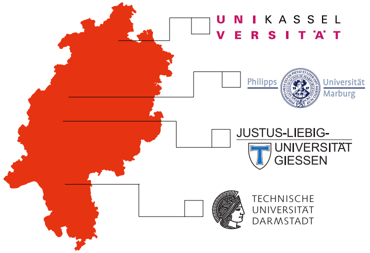
Debugging & Totalview

Hessisches Kompetenzzentrum für Hochleistungsrechnen (HKHLR)

Dr. Christian Iwainsky

V1.0


UNIKASSEL VERSITÄT
Philipps Universität Marburg
JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
TECHNISCHE UNIVERSITÄT DARMSTADT

HKHLR is funded by the Hessian Ministry of Sciences and Arts

Introduction to debugging and Totalview Part V

Topics

- Classic userinterface,
- Basic parallel debugging techniques,
- Controlling individual threads, group of threads, processes,
- Cross thread / process data inspection, and
- OpenMP and MPI debugging.

Demo 4

This video will be discussing Totalview using the program found in the **demo04** folder. Demo 4 integrates the area of a half-circle to compute PI. This demonstrator is without errors to demonstrate parallel debugging techniques.

- Makefile
- demo04.cc — Original sourcefile for *Part V*
- demo04A.cc — Simple OpenMP variant of demo04.cc.
- readme.md

The makefile has 5 targets:*demo04.exe*,*demo04A.exe*, ... , *demo04C.exe* and *clean*.
The program has no input.
Please consult readme.md for more details.

Totalview: Control Groups

- Group (control): all entities unter control of Totalview
- Group (share): all entities sharing the same binary
- Group (worker): all entities considerd worker processes, such as MPI ranks
- Group (lockstep): all entries with the same program counter as the focues entity
- Process P: a single process P
- Process (worker): all worker threads in the current process
- Process (lockstep): all threads in the current process with the same program counter
- Thread $P.T$: thread T of process P

Demo 5

We will discuss more features of Totalview using the program found in the **demo05** folder: Demo 5 is a MPI-Parallelized Program.

- Makefile
- demo05.cc

Original sourcefile

The makefile has 2 targets:*demo05.exe* and *clean*. You need to load an MPI module for using this program (e.g. `module load openmpi`)
The program has no input.

shell

```
>$ mpirun -n 2 ./demo05.exe
```

HESSEN

Used MPI Functions

Short explanation of the used MPI Functions in the example code:

- `MPI_Init`, `MPI_Finalize` initialize/finalize the MPI Library.
- `MPI_Send(buffer,count,datatype,destination,tag,communicator,status)`: Sends a message of *count* entries fo type *dataype* from *buffer* to the *destination* process using the given message *tag* and *communicator*.
- `MPI_Recv(buffer,count,datatype,source,tag,communicator,status)`: Wait for a message from the *source* process with the given message *tag* and receive it into the *buffer*.
- `MPI_Comm_Rank(communicator,rank)`: tells how many processes (*size*) are in the given in the current *communicator*
- `MPI_Comm_size(communicator,rank)`: tells the unique number of the calling process (*rank*) in the given in the current *communicator*

This segments contents:

- Classic userinterface,
- Basic parallel debugging techniques,
- Controlling individual threads, group of threads, processes,
- Cross thread / process data inspection, and
- OpenMP and MPI debugging.