

---

# DPGEN

## Deep Potential

Sep 04, 2022



## PARAMETERS

<b>1</b>	<b>dpngen run machine parameters</b>	<b>1</b>
<b>2</b>	<b>This is a blank file to test sphinx-build</b>	<b>31</b>
<b>3</b>	<b>How to contribute</b>	<b>33</b>



## DPGEN RUN MACHINE PARAMETERS

### **run\_mdata:**

type: dict

argument path: run\_mdata

machine.json file

### **api\_version:**

type: str

argument path: run\_mdata/api\_version

Please set to 1.0

### **train:**

type: dict

argument path: run\_mdata/train

Parameters of command, machine, and resources for train

### **command:**

type: str

argument path: run\_mdata/train/command

Command of a program.

### **machine:**

type: dict

argument path: run\_mdata/train/machine

### **batch\_type:**

type: str

argument path: run\_mdata/train/machine/batch\_type

The batch job system type. Option: PBS, Lebesgue, DpCloudServer, Shell, DistributedShell, Torque, Slurm, SlurmJobArray, LSF

### **local\_root:**

type: `NoneType | str`

argument path: `run_mdata/train/machine/local_root`

The dir where the tasks and relating files locate. Typically the project dir.

**remote\_root:**

type: `NoneType | str`, optional

argument path: `run_mdata/train/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

**clean\_asynchronously:**

type: `bool`, optional, default: `False`

argument path: `run_mdata/train/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context\_type*, different sub args are accepted.

**context\_type:**

type: `str` (flag key)

argument path: `run_mdata/train/machine/context_type`

possible choices: `DpCloudServerContext`, `HDFSContext`, `LocalContext`, `LazyLocalContext`, `SSHContext`, `LebesgueContext`

The connection used to remote machine. Option: `DpCloudServerContext`, `HDFSContext`, `LebesgueContext`, `LocalContext`, `LazyLocalContext`, `SSHContext`

When *context\_type* is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

**remote\_profile:**

type: `dict`

argument path: `run_mdata/train/machine[DpCloudServerContext]/remote_profile`

The information used to maintain the connection with remote machine.

**email:**

type: `str`

argument path:

`run_mdata/train/machine[DpCloudServerContext]/remote_profile/email`

Email

**password:**

type: `str`

argument path:

`run_mdata/train/machine[DpCloudServerContext]/remote_profile/password`

Password

**program\_id:**

type: int, alias: *project\_id*

argument path:

run\_mdata/train/machine[DpCloudServerContext]/remote\_profile/program\_id

Program ID

**keep\_backup:**

type: bool, optional

argument path: run\_mdata/train/machine[DpCloudServerContext]/

remote\_profile/keep\_backup

keep download and upload zip

**input\_data:**

type: dict

argument path:

run\_mdata/train/machine[DpCloudServerContext]/remote\_profile/input\_data

Configuration of job

When *context\_type* is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/train/machine[HDFSContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/train/machine[LocalContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/train/machine[LazyLocalContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

**remote\_profile:**

type: dict

argument path: `run_mdata/train/machine[SSHContext]/remote_profile`

The information used to maintain the connection with remote machine.

**hostname:**

type: str

argument path: `run_mdata/train/machine[SSHContext]/remote_profile/hostname`

hostname or ip of ssh connection.

**username:**

type: str

argument path: `run_mdata/train/machine[SSHContext]/remote_profile/username`

username of target linux system

**password:**

type: str, optional

argument path: `run_mdata/train/machine[SSHContext]/remote_profile/password`

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

**port:**

type: int, optional, default: 22

argument path: `run_mdata/train/machine[SSHContext]/remote_profile/port`

ssh connection port.

**key\_filename:**

type: `NoneType` | str, optional, default: None

argument path:

`run_mdata/train/machine[SSHContext]/remote_profile/key_filename`

key filename used by ssh connection. If left None, find key in `~/.ssh` or use password for login

**passphrase:**

type: `NoneType` | str, optional, default: None

argument path:

`run_mdata/train/machine[SSHContext]/remote_profile/passphrase`

passphrase of key used by ssh connection

**timeout:**

type: int, optional, default: 10

argument path: `run_mdata/train/machine[SSHContext]/remote_profile/timeout`



timeout of ssh connection

**totp\_secret:**

type: `NoneType | str`, optional, default: `None`

argument path:

`run_mdata/train/machine[SSHContext]/remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

**tar\_compress:**

type: `bool`, optional, default: `True`

argument path:

`run_mdata/train/machine[SSHContext]/remote_profile/tar_compress`

The archive will be compressed in upload and download if it is `True`. If not, compression will be skipped.

When *context\_type* is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

**remote\_profile:**

type: `dict`

argument path: `run_mdata/train/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

**email:**

type: `str`

argument path:

`run_mdata/train/machine[LebesgueContext]/remote_profile/email`

Email

**password:**

type: `str`

argument path:

`run_mdata/train/machine[LebesgueContext]/remote_profile/password`

Password

**program\_id:**

type: `int`, alias: *project\_id*

argument path:

`run_mdata/train/machine[LebesgueContext]/remote_profile/program_id`

Program ID

**keep\_backup:**

type: bool, optional  
argument path:  
run\_mdata/train/machine[LebesgueContext]/remote\_profile/keep\_backup  
keep download and upload zip

**input\_data:**

type: dict  
argument path:  
run\_mdata/train/machine[LebesgueContext]/remote\_profile/input\_data  
Configuration of job

**resources:**

type: dict  
argument path: run\_mdata/train/resources

**number\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/train/resources/number\_node  
The number of node need for each *job*

**cpu\_per\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/train/resources/cpu\_per\_node  
cpu numbers of each node assigned to each job.

**gpu\_per\_node:**

type: int, optional, default: 0  
argument path: run\_mdata/train/resources/gpu\_per\_node  
gpu numbers of each node assigned to each job.

**queue\_name:**

type: str, optional, default: \*\*\*\*  
argument path: run\_mdata/train/resources/queue\_name  
The queue name of batch job scheduler system.

**group\_size:**

type: int  
argument path: run\_mdata/train/resources/group\_size  
The number of *tasks* in a *job*. 0 means infinity.

**custom\_flags:**

type: list, optional  
argument path: run\_mdata/train/resources/custom\_flags

The extra lines pass to job submitting script header

**strategy:**

type: dict, optional  
argument path: run\_mdata/train/resources/strategy

strategies we use to generation job submitting scripts.

**if\_cuda\_multi\_devices:**

type: bool, optional, default: False  
argument path: run\_mdata/train/resources/strategy/if\_cuda\_multi\_devices

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable CUDA\_VISIBLE\_DEVICES to different task. Usually, this option will be used with Task.task\_need\_resources variable simultaneously.

**ratio\_unfinished:**

type: float, optional, default: 0.0  
argument path: run\_mdata/train/resources/strategy/ratio\_unfinished

The ratio of *jobs* that can be unfinished.

**para\_deg:**

type: int, optional, default: 1  
argument path: run\_mdata/train/resources/para\_deg

Decide how many tasks will be run in parallel.

**source\_list:**

type: list, optional, default: []  
argument path: run\_mdata/train/resources/source\_list

The env file to be sourced before the command execution.

**module\_purge:**

type: bool, optional, default: False  
argument path: run\_mdata/train/resources/module\_purge

Remove all modules on HPC system before module load (module\_list)

**module\_unload\_list:**

type: list, optional, default: []  
argument path: run\_mdata/train/resources/module\_unload\_list

The modules to be unloaded on HPC system before submitting jobs

**module\_list:**

type: list, optional, default: []

argument path: run\_mdata/train/resources/module\_list

The modules to be loaded on HPC system before submitting jobs

**envs:**

type: dict, optional, default: {}

argument path: run\_mdata/train/resources/envs

The environment variables to be exported on before submitting jobs

**wait\_time:**

type: int | float, optional, default: 0

argument path: run\_mdata/train/resources/wait\_time

The waiting time in second after a single *task* submitted

Depending on the value of *batch\_type*, different sub args are accepted.

**batch\_type:**

type: str (flag key)

argument path: run\_mdata/train/resources/batch\_type

possible choices: LSF, Shell, PBS, Torque, Slurm, SlurmJobArray, DpCloudServer, Lebesgue, DistributedShell

The batch job system type loaded from machine/batch\_type.

When *batch\_type* is set to LSF (or its alias *lsf*):

**kwargs:**

type: dict

argument path: run\_mdata/train/resources[LSF]/kwargs

Extra arguments.

**gpu\_usage:**

type: bool, optional, default: False

argument path: run\_mdata/train/resources[LSF]/kwargs/gpu\_usage

Choosing if GPU is used in the calculation step.

**gpu\_new\_syntax:**

type: bool, optional, default: False

argument path: run\_mdata/train/resources[LSF]/kwargs/gpu\_new\_syntax

For LFS >= 10.1.0.3, new option -gpu for #BSUB could be used. If False, and old syntax would be used.

**gpu\_exclusive:**

type: bool, optional, default: True

argument path: run\_mdata/train/resources[LSF]/kwargs/gpu\_exclusive

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

**custom\_gpu\_line:**

type: NoneType | str, optional, default: None

argument path: run\_mdata/train/resources[LSF]/kwargs/custom\_gpu\_line

Custom GPU configuration, starting with #BSUB

When *batch\_type* is set to Shell (or its alias shell):

**kwargs:**

type: dict, optional

argument path: run\_mdata/train/resources[Shell]/kwargs

This field is empty for this batch.

When *batch\_type* is set to PBS (or its alias pbs):

**kwargs:**

type: dict, optional

argument path: run\_mdata/train/resources[PBS]/kwargs

This field is empty for this batch.

When *batch\_type* is set to Torque (or its alias torque):

**kwargs:**

type: dict, optional

argument path: run\_mdata/train/resources[Torque]/kwargs

This field is empty for this batch.

When *batch\_type* is set to Slurm (or its alias slurm):

**kwargs:**

type: dict, optional

argument path: run\_mdata/train/resources[Slurm]/kwargs

Extra arguments.

**custom\_gpu\_line:**

type: NoneType | str, optional, default: None

argument path: run\_mdata/train/resources[Slurm]/kwargs/custom\_gpu\_line

Custom GPU configuration, starting with #SBATCH

When *batch\_type* is set to `SlurmJobArray` (or its alias `slurmjobarray`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/train/resources[SlurmJobArray]/kwargs`

Extra arguments.

**custom\_gpu\_line:**

type: `NoneType` | str, optional, default: None

argument path:

`run_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with `#SBATCH`

When *batch\_type* is set to `DpCloudServer` (or its alias `dpcloudserver`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/train/resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to `Lebesgue` (or its alias `lebesgue`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/train/resources[Lebesgue]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to `DistributedShell` (or its alias `distributedshell`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/train/resources[DistributedShell]/kwargs`

This field is empty for this batch.

**model\_devi:**

type: dict

argument path: `run_mdata/model_devi`

Parameters of command, machine, and resources for `model_devi`

**command:**

type: str

argument path: `run_mdata/model_devi/command`

Command of a program.

**machine:**

type: dict  
argument path: run\_mdata/model\_devi/machine

**batch\_type:**

type: str  
argument path: run\_mdata/model\_devi/machine/batch\_type  
The batch job system type. Option: PBS, Lebesgue, DpCloudServer, Shell, DistributedShell, Torque, Slurm, SlurmJobArray, LSF

**local\_root:**

type: NoneType | str  
argument path: run\_mdata/model\_devi/machine/local\_root  
The dir where the tasks and relating files locate. Typically the project dir.

**remote\_root:**

type: NoneType | str, optional  
argument path: run\_mdata/model\_devi/machine/remote\_root  
The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

**clean\_asynchronously:**

type: bool, optional, default: False  
argument path: run\_mdata/model\_devi/machine/clean\_asynchronously  
Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context\_type*, different sub args are accepted.

**context\_type:**

type: str (flag key)  
argument path: run\_mdata/model\_devi/machine/context\_type  
possible choices: DpCloudServerContext, HDFSContext, LocalContext, LazyLocalContext, SSHContext, LebesgueContext  
The connection used to remote machine. Option: DpCloudServerContext, HDFSContext, LebesgueContext, LocalContext, LazyLocalContext, SSHContext

When *context\_type* is set to DpCloudServerContext (or its aliases dpcloudservercontext, DpCloudServer, dpcloudserver):

**remote\_profile:**

type: dict  
argument path:  
run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile

The information used to maintain the connection with remote machine.

**email:**

type: str  
argument path: run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile/email  
Email

**password:**

type: str  
argument path: run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile/password  
Password

**program\_id:**

type: int, alias: *project\_id*  
argument path: run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile/program\_id  
Program ID

**keep\_backup:**

type: bool, optional  
argument path: run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile/keep\_backup  
keep download and upload zip

**input\_data:**

type: dict  
argument path: run\_mdata/model\_devi/machine[DpCloudServerContext]/remote\_profile/input\_data  
Configuration of job

When *context\_type* is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

**remote\_profile:**

type: dict, optional  
argument path: run\_mdata/model\_devi/machine[HDFSContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

**remote\_profile:**

type: dict, optional



argument path: `run_mdata/model_devi/machine[LocalContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

**remote\_profile:**

type: dict, optional

argument path: `run_mdata/model_devi/machine[LazyLocalContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

**remote\_profile:**

type: dict

argument path: `run_mdata/model_devi/machine[SSHContext]/remote_profile`

The information used to maintain the connection with remote machine.

**hostname:**

type: str

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/hostname`

hostname or ip of ssh connection.

**username:**

type: str

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/username`

username of target linux system

**password:**

type: str, optional

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/password`

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

**port:**

type: int, optional, default: 22

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/port`

ssh connection port.

**key\_filename:**

type: `NoneType` | `str`, optional, default: `None`

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/key_filename`

key filename used by ssh connection. If left `None`, find key in `~/.ssh` or use password for login

**passphrase:**

type: `NoneType` | `str`, optional, default: `None`

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/passphrase`

passphrase of key used by ssh connection

**timeout:**

type: `int`, optional, default: `10`

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/timeout`

timeout of ssh connection

**totp\_secret:**

type: `NoneType` | `str`, optional, default: `None`

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

**tar\_compress:**

type: `bool`, optional, default: `True`

argument path:

`run_mdata/model_devi/machine[SSHContext]/remote_profile/tar_compress`

The archive will be compressed in upload and download if it is `True`. If not, compression will be skipped.

When *context\_type* is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

**remote\_profile:**

type: `dict`

argument path: `run_mdata/model_devi/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

**email:**

type: `str`

argument path:  
run\_mdata/model\_devi/machine[LebesgueContext]/remote\_profile/email  
Email

**password:**

type: str  
argument path:  
run\_mdata/model\_devi/machine[LebesgueContext]/remote\_profile/password  
Password

**program\_id:**

type: int, alias: *project\_id*  
argument path:  
run\_mdata/model\_devi/machine[LebesgueContext]/remote\_profile/program\_id  
Program ID

**keep\_backup:**

type: bool, optional  
argument path: run\_mdata/model\_devi/machine[LebesgueContext]/  
remote\_profile/keep\_backup  
keep download and upload zip

**input\_data:**

type: dict  
argument path:  
run\_mdata/model\_devi/machine[LebesgueContext]/remote\_profile/input\_data  
Configuration of job

**resources:**

type: dict  
argument path: run\_mdata/model\_devi/resources

**number\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/model\_devi/resources/number\_node  
The number of node need for each *job*

**cpu\_per\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/model\_devi/resources/cpu\_per\_node  
cpu numbers of each node assigned to each job.

**gpu\_per\_node:**

type: int, optional, default: 0

argument path: run\_mdata/model\_devi/resources/gpu\_per\_node

gpu numbers of each node assigned to each job.

**queue\_name:**

type: str, optional, default: ""

argument path: run\_mdata/model\_devi/resources/queue\_name

The queue name of batch job scheduler system.

**group\_size:**

type: int

argument path: run\_mdata/model\_devi/resources/group\_size

The number of *tasks* in a *job*. 0 means infinity.

**custom\_flags:**

type: list, optional

argument path: run\_mdata/model\_devi/resources/custom\_flags

The extra lines pass to job submitting script header

**strategy:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources/strategy

strategies we use to generation job submitting scripts.

**if\_cuda\_multi\_devices:**

type: bool, optional, default: False

argument path:

run\_mdata/model\_devi/resources/strategy/if\_cuda\_multi\_devices

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS.If true, dpdispatcher will manually export environment variable CUDA\_VISIBLE\_DEVICES to different task.Usually, this option will be used with Task.task\_need\_resources variable simultaneously.

**ratio\_unfinished:**

type: float, optional, default: 0.0

argument path: run\_mdata/model\_devi/resources/strategy/ratio\_unfinished

The ratio of *jobs* that can be unfinished.

**para\_deg:**

type: int, optional, default: 1

argument path: `run_mdata/model_devi/resources/para_deg`

Decide how many tasks will be run in parallel.

**source\_list:**

type: `list`, optional, default: `[]`

argument path: `run_mdata/model_devi/resources/source_list`

The env file to be sourced before the command execution.

**module\_purge:**

type: `bool`, optional, default: `False`

argument path: `run_mdata/model_devi/resources/module_purge`

Remove all modules on HPC system before module load (`module_list`)

**module\_unload\_list:**

type: `list`, optional, default: `[]`

argument path: `run_mdata/model_devi/resources/module_unload_list`

The modules to be unloaded on HPC system before submitting jobs

**module\_list:**

type: `list`, optional, default: `[]`

argument path: `run_mdata/model_devi/resources/module_list`

The modules to be loaded on HPC system before submitting jobs

**envs:**

type: `dict`, optional, default: `{}`

argument path: `run_mdata/model_devi/resources/envs`

The environment variables to be exported on before submitting jobs

**wait\_time:**

type: `int` | `float`, optional, default: `0`

argument path: `run_mdata/model_devi/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch\_type*, different sub args are accepted.

**batch\_type:**

type: `str` (flag key)

argument path: `run_mdata/model_devi/resources/batch_type`

possible choices: LSF, Shell, PBS, Torque, Slurm, SlurmJobArray, DpCloudServer, Lebesgue, DistributedShell

The batch job system type loaded from machine/*batch\_type*.

When *batch\_type* is set to LSF (or its alias *lsf*):

**kwargs:**

type: dict

argument path: `run_mdata/model_devi/resources[LSF]/kwargs`

Extra arguments.

**gpu\_usage:**

type: bool, optional, default: False

argument path: `run_mdata/model_devi/resources[LSF]/kwargs/gpu_usage`

Choosing if GPU is used in the calculation step.

**gpu\_new\_syntax:**

type: bool, optional, default: False

argument path: `run_mdata/model_devi/resources[LSF]/kwargs/gpu_new_syntax`

For LFS  $\geq$  10.1.0.3, new option `-gpu` for `#BSUB` could be used. If False, and old syntax would be used.

**gpu\_exclusive:**

type: bool, optional, default: True

argument path: `run_mdata/model_devi/resources[LSF]/kwargs/gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

**custom\_gpu\_line:**

type: `NoneType` | str, optional, default: None

argument path: `run_mdata/model_devi/resources[LSF]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with `#BSUB`

When *batch\_type* is set to `Shell` (or its alias `shell`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/model_devi/resources[Shell]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to `PBS` (or its alias `pbs`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/model_devi/resources[PBS]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to `Torque` (or its alias `torque`):

**kwargs:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources[Torque]/kwargs

This field is empty for this batch.

When *batch\_type* is set to Slurm (or its alias slurm):

**kwargs:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources[Slurm]/kwargs

Extra arguments.

**custom\_gpu\_line:**

type: NoneType | str, optional, default: None

argument path:

run\_mdata/model\_devi/resources[Slurm]/kwargs/custom\_gpu\_line

Custom GPU configuration, starting with #SBATCH

When *batch\_type* is set to SlurmJobArray (or its alias slurmjobarray):

**kwargs:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources[SlurmJobArray]/kwargs

Extra arguments.

**custom\_gpu\_line:**

type: NoneType | str, optional, default: None

argument path:

run\_mdata/model\_devi/resources[SlurmJobArray]/kwargs/custom\_gpu\_line

Custom GPU configuration, starting with #SBATCH

When *batch\_type* is set to DpCloudServer (or its alias dpcloudserver):

**kwargs:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources[DpCloudServer]/kwargs

This field is empty for this batch.

When *batch\_type* is set to Lebesgue (or its alias lebesgue):

**kwargs:**

type: dict, optional

argument path: run\_mdata/model\_devi/resources[Lebesgue]/kwargs

This field is empty for this batch.

When *batch\_type* is set to `DistributedShell` (or its alias `distributedshell`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/model_devi/resources[DistributedShell]/kwargs`

This field is empty for this batch.

**fp:**

type: dict

argument path: `run_mdata/fp`

Parameters of command, machine, and resources for fp

**command:**

type: str

argument path: `run_mdata/fp/command`

Command of a program.

**machine:**

type: dict

argument path: `run_mdata/fp/machine`

**batch\_type:**

type: str

argument path: `run_mdata/fp/machine/batch_type`

The batch job system type. Option: PBS, Lebesgue, DpCloudServer, Shell, DistributedShell, Torque, Slurm, SlurmJobArray, LSF

**local\_root:**

type: `NoneType` | str

argument path: `run_mdata/fp/machine/local_root`

The dir where the tasks and relating files locate. Typically the project dir.

**remote\_root:**

type: `NoneType` | str, optional

argument path: `run_mdata/fp/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

**clean\_asynchronously:**

type: bool, optional, default: False

argument path: `run_mdata/fp/machine/clean_asynchronously`



Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context\_type*, different sub args are accepted.

**context\_type:**

type: str (flag key)

argument path: run\_mdata/fp/machine/context\_type

possible choices: DpCloudServerContext, HDFSContext, LocalContext, LazyLocalContext, SSHContext, LebesgueContext

The connection used to remote machine. Option: DpCloudServerContext, HDFSContext, LebesgueContext, LocalContext, LazyLocalContext, SSHContext

When *context\_type* is set to DpCloudServerContext (or its aliases dpcloudservercontext, DpCloudServer, dpcloudserver):

**remote\_profile:**

type: dict

argument path: run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile

The information used to maintain the connection with remote machine.

**email:**

type: str

argument path:

run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile/email

Email

**password:**

type: str

argument path:

run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile/password

Password

**program\_id:**

type: int, alias: *project\_id*

argument path:

run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile/program\_id

Program ID

**keep\_backup:**

type: bool, optional

argument path:

run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile/keep\_backup

keep download and upload zip

**input\_data:**

type: dict

argument path:

run\_mdata/fp/machine[DpCloudServerContext]/remote\_profile/input\_data

Configuration of job

When *context\_type* is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/fp/machine[HDFSContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/fp/machine[LocalContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

**remote\_profile:**

type: dict, optional

argument path: run\_mdata/fp/machine[LazyLocalContext]/remote\_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When *context\_type* is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

**remote\_profile:**

type: dict

argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile

The information used to maintain the connection with remote machine.

**hostname:**

type: str

argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/hostname

hostname or ip of ssh connection.

**username:**

type: str  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/username  
username of target linux system

**password:**

type: str, optional  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/password  
(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

**port:**

type: int, optional, default: 22  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/port  
ssh connection port.

**key\_filename:**

type: NoneType | str, optional, default: None  
argument path:  
run\_mdata/fp/machine[SSHContext]/remote\_profile/key\_filename  
key filename used by ssh connection. If left None, find key in ~/.ssh or use password for login

**passphrase:**

type: NoneType | str, optional, default: None  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/passphrase  
passphrase of key used by ssh connection

**timeout:**

type: int, optional, default: 10  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/timeout  
timeout of ssh connection

**totp\_secret:**

type: NoneType | str, optional, default: None  
argument path: run\_mdata/fp/machine[SSHContext]/remote\_profile/totp\_secret  
Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

**tar\_compress:**

type: bool, optional, default: True  
argument path:  
run\_mdata/fp/machine[SSHContext]/remote\_profile/tar\_compress  
The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When *context\_type* is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

**remote\_profile:**

type: dict

argument path: `run_mdata/fp/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

**email:**

type: str

argument path: `run_mdata/fp/machine[LebesgueContext]/remote_profile/email`

Email

**password:**

type: str

argument path:

`run_mdata/fp/machine[LebesgueContext]/remote_profile/password`

Password

**program\_id:**

type: int, alias: *project\_id*

argument path:

`run_mdata/fp/machine[LebesgueContext]/remote_profile/program_id`

Program ID

**keep\_backup:**

type: bool, optional

argument path:

`run_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup`

keep download and upload zip

**input\_data:**

type: dict

argument path:

`run_mdata/fp/machine[LebesgueContext]/remote_profile/input_data`

Configuration of job

**resources:**

type: dict

argument path: `run_mdata/fp/resources`

**number\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/fp/resources/number\_node  
The number of node need for each *job*

**cpu\_per\_node:**

type: int, optional, default: 1  
argument path: run\_mdata/fp/resources/cpu\_per\_node  
cpu numbers of each node assigned to each job.

**gpu\_per\_node:**

type: int, optional, default: 0  
argument path: run\_mdata/fp/resources/gpu\_per\_node  
gpu numbers of each node assigned to each job.

**queue\_name:**

type: str, optional, default: ""  
argument path: run\_mdata/fp/resources/queue\_name  
The queue name of batch job scheduler system.

**group\_size:**

type: int  
argument path: run\_mdata/fp/resources/group\_size  
The number of *tasks* in a *job*. 0 means infinity.

**custom\_flags:**

type: list, optional  
argument path: run\_mdata/fp/resources/custom\_flags  
The extra lines pass to job submitting script header

**strategy:**

type: dict, optional  
argument path: run\_mdata/fp/resources/strategy  
strategies we use to generation job submitting scripts.

**if\_cuda\_multi\_devices:**

type: bool, optional, default: False  
argument path: run\_mdata/fp/resources/strategy/if\_cuda\_multi\_devices  
If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS.If true, dpdispatcher will manually export environment variable CUDA\_VISIBLE\_DEVICES to different task.Usually, this option will be used with Task.task\_need\_resources variable simultaneously.

**ratio\_unfinished:**

type: float, optional, default: 0.0

argument path: run\_mdata/fp/resources/strategy/ratio\_unfinished

The ratio of *jobs* that can be unfinished.

**para\_deg:**

type: int, optional, default: 1

argument path: run\_mdata/fp/resources/para\_deg

Decide how many tasks will be run in parallel.

**source\_list:**

type: list, optional, default: []

argument path: run\_mdata/fp/resources/source\_list

The env file to be sourced before the command execution.

**module\_purge:**

type: bool, optional, default: False

argument path: run\_mdata/fp/resources/module\_purge

Remove all modules on HPC system before module load (module\_list)

**module\_unload\_list:**

type: list, optional, default: []

argument path: run\_mdata/fp/resources/module\_unload\_list

The modules to be unloaded on HPC system before submitting jobs

**module\_list:**

type: list, optional, default: []

argument path: run\_mdata/fp/resources/module\_list

The modules to be loaded on HPC system before submitting jobs

**envs:**

type: dict, optional, default: {}

argument path: run\_mdata/fp/resources/envs

The environment variables to be exported on before submitting jobs

**wait\_time:**

type: int | float, optional, default: 0

argument path: run\_mdata/fp/resources/wait\_time

The waiting time in second after a single *task* submitted

Depending on the value of *batch\_type*, different sub args are accepted.

**batch\_type:**

type: str (flag key)

argument path: run\_mdata/fp/resources/batch\_type

possible choices: LSF, Shell, PBS, Torque, Slurm, SlurmJobArray, DpCloudServer, Lebesgue, DistributedShell

The batch job system type loaded from machine/batch\_type.

When *batch\_type* is set to LSF (or its alias *lsf*):

**kwargs:**

type: dict

argument path: run\_mdata/fp/resources[LSF]/kwargs

Extra arguments.

**gpu\_usage:**

type: bool, optional, default: False

argument path: run\_mdata/fp/resources[LSF]/kwargs/gpu\_usage

Choosing if GPU is used in the calculation step.

**gpu\_new\_syntax:**

type: bool, optional, default: False

argument path: run\_mdata/fp/resources[LSF]/kwargs/gpu\_new\_syntax

For LFS >= 10.1.0.3, new option -gpu for #BSUB could be used. If False, and old syntax would be used.

**gpu\_exclusive:**

type: bool, optional, default: True

argument path: run\_mdata/fp/resources[LSF]/kwargs/gpu\_exclusive

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

**custom\_gpu\_line:**

type: NoneType | str, optional, default: None

argument path: run\_mdata/fp/resources[LSF]/kwargs/custom\_gpu\_line

Custom GPU configuration, starting with #BSUB

When *batch\_type* is set to Shell (or its alias *shell*):

**kwargs:**

type: dict, optional

argument path: run\_mdata/fp/resources[Shell]/kwargs

This field is empty for this batch.

When *batch\_type* is set to PBS (or its alias pbs):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[PBS]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to Torque (or its alias torque):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[Torque]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to Slurm (or its alias slurm):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[Slurm]/kwargs`

Extra arguments.

**custom\_gpu\_line:**

type: `NoneType` | str, optional, default: None

argument path: `run_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with #SBATCH

When *batch\_type* is set to SlurmJobArray (or its alias slurmjobarray):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[SlurmJobArray]/kwargs`

Extra arguments.

**custom\_gpu\_line:**

type: `NoneType` | str, optional, default: None

argument path:

`run_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with #SBATCH

When *batch\_type* is set to DpCloudServer (or its alias dpcloudserver):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[DpCloudServer]/kwargs`



This field is empty for this batch.

When *batch\_type* is set to `Lebesgue` (or its alias `lebesgue`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[Lebesgue]/kwargs`

This field is empty for this batch.

When *batch\_type* is set to `DistributedShell` (or its alias `distributedshell`):

**kwargs:**

type: dict, optional

argument path: `run_mdata/fp/resources[DistributedShell]/kwargs`

This field is empty for this batch.



**THIS IS A BLANK FILE TO TEST SPHINX-BUILD**



## HOW TO CONTRIBUTE

- If you want to add the documentation of a toy model, simply put your file in the directory `doc/toymodels/` and push;
- If you want to add a new directory for a new category of instructions, make a new directory and add it in `doc/index.rst`.
- `genindex`
- `modindex`
- `search`