Разработка и запуск параллельных программ на языках C-DVMH и Fortran-DVMH на суперкомпьютере МВС-100К

DVM-система

17 декабря 2016 г.

Для компиляции и запуска параллельных DVMH-программ на вычислительном комплексе МВС-100К необходимо:

1. Инициализировать версию DVM-системы, установленную на вычислительном комплексе МВС-100К. Выполнить команду:

   ```
   /nethome/admdvm/MVS-100K/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.

   (a) Компиляция программы на C-DVMH осуществляется при помощи команды:

   ```
   ./dvm c [-noH] <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 осуществляется при помощи команды:

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

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

   Например:

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

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

   Замечания.

   i. Опция `-noH` задает режим работы компилятора. При указании опции `-noH` дополнительные нити для распределения работы внутри узла кластера создаваться не будут.

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

   iii. 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.

   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 узле. Переменная `DVMH_PPN`:

```
export DVMH_PPN='2'
```

(c) Число нитей, используемых каждым MPI-процессом. Переменная `DVMH_NUM_THREADS`:

```
export DVMH_NUM_THREADS='4,4'
```

Замечание. Значение данной переменной не учитывается для программ, скомпилированных с опцией `-noH`.

Например, для конфигурации:

```
export DVMH_PPN='2'
export DVMH_NUM_THREADS='4,4'
```

на каждом вычислительном узле суперкомпьютера МВС-100К будут запущены два MPI-процесса, каждый из которых будет использовать 4 нити для распределения работы.

При указании конфигурации:

```
export DVMH_PPN='1'
export DVMH_NUM_THREADS='8'
```

на каждом узле кластера будет запущено по одному MPI-процессу, использующему 8 нитей для распределения работы.

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'
export DVMH_NUM_THREADS='4,4'
```

для выполнения задачи будет выделено два вычислительных узла. На каждом из них будет запущено два MPI-процесса, каждый из которых создаст по 4 нити. Всего для вычислений будет использовано 16 ядер.

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 работе на вычислительном комплексе МВС-100К доступна на сайте.