## Table of Contents


Table of Contents 1

Job Manager for local execution of QuantumATK scripts 2

- Execute QuantumATK simualtions via the Job Manager 2
- Serial execution 8
- Threading 10
- MPI parallelization 13
  - Running from the command line 14
- Machine Manager 15

# Job Manager for local execution of QuantumATK scripts

**Version:** 2017.0

**Downloads & Links**

- PDF version
- PDF version for QuantumATK 2016
- silicon.py
- cnt.py
- mpi_check.py

Table of Contents:

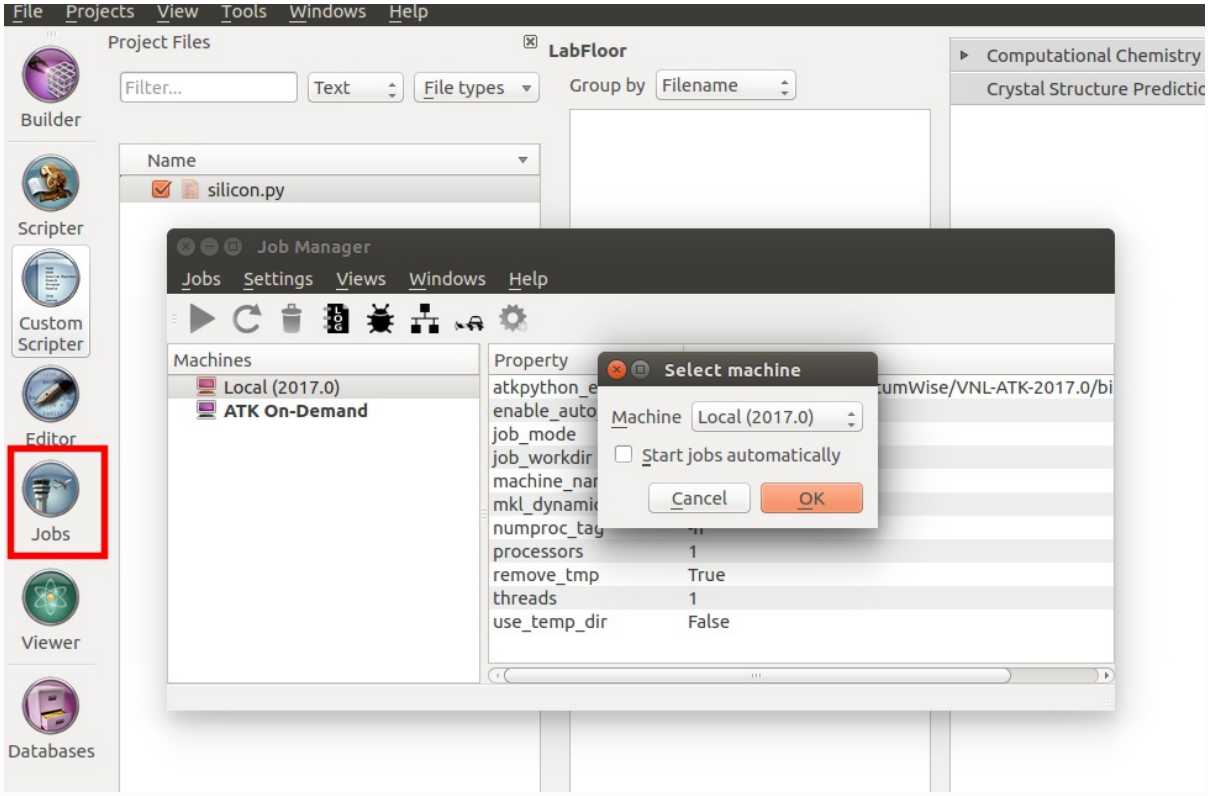

- Execute QuantumATK simualtions via the Job Manager
- Serial execution
- Threading
- MPI parallelization
- Machine Manager


## Execute QuantumATK simualtions via the Job Manager

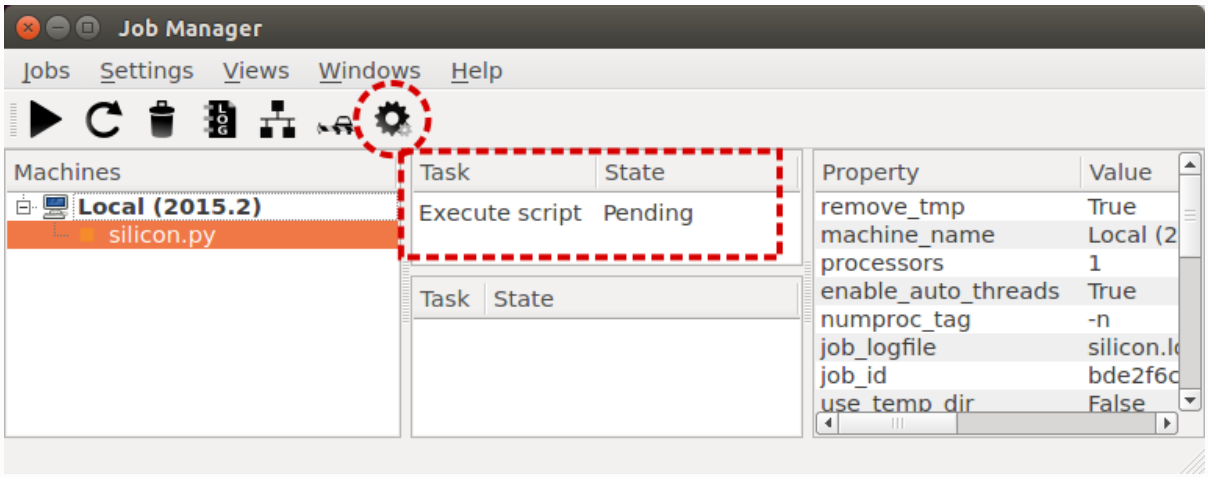
In this section you will learn how to use the **Job Manager** for local execution of QuantumATK scripts. Specifically, you will learn about queuing, running and managing QuantumATK jobs.

Create a new empty project and download the example script silicon.py, which runs an ATK-DFT calculation with very many k-points (31x31x31).

Drop it on the **Job Manager** and select a local machine for the job execution.




The job is now in the task state “pending” with “Threaded parallel (Single process)” default settings. Click the **Job Settings** icon to edit the job settings.



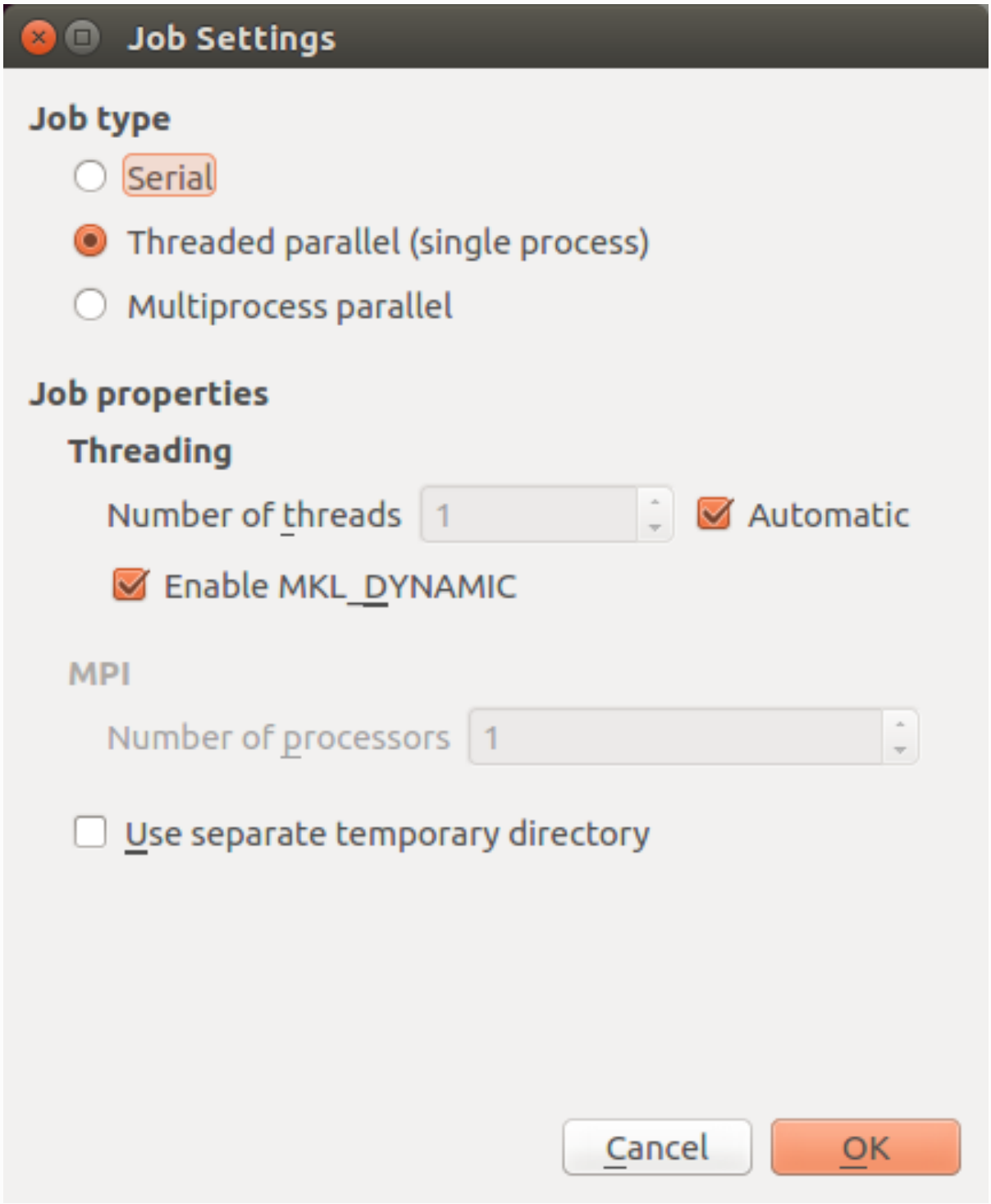

The **Job Settings** widget has three basic panels:

- **Job type;**
  - Serial
  - Threaded parallel (single process)
  - Multiprocess parallel

- **Job properties;**
  - Threading
  - MPI

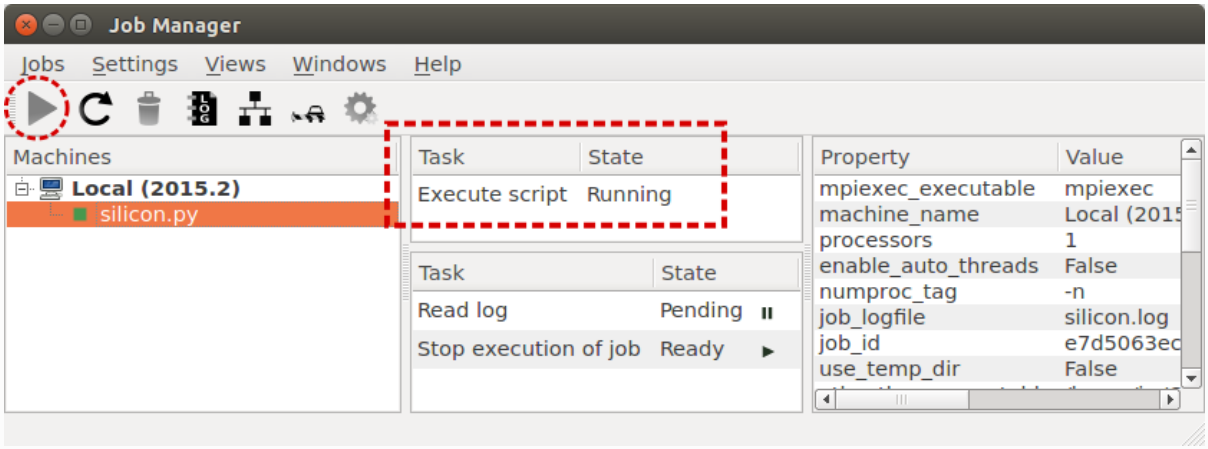
- **Use separate temporary directory** (set this to allow using a non-default working directory for running the job. Note, results will not be appended to existing hdf5 or nc files, but instead a new file

will be created.)

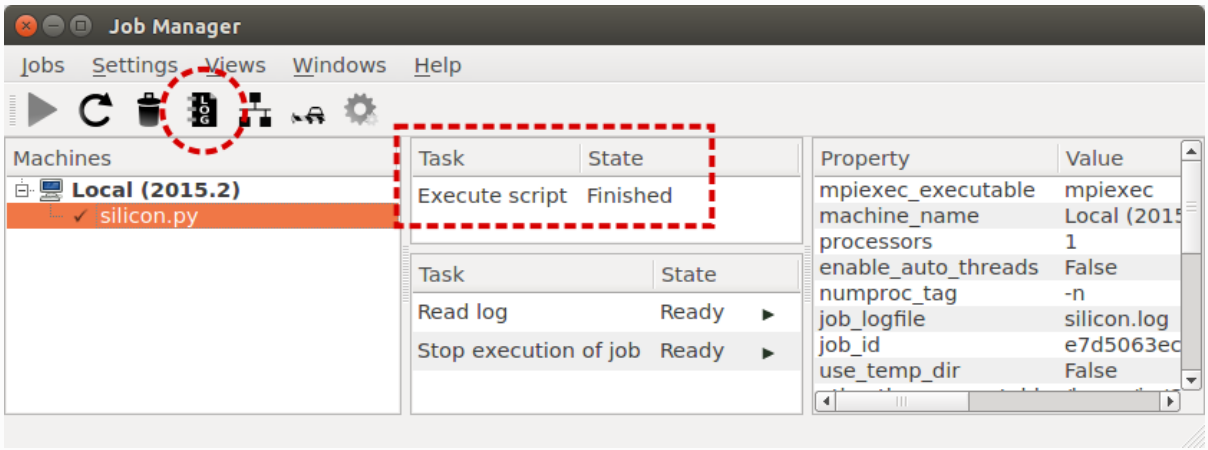


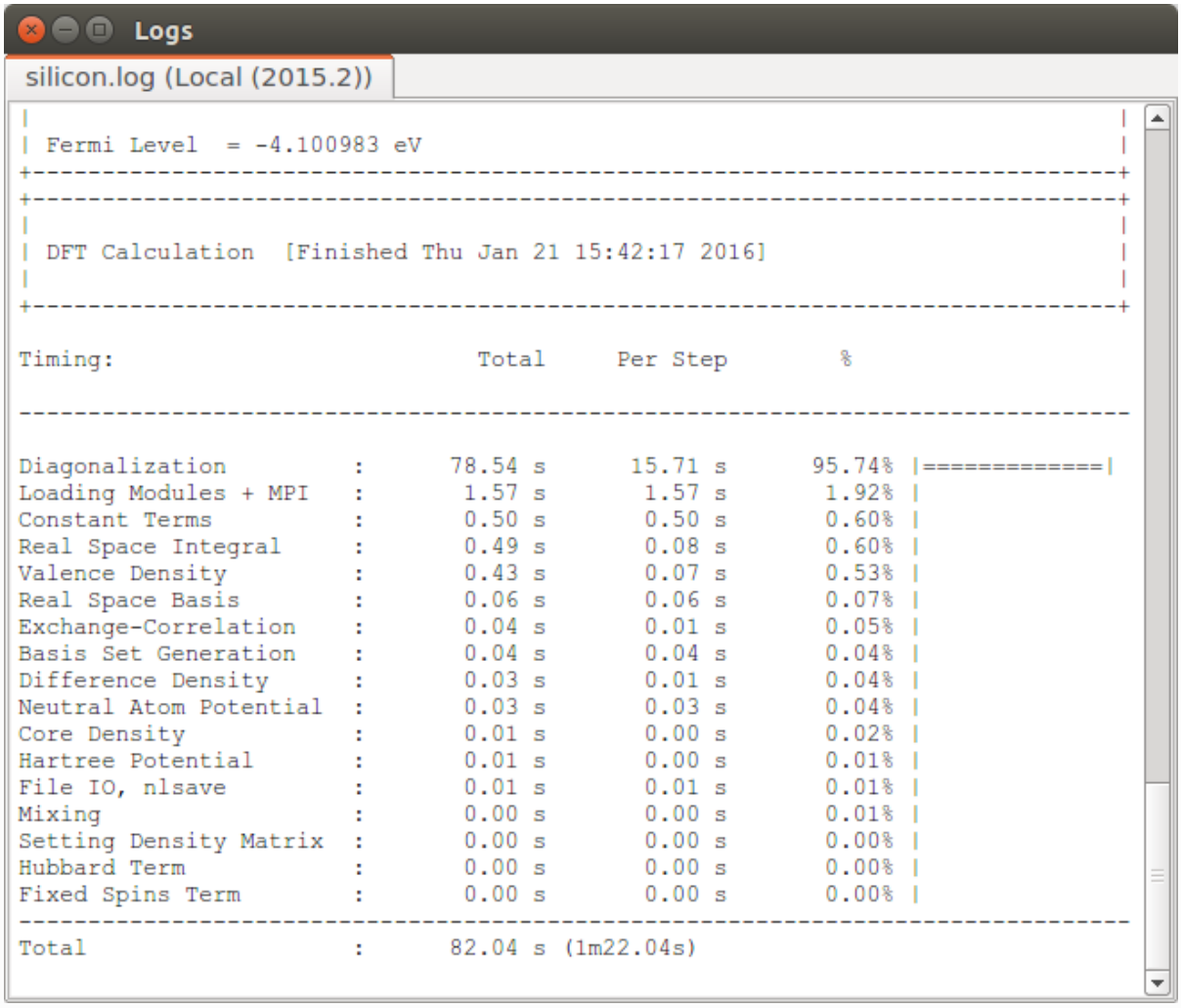

Set these settings according to your needs, and click **OK**.

Back in the **Job Manager**, click the **Run** ▶ icon to start the job. The task state changes from “Pending” to “Running”.

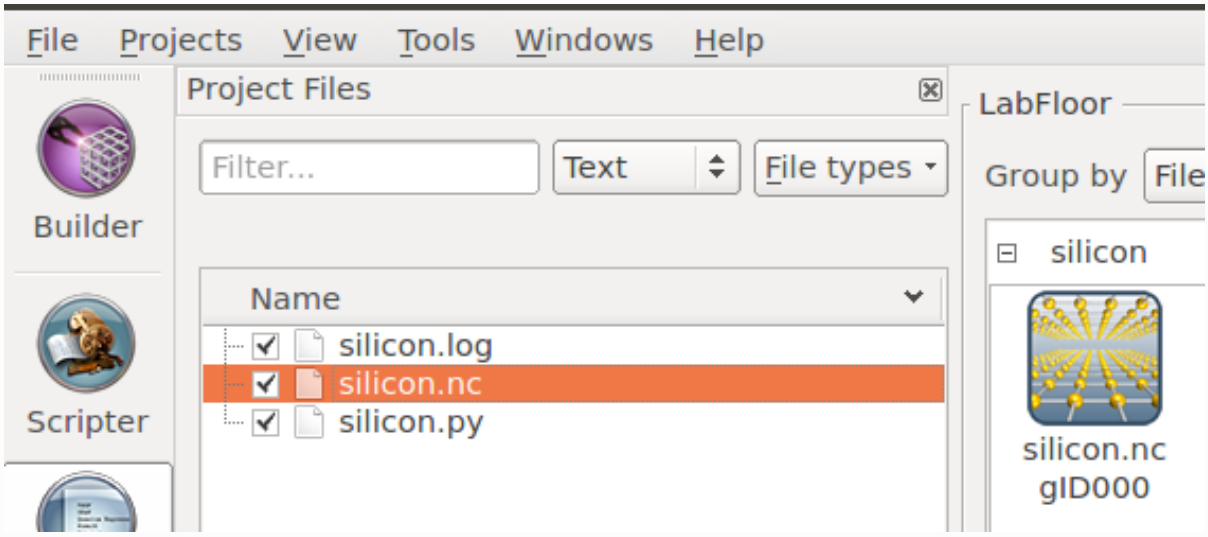



The job finishes after ca. 1 minute (2.5 GHz CPU). Note that the task state changes to “Finished”. You can inspect the job log file by clicking the **LOG** icon: 



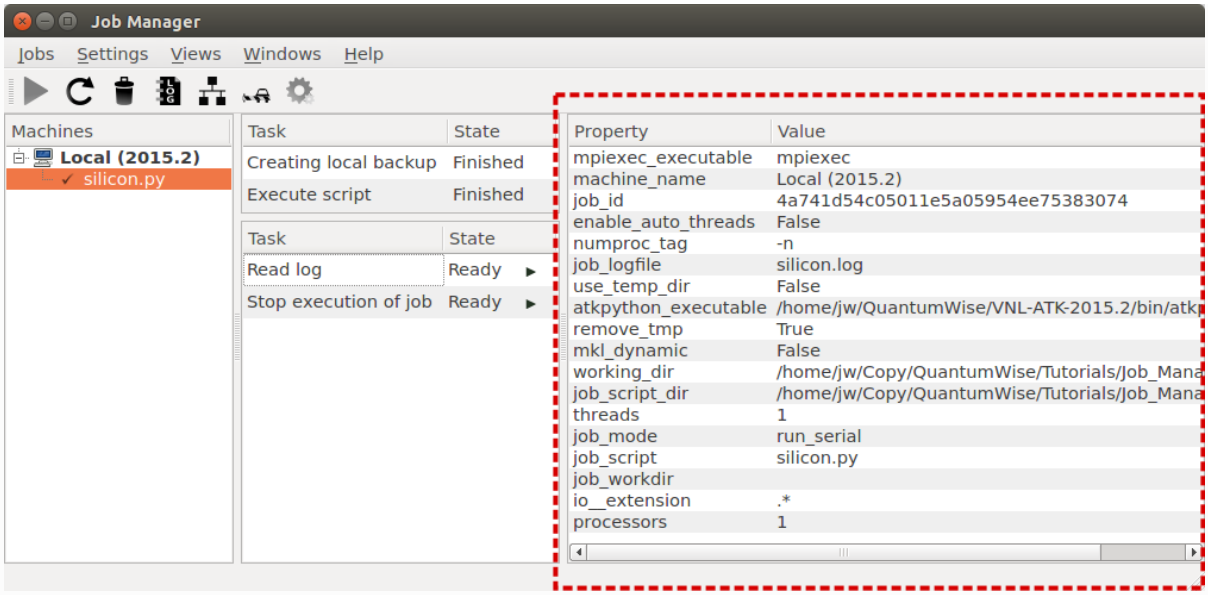



The job output of course appears on the QuantumATK **LabFloor** after job execution.

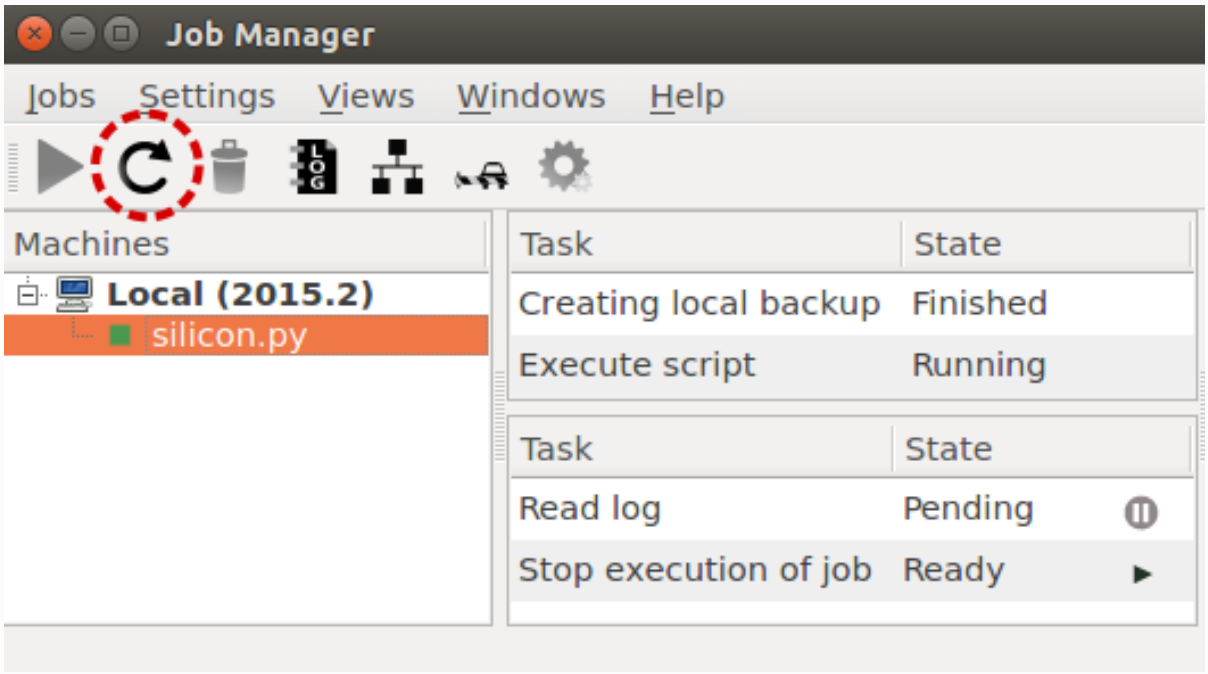



Back in the **Job Manager**, the Property–Value list shows all details of the settings used for job execution, including

- path to the QuantumATK executable;
- name of the Python script and the log file.
- threading and other prallelization options.



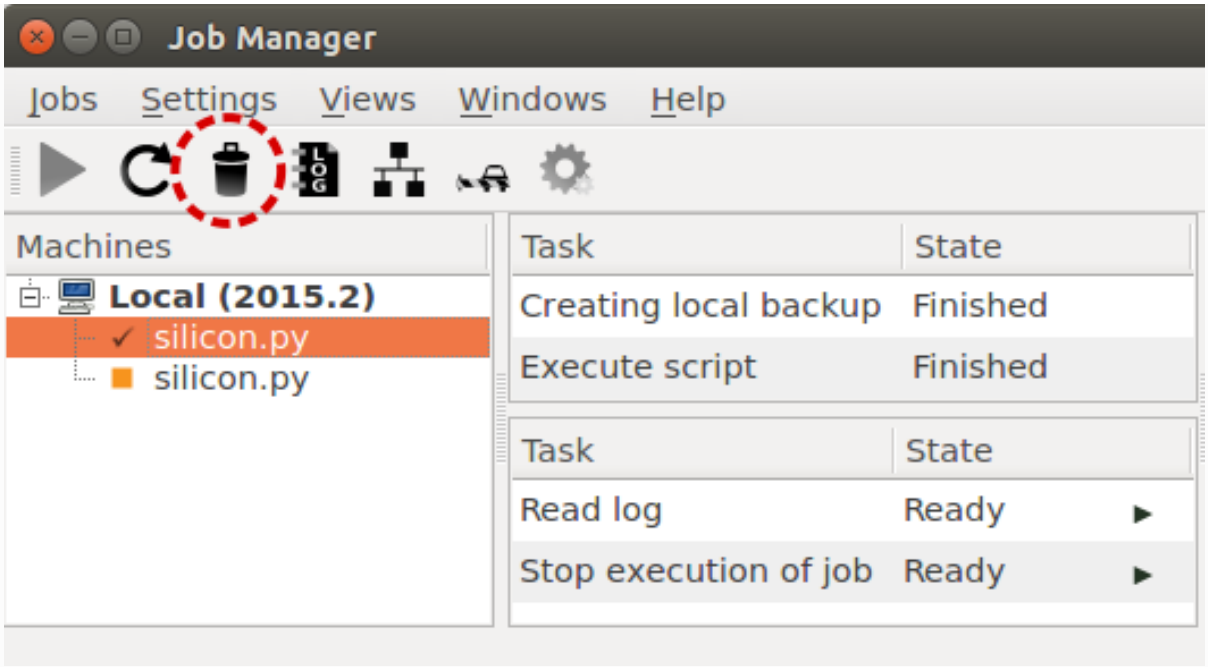

You can use the **Resubmit** icon to resubmit a script. Note that any changes that have been made to the script will be picked up by the new job.




**Note**

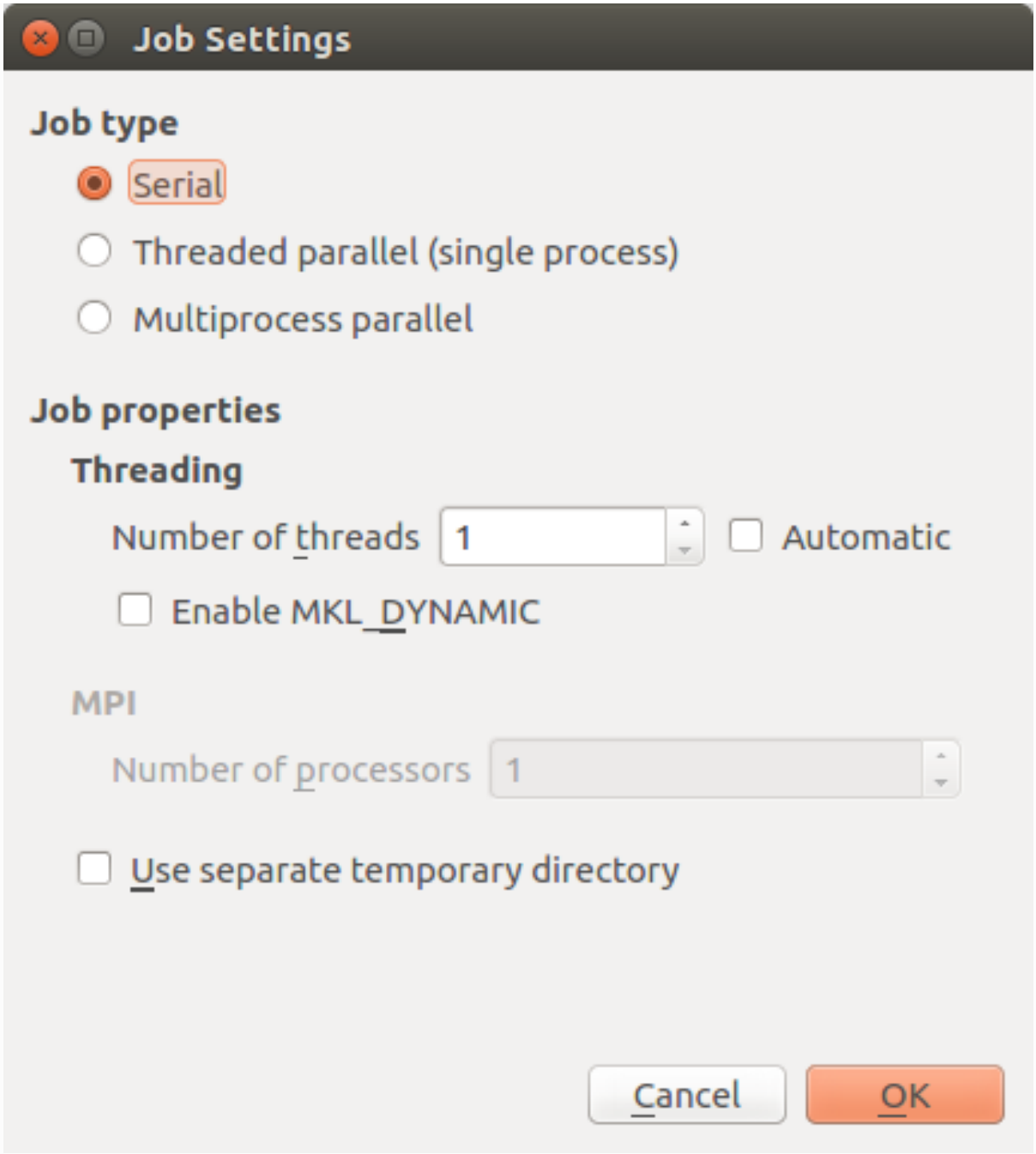
Remember that the default job type is “Threaded parallel (Single process)”. You can change this to “Serial” or “Multiprocess parallel” before starting the job.

Use the **Trash** icon to remove jobs from the job queue.

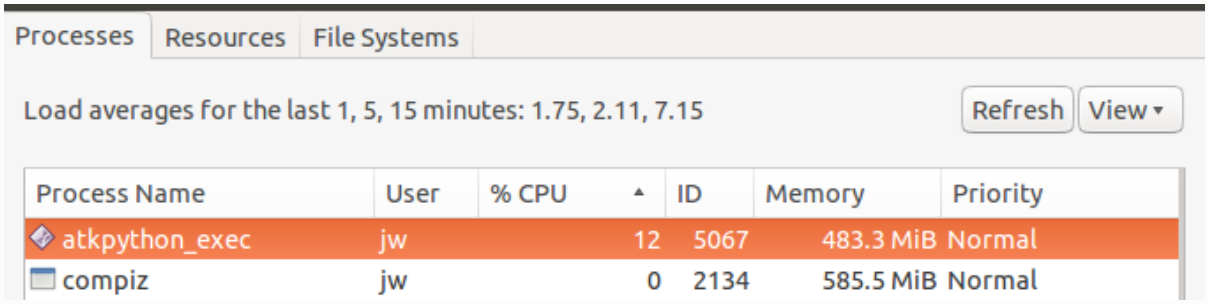



## Serial execution

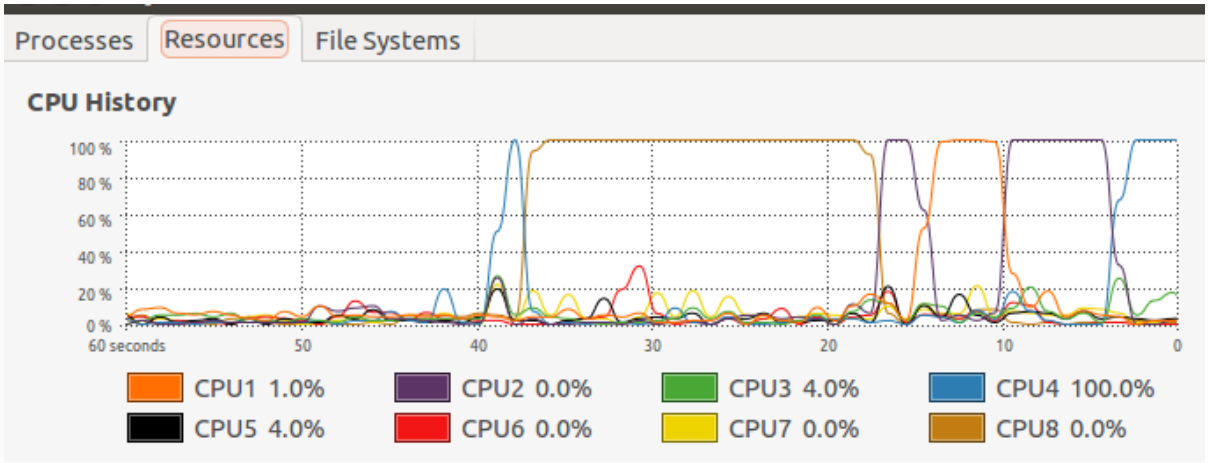
In the **Job Settings** window select a **Serial** job type as shown in the below figure to run on a single porcess with no threading. In fact, note that threading is turned off (number of threads is 1), and MPI parallelization is not available.




If you check the system load during local execution in serial, you should see that the serial job launches only a single computing task on a single CPU core.

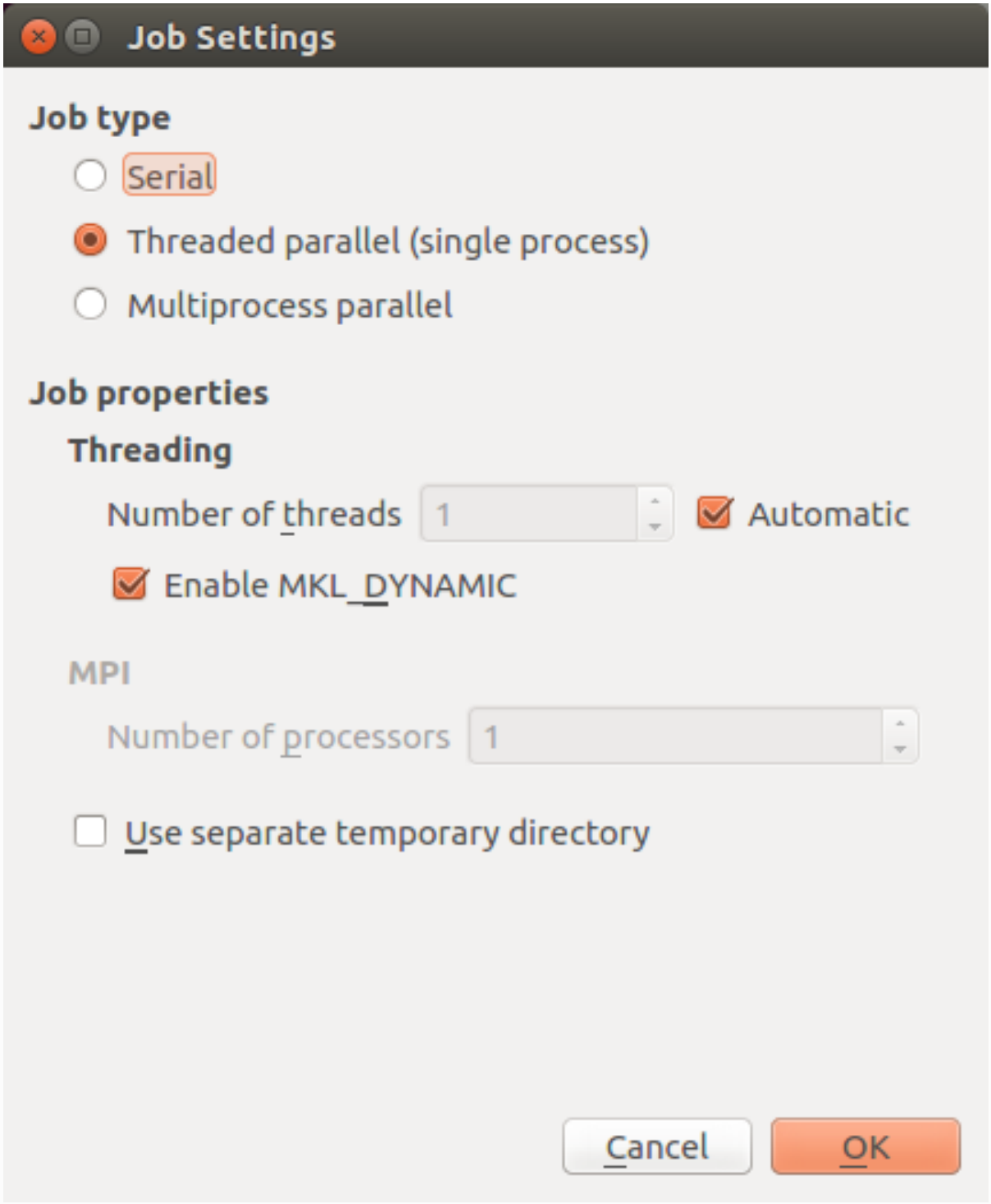



Only one core is used at a time, but the hardware process manager may move the task between cores from time to time.




## Threading

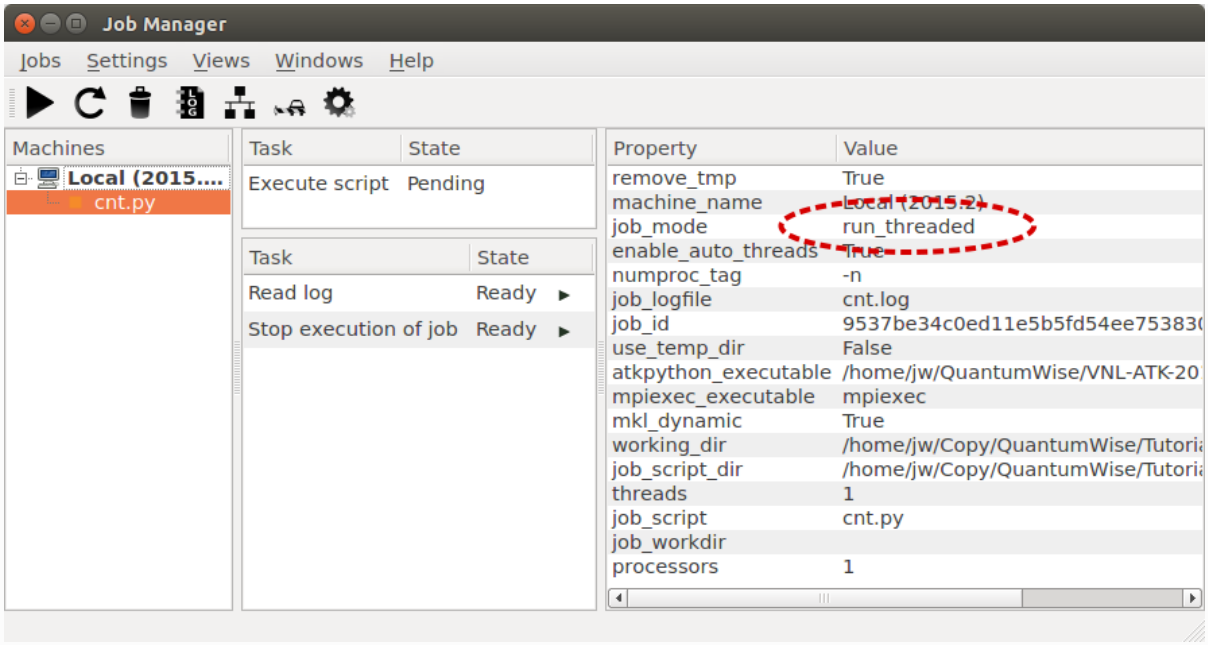
In the **Job Settings** window select a **Threaded parallel (single process)** job type as shown in the below figure to run on a single process with threading.



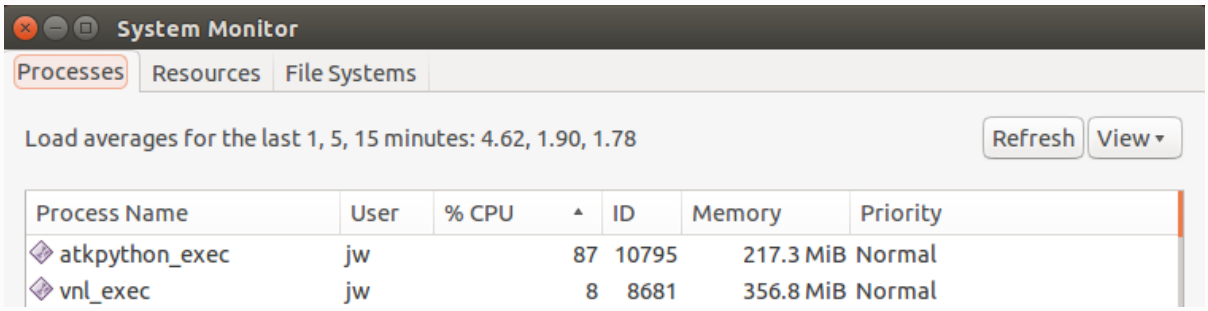

Threading is one way to parallelize a computational job. QuantumATK uses Intel MKL for openMP threading. Note that we do in general recommend MPI parallelization over threading for parallelizing DFT calculations. However, threading is often more efficient for parallelizing **ATK-ForceField** calculations.

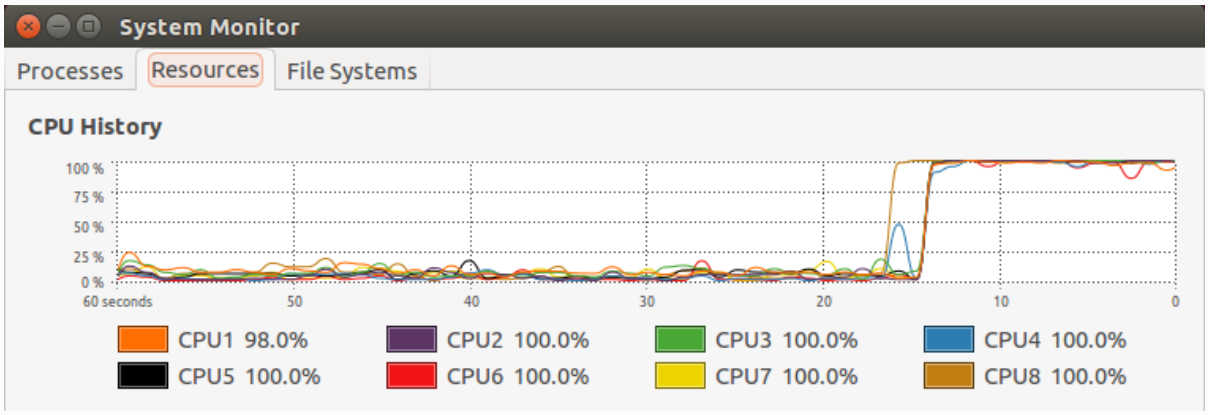
Download the script cnt.py, which uses ATK-ForceField to calculate the dynamical matrix of a multiwall carbon nanotube.

Execute it using the **Job Manager**, and choose job type “Threaded parallel (single process)”. It should be pretty fast.




If you check the system load during execution of the calculation, you should see that only a single **atkpython** process is started, even though several cores appear to be busy. This is because the work load of the one process is split into a number of threads that may be distributed on more cores.




Download the script cnt.py to test the performance of a ATK-ForceField simulations using threading. This specific example will calculate the dynamical matrix of a multiwall carbon nanotube. If you also run the calculation in serial, you will see that the wall-clock time used for evaluating ATK-ForceField forces may decrease significantly when threading is switched on. In the example shown below, the time spent on force calculations is roughly halved.

```
==> cnt_threading.log <==
Timing:
--------------------------------------------------------------------------------
Forces                  :      14.22 s       0.00 s      60.63% |=============|
Loading Modules + MPI   :       1.57 s       1.57 s       6.69% |=|
File IO, nlsave         :       0.10 s       0.05 s       0.43% |
--------------------------------------------------------------------------------

==> cnt_serial.log <==
Timing:
--------------------------------------------------------------------------------
Forces                  :      22.37 s       0.01 s      79.53% |=============|
Loading Modules + MPI   :       1.55 s       1.55 s       5.52% ||
File IO, nlsave         :       0.06 s       0.03 s       0.21% |
--------------------------------------------------------------------------------
```

## MPI parallelization

**Important**

If you are running **ATK 2016 or earlier** you need to have MPI installed on your local machine.

Both the **Linux** and **Windows** versions are compiled against Intel MPI library. Since **ATK 2017** Intel's `mpiexec.hydra` is provided on both Windows and Linux versios - **this is the recommended way to run QuantumATK in parallel**.

In **Job Settings** choose *Multiprocess parallel* and e.g. 4 MPI processes.

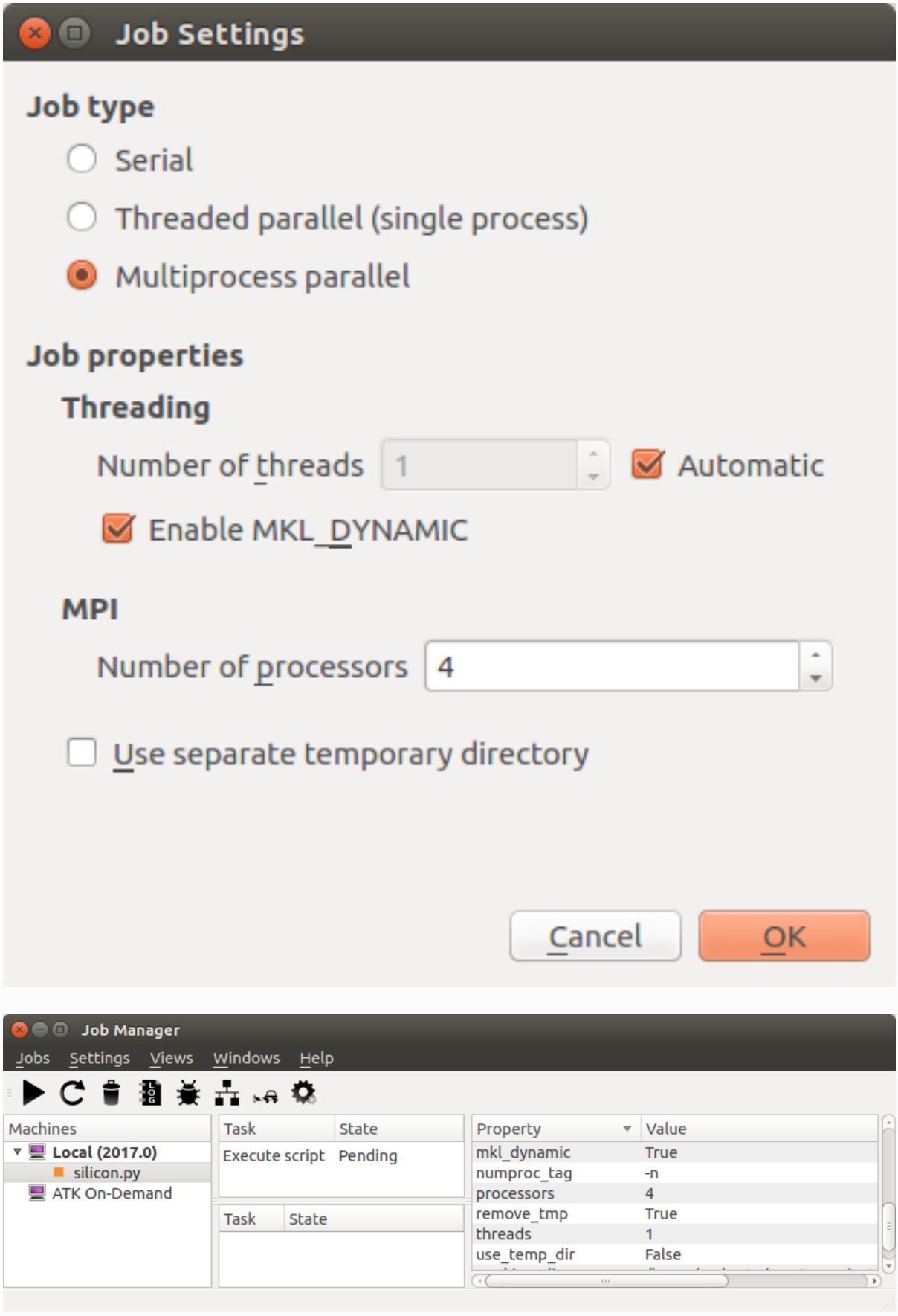



Fig. 22 The Property–Value list shows the name of the MPI executable and that 4 processors are used for MPI.

## Running from the command line

If you wish to run QuantumATK in parallel from the command line you can use the `mpiexec.hydra` binary

shipped with QuantumATK and located in the folder `libexec/mpiexec.hydra` present in your installation folder.

In this case you can run parallel jobs with:

```
$ QW_INSTALLATION_PATH/libexec/mpiexec.hydra -n 4 atkpython atk_script.py
```

**Hint**

Prepend `QW_INSTALLATION_PATH/libexec/` in your `PATH`.

```
$ export PATH=QW_INSTALLATION_PATH/libexec:$PATH
```

This way you will automatically pick up the `mpiexec.hydra` binary shipped with QuantumATK:

```
$ mpiexec.hydra -n 4 atkpython atk_script.py
```

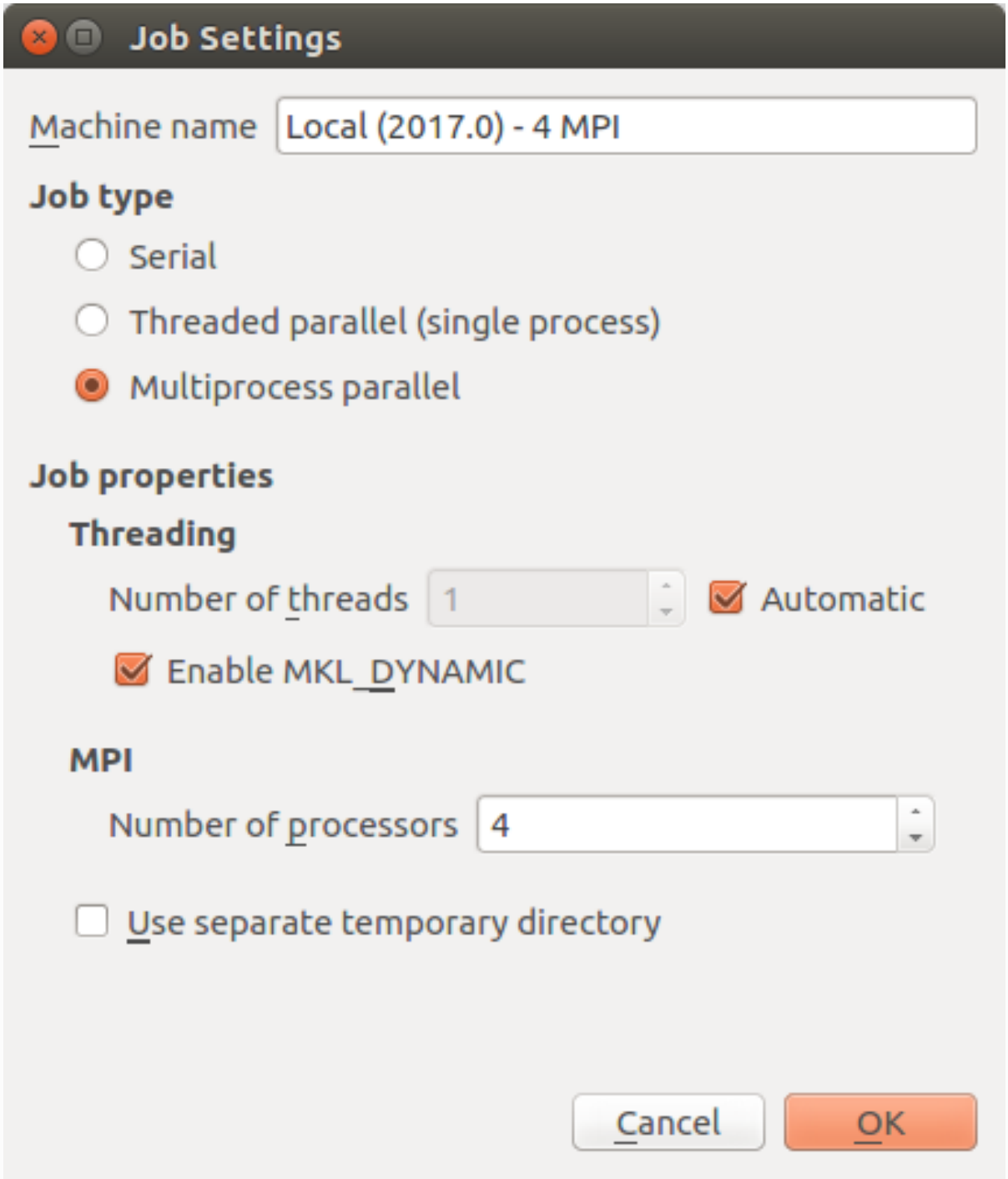
## Machine Manager

It may sometimes be convenient to have a predefined local machine that is set up with MPI parallelization as default mode. You can easily add such a machine yourself.

In the Job Manager main window, click  to open the **Machine Manager**, and click New ▸ Local.

Then edit the default job settings of the new machine in the window that pops up:

- Name the machine, e.g. “Local (2017.0) - 4 MPI”.
- Select *Multiprocess Parallel* as job type.
- Make sure threading is turned off (*Number of threads* = 1)
- Choose the default number of processors, e.g. 4.
- Click *OK* to add the new machine to the Machine Manager.


Job Settings
Machine name
Local (2017.0) - 4 MPI
Job type
Serial
Threaded parallel (single process)
Multiprocess parallel
Job properties
Threading
Number of threads
1
Automatic
Enable MKL_DYNAMIC
MPI
Number of processors
4
Use separate temporary directory
Cancel
OK

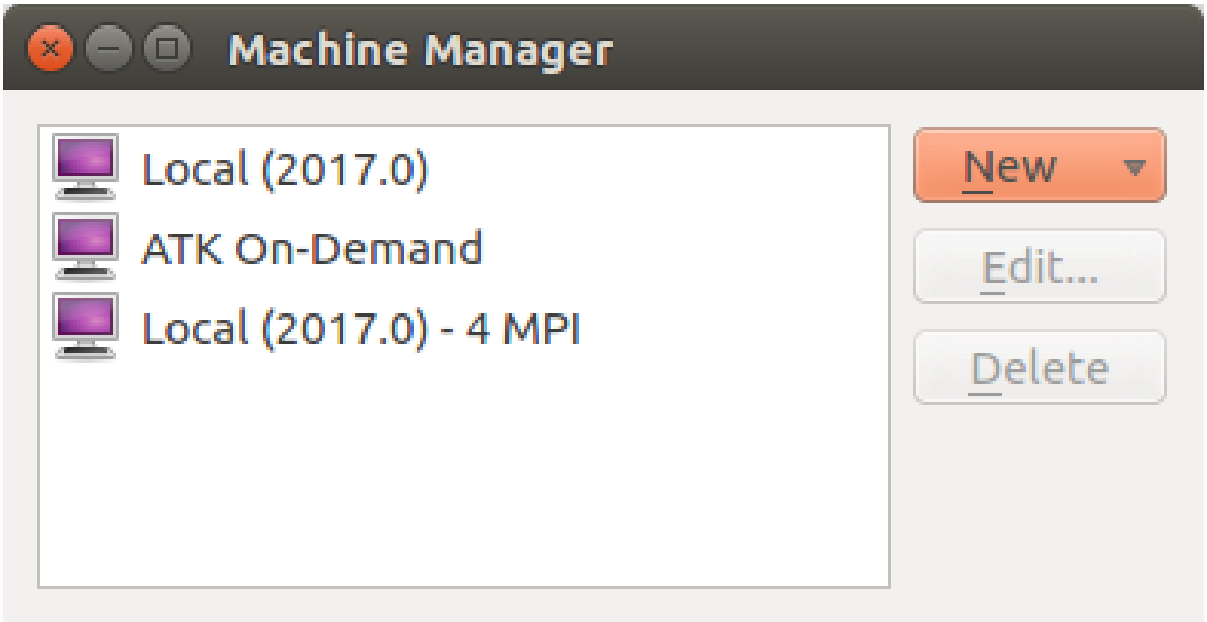



Fig. 23 You can add as many custom machines to the Machine Manager as you like.

Previous Next