроцессора в модели MPI/OpenMP с именем `jac3d_perf.mic`.

   (d) Компиляция программы на Fortran-DVMH для выполнения на сопроцессоре Intel Xeon Phi осуществляется при помощи команды:

```
./dvm f -mmic <program name>.<fdvm-ext>
```

Например:

```
./dvm f -mmic jac3d_perf.fdv
```

где `<fdvm-ext> = fdv|f|ftn|for|f90|f95|f03` В результате выполнения данной команды будет получена исполняемая параллельная программа для сопроцессора в модели MPI/OpenMP с именем `jac3d_perf.mic`.

Замечания.

i. DVMH-программа может иметь стандартное расширение (например, `*.f90` или `*.c`).

ii. Для Fortran-программ формат записи (свободный или фиксированный) определяется автоматически по расширению программы. Для программ, имеющих расширения `*.f90`, `*.f95`, `*.f03`, используется свободный формат, а для всех остальных – фиксированный. Указать формат можно при помощи опций компилятора: `-f90` (свободный) и `-FI` (фиксированный). Например:

```
./dvm f -f90 jac.fdv
```

3. Запуск DVMH-программы на выполнение.

Для запуска DVMH-программ используется СУППЗ, необходимо загрузить модуль СУППЗ: `module load launcher/suppz`. Перед выполнением DVMH-программы необходимо настроить следующие переменные окружения в скрипте `dvm` (более подробно про переменные окружения см. в документации):

(a) Максимальное время выполнения задачи. Переменная `maxtime`:

`export maxtime=10`

Значение по умолчанию – 10 минут. Для более длительного расчета (например, 1 час) необходимо задать:

`export maxtime=60`

(b) Число MPI-процессов, запускаемых на процессоре Intel Xeon и сопроцессорах Intel Xeon Phi. Переменная `DVMH_PPN`:

```
export DVMH_PPN='<HOST_PPN>,<MIC1_PPN>,<MIC2_PPN>'
```

Например: `export DVMH_PPN='2,1,1'`

(c) Число нитей, запускаемых на процессоре Intel Xeon и сопроцессорах Intel Xeon Phi (для каждого MPI-процесса узла). Переменная `DVMH_NUM_THREADS`:

`export DVMH_NUM_THREADS='8,8,240,240'`

Например, для конфигурации (симметричный режим):

```
export DVMH_PPN='2,1,1'
export DVMH_NUM_THREADS='8,8,240,240'
```

На каждом вычислительном узле МВС-10П, используемом при запуске программы, будут запущены два MPI-процесса на процессорах Intel Xeon и по одному MPI-процессу на каждом ускорителе Intel Xeon Phi. Для распределения работы между 8-ми ядрами процессора Intel Xeon будет создано 8 нитей, на каждом сопроцессоре Intel Xeon Phi будет создано 240 нитей.

При задании конфигурации:

```
export DVMH_PPN='0,1,1'
export DVMH_NUM_THREADS='240,240'
```

На каждом выделенном для выполнения задачи вычислительном узле МВС-10П будут использоваться только сопроцессоры Intel Xeon Phi, на каждом из которых будет запущено по одному MPI-процессу и будет создано 240 нитей.

При указании конфигурации:

```
export DVMH_PPN='1,1'
export DVMH_NUM_THREADS='16,240'
```

Один MPI-процесс будет запущен на процессоре Intel Xeon и один MPI-процесс на сопроцессоре Intel Xeon Phi. Данные MPI-процессы создадут 16 и 240 нитей соответственно. Второй сопроцессор Intel Xeon Phi, установленный в узле МВС-10П, использоваться не будет.

При задании конфигурации (native-режим):

```
export DVMH_PPN='0,1'
export DVMH_NUM_THREADS='240'
```

Вся работа будет выполняться на одном сопроцессоре.

После задания указанных выше переменных окружения DVMH-программа может быть запущена на счет при помощи команды:

```
./dvm run <processor matrix> <program name> [<task parameters>]
```

Параметр `<processor matrix>` задает размерность процессорной решетки, на которой будет запущена задача (`<n1> <n2>... <nK>`). При выполнении задачи будет создано `<n1>*<n2>*...*<nK>` MPI-процессов. Эти MPI-процессы будут отображены на процессоры Intel Xeon и сопроцессоры Intel Xeon Phi в соответствии с переменной `DVMH_PPN`. Например, в результате запуска данной команды

```
./dvm run 2 1 2 jac3d_perf
```

будет создана задача, которая использует четыре MPI-процесса. Для конфигурации:

```
export DVMH_PPN='2,1,1'
export DVMH_NUM_THREADS='8,8,240,240'
```

для выполнения задачи будет выделен один вычислительный узел. Два MPI-процесса, использующие 8 нитей, будут выполняться на ядрах процессоров Intel Xeon; еще два MPI-процесса по 240 нитей будут выполняться на сопроцессорах Intel Xeon Phi.

Если свободных ресурсов недостаточно и программа поставлена в очередь (`Task "<task name>.<task number>" queued successfully`), то можно ее исключить из очереди (`mqdel <task name>.<task number>`), опросить количество свободных процессоров (`mfree`) и запустить задачу повторно на меньшем числе процессоров.

Выдачи задачи перенаправляются в файлы `<task name>.<task number>.1/output` и `<task name>.<task number>.1/error`.

Замечание. `<task name>` - это `<program name>.<number of processors>.<task number>`. Поле `<task number>` - это число запусков программы на данном числе процессов из текущей директории.

4. Получение характеристик эффективности выполнения DVMH-программы:

```
./dvm pa <task name>.sts.gz+ <name of output file with characteristics>
```

5. Справочная информация о других DVM-командах доступна:

```
./dvm help
```

Получить подробную информацию о DVM-системе можно с сайта DVM.
Вопросы и замечания следует отправлять по адресу: dvm@keldysh.ru.
Инструкция по использованию СУППЗ на вычислительном комплексе МВС-10П доступна на сайте.