Desmond/GPU for Desmond 3.x Users Documentation

Release 1.0

D. E. Shaw Research

April 23, 2013

CONTENTS

CHAPTER

ONE

RUNNING DESMOND/GPU

1.1 About configuration

In addition to the configuration file sections of Desmond 3.x, Desmond/GPU has an additional configuration section *spatial_order*. Furthermore, the *minimize* application is not available in Desmond/GPU.

Configuration file sections are:

```
app = mdsim|remd|vrun|...
boot = { file = p } # the structure file
global_cell = { ... }
force = { ... }
migration = { ... }
integrator = { ... }
profile = { ... } # for debugging
mdsim = { ... }
vrun = { ... }
remd = { ... }
spatial_order = { ... }
```

1.2 Invoking Desmond/GPU

Desmond/GPU applications are invoked from the command line by the `gdesmond` executable. Use `--include` to specify the configuration file. For example, to invoke `gdesmond` with the configuration file `equil.cfg`:

```
$ gdesmond --include equil.cfg
```

Desmond/GPU supports two different parallel collectives, `MPI` and `MT`, which are selected through the `--collective` command-line argument. The `MPI` collective is the default, and is used when the command-line argument is not supplied.

Use the `--tpp` argument in conjunction with `--collective MT` to run Desmond/GPU on multiple GPUs. To run the configuration `equil.cfg` on 2 GPUs:

```
$ gdesmond --include equil.cfg --collective MT --tpp 2
```

See *Running Desmond/GPU in Parallel* for details on `--collective` and `--tpp`

Due to the characteristics of current GPU architectures and programming models, thorough error handling on the GPU can lead to a significant loss of performance.

By default, Desmond/GPU does not perform comprehensive error checking on the GPU. It can however be put into a "forensic" mode which allows the identification of problems and the catching of errors. This forensic mode incurs a penalty on achievable throughput.

When using the `gdesmond` driver application, *forensics* mode can be set by adding the `--forensics` flag to the command line arguments.

The environment variable `CUDA_VISIBLE_DEVICES` can be used to specify the set of GPUs `gdesmond` can chose from. It is recommended that his mechanism be used at all times, since it can reduce CUDA-Runtime NUMA effects on multi-socket nodes.

Alternatively, the command line flags `--device` and `--exclude-device` can be used to select or exclude a specific GPU, e.g.:

```
$ gdesmond --include equil.cfg --exclude-device 1
```

will remove device 1 from the pool of devices Desmond/GPU can chose from. The recommended method to achieve this effect is to modify `CUDA_VISIBLE_DEVICES`, e.g.:

```
$ CUDA_VISIBLE_DEVICES=0,2,3 gdesmond --include equil.cfg
```

Table 1.1: Desmond command line options

argument	description
`--tpp` N	Sets the number of GPUs to parallelize across in the `MT` collective.
`--collective` name	Selects the collective, Can be `MT` or `MPI` (default).
`--include` file name	Adds configuration information from the given file. Can be given any number of times.
`--cfg` string	Adds configuration information from the given string. Can be given any number of times.
`--forensics`	Run the simulation in forensics mode.
`--restore` file	Restarts the `mdsim` or `remd` applications from a checkpoint. Because these applications are expected to run for long periods of time, during which hardware might fail, they can be set to produce a checkpoint file periodically, from which you can restart
`--profiling`	Generate a timing profile at the end of the simulation. Will have adverse effects on performance.

1.3 Running Desmond/GPU in parallel

Use the `MPI` collective to launch multiple, weakly-interacting simulations, such as in replica-exchange molecular dyanmics (see .....):

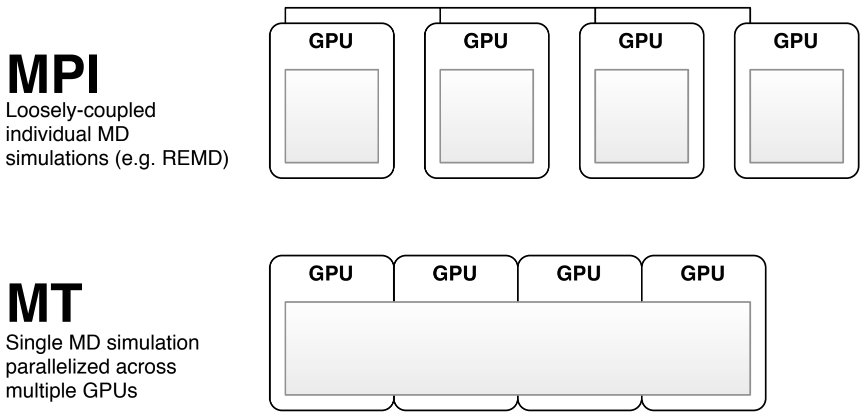
```
$ mpirun -n 4  gdesmond --include remd.cfg --collective MPI
```

For non-REMD simulations with a large number of particles (e.g. more than 50,000), Desmond/GPU can parallelize the work-load across two or four GPUs under certain conditions, which can reduce the running time of the simulation (see *illustration*).

```
$ gdesmond --include large_system.cfg --collective MT --tpp 2
```

The prerequisites for this multi-GPU parallelism are:

- the GPUs have to reside under the same PCIe root-complex (this can be verified by running `lspci -vt`).
- the simulation is run using the `MT` collective


MPI
Loosely-coupled individual MD simulations (e.g. REMD)
GPU
GPU
GPU
GPU
MT
Single MD simulation parallelized across multiple GPUs
GPU
GPU
GPU
GPU

The two collectives, `MT` and `MPI` are mutually exclusive.

1.4 Configuring Desmond/GPU applications

The main Desmond/GPU applications are `mdsim`, `remd`, and `vrun`. All Desmond/GPU applications share one difference compared to their Desmond 3.x equivalents: the `spatial_order` configuration block. Other, application-specific differences are described below.

1.4.1 Spatial order

"Spatial order" is a concept which is not exposed in Desmond 3.x. Spatial ordering defines the way particles are sorted spatially, and affects the pairlist creation algorithm, and thus the nonbonded near term implementation.

The spatial order is specified as follows:

```
spatial_order = auto|cells|conpablo
```

For most cases `spatial_order=auto` is the recommended settings. For certain systems with very large particle density, choosing `spatial_order=conpablo` may lead to better performance.

1.4.2 Differences in the REMD application

Desmond 3.x provides two different REMD versions: `remd` and `remd-graph`. Desmond/GPU provides only one REMD version `remd`, which is equivalent to Desmond 3.x `remd-graph`. Thus in order to run a Desmond 3.x `remd-graph` configuration with Desmond/GPU we need to rename `remd-graph` in the configuration file to `remd`.

The REMD `deltaE` plugin can only be executed at an interval which is a multiple of the migration interval, in other words: `remd.deltaE.interval` must be a multiple of `migration.interval`.

REMD allows different plugin configurations on a per-replica basis; e.g. to specify a different trajectory file name for each replica in Desmond 3.x:

```
remd-graph.T0.plugin.trajectory.name = ...
```

The same thing in Desmond/GPU:

```
remd.T0.remd.plugin.trajectory.name = ...
```

In contrast to Desmond 3.x, it is not possible to run one replica across more than one GPU; on the other hand, Desmond/GPU allows running *multiple* replicas on one GPU, e.g. 8 GPUs can be used to run a 23 replica REMD simulation, if there is enough memory available per GPU to simultaneously hold 3 systems.

1.5 Configuring the built-in plugins

Table 1.2: Desmond built-in plugin support in

plugin	**supported**
anneal	no
Biasing Force	no
e_bias	no
energy_groups	partially
compute_forces	no
eneseq	yes
maeff_output	no
posre_schedule	yes
randomize_velocities	yes
remove_com_motion	yes
trajectory	yes
status	yes

1.5.1 energy_groups

Periodically writes energy to the output file broken down across potential energy terms by each term's Hamiltonian category.

Energy break-down according to "energy group", (`grp_energygrp`) is not supported.

1.5.2 eneseq

The eneseq plugin in Desmond 3.x has an additional option `eneseq.flush_interval`, which control the frequency in which the eneseq output is flushed to file. Setting this value to zero will force eneseq to write out the data every time it is executed. Setting `eneseq.flush_interval` to a large value can be advantageous when the plugin is executed frequently and disk I/O, or network filesystem contention occurs.

CHAPTER

TWO

THE GLOBAL CELL

2.1 Migration

The migration interval (`migration.interval`) must be a multiple of the outer interval, i.e. a multiple of `integrator.respa.outer_timesteps * integrator.dt`.

The migration process is relatively more expensive in Desmond/GPU than in Desmond 3.x. The performance-optimal balance between the choice of `global_cell.r_clone`, `global_cell.margin` and `migration.interval` may therefore be different.

CHAPTER

THREE

CALCULATING FORCE AND ENERGY

3.1 Bonded, pair, and excluded interactions

3.1.1 Flat-bottomed harmonic well

Flat-bottomed harmonic well terms, e.g. `stretch_fbhw`, `angle_fbhw`, `improper_fbhw`, `posre_fbhw`, are not supported.

3.2 Van der Waals and electrostatic interactions

3.2.1 Near interactions

```
force.nonbonded.near = {
  type = default
  taper = none
  r_tap = $R_{\mathrm{tap}}$
  average_dispersion = $\nu$ # optional
}
```

Desmond/GPU does not support tapering of the interaction potentials, and for non-FEP calculations only `force.nonbonded.near.type=default` is required.

3.3 Nonbonded far interactions

GSE is not supported in Desmond/GPU. The configuration block for PME is:

```
force.nonbonded.far = {
  type = pme
  n_k = [ $k_x$ $k_y$ $k_z$ ]
  order = [ 4 4 4 ]
}
```

Since Desmond/GPU‘s PME interpolation order is fixed at 4, the grid sizes used in Desmond/GPU shoud in general be chosen at least as large as in Desmond 3.x.

Currently supported grid sizes are:

```
[  32  32  16 ] [  64  64  64 ] [  96  96  96 ]
[  96  96  64 ] [  48  48  48 ] [  32  64  64 ]
[  64  32  64 ] [  64  64  32 ] [  32  32  64 ]
[  32  64  32 ] [  64  32  32 ] [  32  32  32 ]
[  16  64  64 ] [  64  64  16 ] [  64  64 128 ]
[  64 128 128 ] [ 128 128 128 ] [ 128 128  64 ]
[ 128 128  32 ]
```

CHAPTER

FOUR

CONSTRAINTS

The configuration block for the constraints in Desmond/GPU is

```
force.constraint = {
  tol = δ
  maxit = m
  use_Reich = b_R
}
```

Reich's rigid motion constraint algorithm is used by default for HOH constraints. On some Kepler GPUs (K10, GTX 670, GTX 680), setting `force.constraint.use_Reich=false` can lead to better performance.

Reshake is not supported.

CHAPTER

FIVE

DYNAMICS

5.1 RESPA

For best performance `integrator.respa.outer_timesteps` should be chosen as large as possible.

5.2 Temperature

Temperature groups are not supported in Desmond/GPU.

5.3 Dynamical Systems

The only dynamical systems supported in Desmond/GPU are Brownian motion, Multigrator, and the Concatenator.

5.4 Multigrator

The Multigrator configuration of Desmond/GPU is identical to the configuration in Desmond 3.x, with the limitation that the new "Antithetic" thermostat is not yet supported.

The same Verlet schedules are supported in Desmond/GPU as in Desmond 3.x. Available PLS schedules are: PLS 1,1,3 and PLS 1,3,3.