Разработка и запуск параллельных программ на языках C-DVMH и Fortran-DVMH на экспериментальном вычислительном сервере Tester

DVM-система

22 декабря 2016 г.

Вычислительный сервер Tester предназначен для пользователей, которые только начинают знакомиться с возможностями DVM-системы.

Сервер состоит из 6-ти ядерного 12-ти поточного процессора Intel Core i7-980X Extreme Edition (12M Cache, 3.33 GHz) и двух графических ускорителей NVIDIA GeForce GTX 550 Ti (192 CUDA Cores, 1024 MB GDDR5). Объем оперативной памяти - 12 Гбайт DDR3.

Удаленный терминальный доступ на сервер осуществляется по протоколу SSH версии 2. На Unix-машинах этот протокол поддерживается стандартной командой ssh из пакета openssh. Для Windows рекомендуется программа Putty.

Передача файлов на сервер производится по протоколу SFTP или SCP, реализация которых включена в состав клиента SSH, как для Unix, так и для Windows.

Для компиляции и запуска параллельных DVMH-программ на вычислительном сервере Tester необходимо:

1. Инициализировать версию DVM-системы, установленную на вычислительном сервере. Выполнить команду:

   ```
   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] [-noCuda] <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/CUDA с именем `jac3d_perf`.

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

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

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

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

   (b) Компиляция программы на Fortran-DVMH осуществляется при помощи команды:

   ```
   ./dvm f [-noH] [-noCuda] <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/CUDA с именем `jac3d_perf`.

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

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

```
./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. 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.

Перед выполнением DVMH-программы необходимо настроить следующие переменные окружения в скрипте `dvm` (более подробно про переменные окружения см. в документации):

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

```
export DVMH_NUM_THREADS='12'
```

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

(b) Для каждого MPI-процесса необходимо указать количество используемых графических ускорителей. Переменная `DVMH_NUM_CUDAS`:

```
export DVMH_NUM_CUDAS='2'
```

Замечание. Значение данной переменной не учитывается для программ, скомпилированных с опциями `-noH` или `-noCuda`.

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. Например,

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

В результате выполнения данной команды будут запущены 2 MPI-процесса

При задании конфигурации:

```
export DVMH_NUM_THREADS='0'
export DVMH_NUM_CUDAS='1'
```

вся вычислительная работа будет выполняться только на графических ускорителях. Каждый MPI-процесс будет использовать свой (один) графический ускоритель.

При задании конфигурации:

```
export DVMH_NUM_THREADS='6'
export DVMH_NUM_CUDAS='0'
```

каждый MPI-процесс создаст 6 нитей, между которыми будет распределяться работа, а графические ускорители не будут использоваться для вычислений.

Конфигурация:

```
export DVMH_NUM_THREADS='5'
export DVMH_NUM_CUDAS='1'
```

позволяет загрузить все вычислительные устройства сервера. Каждый MPI-процесс создаст 6 нитей. Пять из которых будут использоваться для вычислений, а одна служебная нить будет осуществлять взаимодействие с графическим ускорителем.

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.