User Guide of MPI-Runner for parallel CORSIKA simulations on computing systems supporting Message Passing Interface

12/04/2013

Sushant Sharma, Gevorg Poghosyan
Steinbuch Centre for Computing

Revision History		
Date	Type	Author
16/11/2011	Initial Version	G. Poghosyan
18/11/2011	Typos and Formatting	D. Heck, G.Poghosyan
21/11/2011	Generalizing the “Getting starting” steps	J.Oehlschlaeger G.Poghosyan
24/11/2011	Changes of output files	G.Poghosyan
05/12/2011	Implementation of debug switch	G.Poghosyan
14/03/2012	Typos and Formatting	J.Oehlschlaeger G.Poghosyan
12/04/2013	Addition of unsetoptionflags	J.Oehlschlaeger D. Heck

Contents

For support in using MPI-Runner or suggestions for improvement of the code, contact:
Dr. Gevorg Poghosyan
Karlsruhe Institute of Technology
Steinbuch Centre for Computing
Head of Simulation Lab Elementary- and Astro- Particle
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
Germany
Tel: +49 721 608 25604.
Fax: +49 721 608 24972
E-Mail: Gevorg.Poghosyan@kit.edu
Web: http://www.scc.kit.edu/personen/gevorg.poghosyan.php

1. Introduction

The second section describes the steps how to start quickly working with MPI-Runner.
In the third section we talk about additional keywords that are required in the CORSIKA steering input file to run the air shower simulation in parallel.
Section four describes constants used in the MPI-Runner source code.
Fifth section explains the outputs produced by the MPI-Runner.
The last section represents the post-processing tools to be used for advanced analysis.

2. Getting Started

This section gives an explanation, how to compile and getting started with the MPI-Runner for parallel CORSIKA simulations.

Step 0:
Run script *src/parallel/unsetoptflags.sh* by typing *'. src/parallel/unsetoptflag.sh'* to exclude all environment variables concerning high compiler optimizations before creating a corsika parallel executable.

Step 1:
Compile the CORSIKA (using *./coconut)* with the options PARALLEL, PARALLELIB and MPIRUNNER.

> *Note: The corsikacompilefile.f and mpi_runner.c will be compiled and linked to the executable mpi_corsikaXXXXXX_runner, where XXXXXX is set upon the chosen CORSIKA version and the selected options. The post-processing code postprocess.c (see step 4) will also be compiled.*

Step 2:
Edit or create the steering input file with the keywords PARALLEL. Also keyword CUTFILE must be used when secondary particle/subshower will be simulated using an additional input file.

> *Note: If the keyword CUTFILE is used, be sure that the corresponding input file exists (*see section INPUTS of this document *for more details). The* **run** *directory of CORSIKA distributive contains a sample steering file named* ***parallel-inputs****.*

Step 3:
Submit the parallel job with the syntax:

[job_submitter] [mpi_executable] [input_file_name] [debug_switch]

> *Note: Job_submitter must be in syntax corresponding to used distributed parallel computation architecture with MPI support. When a debug switch as "T" is used a detailed protocol about all steps done by MPI-Runner will be saved in the FILE "mpirunprotocol.txt"*
>
> *For example, to submit the parallel CORSIKA to* HP XC3000 multi-processor system *at Steinbuch Centre for Computing, using the* HP MPI *batch system, the following command line could be used to run mpi-executable of CORSIKA on 16 processors in parallel; complete job is limited to 10 minutes execution time and 1000 MB for the memory;* parallel-inputs *steering file will be used and a detailed protocol about running steps will be generated*

```
job_submit -p 16 -c d -t 10 -m 1000 \
mpirun mpi_corsika72495Linux_QGSJET_gheisha_runner parallel-inputs
```

Step 4 (optional):
After the run the additional statistic for analysing the running process and results will be generated if the *postprocess* executable is copied and started in same folder where the output files are stored (see section Outputs of this document).

3. Inputs

To run CORSIKA in parallel the keyword *PARALLEL* in the steering (input) file of CORSIKA must be used (see also CORSIKA User's Guide). This keyword must be followed by 4 parameters in one line:

PARALLEL **DECTCUT DECTMAX MPIID LCOUT**

Fortran Format = (A8, 2F, I, L) **Default values = 1000., 1000000., 1, F**

DECTCUT	The lower energy threshold in GeV. All particles below this energy continue to run in the same job.
DECTMAX	The upper energy threshold in GeV. All particles above this threshold will be executed separately as a new job. The rest of the particles with energy between DECTCUT and DECTMAX are simulated together in different groups, such that the energy sum of each group is about DECTMAX.
MPIID	Unique identification number of each parallel task. Optional when running via MPI, but important for parallel simulations with job submission by shell scripts to distinguish sequential CORSIKA runs executed in parallel.
LCOUT	This logical parameter is passed to CORSIKA to prohibit (or enable) the production of CUTFILES with individual or groups of particles in each line. This is relevant when extra simulations of subshowers in separate CORSIKA runs are planned, e.g. to rerun uncompleted or incorrect parts.

For the separate simulation of a subshower the keyword *CUTFILE* in the steering (input) file of CORSIKA needs to be used (see also CORSIKA User's Guide). This has to be given with 3 parameter values in one line:

CUTFILE **CFILINP I1CUTPAR I2CUTPAR**

Fortran Format = (A7, A255, 2I) **Default values = ' ', 0, 0**

CFILINP	Filename to be read in case of secondary shower simulations, containing parameters of individual particles or group of particles in each line.
I1CUTPAR	Index (line number) of 1st particle to be read from the **CFILINP** and processed in the actual run.
I2CUTPAR	Index (line number) of last particle to be read from **CFILINP** and processed in the actual run. In case of only one particle to be processed, **I2CUTPAR** should have the same value as **I1CUTPAR**.

If in the steering file the keyword *CUTFILE* is used, the LPRIM variable used in CORSIKA and MPI-Runner will be set to 0 for secondary particle simulation. By default LPRIM is set to 1 which means a shower from a primary particle will be simulated.

Example:
```
PARALLEL 1000. 1000000. 1 F
CUTFILE DAT000999-741188179-000000060.cut 23 25
```

Note: Here the second line is optional and needed for the simulation for secondary showers for particles 23 to 25 from `DAT000999-741188179-000000060.cut`*.*

4. Constants Integrated in the Source Code of the MPI-Runner

When adapting the code to a given parallel computing system to optimize MPI communication and memory usage, some parameters in the source code could be tuned before compilation:

MASTER	MPI rank of processor designated as MASTER. Default value: 0
MXLIST	Maximum value of new parallel simulations/jobs a SLAVE can request by MASTER in a single request. Default value: 200001
MAX_GROUP_SIZE	Maximum number of particles in a group to be used for MPI communication. If the number of jobs in any group increases at runtime, it can lead to memory leak as the buffer for MPI communication could be too small to transfer all the parameters for starting parallel simulations of subshowers. The actual value at run time depends on the chosen energy thresholds (see DECTCUT and DECTMAX parameters of *PARALLEL* keyword). It is always safe to keep MAX_GROUP_SIZE high. Default value: 200
PARTICLE_INFO_COUNT	Number of parameters that define a particle. Default value: 19

5. Outputs

The MPI-Runner is generating output files in the subdirectory given by the keyword DIRECT of the CORSIKA steering file, as well some details would be stored directly in standard output of system (output file of job), where the simulation is running. This information could be useful when some of the subshowers have been simulated improperly and a part of the simulation should be repeated without the need to start all simulations from primary particle.

corsika_status_start1	This file contains information about the CORSIKA instance requested by the SLAVE. It could be used for debugging the interaction between the MPI-Runner and CORSIKA. The first column gives the unique MPIID of the job. Columns 2 and 3 give the index of the particle in the CUTFILE The 4th column tells whether the job was a primary (1) or a secondary (0) shower. Column 5 is the name of the CUTFILE if needed to be read by this instance of CORSIKA to fill 2nd stack.
corsika_status_finish1	This file has the same content as the file *corsika_status_start1*, but it is printed after the instance of CORSIKA has finished the subshower simulation.
Master2SlaveOrder	This file contains the list of the START messages sent by the MASTER to the SLAVES. The first 2 columns define the unique MPIID of the child and the parent jobs. The 3rd column tells whether the job was a primary (1) or secondary (0) shower. The columns 4 to 6 define the run number, seed, and MPIID which in-turn define the name of the CUTFILE to be read by the CORSIKA shower. Columns 7 and 8 give the indexes of the particles in the CUTFILE
Master2SlaveRecv	This file has the same information and format as the file *Master2SlaveOrder* and its content should match with the *Master2SlaveOrder* if the communication is correct.
queue	This file gives information about all the activities related to QUEUE, i.e. data added to QUEUE and data read from QUEUE and serves for bookkeeping. The first column reports about the type of requests that SLAVES sent to the MASTER as soon as a new subshower simulation is necessary or it is finished and must be removed from the bookkeeping buffer. Possible values are: <u>ADD_REQ</u> – Request for new parallel subshower simulation received. <u>ADDED</u> – Request for new subshower simulation included into QUEUE. <u>DEL_REQ</u> – Request for removing a parallel simulation from QUEUE received.

	DELETED – Parallel simulation is finished and removed from QUEUE. The second and third columns define the unique MPIID of the child process that would be used for a new simulation and the parent jobs – ID of slave/subshower that requested a new parallel simulation. The 4th column tells whether the job was a primary (1) or a secondary (0) shower. The 5th column tells the name of CUTFILE defined using run-number, seed, and MPIID of a new child job for possible simulation using CUTFILE. The 6th and 7th columns are the indexes in CUTFILE that a new particle will have if a simulation must be made using CUTFILE.
queue_add	This file gives the detailed information on the jobs added to the QUEUE. The first 2 columns define the unique MPIID of the child and the parent jobs. The 3rd column tells whether the job was a primary (1) or a secondary (0) shower. The columns 4 to 6 define the run-number, seed, and MPIID. The columns 7 and 8 give the indexes of the particles in case of the CUTFILE will be used. The column 9 gives the Linux time when the job was queued. The columns 11 to 14 give the snapshots of QUEUE as number of jobs running, queued, finished and lost, respectively.
relation1	This file gives the relation between any 2 jobs by backtracking from the child to the parent. The file consists of MPIID of child and parent job in first and second column, respectively.
mpirunprotocol.txt	This file will be generated if the debugging switch as "T" is used: third argument in command line after executable and input file name (see step 3 in section "Getting Started" of this document). It would contain the protocol about communications between MPI nodes including the time in seconds since an arbitrary time in the past and a types of book-keeping/debugging information: CUTFILE: information about particles to be used for running parallel simulations or generating external CUTFILE CUTFILENAME: report about usage of external CUTFILE FINISH: Master got information from slave about complete job REQUEST: Master got request from free slave for new job RESUME: Job waiting in queue sent to a free slave RUN_INFO: about the parameters used for initial start of simulation as primary or secondary shower SLAVE: Slave received order and going to start new parallel run START: Master ordering a free slave to start new parallel run STOP: Slave received order from Master to stop running TIME: Start and end time of separate parallel simulation

Slave2MasterRecv	This file shows the list of the requests received by the MASTER. The contents of this file should match to *Slave2MasterRequest* if the communication was correct.
Slave2MasterRequest	This file gives the list of the REQUEST messages sent by the SLAVES to the MASTER. The first column gives the unique MPIID of the job running on the SLAVE. The 3rd column tells whether the child job should be initiated as a primary (1) or a secondary (0) shower. The columns 4 to 6 define the run number, seed, and MPIID which in-turn define the name of the CUTFILE if it has to be read by the child job. The columns 7 and 8 give the indexes of the particles to be read from this CUTFILE.
status_finish1	This file contains the list of the FINISH messages received by the MASTER, i.e. the list of finished jobs. The first 2 columns define the unique MPIID of the child and the parent jobs. Column 3 is the end time of the shower/sub-shower simulation. Column 4 and 5 give the real start time and the end time of the simulation. The columns 6 to 9 give the snapshot of QUEUE as the number of jobs running, queued, finished, and lost, respectively, during simulations.
status_start	This file gives a list of the jobs initiated by the MASTER, entry is made as soon as a START message is sent to a SLAVE. The first 2 columns define the unique MPIID of the child and parent job. The 3rd column tells whether the job was a primary (1) or a secondary (0) shower. The columns 4 to 6 define the run-number, seed, and MPIID which in-turn defines the name of the CUTFILE to be read by the CORSIKA shower. The columns 7 and 8 give the indexes of the particles in the CUTFILE. The column 9 is the RANK (processor identifier) on which the job will run. Column 10 is a double value which is the start time of the shower/sub-shower (time in *sec* since Jan. 1, 1970). Columns 11 to 14 give the number of jobs running, queued, finished, and lost, respectively.
time.txt	Herewith the completeness of the simulation could be simply checked if this file is generated. It contains the start and stop time of simulation in seconds since an arbitrary time in the past and computation time spend for simulation in minits.

6. Post-Processing

The program ***postprocess*** analyses the output files *queue_add*, *status_start1*, *status_finish1*, *Slave2MasterRecv* produced during simulations by the MPI-Runner (see section Outputs).
Simply copy and run the executable in a directory, where the outputs of MPI runner are stored to produce advanced data for analysing and visualising the parallel simulation procedure.
It will extend following output files (or generate them, if they do not exist):

time.txt	This file displays information about: START TIME, STOP TIME and the TOTAL TIME taken by the simulation; MPIID of the longest job and the time taken by it; size (number of particles) of the largest group requested for parallel simulation; total number of jobs; actual aggregate CPU time consumed by the simulation. *NOTE*: If this information is unavailable, it might be the case that the simulation was not completed correctly.
plot_queue ***plot_idv_time***	These are scripts to be used for visualizing the outputs of MPI-Runner using *GNUPLOT*.
Graph	This script can be used to picture with *GRAPHVIZ* a relation between the jobs.
result	This file is a compilation of all the outputs of MPI Runner in form of 19 columns. This will be generated also if the simulation was not successful. The first 2 columns define the unique MPIID of child and parent job. The column 3 is the MPI rank on which the job ran (if started). The 4rd column tells whether the job was a primary (1) or a secondary (0) shower. The 5th column gives the status of the job: Q = queued, R = running, F = finished. The columns 6 to 8 define the run-number, seed, and MPIID. The columns 9 and 10 give the indexes of the particles in the CUTFILE. Colum 11 gives the size of the group (number of particles in the group) used for a new parallel simulation of a subshower. Columns 12 to 14 give the CPU time of a successful simulation. Columns 15 to 17 give the full time including the one spent for MPI communication. Columns 18 and 19 give the name of CUTFILES read and produced by the CORSIKA instance respectively.