

# Teaching of HKHLR

Hessisches Kompetenzzentrum für Hochleistungsrechnen (HKHLR)

Tim Jammer, Anja Gerbes, René Sitt

30.07.2019



HKHLR is funded by the Hessian Ministry of  
Sciences and Arts



- ▶ In this talk:
  
- ▶ Brief introduction of the HKHLR
- ▶ Skills hessian cluster users need
- ▶ Course offering of HKHLR
  - ▶ Focus: ProTHPC
- ▶ Conclusion



- ▶ Competence Center for High Performance Computing in Hessen
- ▶ Training
- ▶ Consulting
- ▶ Information
- ▶ Monitoring

### Participating universities:



HKHLR is funded by the Hessian Ministry of  
Sciences and Arts



- ▶ K1.1-B System Architectures
  - ▶ Used architecture is different for each location
- ▶ USE1-B Cluster Operating System
  - ▶ No fundamental differences between sites
- ▶ K4-B Job Scheduling
  - ▶ Marburg, Gießen use Grid Engine
  - ▶ Darmstadt, Frankfurt, Kassel use SLURM
  
- ▶ Only some users need
  - ▶ USE3-B Building of Parallel Programs

- ▶ Location specific cluster introduction
  - ▶ Regular cluster introduction at every location (additional if required)
- ▶ ProTHPC
  - ▶ Proficiency Training High Performance Computing
  - ▶ Three times per year
  - ▶ HPC basics
- ▶ HiPerCH
  - ▶ High Performance Computing in Hessen
  - ▶ Annually
  - ▶ Changing advanced or specific HPC topics
- ▶ Other tutorials and workshops as needed
  - ▶ Like courses on C++, OpenMP, MPI, ...

- ▶ 3 times per year
  - ▶ Rotating location in Hestia
  - ▶ Free of charge for all academia
  - ▶ Three days of training
  - ▶ Seperate registration for specific modules
  - ▶ Each trainer may give each course
- 
- ▶ March 2019: Marburg
  - ▶ June 2019: Darmstadt
  - ▶ September 2019: Gießen



- ▶ Linux and Shell Scripting
- ▶ Batch Job Scheduling
- ▶ Building Software Applications in Linux (Make)
- ▶ Introduction to Version Control with GIT
- ▶ Parallel Debugging with Totalview
  
- ▶ Each module contains a large "hands-on" part



- ▶ Half-day course
- ▶ Teaches Skill USE1-B: Cluster Operating System
  - ▶ Command Line
  - ▶ Shell Scripting
  - ▶ Use of the module system

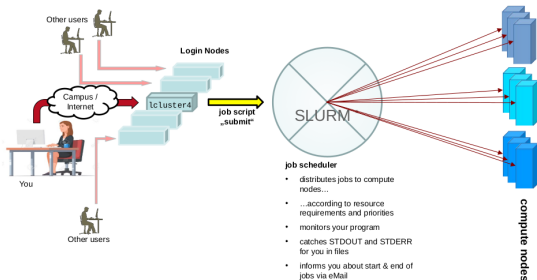
```
#!/bin/bash

for i in 1 2 3 10 11 hundred; do
    echo "Print i = $i"
done
```



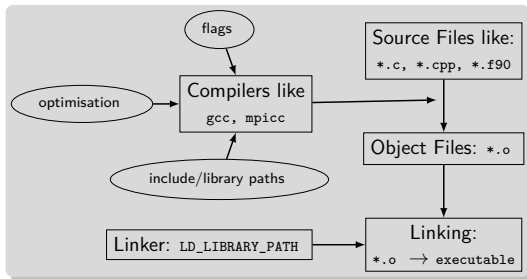


- ▶ Half-day course
- ▶ Teaches Skill K4-B: Job Scheduling
- ▶ Scheduler used (Slurm or Grid Engine) depends on location where the course is offered

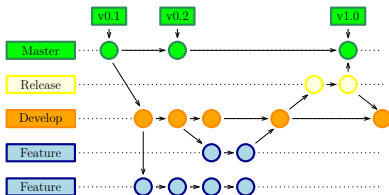


- ▶ Half-day course
- ▶ Teaches Skill USE3-B: Building of Parallel Programs

### Automatic build using make and Makefiles



- ▶ Half-day course
- ▶ Teaches Skill SD3.2-B: Version Control
- ▶ No HPC specific skills
- ▶ Rather version control is useful for all kinds of projects



- ▶ Full-day course
- ▶ Teaches Skill SD2.2-B: Debugging

The screenshot shows the TotalView debugger interface. The main window displays the source code for a C++ program, with a breakpoint set at line 429 of the `Circle::area` function. The code snippet is as follows:

```

420  m_radius = radius;
421  myarea = 2 * PI * m_radius * m_radius;
422  m_area = area();
423
424 |
425
426 // Your basic circle area function
427
428 double Circle::area() {
429  double result;
430  result = PI * m_radius * m_radius; // Our old friend, pi r squared...
431  return result;
432 }
433
434 // A Simple 3-D figure - class exercise: Do a cone figure in the same
435 // fashion. - I forget how to calculate the surface area of a cone :-}
436 class Cylinder : public Circle {
437 public:
438  Cylinder(char *name, double radius, double height);
439  Cylinder(double radius, double height);

```

The 'Stack Trace' window shows the current function call:

```

Function "Circle::area":
Block: "#p1":
Result: 0
Registers for the frame:
Function Circle::area in combined.cxx

```

The 'Action Points' window at the bottom shows a list of active breakpoints:

Line	File	Function	Condition
430	combined.cxx	Circle::area	0x06
430	combined.cxx	Cylinder::Cylinder	10x06...
430	combined.cxx	Cylinder::Cylinder	0x10
430	combined.cxx	Cylinder::Cylinder	0x00
430	combined.cxx	Cylinder::Cylinder	0x00
430	combined.cxx	Cylinder::Cylinder	0x0a
430	combined.cxx	area	0x17
430	combined.cxx	area	0x12

- ▶ K1.1-B System Architectures
  - ▶ Trained by: Cluster Introduction at every Location
- ▶ USE1-B Cluster Operating System
  - ▶ Trained by: ProthPC module 1: Linux and Shell Scripting
- ▶ K4-B Job Scheduling
  - ▶ Trained by: ProthPC module 2: Batch Job Scheduling
  
- ▶ Some Users also need
  - ▶ USE3-B Building of Parallel Programs
  - ▶ Trained by: ProthPC module 3: Building Software Applications in Linux

- ▶ 23.09.19 - 27.09.19 in Darmstadt
- ▶ Focus on Deep Learning on HPC systems
- ▶ Workshop on Fortran modernization in cooperation with NAG.
  
- ▶ More Information: <https://www.hkhlr.de/en/events/hiperch-11-2019-09-23>

- ▶ Very high demand in basic training
- ▶ Participants prefer local courses
  
- ▶ HKHLR developed ProTHPC training to satisfy the need for training of HPC basics in Hestia