

An MPI Application Porting Example: NAMD for Molecular Simulation

Daniele Cesini

INFN-CNAF

Stefano Ottani

CNR-ISOF

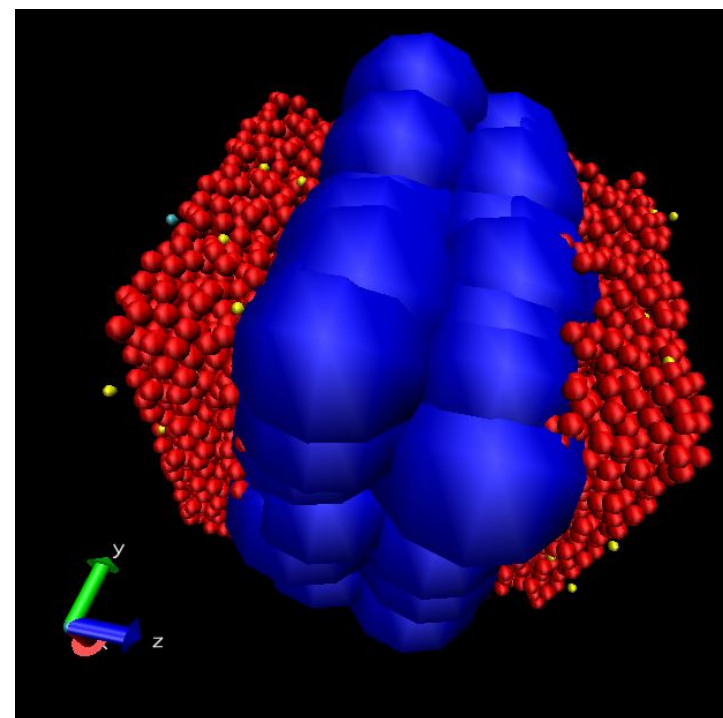
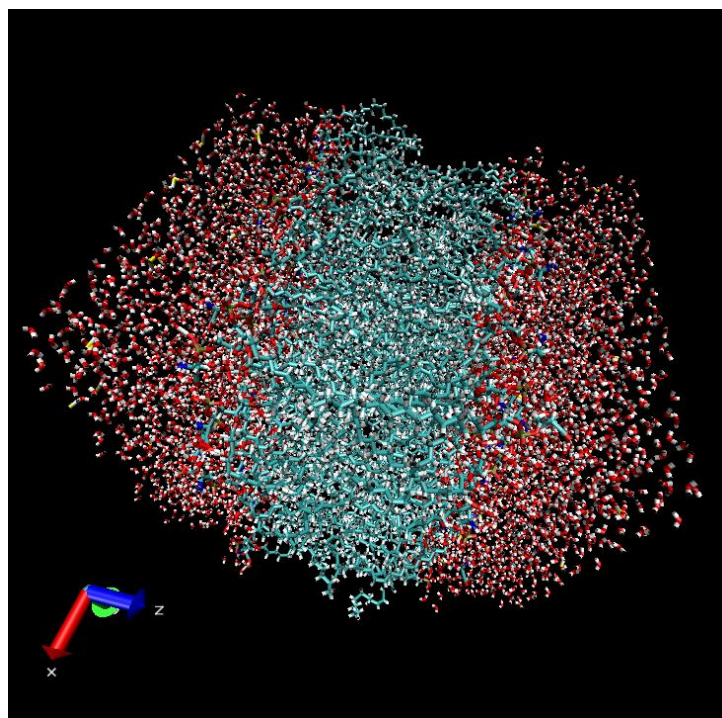
Alessandro Venturini

CNR-ISOF

- **The problem**
 - General Statement
 - Computing Requirement
 - Storage Requirement
- **The Griddification Strategy**
 - How do I submit?
 - Where do I submit (i.e how many available CE)?
 - Where do I put ISB & OSB ?
 - How many replicas?
- **The JDL**
- **The Submitting Application**

Computing Model

- **Long** dynamical simulation of **many independent** big molecules
- Typically one of this molecule contains 35k atoms and 25ns of simulated time were required
- The NAMD package is used for the simulation



- **“Long Simulation” means that MPI jobs are needed**
 - The Grid is not yet very performing on this
 - Few HPC centers – hope will improve in the future
- **“Many independent molecules” is solved submitting many independent jobs**
 - The Grid is (should be) already “perfect” for this

- **“charmrun” was used as process manager**
- **“charmrun” cannot be used on the Grid**
 - need to use mpirun!
- **NAMD needed to be recompiled for MPI (and SL5)**
 - Ok, took some time, but no problem
- **No License needed to run NAMD**

- For a typical molecule, 2.5 ps of simulation time using 8 cores and “charmrun” (2x Intel(R) Xeon(R) CPU E5520 (Nehalem) @ 2.27GHz)
→ 272s – 108,8 s/ps_sim
- 25 ns = 25000ps * 108.8 = 31.5 days walltime (on 8 Nehalem cores using charmrun)
 - 41 days using 8 cores and mpirun
 - 18 days using 32HTcore and mpirun
- Walltime almost linear vs the number of physical cores
- Walltime linear vs simulation time
- mpirun slower than charmrun → but it's our only possibility
- No particular requirements on memory

CHARMRUN

Run Time	NP	sim time (ps)	Proc Type	Walltime (s)
charm	4	2,5	2 x E5410 @ 2.33GHz	469
charm	8	2,5	2 x E5410 @ 2.33GHz	279
charm	8	250	2 x E5410 @ 2.33GHz	27643
charm	8	2,5	2 x E5520 @ 2.27GHz	272
charm	16HT	2,5	2 x E5520 @ 2.27GHz	147
charm	16HT	250	2 x E5520 @ 2.27GHz	23754

MPIRUN

Run Time	NP	sim time (ps)	Proc Type	Walltime (s)
mpirun	4	2,5	2 x E5410 @ 2.33GHz	568
mpirun	8	2,5	2 x E5410 @ 2.33GHz	360
mpirun	4	2,5	2 x E5520 @ 2.27GHz	469
mpirun	8	2,5	2 x E5520 @ 2.27GHz	275
mpirun	16HT	2,5	2 x E5520 @ 2.27GHz	255
mpirun	32HT	250	2x(2 x E5520 @ 2.27GHz)	15666

- **ISB**
 - 1 executable 8.3MB
 - 6 input data files → 9.3MB
 - 1 conf file (few bytes)

- **OSB**
 - Log file 1.2 MB (500ps)
 - OUTPUT data (containing trajectory) 0.39MB/frame
 - can be big depending on the frame frequency (set to 200fs)
 - *975MB for a 500ps run*
 - std.out / std.err (few bytes)
 - Restart file (3 files) ~ 3MB

- **A single MPI job will be too long**
 - Most of the CE queue are time limited
 - Everything can happen to our job in the long run
- **Submit N jobs of $25/N$ ns_sim each**
 - The output of a job is the input of the next one
- **Two fundamental parameters:**
 - The number N of Jobs (i.e the sim_time for each Job)
 - The number of CPU requested for each job (CPU_NUM)
 - Higher CPU_NUM means shorter runtime per job
 - Higher CPU_NUM means longer waiting time for the job in CE queues



- **Which CPU_NUM and N?**
 - Let's start with **CPU_NUM=32** and **N=50** (500ps_sim each)
 - ~8.5 hours walltime per job (Remember about 18days to complete)
 - ~1GB output per job(frame frequency 200fs – 2.5kframes)
- **Data management and replicas**
 - Executable and input data on a SE (~20MB in total)
 - No need for data requirements to run in a CE close to the data
 - Output and restart file on a SE (~1GB)
 - **Restart file of a job will be the input of the next one (few MB)**
 - Output (and restart files) replicated to have a backup copy
 - At least one copy in the close SE
 - Conf file, log file std.out /str.err on ISB & OSB via WMS (~1MB)

```

JobType = "Normal";
CPUNumber = <NUM_CPU>;
Executable = "mpi-start-wrapper.sh";
Arguments = "namd2 OPENMPI";
StdOutput = "namd2<n>.out";
StdError = "namd2<n>.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","namd2_conf file.conf"};
OutputSandbox = {"namd2_<n>.err","namd2_<n>.out","namd2_<n>.log"};
    
```

Contains the <n> implicitly
because contains the run length
And input file names

Requirements =

```

Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment)
    
```

```

Prologue="fetch_exec_and_data.sh";
PrologueArguments="<Molecule_Name> <n>"
Epilogue="put_data.sh";
EpilogueArguments="<Molecule_Name> <n>"
    
```

Fetch from the Grid input data and
executable based on a naming
convention for the lfn depending on <n>
and molecule name.
i.e. input_<mol_name>_<n>

```

RetryCount=6;
ShallowRetryCount = 6;
;
    
```

Pay attention to create
a safe Prologue and
Epilogue

Upload to the Grid output and restart
files based on the same naming
convention for the lfn
i.e. input_<mol_name>_<n+1>

- **2 CEs matching the JDL for the given VO**
 - MPI Requirements exclude many CEs
 - Only CREAM CE matched
- **Fortunately both CEs have more than 32 computing nodes**
 - We can (try to) use `CPU_NUM = 32`



The submitting application python pseudocode



```
from multiprocessing import Process

class common_data :
    CPU_NUM = 32
    N= 50
    common_conf_param = {'param1' : p1 , 'param2' : p2 ...}
    SE_NAMES = [SE1,SE2...]

cd = common_data
molecule_list = [mol1,mol2,mol3]

for molecule in molecule_list:

    if __name__ == '__main__':
        p = Process(target=run_molecule, args=(molecule,cd))
        p.start()
        p.join()
```

```
def run_molecule(molecule,cd):
    tmp = upload_input_t0('cd.SE_NAMES[1]',molecule)
    #create the input t0 file on SE_NAME, if not exists
    #lfn = input_molecule_t0
    tmp = replica_input_t0('cd.SE_NAMES[2]',molecule)

    for n in range(1 , N + 1):
        conf_file = create_conf_file(molecule, cd.n,
                                     cd.common_conf_param)
        jdl = create_jdl(cd.NUM_CPU, cd.N,n,conf_file)
        jobid = submit_n(jdl)
        jobid_list.append(jobid)

    while True:
        sleep(300)
        if check_grid_status(job_id) == 'Done':
            result = get_output(jobid)
            break
        if check_grid_status(job_id) == 'Aborted':
            print "Some Error"
            exit(1)
        ##### you need a timeout check here! #####
```

- Another way to approach the submission problem is to use a DAG for each molecule
- All the CE that matched the JDL were CREAM CEs
- Dag jobs are still not supported in CREAM CEs
- We have to wait to use DAGs for this application

- **Fine tuning of CPU_NUM and N**
 - Minimizing the waiting time on the CE queues keeping jobs walltime as short as possible
- **Automatic error recovery in case of aborted jobs**
- **Try to increase the number of sites supporting the given VO and MPI**



Useful Links



- **NAMD Project home page**

<http://www.ks.uiuc.edu/Research/namd/>

- **JDL Attributes Specification**

<https://edms.cern.ch/document/590869/1>

- **MPI in gLite**

<http://grid.ie/mpi/wiki/JobSubmission>

http://egee-uig.web.cern.ch/egee-uig/production_pages/MPIJobs.html