Report #5

Single-event OSTRICH-SWMM

Two documents can be used as support for setting the files needed for OSTRICH-SWMM. Kristina Macro developed a detailed tutorial on a simple SWMM model with one outlet and three subcatchments. OSTRICH Manual, developed by Dr. Matott, has detailed descriptions on how OSTRICH interprets the input files and gives parsing guidelines that can be used for setting up the input files. Nonetheless, the connection between OSTRICH and SWMM is not documented, which makes hard for a new user scale Kristina's small example to large-scale study areas. I recommend we develop a detailed Manual on how OSTRICH-SWMM works. On the other hand, there is no documentation on how to run OSTRICH-SWMM in CCR. This document partially close these gaps. Hence, it can be used for guidance for future OSTRICH-SWMM implementations.

There are six steps that must be followed to make a new SWMM model run in OSTRICH SWMM. These steps are detailed in Kristina's tutorial and are summarized as follows. Some notes are added, extending the content of Kristina's instructions.

1. Edit the input file and save as template (I suggest not using periods in the template name. e.g., save the file as model_template.inp instead of model.template.inp)
2. Edit the ostIn.txt file according to the parameters, variables, constraints and objective functions. (Observe that the number of integer parameters have to be the same in the model_input_parameters.json.tpl
3. Edit the model_parameters.input.json.tlp (same number of lines as integer parameters)
4. Edit model_input_parameters.json. Observe here that the name of the subcatchments is that of the model catchment, not the name of the variable (see Apendix A)
5. Change model_ostrich-swmm-config.json (I suggest not to change the binary output file nor the input file names (see Appendix A)

Once all the required files are modified, we are to decide where in CCR we are to run OSTRICH-SWMM. We can access to ccr from the command window of through the web server "ccr on demand". We have two cluster options:

1. Run from academic cluster (vortex)
2. Run for industry cluster (presto)

The difference on these clusters are the machines they have available. In general (at least until now) the academic cluster has more powerful machines, but they are serving more users (creating large waiting times). Industry cluster machines are also good, and are most of the time idle, so ours jobs start running almost immediately (no need to wait in the queue). The machines available in each cluster are listed in Tables 1 and 2.

Table 1. Academic cluster specs

	Front-end server for UB-HPC cluster	Dell 8-core Compute Nodes	IBM 8-core Compute Nodes	Dell 12-core Compute Nodes	Dell 16-core Compute Nodes	IBM 32-core Large Memory Nodes	Dell 32-core Large Memory Nodes		Dell 32-core GPU nodes	Dell 32-core "Skylake" Compute Nodes
Partition						largemem	largemem		gpu	skylake
Number of nodes		128	128	372	32	8	1	16	16	86
Number of proc	32	8	8	12	16	32	32	32	16	16
Number of threads									32	32
Processor (GHz)	2.10	8x2.13	8 x 2.27	12 x 2.40	16x2.20	32x2.20	32x2.13	16	32 x 2.10	32 x 2.10

Table 2. Industry cluster specs

	Compute nodes parallel	Compute nodes serial
Partition	144	72
Number of nodes	16	16
Number of processor Cores	8	8
Number of threads		
Processor (GHz)	2.60	2.60

Also, it is important to keep in mind that each partition has a limit per user (and time limit as well). For us, time is a limitation, since each SWMM run is estimated to take two ours (event-based)

Table 3. Academic cluster - Limits per user and time limit

Partition Name	Time Limit	Default Number CPUs	Job Submission Limit/user
debug	1 hour	1	4
general-compute	72 hours	1	1000
gpu	72 hours	1	4*
largemem	72 hours	1	32
skylake	72 hours	1	1000

Table 4. Industry cluster - Limits per user and time limit

INDUSTRY			
Partition Name	Time Limit	Default Number CPUs	Job Submission Limit/user
compute	72 hours	1	available only to industrial partners
compute	72 hours	1	

Once decided in which cluster OSTRICH-SWMM is going to run we must decide wether running serial or in parallel. We are in the process of timing these differences. The boxes below specify the slurms that run each of these jobs.

Serial (set the threads of SWMM input file to 1)

```
#!/bin/bash
#SBATCH --error=ostrich.stderr
#SBATCH --output=ostrich.stdout
#SBATCH --job-name=Tutorial
#SBATCH --mail-user=mtorresc@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --time=24:00:00
#SBATCH --cpus-per-task=1
#SBATCH --nodes=2
#SBATCH --tasks-per-node=12

module load intel
module load intel-mpi
module load ostrich
ulimit -s unlimited

# config intel MPI
export I_MPI_DEBUG=4
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

# launch distributed program
srun --ntasks-per-node=12 --cpus-per-task=1 OstrichMPI
echo "SLURM_JOBID="$SLURM_JOBID
```

Parallel (set the threads of SWMM input file to 4 OR 6). According to our tests, 4 works optimally.

```
#!/bin/bash
#SBATCH --error=ostrich.stderr
#SBATCH --output=ostrich.stdout
#SBATCH --job-name=PDDs_4cpus-per-task
#SBATCH --mail-user=mtorresc@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --time=03:00:00
#SBATCH --cpus-per-task=4
#SBATCH --nodes=2
#SBATCH --tasks-per-node=4
#SBATCH --constraint=IB

module load intel
module load intel-mpi
module load ostrich
module load swmm
ulimit -s unlimited

# config intel MPI
export I_MPI_DEBUG=4
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

export OPENMP_NUM_THREADS=SLURM_CPUS_PER_TASK
```

Remember that when running in presto (industry cluster) this have to be added at the beginning of the slurm:

#SBATCH --account=bsa

#SBATCH --partition=industry
#SBATCH --qos=industry

****

The test running in this moment include: The whole City of Buffalo Model, 1 CSO outfall, 2 different rain barrels, 5 locations. 320 iterations using parallel DDS (following Dr. Matott suggestion).

We have to discuss if it is possible to increase the run time in ccr, because today the system is not allowing submitting jobs for longer than 24 hours. I believe our run will take considerably longer.

Still, the following tests are up and running

DDS in the academic cluster
Parallel DDS in the academic cluster
Parallel DDS in the industry cluster

Appendix A

#1. Edit input files adding and save as model.template.inp

```
[LID_CONTROLS]
;;          Type/Layer Parameters
;;--------------- ---------- ----------
RB1         RB
RB1         STORAGE    36      1.0     0       0
RB1         DRAIN      58.5    0.5     0       6
RB2         RB
RB2         STORAGE    48      1.0     0       0
RB2         DRAIN      26      0.5     0       6

[LID_USAGE]
;;Subcatchment LID Process Number  Area      Width     InitSatur  FromImprv  ToPerv Report File Drain
to
;;--------------- --------------- ------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
```

#2. Edit OstIn.txt

```
ProgramType ParallelDDS
ModelExecutable ./ostrich-swmm.sh
ModelSubdir mod
ObjectiveFunction GCOP
PreserveModelOutput no

BeginFilePairs
model_input_parameters.json.tpl;  model_input_parameters.json
EndFilePairs

BeginExtraFiles
model-ostrich-swmm-config.json
model.template.inp
ET_BSA_07012016-01012018.dat
#RAIN_1993TY.RFF
EndExtraFiles

BeginIntegerParams
_NRB1_0_        0       0       100
_NRB2_0_        0       0       100
_NRB1_1_        0       0       100
_NRB2_1_        0       0       100
EndIntegerParams

BeginCombinatorialParams
_SUBCAT_0_      string  HStreet_10124-2         1       HStreet_10124-2
_SUBCAT_1_      string  HStreet_102     1       HStreet_102
EndCombinatorialParams
```

```
BeginResponseVars
#name   filename          keyword       line   col    token  aug?
#NCSO    model_results.csv ; node_name      1     2      ','   yes
FVOL     model_results.csv ; node_name      1     3      ','   yes
#FDUR    model_results.csv ; node_name      1     4      ','   yes
NRB1     num_lid.csv ;       Subcat_Name      1          2          ','   no
SUM_NRB1          num_lid.csv ;       Subcat_Name      4          2          ','   no
SUM_NRB2          num_lid.csv ;       Subcat_Name      4          3          ','   no
SUM_NRB3          num_lid.csv ;       Subcat_Name      4          3          ','   no
#ExcessRB1        num_lid.csv ;       Subcat_Name      5          2          ','   no
#ExcessRB2        num_lid.csv ;       Subcat_Name      5          3          ','   no
EndResponseVars

BeginTiedRespVars
#real cost of lids, cost = 150*RB1 + 400*RB2 + 500*RB3
#Real_Cost 1 SUM_NRB1 wsum 150
Real_Cost 3 SUM_NRB1 SUM_NRB2 SUM_NRB3 wsum 150 400 500
EndTiedRespVars

BeginGCOP
CostFunction FVOL
#CostFunction NCSO
#CostFunction Real_Cost
#PenaltyFunction APM
EndGCOP

BeginConstraints
COST     general  1E6       0.00      10000    Real_Cost
#Excess_Con1 general 1E6 0.00      0.00      ExcessRB1
#Excess_Con2 general 1E6 0.00      0.00      ExcessRB2
EndConstraints

BeginDDSAlg
PerturbationValue 0.2
MaxIterations  320
UseRandomParamValues
EndDDSAlg
```

#3. Edit model_parameters.input.json.tlp

```
{
        "lids": [
                {"width": 0, "fromImp": 1, "location": {"subcatchment": "_SUBCAT_0_"}, "area": 3.342, "toPerv":
1, "type": "RB1", "number": _NRB1_0_, "initSat": 0},
                {"width": 0, "fromImp": 1, "location": {"subcatchment": "_SUBCAT_1_"}, "area": 3.342, "toPerv":
1, "type": "RB1", "number": _NRB1_1_, "initSat": 0}
        ],

        "roofs": [
```

{"slope": 40, "NPerv": 0.1, "number": _NRB1_0_, "PctImperv": 100, "PctZero": 100, "area": 1655.23, "width": 43, "location": {"subcatchment": "_SUBCAT_0_"}, "OutID": "RB1", "NImp": 0.0115, "type": "RF1"},
{"slope": 40, "NPerv": 0.1, "number": _NRB1_1_, "PctImperv": 100, "PctZero": 100, "area": 1655.23, "width": 43, "location": {"subcatchment": "_SUBCAT_1_"}, "OutID": "RB1", "NImp": 0.0115, "type": "RF1"}
]
}

#4. Edit model_input_parameters.json

{
"lids": [
{
"location": {
"subcatchment": "HStreet_102"
},
"type": "RB1",
"number": 1,
"area": 36,
"width": 0,
"initSat": 0,
"fromImp": 0,
"toPerv": 1
},
{
"location": {
"subcatchment": "HStreet_102"
},
"type": "RB2",
"number": 1,
"area": 48,
"width": 0,
"initSat": 0,
"fromImp": 0,
"toPerv": 1
},
{
"location": {
"subcatchment": "HStreet_102"
},
"type": "RB3",
"number": 1,
"area": 65,
"width": 0,
"initSat": 0,
"fromImp": 0,
"toPerv": 1
},
]

}

#5. Change model_ostrich-swmm-config.json


```
{
  "binary_output_path": "LIDModel.out",
  "input_template_path": "model.template.inp",
  "input_parameters_path": "model_input_parameters.json",
  "input_path": "LIDModel.inp",
  "summary_dir": ".",
  "extract": {
    "steps": [
      {
        "type": "node",
        "enabled": true,
        "output_path": "model_results.csv",
        "statistics": [
          "node_name",
          "num_flow_events",
          "total_flow_volume",
          "total_flow_duration",
          "first_flow_start",
          "first_flow_end",
          "first_flow_duration",
          "first_flow_volume",
          "last_flow_start",
          "last_flow_end",
          "last_flow_duration",
          "last_flow_volume",
          "max_volume_flow_start",
          "max_volume_flow_end",
          "max_volume_flow_duration",
          "max_volume_flow_volume",
          "max_duration_flow_start",
          "max_duration_flow_end",
          "max_duration_flow_duration",
          "max_duration_flow_volume"
        ],
        "nodes": ["102"
        ],
        "event_threshold_flow_rate": 25
      }
    ]
  }
}
```