

Development and launch of parallel C-DVMH and Fortran-DVMH programs on the experimental computing server Tester

DVM System

December 22, 2016

The computing server Tester is intended for users who only begin to study DVM system possibilities.

The server consists of 6-cores 12-threads processor Intel Core i7-980X Extreme Edition (Cache 12M, 3.33 GHz) and two graphic accelerators NVIDIA GeForce GTX 550 Ti (192 CUDA Cores, 1024 MB GDDR5). The volume of random access memory is 12 Gbytes DDR3.

Remote terminal access to the server is performed by SSH protocol, version 2. On Unix machines this protocol is supported by standard command `ssh` from `openssh` package. For Windows the **Putty** program is recommended.

The file transfer to the server is performed according to SFTP or SCP protocol. Their implementation is included in content of SSH client, both for Unix, and for Windows.

To compile and launch parallel DVMH-programs on the computing server Tester it is necessary:

1. To initiate DVM system version installed on the computing server. To perform the command:

```
dvminit
```

As a result of given command performing, `dvm` and `usr.par` files necessary for DVMH-program compilation and execution will be copied to the current directory of a user, and also the symbolic reference `~dvm.current` to the installed DVM system version will be created. Demonstrational DVMH-programs are available in the directory `~dvm.current/dvm_sys/demo` (files with `*.cdv` and `*.fdv` extensions).

2. Compilation of DVMH-programs.

- (a) Compilation of C-DVMH program is performed by the command:

```
./dvm c [-noH] [-noCuda] <program name>.<cdvm-ext>,
```

where `<cdvm-ext>` = `cdv|c|h`

For example,

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

The result of the command is ready-for-run parallel program in MPI/OpenMP/CUDA model with the name `jac3d_perf`.

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

The result of the command is ready-for-run parallel program in MPI/OpenMP model with the name `jac3d_perf`.

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

The result of the command is ready-for-run parallel program in MPI model with the name `jac3d_perf`.

- (b) Compilation of Fortran-DVMH program is performed by the command:

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

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

For example,

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

The result of the command is ready-for-run parallel program in MPI/OpenMP/CUDA model with the name `jac3d_perf`.

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

The result of the command is ready-for-run parallel program in MPI/OpenMP model with the name `jac3d_perf`.

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

The result of the command is ready-for-run parallel program in MPI model with the name `jac3d_perf`.

Notes.

- i. DVMH-program can have standard extension (for example, `*.f90` or `*.c`).
- ii. For Fortran-programs the record format (free or fixed) is determined automatically by the program extension. For the programs having extensions `*.f90`, `*.f95`, `*.f03`, the free format is used, and for all remaining – the fixed one. It is possible to specify a format using compiler options: `-f90` (free) and `-FI` (fixed). For example,

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

3. Launching DVMH-program.

Before DVMH program start it is necessary to set the following environment variables in `dvm` script (details see in [documentation](#)):

- (a) A number of the threads used by each MPI process. `DVMH_NUM_THREADS` variable:

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

Note. Value of this variable is ignored for the programs compiled with the option `-noH`.

- (b) For each MPI process it is necessary to specify a number of used graphic accelerators. `DVMH_NUM_CUDAS` variable:

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

Note. Value of this variable is ignored for the programs compiled with option `-noH` or `-noCuda`.

After setting of environment variables listed above, the DVMH-program can be launched by the command:

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

The parameter `<processor matrix>` sets rank of a processor grid on which the task will be run (`<n1>` `<n2>`... `<nK>`). During the task execution `<n1>*<n2>*...*<nK>` MPI processes will be created. For example,

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

As a result of this command 2 MPI processes will be launched.

For the configuration:

```
export DVMH_NUM_THREADS='0'
```

```
export DVMH_NUM_CUDAS='1'
```

All computational job will be done only on graphic accelerators. Every MPI process will use its own (single) graphic accelerator.

When specifying the configuration:

```
export DVMH_NUM_THREADS='6'
```

```
export DVMH_NUM_CUDAS='0'
```

Each MPI process will create 6 threads and job will be distributed between them, but graphic accelerators will not be used for computations.

Configuration:

```
export DVMH_NUM_THREADS='5'
```

```
export DVMH_NUM_CUDAS='1'
```

This configuration allows to load all computing devices of the server. Each MPI process will create 6 threads. Five from them will be used for computations, and one official thread will perform interaction with the graphic accelerator.

4. Obtaining efficiency characteristics of DVMH-program execution. Command:

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

5. The help information about other DVM commands is available as:

```
./dvm help
```

The detailed information about DVM system is on [DVM](#) site.

Questions and notes should be sent to the address: dvm@keldysh.ru.