Разработка и запуск параллельных программ на языках C-DVMH и Fortran-DVMH на суперкомпьютере «ЛОМОНОСОВ»

DVM-система

17 декабря 2016 г.

Для компиляции и запуска параллельных DVMH-программ на суперкомпьютере «ЛОМОНОСОВ» необходимо:

1. Загрузить модули для компилятора, реализации MPI, CUDA, системы очередей и, возможно, другие модули, необходимые для Вашей программы:

```
module add intel/15.0.090
module add impi/5.0.1
module add cuda/6.5.14
module add slurm/2.5.6
```

   Замечание. Загруженные модули не сохраняются между сессиями, если Вам нужно всегда использовать один набор модулей, допишите нужные команды в файл ~/.bashrc.

2. Инициализировать версию DVM-системы, установленную на суперкомпьютере «ЛОМОНОСОВ». Выполнить команду:

```
/mnt/msu/users/admdvm/DVM/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`).

   Замечание. На суперкомьютере «ЛОМОНОСОВ» запуск программ осуществляется только из каталога ~/`_scratch` и его подкаталогов. Только этот каталог будет виден на вычислительном узле и именно он будет считаться домашним каталогом. Поэтому инициализацию DVM-системы рекомендуется выполнять в каталоге ~/`_scratch` или его подкаталогах. Иначе перед запуском Вам необходимо будет скопировать все данные, скрипт `dvm` и исполняемые файлы в каталог ~/`_scratch`.

3. Компиляция DVMH-программ.

   Компиляция осуществляется на узле `compiler`. Предварительно нужно зайти на него командой `ssh compiler`. Вы можете зайти на этот узел сразу, указав в ssh-клиенте адрес `compiler.lomonosov.parallel.ru` вместо `lomonosov.parallel.ru`.

   (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
```

4. 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) Название раздела системы очередей, в который будет поставлена задача. Переменная `queuemode`:

```
export queuemode='regular4'
```

(c) Число MPI-процессов, запускаемых на узле. Переменная `DVMH_PPN`:

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

(d) 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`.

(e) 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`.

Рекомендуемые конфигурации:

Для раздела `gpu` можно использовать конфигурацию:

```
export DVMH_PPN='2'
export DVMH_NUM_THREADS='0'
export DVMH_NUM_CUDAS='1'
```

Для такой конфигурации на каждом вычислительном узле суперкомпьютера «ЛОМОНОСОВ» будут запущены два MPI-процесса, каждый из которых будет использовать для вычислений свой графический ускоритель.

Для раздела `regular6` можно использовать следующую конфигурацию:

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

Для такой конфигурации на каждом вычислительном узле суперкомпьютера «ЛОМОНОСОВ» будут запущены два MPI-процесса по 6 нитей. Графические ускорители использоваться не будут.

Для раздела `regular4` можно использовать следующую конфигурацию:

```
export DVMH_PPN='8'
export DVMH_NUM_THREADS='1'
export DVMH_NUM_CUDAS='0'
```

Для такой конфигурации на каждом вычислительном узле суперкомпьютера «ЛОМОНОСОВ» будут запущены 8 MPI-процессов. Распределение работы между нитями и графическими ускорителями производиться не будет.

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 (`Submitted batch job <JOBID>; squeue -j <JOBID>`), то можно ее исключить из очереди (`scancel <JOBID>`), опросить количество свободных процессоров/разделов (`sinfo`) 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>.<start number>`. Поле `<task number>` - это число запусков программы на данном числе процессов из текущей директории.

5. Получение характеристик эффективности выполнения DVMH-программы:

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

6. 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 на суперкомпьютере «ЛОМОНОСОВ» доступна на сайте.