

Разработка и запуск параллельных программ на языках C-DVMH и Fortran-DVMH на гибридном вычислительном комплексе K-100

DVM-система

17 декабря 2016 г.

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

1. Инициализировать версию DVM-системы, установленную на вычислительном комплексе K-100. 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) Максимальное время выполнения задачи. Переменная `maxtime`:

```
export maxtime=30
```

Значение по умолчанию – 30 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'
```

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

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

```
export DVMH_NUM_CUDA='3'
```

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

Один узел вычислительного узла K-100 содержит два 6-ти ядерных процессора Intel Xeon X5670 и три графических ускорителя NVidia Fermi C2050.

Рекомендуемые конфигурации:

При указании конфигурации

```
export DVMH_PPN='3'  
export DVMH_NUM_THREADS='0'  
export DVMH_NUM_CUDA='1'
```

на каждом вычислительном узле компьютера K-100 будут запущены три MPI-процесса, каждый из которых будет использовать для вычислений свой графический ускоритель.

При указании конфигурации

```
export DVMH_PPN='2'  
export DVMH_NUM_THREADS='6'  
export DVMH_NUM_CUDA='0'
```

на каждом вычислительном узле компьютера K-100 будут запущены два MPI-процесса по 6 нитей. Графические ускорители использоваться не будут.

При указании конфигурации:

```
export DVMH_PPN='12'  
export DVMH_NUM_THREADS='1'  
export DVMH_NUM_CUDA='0'
```

на каждом вычислительном узле компьютера K-100 будут запущены 12 MPI-процессов. Распределение работы между нитями и графическими ускорителями производиться не будет.

При указании конфигурации:

```
export DVMH_PPN='1'  
export DVMH_NUM_THREADS='12'  
export DVMH_NUM_CUDAS='3'
```

будут задействованы все вычислительные ресурсы узла. На каждом узле будет запущен один MPI-процесс, который будет использовать 3 графических ускорителя и 12 нитей.

При указании конфигурации:

```
export DVMH_PPN='2'  
export DVMH_NUM_THREADS='6'  
export DVMH_NUM_CUDAS='1'
```

на каждом вычислительном узле компьютера К-100 будут запущены два MPI-процесса, каждый из которых будет использовать по 6 нитей и 1 графическому ускорителю.

Данная конфигурация является наиболее предпочтительной из-за архитектуры суперкомпьютера К100.

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, 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='0'  
export DVMH_NUM_CUDAS='1'
```

для выполнения задачи будет выделено два вычислительных узла. На каждом из них будет запущено два MPI-процесса, каждый из которых будет использовать один графический ускоритель. Всего для вычислений будет использовано 4 графических ускорителя.

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 К-100 доступна на [сайте](#).