

Using Wolfram Mathematica on the HybriLIT Platform to Model Heavy Meson Properties in Effective QCD.

Хмелев А.В., Кокаев Д.А.,Стрельцова О.И., Зуев М.И., Калиновский Ю.Л., Матвеев М.А., Киракосян М.Х.

Лаборатория информационных технологий им. М. Г. Мещерякова, Объединённый Институт Ядерных Исследований

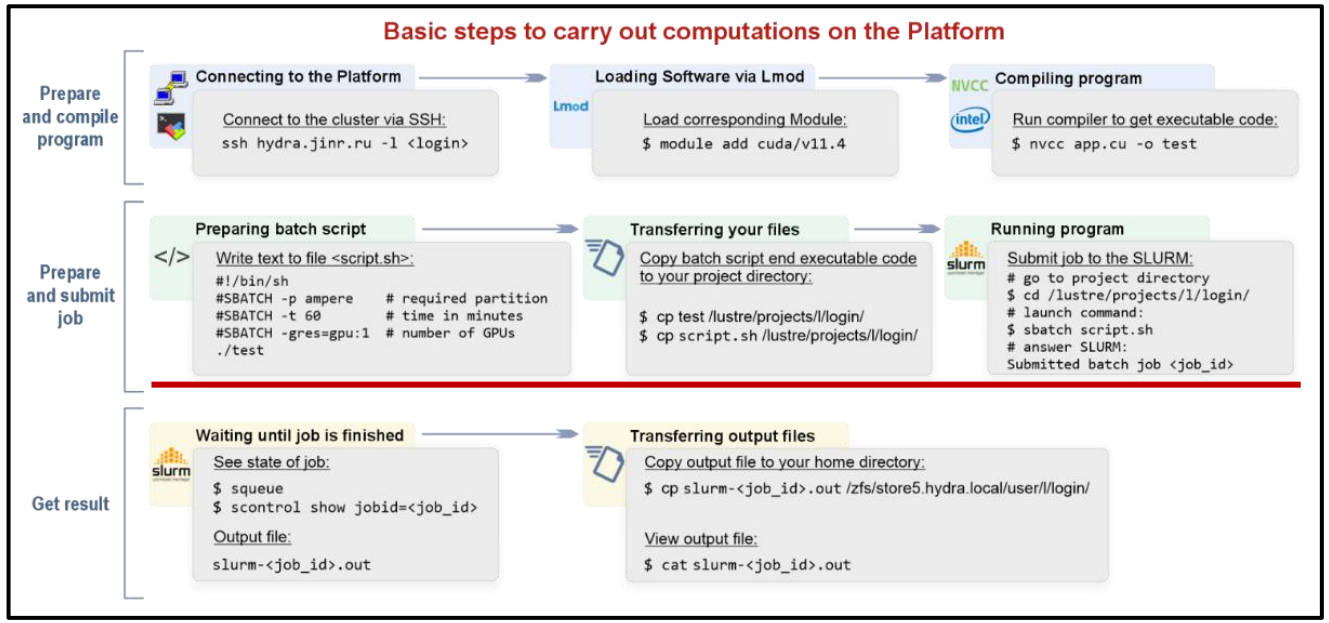
SLURM (Slurm Workload Manager)

Main SLURM workload manager commands:

- **sbatch** — command to submit an application/job to the queue;
- **squeue** — command to view the list of running applications/jobs in the queue;
- **sinfo** — command to view the status of compute nodes and queues. Compute nodes can be in one of the following states:

1. idle – the node is free/idle;
2. alloc – the node is in use by a process/allocated;
3. mix – the node is partially occupied and partially free;
4. down, drain, drng – the node is blocked/drained.

- **scancel** — command to cancel/remove an application/job from the queue.
- **scontrol** — used to view or modify the state of SLURM, including: jobs, nodes, and queues.



Prepare the program files with the **`_.m`** extension and a launch script with the **`_.sh`** extension.

→ Move the prepared files to the **`/lustre/projects/l/login`** directory

→ Submit a job to the **SLURM** workload manager

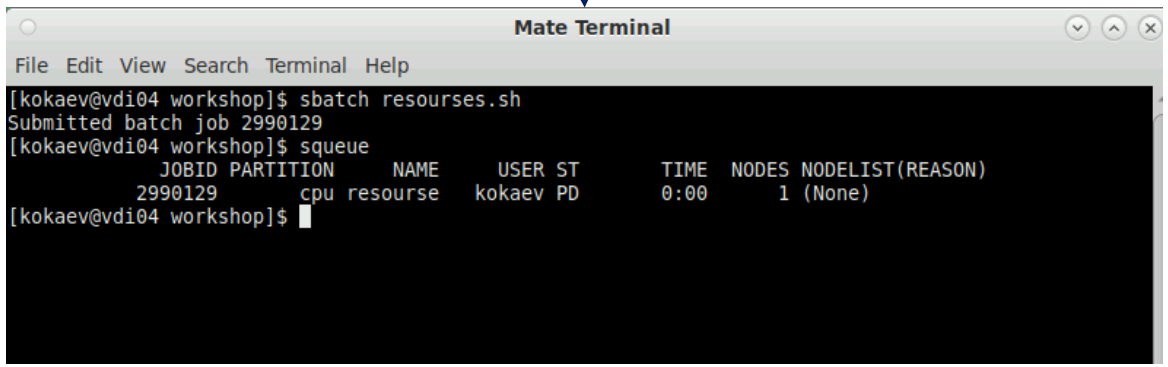
Task launch procedure

Example program **tutorial.m**

```
 (* 1. Initialization of constants and parameters *)
Nc = 3; Nf = 1;
pmin = 0.000001; pmax = 10.0;

(* Definition of physical functions *)
(* (phi, Equark, SolveMassGap, CalculateNorm, ...) *)

FuncSystemEquationAtT[T_, mq_, L_, D0_] := Module[{dm, m, M, Nv, fv},
(* Basic calculations (solving equations) *)
{T, m, 2*m, M, Nv, fv}
];

(* 2. Launching parallel computations *)
(* Checking sub-core availability and license count *)
slurmNTasks = ToExpression[Environment["SLURM_NTASKS"]];
slurmCpusPerTask = ToExpression[Environment["SLURM_CPUS_PER_TASK"]];
totalCores = slurmNTasks * slurmCpusPerTask;

licenseLimit = $MaxLicenseSubprocesses;
If[! NumericQ[licenseLimit], licenseLimit = Infinity];

nKernels = Min[totalCores - 1, licenseLimit];

LaunchKernels[nKernels];
Print[" KernelCount: ", $KernelCount];

(* 3. Preparation of the parameter grid *)
mqGrid = Range[1.1, 1.4, 0.05];
LGrid = Range[1.0, 2.0, 0.1];
D0Grid = Range[0.6, 1.4, 0.1];
grid = Flatten[Table[{mq, L, D0}, {mq, mqGrid}, {L, LGrid}, {D0, D0Grid}],2];

(* 4. Parallel parameter scanning *)
gridResults = ParallelMap[EvalPoint, grid, Method -> "CoarsestGrained"];

(* Searching for the best parameters and outputting the result *)
best = First@MinimalBy[gridResults, #[[2]] &];
Print["Best parameters: ", best[[1]]];

(* 5. Program termination *)
CloseKernels[]; (* Disabling parallel cores *)
Quit[];         (* Terminating the Mathematica kernel *)
```

Example of script text

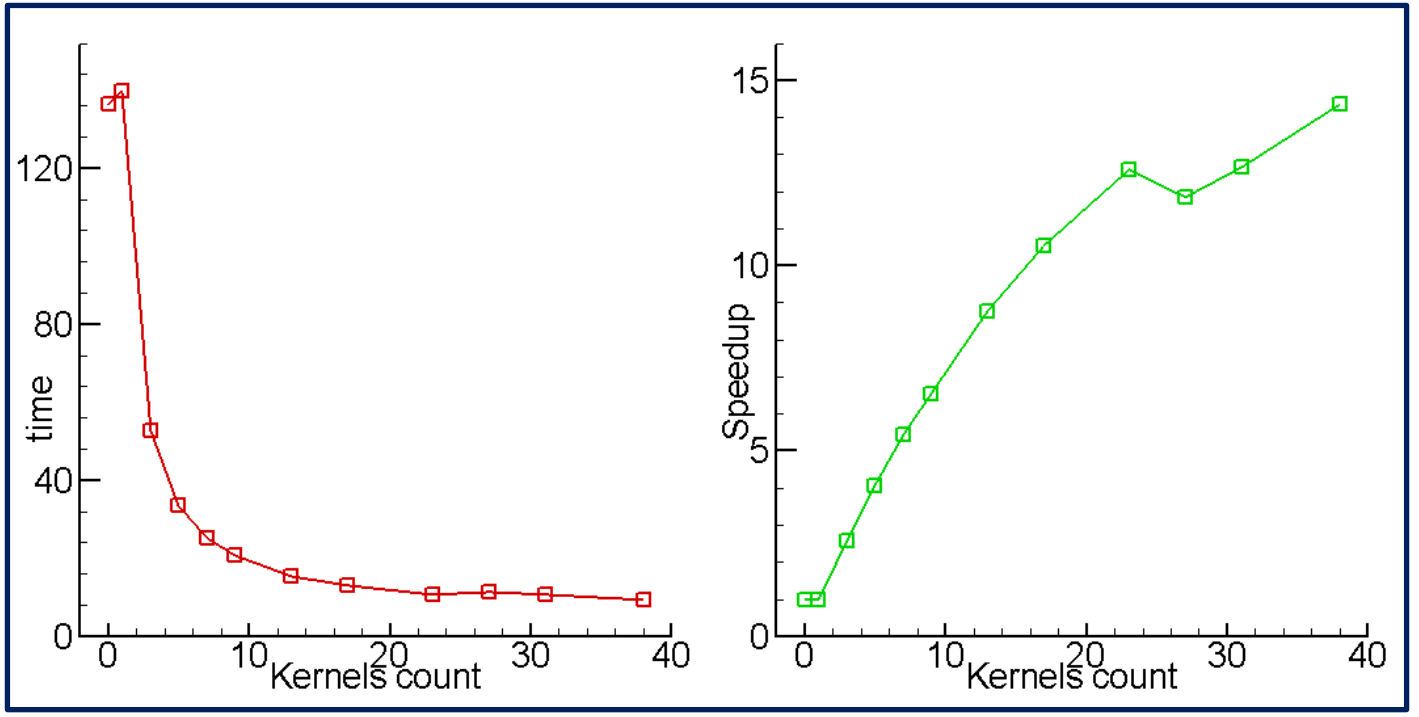
```
#!/bin/sh
#SBATCH -p cpu                          # queue to be used

#SBATCH --job-name=tutorial_task        # sets the job name in the queue
#SBATCH --output= tutorial_task _%j.out # sets the name of the standard output file
#SBATCH --error= tutorial_task _%j.err  # sets the name of the standard error output file

#SBATCH -t 10                           # reserved computation time (10 minutes)

#SBATCH --nodes=1                       # request the number of nodes per job
#SBATCH --ntasks=1                      # request the number of tasks
#SBATCH --cpus-per-task=1               # request 1 CPUs (processors) per task

module add Mathematica/v14.1            # load the Mathematica module
math -script tutorial.m                 # tutorial.m – the name of the file with the program
                                        listing
```



Launching a parallel task

```
#!/bin/sh
#SBATCH -p gpu_k80                      # queue to be used

#SBATCH --job-name=tutorial_task        # sets the job name in the queue
#SBATCH --output= tutorial_task _%j.out # sets the name of the standard output file
#SBATCH --error= tutorial_task _%j.err  # sets the name of the standard error output file

#SBATCH -t 10                           # reserved computation time (10 minutes)

#SBATCH --nodes=1                       # request the number of nodes per job
#SBATCH --ntasks=1                      # request the number of tasks
#SBATCH --cpus-per-task=40              # request 40 CPUs (processors) per task

module add Mathematica/v14.1            # load the Mathematica module
math -script tutorial.m                 # tutorial.m – the name of the file with the program listing
```

By changing the **`cpu-per-task`** parameter, you can request a larger number of physical processors per task.



Dependence of computation time and speedup on the number of cores