Development and launch of parallel C-DVMH and Fortran-DVMH programs on supercomputer MVS-10P

DVM System

December 17, 2016

To compile and launch parallel DVMH-programs on supercomputer MVS-10P it is necessary:

1. To initiate DVM system version installed on supercomputer MVS-10P. To perform the command:

   ```
   /nethome/admdvm/MVS-10P/bin/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. The compute node of MVS-10P consists of two 8-core processors Intel Xeon E5-2690 and two coprocessors Intel Xeon Phi 7110X. The executable codes for the processor and for the coprocessor are differ

   (a) Compilation of C-DVMH program for execution on Intel Xeon processor is performed by the command:

   ```
   ./dvm c <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 model with the name `jac3d_perf`.

   (b) Compilation of Fortran-DVMH program for execution on Intel Xeon processor is performed by the command:

   ```
   ./dvm f <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 model with the name `jac3d_perf`.

   (c) Compilation of C-DVMH program for execution on Intel Xeon Phi coprocessor is performed by the command:

   ```
   ./dvm c -mmic <program name>.<cdvm-ext>,
   ```

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

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

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

   (d) Compilation of Fortran-DVMH program for execution on Intel Xeon Phi coprocessor is performed by the command:

   ```
   ./dvm f -mmic <program name>.<fdvm-ext>
   ```

   For example,

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

where <fdvm-ext> = fdv|f|ftn|for|f90|f95|f03. The result of the command is ready-for-run parallel program for coprocessor in MPI/OpenMP model with the name jac3d_perf.mic.

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.

For DVMH programs launching SUPPZ is used. It is necessary to load SUPPZ module: module load launcher/suppz. Before DVMH program start it is necessary to set the following environment variables in dvm script (details see in documentation):

(a) Maximum runtime of the program - maxtime variable:

```
export maxtime=10
```

Default value is 30 minutes. For more long calculations (for example, 1 hour) it is necessary to set:

```
export maxtime=60
```

(b) A number of MPI processes launched on Intel Xeon processor and on Intel Xeon Phi coprocessors. DVMH_PPN variable:

```
export DVMH_PPN='<HOST_PPN>,<MIC1_PPN>,<MIC2_PPN>'
```

For example, export DVMH_PPN='2,1,1'

(c) A number of the threads launched on Intel Xeon processor and on Intel Xeon Phi coprocessors (for each MPI process of a node). DVMH_NUM_THREADS variable:

```
export DVMH_NUM_THREADS='8,8,240,240'
```

For example, for the configuration (symmetric mode):

```
export DVMH_PPN='2,1,1'
export DVMH_NUM_THREADS='8,8,240,240'
```

On each compute node of MVS-10P used by the program two MPI processes on Intel Xeon processors and one MPI process on each Intel Xeon Phi coprocessor will be launched. For job distribution between 8 cores of the Intel Xeon processor 8 threads will be created, and 240 threads will be created on each Intel Xeon Phi coprocessor.

When specifying the configuration:

```
export DVMH_PPN='0,1,1'
export DVMH_NUM_THREADS='240,240'
```

Only Intel Xeon Phi coprocessors will be used on each compute node of MVS-10P provided for the task. One MPI process will be launched and 240 threads will be created on each of the nodes.

When specifying the configuration:

```
export DVMH_PPN='1,1'
export DVMH_NUM_THREADS='16,240'
```

One MPI process will be launched on Intel Xeon processor and one MPI process will be launched on Intel Xeon Phi coprocessor. These MPI processes will create 16 and 240 threads correspondently. The second Intel Xeon Phi coprocessor in the MVS-10P node won't be used.

When specifying the configuration (native-mode):

```
export DVMH_PPN='0,1'
export DVMH_NUM_THREADS='240'
```

Full job will be performed on one coprocessor.

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. These MPI processes will be mapped on Intel Xeon processors and Intel Xeon Phi coprocessors according to `DVMH_PPN` variable.

For example,

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

As a result of the command the task which uses four MPI processes will be created. For the configuration:

```
export DVMH_PPN='2,1,1'
export DVMH_NUM_THREADS='8,8,240,240'
```

one compute node will be provided for the task execution. Two MPI processes using 8 threads will be executed on the cores of Intel Xeon processors and two MPI processes with 240 threads per process will be executed on Intel Xeon Phi coprocessors.

If there are not enough free resources and the program was queued (`Task "<task name>.<task number>" queued successfully`), it is possible to delete it from queue (`mqdel <task name>.<task number>`), to get information about a number of free processors (`mfree`) and to restart the task on smaller number of processors.

Outputs of the task are redirected to the files `<task name>.<task number>.1/output` and `<task name>.<task number>.1/error`.

Note. `<task name>` is `<program name>.<number of processors>.<task number>`. The field `<task number>` is a number of the program starts on given number of processes in the current directory.

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.
The instruction how to use SUPPZ on supercomputer MVS-10P is available on the site.