bw|HPC

Batch System Introduction

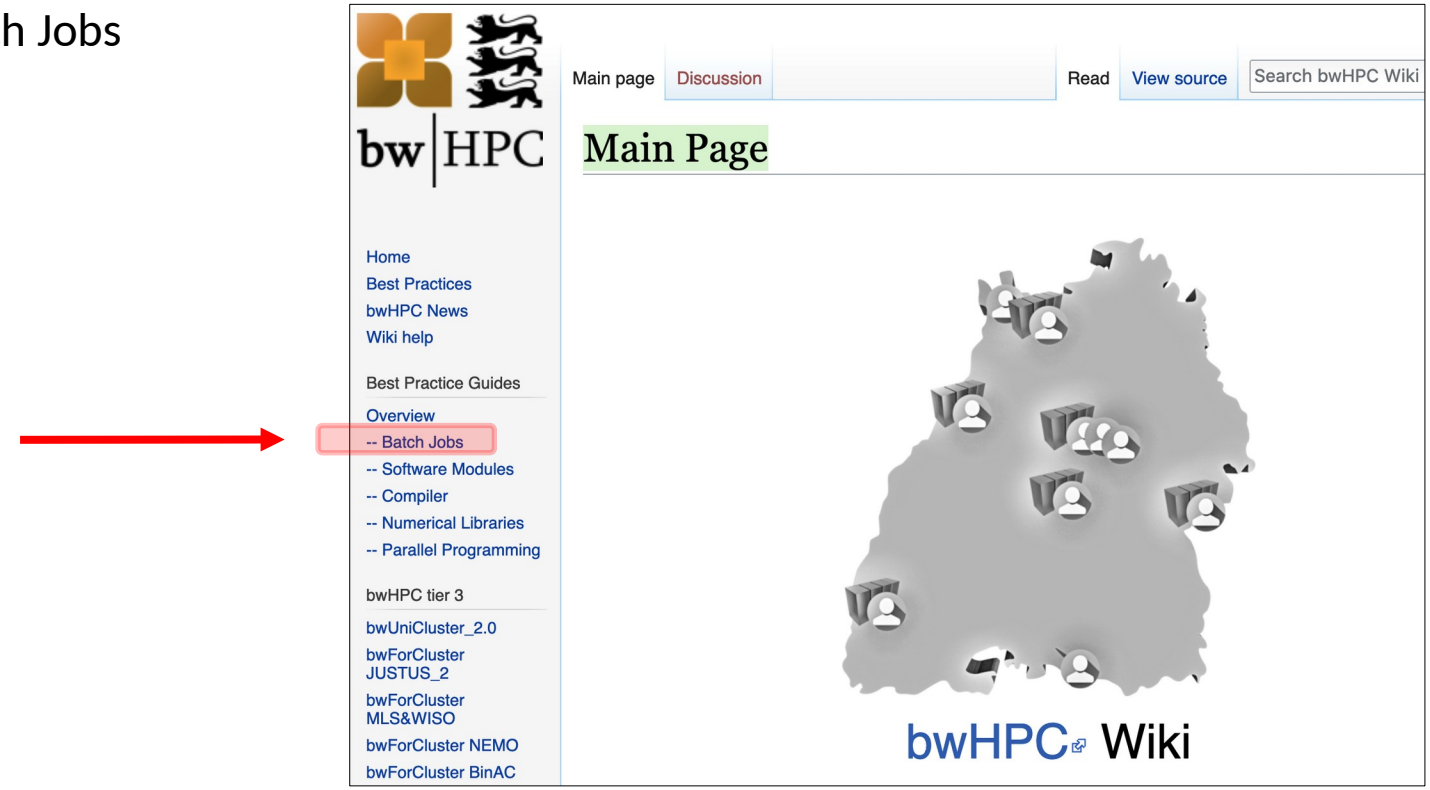
- **Andreas Baer, KIT - SCC**

Funding:

Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

Reference: bwHPC Wiki

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



Material: Slides & Scripts

- https://indico.scc.kit.edu/e/bwhpc_course_2022-10-13
- @bwUniCluster:
  - /opt/bwhpc/common/workshops/2022-10-13

How to read the following slides

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

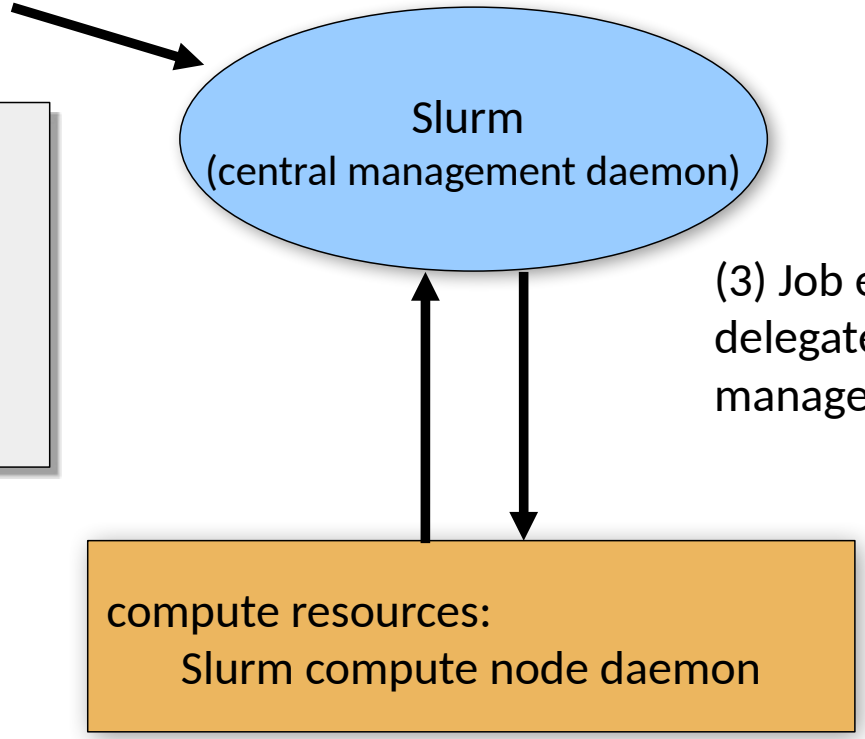
Resource and workload manager (1)

(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 exectuted 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: Sbatch options

- 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.
- sbatch options: command line or in your job script

Command line	Script	Purpose
`-N` *nodes*	`#SBATCH --nodes=`*nodes*	**Number of nodes to be used.**
`-n` *tasks*	`#SBATCH --ntasks=`*tasks*	**Number of tasks to be launched.**
`-J` *name*	`#SBATCH --job-name=`*name*	**Gives a user specified name to the job.**
`-p` *queue*	`#SBATCH --partition=`*queue*	**Defines the queue class**
`--mail-type=`*type*	`#SBATCH --mail-type=`*type*	**Send email when job begins (BEGIN), ends (END), aborts (FAIL), at each event (ALL) ...**

1./2. Job Submit: Important Resource Parameters

Sbatch Resource Options		
Command line	Script	Purpose
`-t` *time*	`#SBATCH --time=`*time*	Wallclock time limit
`-N` *nodes*	`#SBATCH --node=`*nodes*	Number of nodes to be used
`-n` *tasks*	`#SBATCH --ntasks=`*tasks*	Number of tasks to be launched
`-c` *count*	`#SBATCH --cpus-per-task=`*count*	Number of CPUs required per (MPI-)task
`--ntasks-per-node=`*count*	`#SBATCH --ntasks-per-node=`*count*	Number of (MPI-)tasks per node
`--mem=`*MB_value*	`#SBATCH --mem=`*MB_value*	Memory in MegaByte per node
`--mem-per-cpu=`*MB_value*	`#SBATCH --mem-per-cpu=`*MB_value*	Memory required per allocated core

- 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.
- sbatch options: command line or in your job script

1./2. Job Submit: Important Resource Parameters

Sbatch Resource Options		
Command line	Script	Purpose
`-t time`	`#SBATCH --time=time`	Wallclock time limit
`-N nodes`	`#SBATCH --node=nodes`	Number of nodes to be used
`-n tasks`	`#SBATCH --ntasks=tasks`	Number of tasks to be launched
`-c count`	`#SBATCH --cpus-per-task=count`	Number of CPUs required per (MPI-)task
`--ntasks-per-node=count`	`#SBATCH --ntasks-per-node=count`	Number of (MPI-)tasks per node
`--mem=MB_value`	`#SBATCH --mem=MB_value`	Memory in MegaByte per node
`--mem-per-cpu=MB_value`	`#SBATCH --mem-per-cpu=MB_value`	Memory required per allocated core

Use these options in the job script:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 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>
```

bwUniCluster 2.0

	Thin nodes	HPC nodes	Fat nodes	GPU nodes
Number of nodes	100 + 60	360	6	14 (4 GPUs each) 10 (8 GPUs each)
Number of sockets	2	2	4	2
Number of cores per node	40	40	80	40
Main memory per node	96 GB / 192 GB	96 GB	3 TB	384 GB 768 GB
Local SSD	960 GB SATA	960 GB SATA	4.8 TB NVMe	3.2 TB NVMe 6.4 TB NVMe
Interconnect (InfiniBand)	HDR 100 (blocking)	HDR 100	HDR	HDR

- 4 login nodes
- Some additional nodes that will be turned of by end of October

sbatch -p *queues* (Important queues of bwUniCluster 2.0)

queue	*default resources*	*MIN resources*	*MAX resources*
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

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
```

Tutorial 1a – bwUniCluster

- **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

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

[?????]
```

- **2)** After defining these options, insert the command to be exectuted 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
```

  - You can use "`squeue`" 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

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
```

- 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
```

- **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
```

- 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

Job life cycle: 3. Processing (1)

- 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.

    ```
    squeue
    ```

  - 3.c. Login to compute node:

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

- „Modifying“ via:
  - 4. 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)
   Requeue=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

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 – squeue

- After submission of your script **submit_script.sh**

```
$ squeue

   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 1487570 dev_singl submit_s    ab1234  R       0:05      1 uc2n362
```

```
$ squeue --long

  JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
1487570 dev_singl submit_s    ab1234  RUNNING       2:49     10:00      1 uc2n362
```

- 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      uc2n362
```

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@uc2n997:~$ sbatch submit_script.sh
Submitted batch job 1487652

ab1234@uc2n997:~$ squeue
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
1487652 dev_singl submit_s   ab1234  R       2:42      1 uc2n0362

ab1234@uc2n997:~$ srun --jobid=1487652 --pty /usr/bin/bash
ab1234@uc2n362:~$
```

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

```
ab1234@uc2n362:~$
slurmstepd: error: *** STEP 1487652.2 ON uc2n362 CANCELLED AT 2021-10-
13T10:35:52 ***
exit
srun: Job step aborted: Waiting up to 32 seconds for job step to finish.

ab1234@uc2n997:~$
```

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_singl submit_s    ab1234  R       2:42      1 uc2n362
```

  - Check with `sacct` :

```
$ sacct -j 1487683
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1487683      submit_sc+ dev_single        kit          2 CANCELLED+      0:0
1487683.bat+      batch                   kit          2  CANCELLED
1487683.ext+     extern                   kit          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
```

- 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@uc2n997$ salloc -p ws -n 1 -t 10 --mem=2000

salloc: Granted job allocation 1487738
salloc: Waiting for resource configuration
salloc: Nodes uc2n362 are ready for job

ab1234@uc2n362:~$ {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 uc2n362 CANCELLED AT 2021-10-
13T13:02:41 DUE TO TIME LIMIT ***
exit
srun: error: uc2n362: task 0: Exited with exit code 127

ab1234@uc2n997:~$
```

Job is waiting to start, Do Not interupt the command

Job running. You are now on a compute node

Requested time for the interactive job ran out

Back on the login node