bw|HPC

Advanced Topics: Usage of bwHPC and ForHLR clusters

Samuel Braun, SCC, KIT

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Hochschule für Technik Stuttgart

UNI FREIBURG

UNIVERSITÄT HOHENHEIM 1818

Hochschule Esslingen University of Applied Sciences

Universität Konstanz

UNIVERSITÄT MANNHEIM

Universität Stuttgart

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

KIT Karlsruher Institut für Technologie

ulm university universität ulm

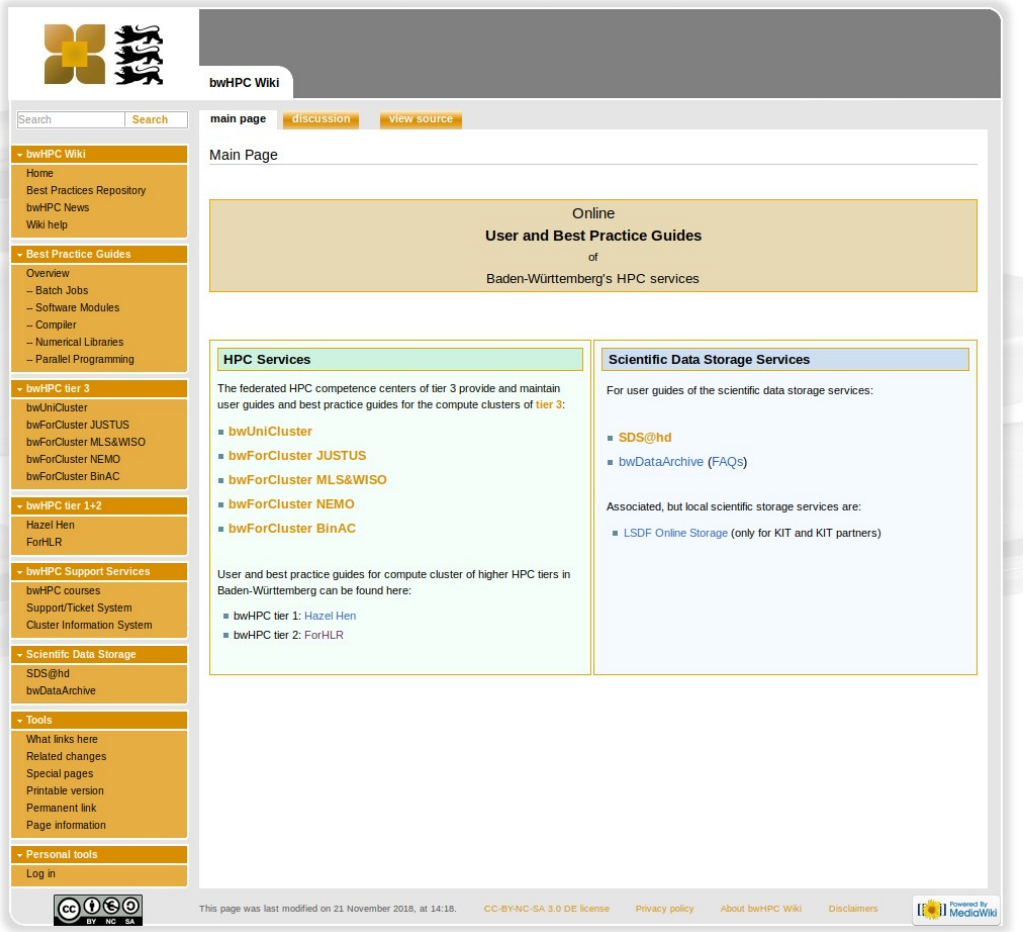
Funding: Baden-Württemberg MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

Outline

- Access and Data transfer topics
  - Access + rights, auto logout
  - Hardware-accelerated visualization @ bwUniCluster, ForHLR
  - Best practice: data sharing
- Architecture topics
  - Cluster topology, interconnect
  - Best practice: parallel file system
- Software topics
  - Best practice: installing own software
  - Best practice: compiling code
- Questions from participants

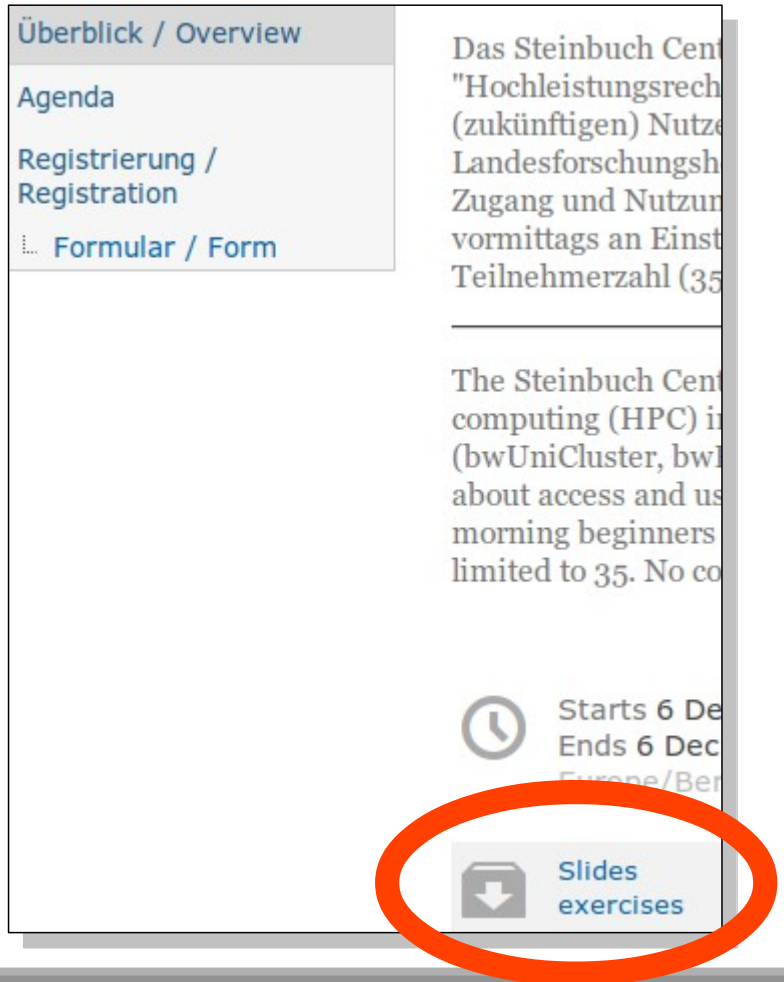
Reference: bwHPC Best Practices Repository

- Most information given by this talk can be found at https://www.bwhpc-c5.de/wiki:



Where to get the slides?

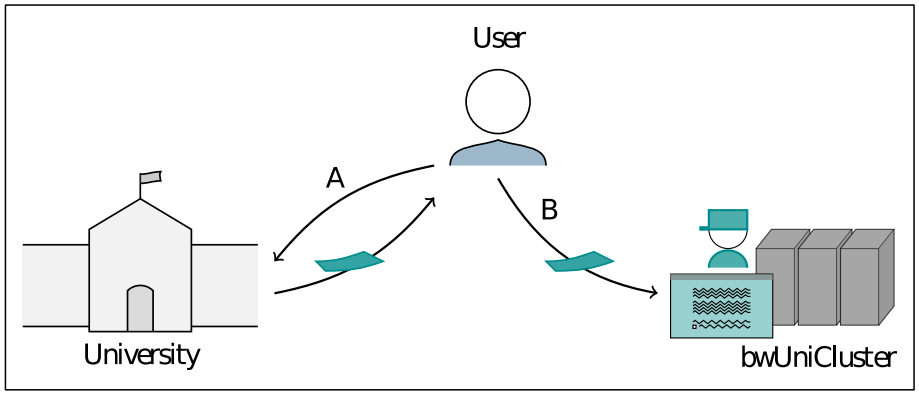
- https://indico.scc.kit.edu/e/bwhpc_course_2019-04-10
- uc1:/pfs/data1/software_uc1/bwhpc/kit/workshop/2019-04-10



Access and Data transfer

Access – bwUniCluster & extension

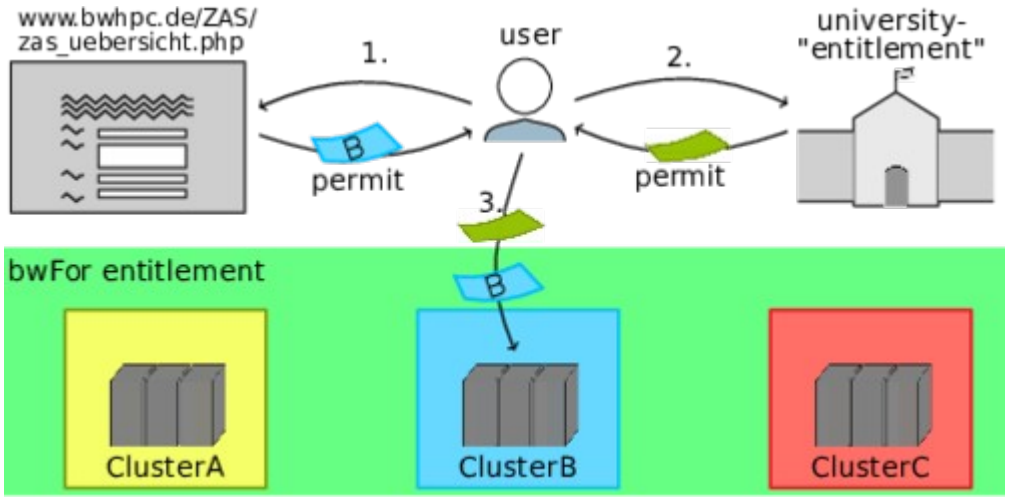
- Registration:
  1. bwUniCluster entitlement
  2. https://bwidm.scc.kit.edu/
  3. questionnaire



- Login:
  - @ „old partition“: `$ ssh <UserID>@uc1.scc.kit.edu`
  - @ „extension“: `$ ssh <UserID>@uc1e.scc.kit.edu`
  - Any difference?
    - concerning compilate code
    - but not concerning job submission
- Auto logout
  - Variable “TMOUT“ is set for 10 hours.
  - → If the user is continuously 10 hours inactive then he/she will be automatically logged out

Access – bwForClusters

- Registration:
  1. Central Application Site (ZAS)
  2. bwForCluster entitlement
  3. bwForCluster personal registration



- Login:
  - @ JUSTUS
  - @ MLS&WISO
  - @ NEMO
  - @ BinAC

```
$ ssh <UserID>@justus.uni-ulm.de
```

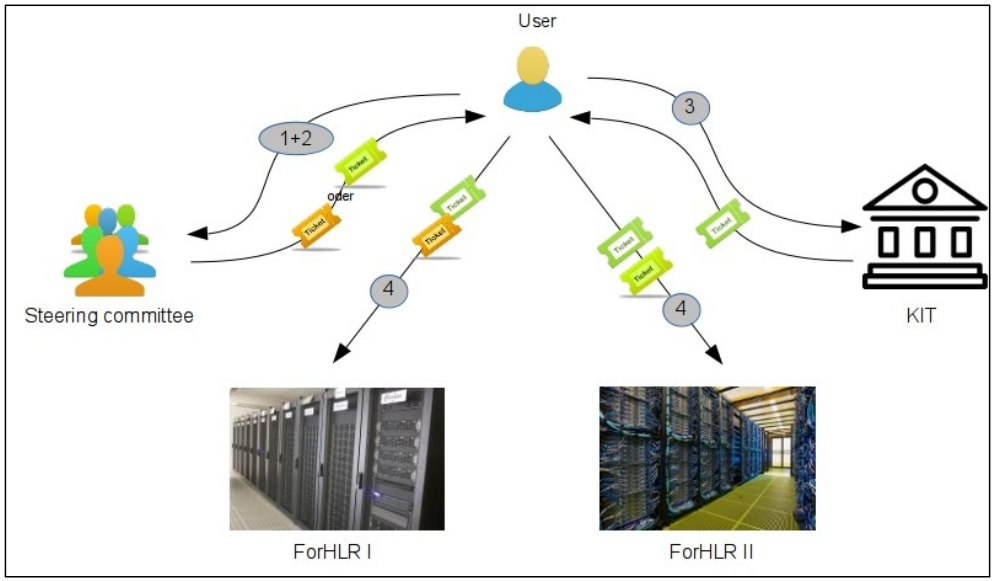
```
$ ssh <UserID>@bwforcluster.bwservices.uni-heidelberg.de
```

```
$ ssh <UserID>@login.nemo.uni-freiburg.de
```

```
$ ssh <UserID>@login0{1,2,3}.binac.uni-tuebingen.de
```

Access – ForHLR I and II

- Registration:
  1. Online Proposal Form
  2. Peer reviewed proposal
  3. ForHLR access form
  4. https://bwidm.scc.kit.edu/



- Login:
  - @ ForHLR I : `$ ssh <UserID>@fh1.scc.kit.edu`
  - @ ForHLR II: `$ ssh <UserID>@fh2.scc.kit.edu`

- Auto logout
  - Variable “TMOUT“ is set for 10 hours.

- **IMPORTANT:** A status report must be provided annually (10-15 pages)!

Password-less Login (linux + macOS)

- SSH private + public key pair

```
$ ssh-keygen -t rsa
```

→ Important: by all means ***secure*** private key ***with passphrase***

- Transfer, e.g. bwUniCluster

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub <UserID>@uc1.scc.kit.edu
```

- Store passphrase in *key-store* (Ubuntu: Gnome Keyring, Mac: Keychain)

```
Ubuntu: $ eval `ssh-agent -s`
```

```
Ubuntu: $ ssh-add
```

- Login, e.g. bwUniCluster

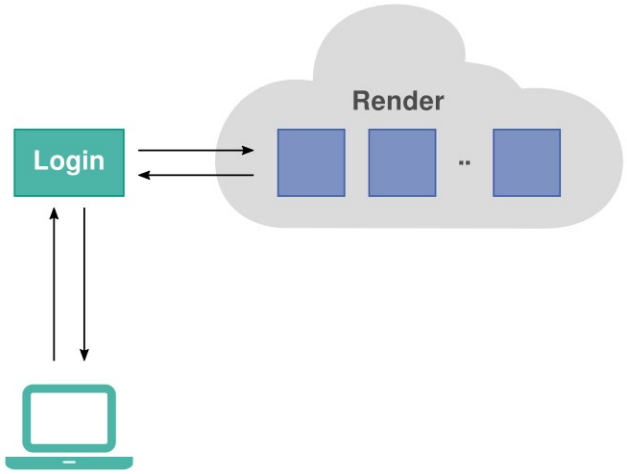
```
$ ssh <UserID>@uc1.scc.kit.edu
```

→ this requires now each time at actual login to ask for passphrase

→ for active sessions if passphrase in keystore, no login password anymore required

Remote Visualization (1)

- The Linux 3D graphics stack is based on X11 and OpenGL. This has some drawbacks in conjunction with remote visualization.
  - Rendering takes place on the client, not the cluster
  - Whole 3D model must be transferred via network to the client
  - Many round trips in the X11 protocol negatively influence interactivity
  - X11 is not available on non-Linux platforms
  - Compatibility problems between client and cluster can occur
- To avoid all those problems the module „start_vnc_desktop“ is provided on bwUniCluster and ForHLR II for remote visualization. More details at bwHPC wiki

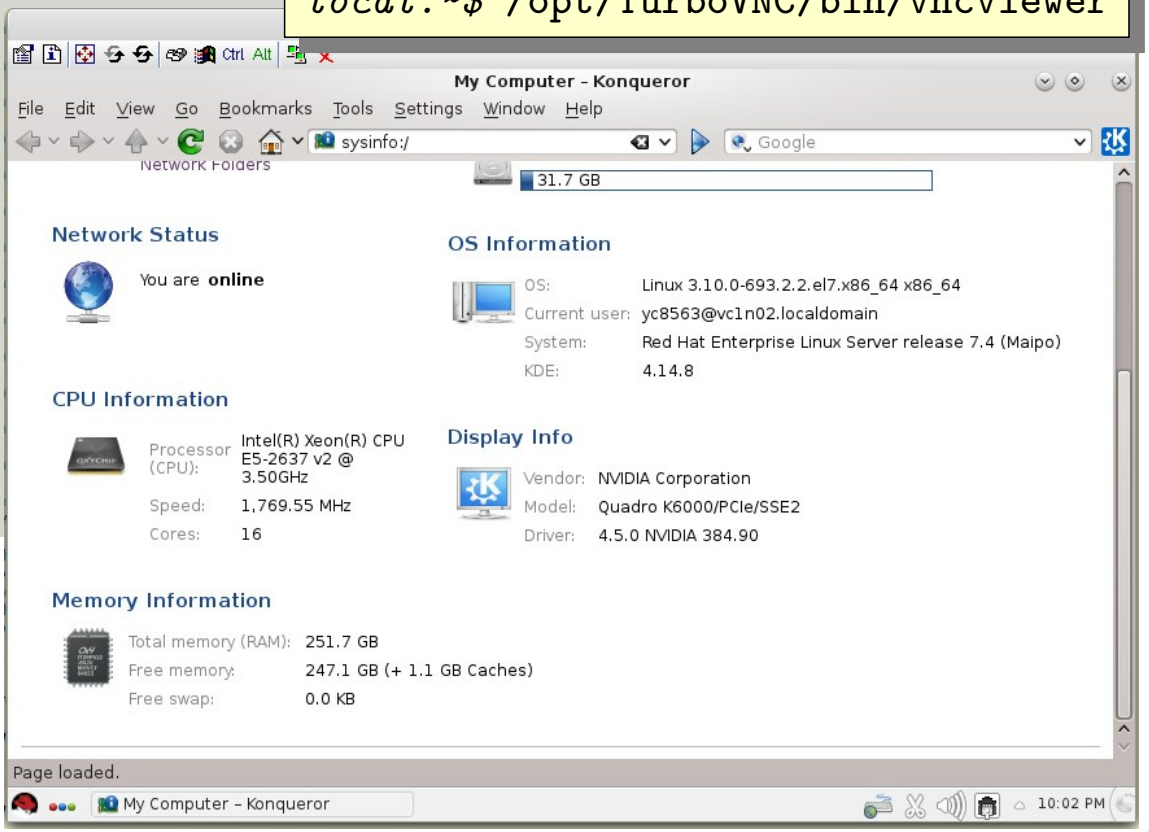


Remote Visualization (2)

```
uc1:~$ start_vnc_desktop --hw-rendering
```

```
Hint for TurboVNC Viewer users (command line):
==============================================
  vncviewer ExtSSH=1 Via=yc8563@uc1.scc.kit.edu Server=vc1n02:1 Password=AgGQmo8z

Hint for TurboVNC Viewer users (GUI)
====================================
  Fill in the following entry field:
  VNC server:    vc1n02:5901

  Click "Options" and choose tab "Security".
  Fill in the following entry fields:

  Gateway (SSH server or UltraVNC repeater)
  SSH user:      yc8563
  Host:          uc1.scc.kit.edu
  Click "OK"

  Click "Connect"

  VNC Authentication:
  Password:      AgGQmo8z

Hint for installing VNC viewer:
===============================
/usr/bin/start_vnc_desktop --help-client
```

```
local:~$ /opt/TurboVNC/bin/vncviewer
```



bwHPC

Best practice: Data Sharing (1)

- How to share data with another person on the same cluster?

  1. Do not share folders in your $HOME, use workspaces!

```
$ ws_allocate sharing 30
Workspace created. Duration is 720 hours.
Further extensions available: 3
/pfs/work2/workspace/scratch/ab1234-sharing-0
$ ls -ld $(ws_find sharing)
drwx------ 2 ab1234 xyz 4096 Oct  9 00:42 /pfs/work2/workspace/scratch/ab1234-sharing-0
```

  → workspace is private!

  2. Adjust permissions to your needs:

     a.) Allow all users of your group to have access

```
$ chmod g+x $(ws_find sharing)
$ ls -ld $(ws_find sharing)
drwx--x--- 2 ab1234 xyz 4096 Oct  9 00:42 /pfs/…/ab1234-sharing-0
```

Best practice: Data Sharing (2)

- How to share data with another person on the same cluster?

  2. Adjust permissions to your needs:

     b.) Allow **another user** to have **full access** **but force group inheritance**

```
$ chmod g+s $(ws_find sharing)
$ ls -ld $(ws_find sharing)
drwx--S--- 2 ab1234 xyz 4096 Oct  9 00:42 /pfs/…/ab1234-sharing-0
```

→ use ACL (access control lists)

To add group uvw:

```
$ setfacl -m g:uvm:rwx ...
```

```
$ setfacl -m u:cd5678:rwx $(ws_find sharing)

$ ls -ld $(ws_find sharing)
drwxrws---+ 2 ab1234 xyz 4096 Oct  9 00:42 /pfs/…/ab1234-sharing-0

$ getfacl $(ws_find sharing)
…
# owner: ab1234
# group: xyz
# flags: -s-
user::rwx
user:cd5678:rwx
group::---
…
```

Best practice: Data Sharing (3)

- How to share data with another person on the same cluster?

  2. Adjust permissions to your needs:

     c.) Revoke other user‘s access to workspace „sharing“

```
$ setfacl -x u:cd5678 $(ws_find sharing)

$ ls -ld $(ws_find sharing)
drwx--S---+ 2 ab1234 xyz 4096 Oct  9 00:42 /pfs/…/ab1234-sharing-0

$ getfacl $(ws_find sharing)
…
# owner: ab1234
# group: xyz
# flags: -s-
user::rwx
group::---
…
```

Best practice: Data Sharing (4)

- Shortcut for Data Sharing 1 – 3: *ws_share*

```
$ module load system/ws_addon
$ ws_share

  USAGE    : ws_share [-options <argument>] <workspace_1> ... <workspace_N>

  -h, --help               Print this message
  -v, --verbose            Verbose printout
  -n, --dryrun             Do a dry run

  -u, --user=<usernames>   Comma separated list of users
                           that given workspace(s) has to be shared with
  -g, --group=<groupnames> Comma separated list of groups
                           that given workspace(s) has to be shared with

  -t, --sharetype=<option> Type of sharing workspaces with following options:
                           All added permissions will be NEVER MORE then owner's permission on files/directory in workspace(s)!
                              default   = All directories can be accessed (r-x) but new user/group can not add files
                              dir-w     = puts write permissons on all directories (rwx) if owner's directory has write permissions
                                          while permission of owner's files are unaltered
                                          and new user/group can add files
                              replicate = replicate owner's permissions on dirs/files.
                                          Depending on owner's dir permissions new user/group can add files

  -s, --status             Print sharing (.i.e., ACL) permissions
  -r, --revoke-all         Revoke recursively all sharing (.i.e., ACL) permissions
```

Loading custom environment

- Automatically customize your CLI session
  - If you always need particular global variables, aliases, bash functions
  - Add to $HOME/.bashrc
    - e.g., own variable to point your workspace „sharing“

```
export WS=$(ws_find sharing)
```

```
$ ssh ab1234@uc1
$ echo $WS
/pfs/work2/workspace/scratch/ab1234-sharing-0
```

    - e.g. own function to jump to your workspace „sharing“

```
goto_ws(){ cd $(ws_find sharing) ; }
```

Do not omit the spaces

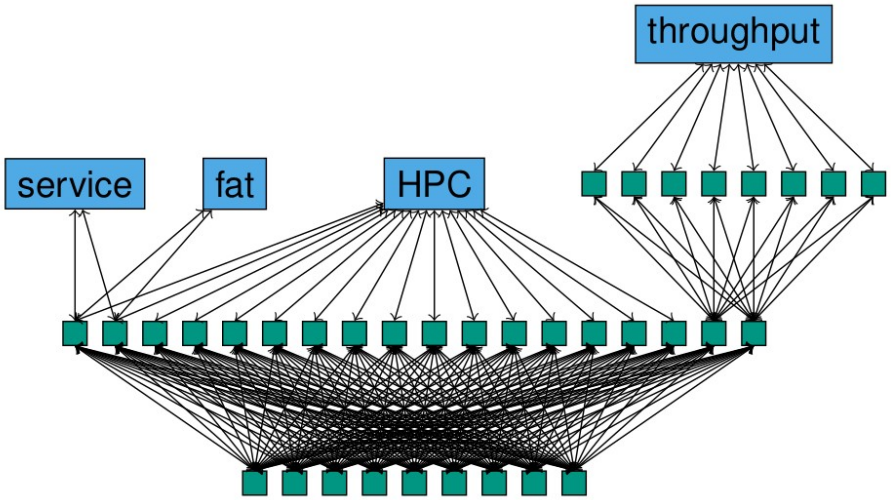
```
$ ssh ab1234@uc1
$ type goto_ws
goto_ws is a function
goto_ws ()
{
    cd $(ws_find sharing)
}
$ goto_ws
```

Architecture

HPC Cluster Architecture: bwUniCluster

- Node types:
  - Login nodes
    - Thin / Fat and Broadwell
  - Compute Nodes
    - Thin, Fat and Broadwell
  - Remote Vis Nodes
  - File Server Nodes
  - Administrative Server Nodes

  → all connected by interconnect



- More details at:

  http://www.bwhpc-c5.de/wiki/index.php/BwUniCluster_Hardware_and_Architecture

Hyper-threading

- Simultaneous multithreading (SMT)
- Physical cores -> doubling into logical cores
  - 2 logical cores share execution part (engine, cache, bus) of 1 physical core

- But: only max number of physical cores requestable by queueing system
  - e.g. bwUniCluster thin nodes:

```
msub -l nodes=1:ppn=32
msub -l nodes=1:ppn=16
```

- Other half of logical cores reserved to OS background task

  → increases overall performance, OS jitter are better distributed

Job I/O statistics

- PFS I/O statistics of a batch job can be collected
  - 1. Determine the PFS name of your job directory, e.g. your workspace

```
$ df --output=source <job_directory> | sed 1d | cut -d: -f3
/pfs2wor2
```

  - 2. Add PFS name during job submission:

```
$ msub -W lustrestats:<PFS_name> ...
```

  - Optional: Get results by e-mail

```
$ msub -W lustrestats:<PFS_name> -M name@domain ...
```

Best Practice: Parallel file systems (1)

- What not to do:
  - Do not run jobs in $HOME → use workspaces
  - Do not generate +10000 files on workspaces → change application code
  - Do not run any kind of database on PFS
  - Do not use PFS for your entire research data storage → clear out periodically
    → use LSDF Online Storage, SDS@hd, bwDataArchive

- Rev: How to check your quota:
  - $HOME @ bwUniC., ForHLR:

```
$ lfs quota -u $(whoami) $HOME
```

- New: Send reminder for workspace renewal

```
$ ws_send_ical.sh <workspace> <email>
```

Best Practice: Parallel file systems (2)

- Improving Performance of PFS:
  - On PFS files striped over storage subsystems, i.e. large files are separated into stripes and distributed to different storage subsystems
    - Stripe size = size of chunks (default = 1 MB)
    - Stripe count = number of used storage subsystems per file/directory

      → New files and subdirs inherit stripe count from parent dir
      - default: stripe_count_my_$HOME = 1; stripe_count_my_$WORK = 2

    - Get stripe count:

```
$ lfs getstripe    <my_file>
$ lfs getstripe -d <my_dir>
```

    - Set stripe count:

```
$ lfs setstripe -c<num> <my_file/my_dir>
```

      - num = -1 → use all available storage subsystems ($HOME=20, $WORK=40)
      - New stripe count of parent dir -> stripe count of existing files inside unchanged
        - To recursively change → copy all content to new directory
      - New stripe count is not saved in PFS backup

Best Practice: Parallel file systems (3)

- Improving Performance of PFS:
  - When to change stripe count?
    - Many tasks use few huge files

      → stripe count = -1
      - To avoid overlapping issues:

        → setting tasks to use *N**1MB blocks
    - Enhancing throughput of a single file by ONE task

      → stripe count between 2 and 8
  - General rules:
    - Transfer data in large blocks and store in few large files
    - Make use of large caches, i.e., collect large blocks and write them sequentially at once
    - Avoid competitive file access

Software

Best Practice: Installing Own Software

- Check list:
  - Legal issues: do you have licence for your software?
  - Disk space?
    - Check if software would exceed quota
  - Installation procedure?
    - If compilation exceeds 10 min
      - Install via interactive batch job on a compute node
    - Never use: *make -j* → but: *make -j <number>*
    - Never simply use binaries on different architecture
      - But: recompile or compile supporting multiple architecture
    - Use guides stored in software module files
- Help:
  - Contact support (bwTicketPortal),
  - apply for Tiger Team Support

Best Practice: Compiling code

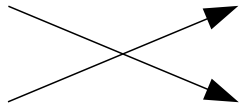
- Details to compilation and Makefile tutorial:
  - See today‘s talk „Compile, Makefiles“

- Clusters with different architecture generations

  e.g. bwUniCluster
  - Thin + Fat nodes (Sandy Bridge) vs. Broadwell nodes

    a) Compile code on corresponding login node

```
uc1:~$ icc/ifort -xHost ...
uc1e:~$ icc/ifort -xHost ...
```



Run @uc1 → crash

Run @uc1e → slower

    b) Compile code including multiple, feature-specific code paths

```
uc1e:~$ icc/ifort -xCORE-AVX2 -axAVX...
```

Thank you for your attention!

Questions?