TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

Center for Information Services and High Performance Computing (ZIH)

Session 4: Advanced Use Cases and Discussion

Dr. Matthias S. Müller (RWTH Aachen University)
Tobias Hilbrich (Technische Universität Dresden)
Joachim Protze (RWTH Aachen University, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de

ZIH
Center for Information Services & High Performance Computing

Content

- MUST
  - Frontend/Backend
  - Application Crash Handling
  - Scalability
  - Integrations
- Vampir
  - Tracing at Large Scale
  - Paradigms: Xeon Phi, OpenSHMEM
  - External Energy Counters
  - GPGPU Critical Paths

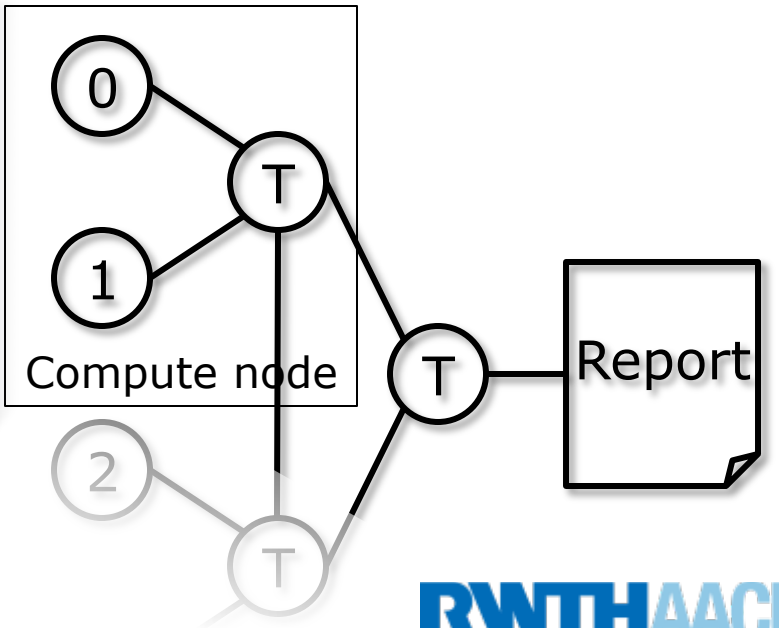
MUST: Using batch systems

- The tool setup phase of MUST might be triggered independent from the application/tool execution
- `mustrun --must:prepare` starts the setup
- `mustrun --must:run` starts the execution
- Both commands need the same options

MUST: Application crash handling

- In case of an error, MUST cannot rely on a possible broken MPI communication
- Activate the alternate communication system with: `--must:nodesize <#processes per node>`

Application crash handling:
- One tool process per compute node
- Application processes and the single tool node use immediate + asynchronous shared memory communication



MUST: Options for scalability

- To increase scalability of the tool you might add more tool processes for distributed analysis
- Activate distributed analysis:
  `--must:distribute`
- Set the branching factor of the analysis tree with:
  `--must:fanin <number>` default is 16
- Query the number of total processes with:
  `--must:info`

MUST: Integrations

- MUST integrates
  - Dyninst to create stacktraces for error situations.
  - Graphviz to create graphs of deadlock or datarace situations