

Batch System - Introduction

Andreas Baer, SCC, KIT



Reference: bwHPC Wiki

- All information given in this talk can be found at wiki.bwhpc.de:

- Batch Jobs



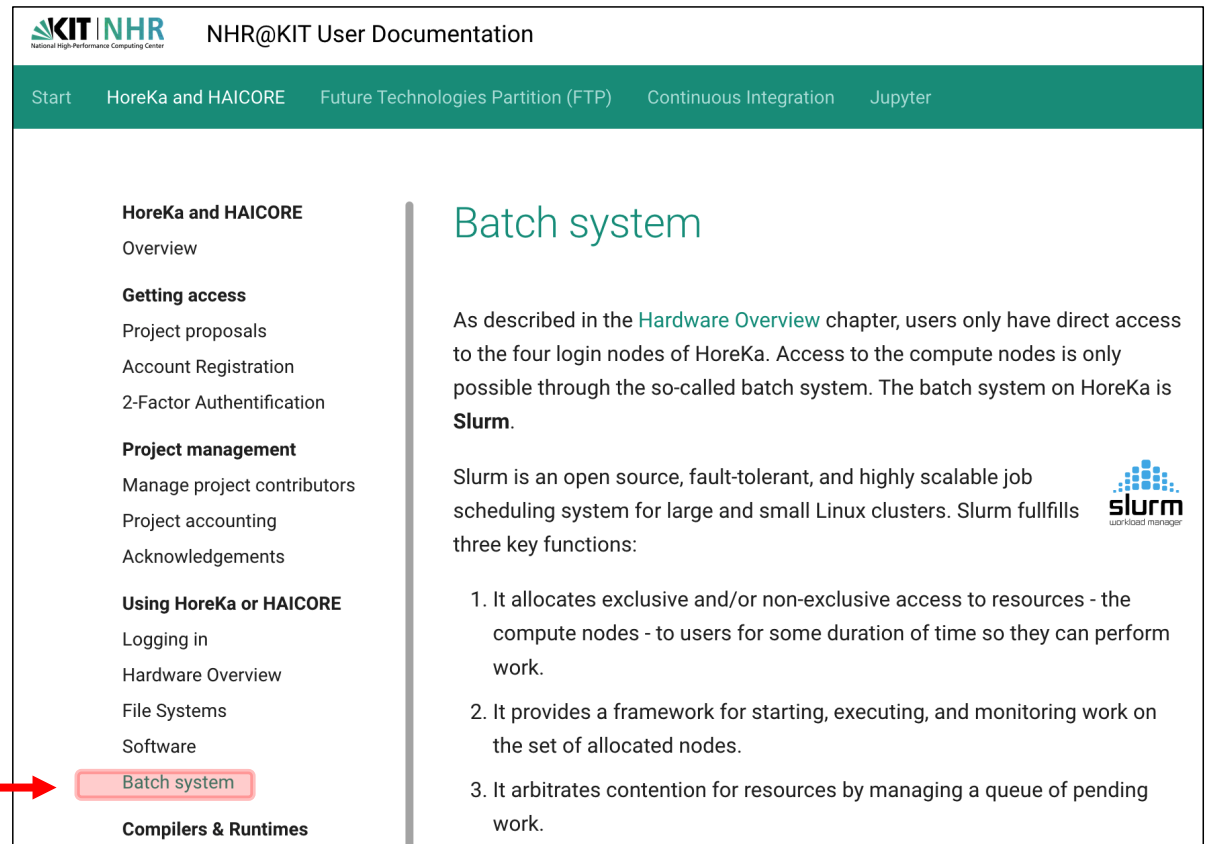
A screenshot of the bwHPC Wiki Main Page. The page features a header with navigation links: 'Main page', 'Discussion', 'Read', 'View source', and a search box labeled 'Search bwHPC Wiki'. The main content area is titled 'Main Page' and contains a large graphic of a map of Baden-Württemberg with several server rack icons and user avatars scattered across it. Below the map is the text 'bwHPC Wiki'. On the left side, there is a sidebar with the bwHPC logo and a list of links. The 'Batch Jobs' link is highlighted with a red box, and a red arrow points to it from the left. The sidebar links include: Home, Best Practices, bwHPC News, Wiki help, Best Practice Guides, Overview, -- Batch Jobs, -- Software Modules, -- Compiler, -- Numerical Libraries, -- Parallel Programming, bwHPC tier 3, bwUniCluster_2.0, bwForCluster JUSTUS_2, bwForCluster MLS&WISO, bwForCluster NEMO, and bwForCluster BinAC.

Reference: [NHR@KIT](https://www.nhr.kit.edu/userdocs/horeka/batch/) User documentation

All information given in this talk can be found at

<https://www.nhr.kit.edu/userdocs/horeka/batch/>:

■ Batch Jobs



KIT | NHR
National High-Performance Computing Center

NHR@KIT User Documentation

Start HoreKa and HAICORE Future Technologies Partition (FTP) Continuous Integration Jupyter

HoreKa and HAICORE
Overview

Getting access
Project proposals
Account Registration
2-Factor Authentication

Project management
Manage project contributors
Project accounting
Acknowledgements

Using HoreKa or HAICORE
Logging in
Hardware Overview
File Systems
Software

Batch system


Compilers & Runtimes

Batch system

As described in the [Hardware Overview](#) chapter, users only have direct access to the four login nodes of HoreKa. Access to the compute nodes is only possible through the so-called batch system. The batch system on HoreKa is **Slurm**.

Slurm is an open source, fault-tolerant, and highly scalable job scheduling system for large and small Linux clusters. Slurm fulfills three key functions:

1. It allocates exclusive and/or non-exclusive access to resources - the compute nodes - to users for some duration of time so they can perform work.
2. It provides a framework for starting, executing, and monitoring work on the set of allocated nodes.
3. It arbitrates contention for resources by managing a queue of pending work.



Material: Slides & Scripts

- https://indico.scc.kit.edu/e/hpc_course_2023-10-16
- @bwUniCluster: /opt/bwhpc/common/workshops/2023-10-16
- @HoreKa: /software/all/workshop/2023-10-16

How to read the following slides

Abbreviation/Colour code	Full meaning
<code>\$ command -option value</code>	<code>\$</code> = prompt of the interactive shell The full prompt may look like: <code>user@machine:path\$</code> The command has been entered in the interactive shell session
<code><integer></code> <code><string></code>	<code><></code> = Placeholder for integer, string etc

Batch System

Resource management

- Jobs are **NOT** executed by the user
- Instead, there is a management system (Batch System)
 - **workload manager (scheduler)**
 - scheduling, managing, monitoring, reporting
 - SLURM (HoreKa, bwUniCluster 2.0, 2x bwForCluster)
 - MOAB (2x bwForCluster)
 - **resource manager**
 - control over jobs and distributed compute nodes
 - SLURM (HoreKa, bwUniCluster 2.0, 2x bwForCluster)
 - TORQUE (2x bwForCluster)

Resource and workload manager (1)

(1) User creates a **job script** and submits it to Slurm via the “**sbatch**” command

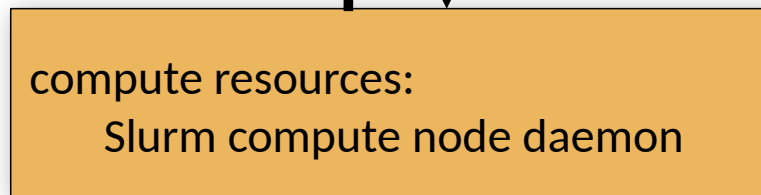
```
#!/bin/bash
#SBATCH -p dev_cpuonly
#SBATCH -N 1 -n 1
#SBATCH -t 00:01:00
#SBATCH -mem-per-cpu=500

./your_simulation
```

(2) Slurm parses the job script:
→ where & when to run job



(3) Job execution:
delegated to resource
manager on the node



(4) The resource manager executes the job and communicates status information to nodes

Resource and workload manager (2)

■ All clusters:

- compute job will **only** be processed by the batch system
- Running **jobs** on **login nodes** **not** allowed

■ Waiting time:

- depends on:
 - your job demands
 - your demand history
 - ONLY bwUniCluster 2.0: your university's share

Job's life cycle

■ 1. Preprocessing: Setup `job_script.sh`:

```
#!/bin/bash
#SBATCH -p dev_single
#SBATCH -N 1 -n 1
#SBATCH -t 00:01:00
#SBATCH -mem-per-cpu=500

./your_simulation
```

1) Options for the job

2) Actual work to be executed on the cluster

■ 2. Submit: ONLY with “`sbatch`” (for interactive jobs: “`salloc`”)

```
$ sbatch job_script.sh
<job_ID>
```

■ 3. Processing:

```
$ squeue
<job_ID> STATE: "PENDING" → "RUNNING" → "COMPLETED"
```

■ 4. Postprocessing: Job is finished → check output (default job name)

```
bwUniCluster: slurm-<job_ID>.out
HoreKa       : slurm-<job_ID>.out
```

1./2. Job submit

1./2. Job Submit: Important Resource Parameters

Sbatch Resource Options

Command line	Script	Purpose
<code>-t time</code>	<code>#SBATCH --time=time</code>	Wallclock time limit
<code>-N nodes</code>	<code>#SBATCH --node=nodes</code>	Number of nodes to be used
<code>-n tasks</code>	<code>#SBATCH --ntasks=tasks</code>	Number of tasks to be launched
<code>-c count</code>	<code>#SBATCH --cpus-per-task=count</code>	Number of CPUs required per (MPI-)task
<code>--ntasks-per-node=count</code>	<code>#SBATCH --ntasks-per-node=count</code>	Number of (MPI-)tasks per node
<code>--mem=MB_value</code>	<code>#SBATCH --mem=MB_value</code>	Memory in MegaByte per node
<code>--mem-per-cpu=MB_value</code>	<code>#SBATCH --mem-per-cpu=MB_value</code>	Memory required per allocated core
<code>-p queue</code>	<code>#SBATCH --partition=queue</code>	Defines the queue class

Use these options in the job script:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --ntasks=20
#SBATCH --time=00:02:00
#SBATCH --mem=500mb
...
```

Or use them with sbatch:

```
$ sbatch -N 1 -n 20 -t 2 --mem=500 <job_script>
```

1./2. Job Submit: Important Resource Parameters

Sbatch Resource Options

Command line	Script	Purpose
<code>-t time</code>	<code>#SBATCH --time=time</code>	Wallclock time limit
<code>-N nodes</code>	<code>#SBATCH --node=nodes</code>	Number of nodes to be used
<code>-n tasks</code>	<code>#SBATCH --ntasks=tasks</code>	Number of tasks to be launched
<code>-c count</code>	<code>#SBATCH --cpus-per-task=count</code>	Number of CPUs required per (MPI-)task
<code>--ntasks-per-node=count</code>	<code>#SBATCH --ntasks-per-node=count</code>	Number of (MPI-)tasks per node
<code>--mem=MB_value</code>	<code>#SBATCH --mem=MB_value</code>	Memory in MegaByte per node
<code>--mem-per-cpu=MB_value</code>	<code>#SBATCH --mem-per-cpu=MB_value</code>	Memory required per allocated core
<code>-p queue</code>	<code>#SBATCH --partition=queue</code>	Defines the queue class

- https://wiki.bwhpc.de/e/BwUniCluster_2.0_Slurm_common_Features#sbatch_Command_Parameters on bwUniCluster 2.0 and
- <https://www.nhr.kit.edu/userdocs/horeka/batch/> on HoreKa respectively.

bwUniCluster 2.0

	HPC nodes ("Thin" / "HPC" / "IceLake")	Fat nodes	GPU nodes	IceLake + GPU x4
Number of nodes	100 + 60 / 360 / 272	6	14 (4 GPUs each) 10 (8 GPUs each)	15 (4 GPUs each)
Number of sockets	2 / 2 / 2	4	2	2
Number of cores per node	40 / 40 / 60	80	40	64
Main memory per node	96 GB, 192 GB / 96 GB / 256 GB	3 TB	384 GB 768 GB	512 GB
Local SSD	960 GB SATA / 1,8 TB NVMe	4.8 TB NVMe	3.2 TB NVMe 6.4 TB NVMe	6.4 TB NVMe
Interconnect (InfiniBand)	HDR 100 (blocking) / HDR 100 / HDR 200	HDR	HDR	HDR
GPUs			NVIDIA Tesla V100	NVIDIA A100 / NVIDIA H100

0 4 login nodes

HoreKa

	Normal nodes	Extra-Large nodes	Accelerator nodes
Number of nodes	570+32	8	167
Sockets per node	2	2	2
Number of cores per node	76	76	76
Main memory per node	256/512 GB	4096 GB	512 GB
Interconnect (InfiniBand)	HDR	HDR	HDR
Local SSD	960 GB NVMe	7x 3.84 TB NVMe	960 GB NVMe

- 0 4 login nodes
- 0 Accelerator nodes incorporate 4 NVIDIA A100-40 GPUs

sbatch -p queues (Important queues of bwUniCluster 2.0)

<i>queue</i>	<i>default resources</i>	<i>MIN resources</i>	<i>MAX resources</i>
dev_single	time=10, mem-per-cpu=1125mb		time=30, nodes=1, mem=180000mb, ntasks-per-node=40
single	time=30, mem-per-cpu=1125mb		time=72:00:00, nodes=1, mem=180000mb, ntasks-per-node=40
multiple	time=30, mem-per-cpu=1125mb	nodes=2	time=72:00:00, nodes=128, mem=90000mb, ntasks-per-node=40
gpu_4	time=10, mem-per-cpu=2178mb, cpu-per-gpu=20	nodes=2	time=48:00:00, nodes=14, mem=376000, ntasks-per-node=40
fat	time=10, mem-per-cpu=18750mb		time=72:00:00, nodes=1, ntasks-per-node=80

- https://wiki.bwhpc.de/e/BwUniCluster_2.0_Batch_Queues
- Time and memory will be automatically chosen if not set

sbatch -p queues (All queues of HoreKa)

<i>queue</i>	<i>default resources</i>	<i>MIN resources</i>	<i>MAX resources</i>
dev_cpuonly	time=10, ntasks=1, mem-per-cpu=1600mb	nodes=1, ntasks=1	time=60, nodes=2, mem=243200mb, ntasks=152
cpuonly	time=10, ntasks=152, mem-per-cpu=1600mb, mem=243200mb	nodes=1, ntasks=152	time=72:00:00, nodes=192, mem=501600mb, ntasks=152
dev_accelerated	time=10, ntasks=1, grep=gpu:1, mem-per-cpu=3300mb, def-mem-per-gpu=125400mb, def-cpu-per-gpu=38	nodes=1, ntasks=1, gres=gpu:1	time=1:00:00, nodes=1, mem=501600mb, ntasks=152, gres=gpu:4
accelerated	time=10, ntasks=152, grep=gpu:4, mem=501600mb, mem-per-cpu=3300mb, def-mem-per-gpu=125400mb, def-cpu-per-gpu=38	nodes=1, ntasks=152, gres=gpu:4	time=48:00:00, nodes=56, ntasks=152, gres=gpu:4, mem=501600mb

■ <https://www.nhr.kit.edu/userdocs/horeka/batch/>

■ Time and memory will be automatically chosen if not set

Available Resources

■ Check available resources via

```
$ sinfo_t_idle
```

```
xy_ab1234@bwunicluster:~$ sinfo_t_idle  
  
Partition dev_single      :      6 nodes idle  
Partition single         :      0 nodes idle  
Partition dev_multiple   :      8 nodes idle  
Partition multiple       :      3 nodes idle  
Partition fat            :      0 nodes idle  
Partition dev_multiple_e :      8 nodes idle  
Partition multiple_e     :      3 nodes idle  
Partition dev_special    :      2 nodes idle  
Partition special       :      0 nodes idle  
Partition gpu_4          :      0 nodes idle  
Partition dev_gpu_4      :      1 nodes idle  
Partition gpu_8          :      0 nodes idle
```

```
xy_ab1234@horeka:~$ sinfo_t_idle  
  
Partition dev_cpuonly    :      1 nodes idle  
Partition cpuonly       :     101 nodes idle  
Partition dev_accelerate :      1 nodes idle  
Partition accelerate    :      83 nodes idle
```

Tutorial 1a

- **Goal:** Use the Batch System to execute “`printenv`” on the cluster
- **1)** Create a file „`submit_script.sh`“ and set the following options in the submit script:
 - 1 task
 - 500 MB memory
 - Time: 5 minutes
- **2)** After defining these options, insert the command to be executed at the end of the jobscript (“`printenv`”)
- **3)** Save the jobscript and submit it to the Batch System with

```
$ sbatch -p single --reservation=ws submit_script.sh # bwunicluster
```

```
$ sbatch -p cpuonly --reservation=ws submit_script.sh # horeka
```

 - You can use “`queue`” to see the status of your job
- **4)** Look in the output file of your job (`slurm-<jobID>.out`) for variables starting with „`SLURM_`“. These can be used to get information on how the job was started

```
#!/bin/bash
#SBATCH [???]
#SBATCH --time=00:05:00
#SBATCH --mem=500

[?????]
```

Tutorial 1a – Solution

- Create a file named „**submit_script.sh**“ with the following content:

```
#!/bin/bash
#SBATCH -n 1
#SBATCH --time=00:05:00
#SBATCH --mem=500

printenv
```

- Save the file and submit it with („**--reservation=ws**“ only for this workshop)

```
$ sbatch -p single --reservation=ws submit_script.sh # bwunicluster
```

```
$ sbatch -p cpuonly --reservation=ws submit_script.sh # horeka
```

- In the output file, you can find the SLURM variables:

- For example: „**SLURM_JOB_PARTITION=cpuonly**“ means:

In the job script, we have not defined a partition but the job was submitted to the „cpuonly“ partition with sbatch command

Tutorial 1b

- 1) Modify your submit script so that instead of “printenv” the value of „**SLURM_NPROCS**“ is printed (Hint: Use **echo**)
- Submit your job again, but this time use sbatch to specify the number of processes:

```
$ sbatch -p single --reservation=ws -n 4 submit_script.sh # bwunicluster
```

```
$ sbatch -p cpuonly --reservation=ws -n 4 submit_script.sh # horeka
```

- 2) Check in your output file if the number of processes is „1“ as specified in the submit script or „4“ as specified directly with **sbatch**

Tutorial 1b - Solution

- Modify your submit script to print the variable **SLURM_NPROCS**

```
#!/bin/bash
#SBATCH -n 1
#SBATCH --time=00:01:00
#SBATCH --mem=500

echo $SLURM_NPROCS
```

- Save the file and submit it with

```
$ sbatch -p single --reservation=ws -n 4 submit_script.sh # bwunicluster
```

```
$ sbatch -p cpuonly --reservation=ws -n 4 submit_script.sh # horeka
```

- In the output file the number of processes is printed:

slurm-<job-ID>.out

```
4
```

The options given directly to sbatch take precedence over the options in the submit script

3. Processing

3. Processing

- After submit command: if successful it returns <job-ID>

```
$ sbatch submit_script.sh  
  
Submitted batch job 1487560
```

- „Monitoring“ via:

- 3.a.

```
scontrol show job <job-ID>
```

- 3.b.

```
queue
```

```
sacct
```

- 3.c. Login to compute node:

```
srun --jobid=<id> --pty [--overlap] /usr/bin/bash
```

- „Modifying“ via:

- 3.d Cancel your job

```
scancel <job-ID>
```

3.a. Processing – scontrol show job (1)

- After submission of your script `submit_script.sh`

```
$ scontrol show job <job-ID>
```

```
JobId=1487569 JobName=submit_script.sh
```

```
UserId=ab1234(27049) GroupId=scc(12345)
```

```
Priority=4298 Nice=0 Account=kit QOS=normal
```

```
JobState=RUNNING Reason=Prolong Dependency=(null)
```

```
Queue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
```

```
RunTime=00:00:19 TimeLimit=00:10:00 TimeMin=N/A
```

```
SubmitTime=2021-10-13T00:06:58 EligibleTime=2021-10-13...
```

```
AccrueTime=2021-10-13T00:06:59
```

```
StartTime=2021-10-13T00:06:59 EndTime=2021-10-13T00:16:59 ...
```

```
SuspendTime=None SecsPreSuspend=0 LastSchedEval=...
```

```
Partition=dev_single AllocNode:Sid=uc2n997:2170796
```

```
ReqNodeList=(null) ExcNodeList=(null)
```

```
NodeList=uc2n362
```

```
BatchHost=uc2n362
```

```
...
```

1) Consumed resources will be booked on your university / project

2) Your job state

3) Your time logging

4) Your selected partition

5) Your node list and Node on which job started

3.a. Processing – scontrol show job (2)

- After submission of your script `submit_script.sh`

```
$ scontrol show job <job-ID>
```

```
JobId=1487569 JobName=submit_script.sh
```

```
...
```

```
NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=1 ...
```

```
TRES=cpu=2,mem=2250M,node=1,billing=2
```

```
Socks/Node=* NtasksPerN:B:S:C=0:0:*:1 CoreSpec=*
```

```
MinCPUsNode=1 MinMemoryCPU=1125M MinTmpDiskNode=0
```

```
Features=(null) DelayBoot=00:00:00
```

```
OverSubscribe=OK Contiguous=0 Licenses=(null) ...
```

```
Command=/pfs/data5/home/kit/scc/ab1234/submit_script.sh
```

```
WorkDir=/pfs/data5/home/kit/scc/ab1234
```

```
StdErr=/pfs/data5/home/kit/scc/ab1234/slurm-1487569.out
```

```
StdIn=/dev/null
```

```
StdOut=/pfs/data5/home/kit/scc/ab1234/slurm-1487569.out
```

```
Power=
```

```
NtasksPerTRES:0
```

1) Your requested nodes & CPU cores

2) Your job memory

3) Actual node policy

4) Your submit directory & submit script

5) Your job standard output and error log file

3.b. Processing – queue

- After submission of your script `submit_script.sh`

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
1487570	dev_cpuon	submit_s	ab1234	R	0:05	1	hkn0301

```
$ squeue --long
```

JOBID	PARTITION	NAME	USER	STATE	TIME	TIME_LIMI	NODES	NODELIST(REASON)
1487570	dev_cpuon	submit_s	ab1234	RUNNING	2:49	10:00	1	hkn0301

- Job states:

- PD = PENDING
- R = RUNNING
- CD = COMPLETED
- ...
- F = FAILED
- CA = CANCELLED



- When job is pending: **expected start time?**

```
$ squeue --start
```

JOBID	...	START_TIME	SCHEDNODES
1487570	...	2021-10-14T10:10:10	hkn0301

3.b. Processing – sacct

- After submission of your script `submit_script.sh`
 - Display accounting data of your job and job steps?

```
$ sbatch submit_script.sh  
Submitted batch job 1487652
```

```
$ sacct -j 1487652
```

JobID	JobName	Partition	Account	AllocCPUS	State	ExitCode
1487652	submit_sc+	dev_single	kit	2	RUNNING	0:0
1487652.bat+	batch		kit	2	RUNNING	0:0
1487652.ext+	extern		kit	2	RUNNING	0:0
1487652.0	hostname		kit	2	COMPLETED	0:0
1487652.1	bash		kit	2	RUNNING	0:0

3.c. Processing – Compute node login

- Once your job is running (state=**R**),
 - login to dededicated nodes is possible:

```
ab1234@hkn1990:~$ sbatch submit_script.sh
Submitted batch job 1487652

ab1234@hkn1990:~$ squeue
  JOBID PARTITION      NAME      USER ST       TIME  NODES NODELIST(REASON)
 1487652 dev_cpuon  submit_s  ab1234  R        2:42     1 hkn0301

ab1234@hkn1990:~$ srun --jobid=1487652 --pty [--overlap] /usr/bin/bash
ab1234@hkn0301:~$
```

- The command srun adds another step to your job,
 - Once main jobs finishes, this job step is cancelled automatically:

```
ab1234@hkn0301:~$
slurmstepd: error: *** STEP 1487652.2 ON hkn0301 CANCELLED AT 2021-10-13T10:35:52 ***
exit
srun: Job step aborted: Waiting up to 32 seconds for job step to finish.

ab1234@hkn1990:~$
```

3.d. Processing – Change status of your job

- Once your job is submitted (state=**PD** or **R**),
 - You can cancel your job

```
$ sbatch submit_script.sh
Submitted batch job 1487683
```

```
$ scancel 1487683
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
1487683	dev_cpuon	submit_s	ab1234	R	2:42	1	hkn0301

- Check with sacct :

```
$ sacct -j 1487683
```

JobID	JobName	Partition	Account	AllocCPUS	State	ExitCode
1487683	submit_sc+	dev_cpuon+	hk-project+	2	CANCELLED+	0:0
1487683.bat+	batch		hk-project+	2	CANCELLED	
1487683.ext+	extern		hk-project+	2	CANCELLED	

Tutorial 2

- Modify your submit script so that it executes a command to wait for 600 seconds (**sleep 600**)
- Set **walltime** to **10 minutes** and give your job a name
- Submit your job script with **sbatch**
- Use **squeue** to check the status of your job
- Use **scontrol show job** to see from which directory you started your job
- Use **scancel <job-ID>** to cancel your job

Tutorial 2 - Solution

■ Modify your submit script

```
#!/bin/bash
#SBATCH -N 1 -n 1
#SBATCH -t 00:10:00
#SBATCH --mem-per-cpu=500
#SBATCH -J myJobName

sleep 600
```

■ Save the file and submit it with

```
$ sbatch -p single --reservation=ws submit_script.sh # bwunicluster
```

```
$ sbatch -p cpuonly --reservation=ws submit_script.sh # horeka
```

■ Get your start directory of your jobs

```
$ scontrol show job 1487685 | grep WorkDir
WorkDir=/pfs/data5/home/kit/scc/ab1234/workshop
```

■ Use `scancel <job-ID>` to cancel your job

Interactive jobs

- Jobs on login nodes are not permitted
- **Solution: interactive slurm jobs**
 - Access compute nodes and work on them interactively

```
ab1234@hkn1990$ salloc -p cpuonly -n 1 -t 10 --mem=2000
```

```
salloc: Granted job allocation 1487738  
salloc: Waiting for resource configuration  
salloc: Nodes hkn0301 are ready for job
```

```
ab1234@hkn0301:~$ {Now you can work on the compute node}
```

```
salloc: Job 1487738 has exceeded its time limit and its allocation has been revoked.  
srun: Job step aborted: Waiting up to 32 seconds for job step to finish.  
slurmstepd: error: *** STEP 1487738.interactive ON hkn0301 CANCELLED AT 2021-10-  
13T13:02:41 DUE TO TIME LIMIT ***  
exit
```

```
srun: error: hkn0301: task 0: Exited with exit code 127
```

```
ab1234@hkn1990:~$
```

Job is waiting to start, Do Not interrupt the command

Job running. You are now on a compute node

Requested time for the interactive job ran out

Back on the login node

- Thank you for your attention!
 - Questions?