TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

Practical I/O-Analysis

EIUG-Workshop 26.9.2017 Hamburg

Sebastian Oeste (sebastian.oeste@tu-dresden.de)

Holger Brunst (holger.brunst@tu-dresden.de)

ZIH

Center for Information Services & High Performance Computing

Agenda

- 1 Motivation
- 2 Linux I/O
- 3 Score-P / Vampir I/O-Analysis

HPC I/O State of the Art

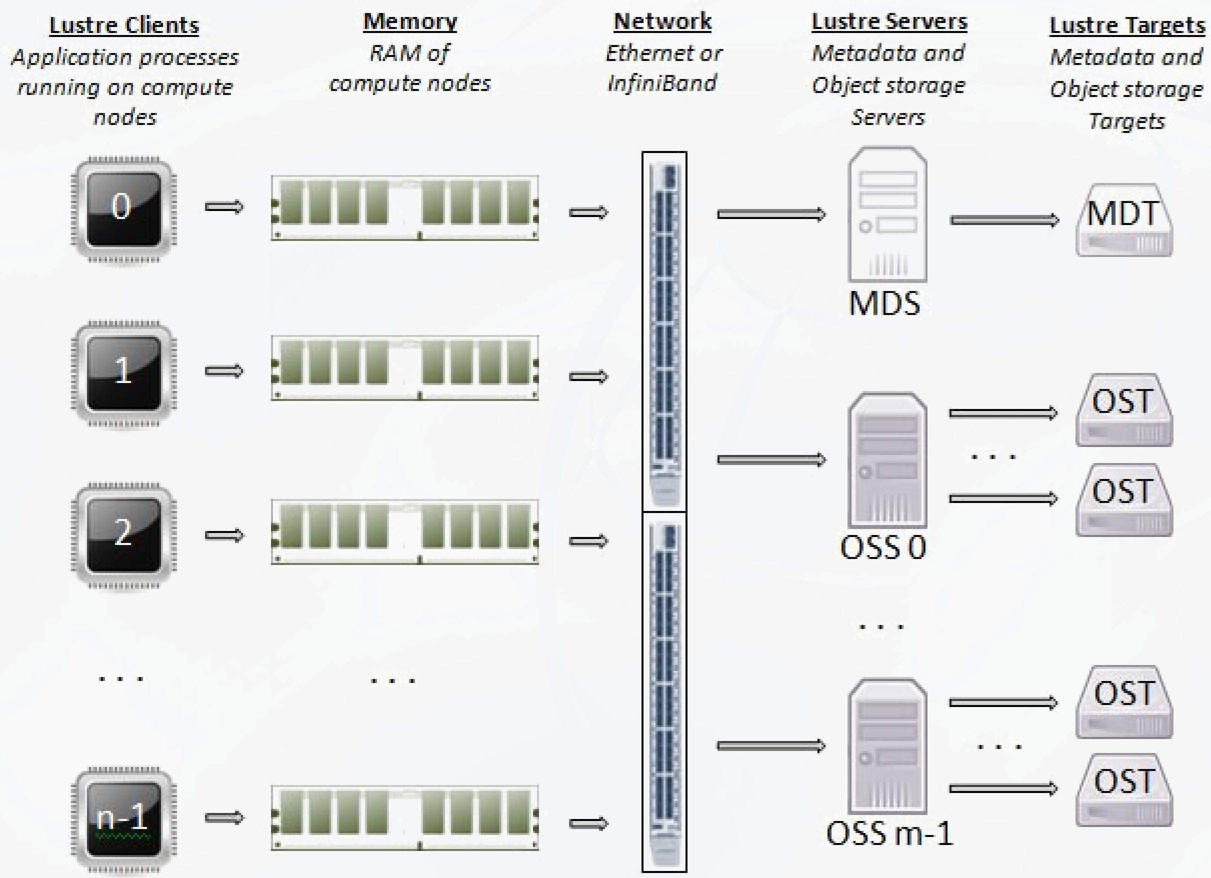
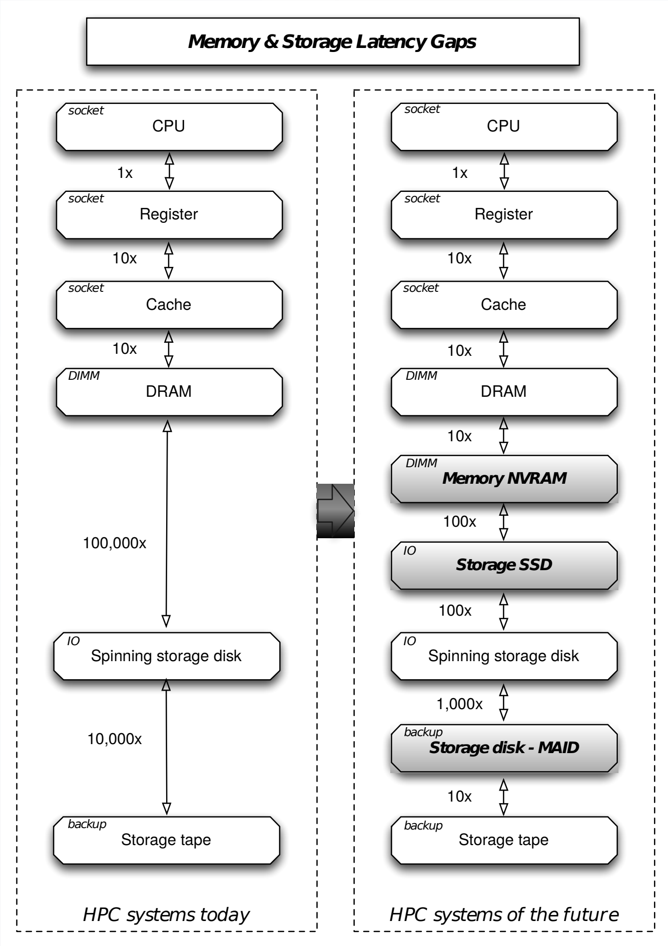


Image: www.nics.tennessee.edu

New members in the memory hierarchy

- New memory technology
- Changes the memory hierarchy we have
- Impact on applications e.g. simulations?
- I/O performance is one of the critical components for scaling applications



The Linux I/O Stack

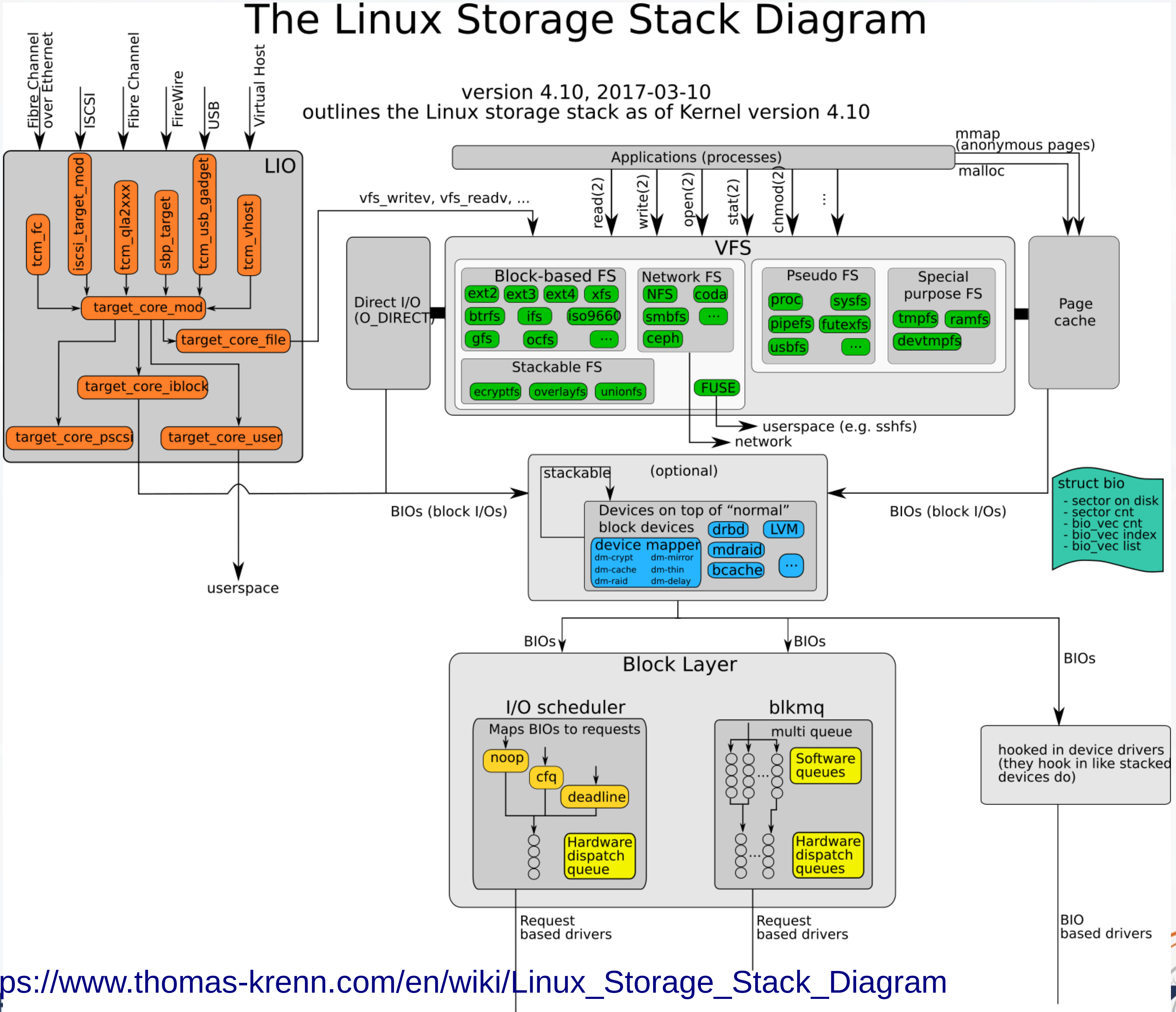


Image: https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram

Tools

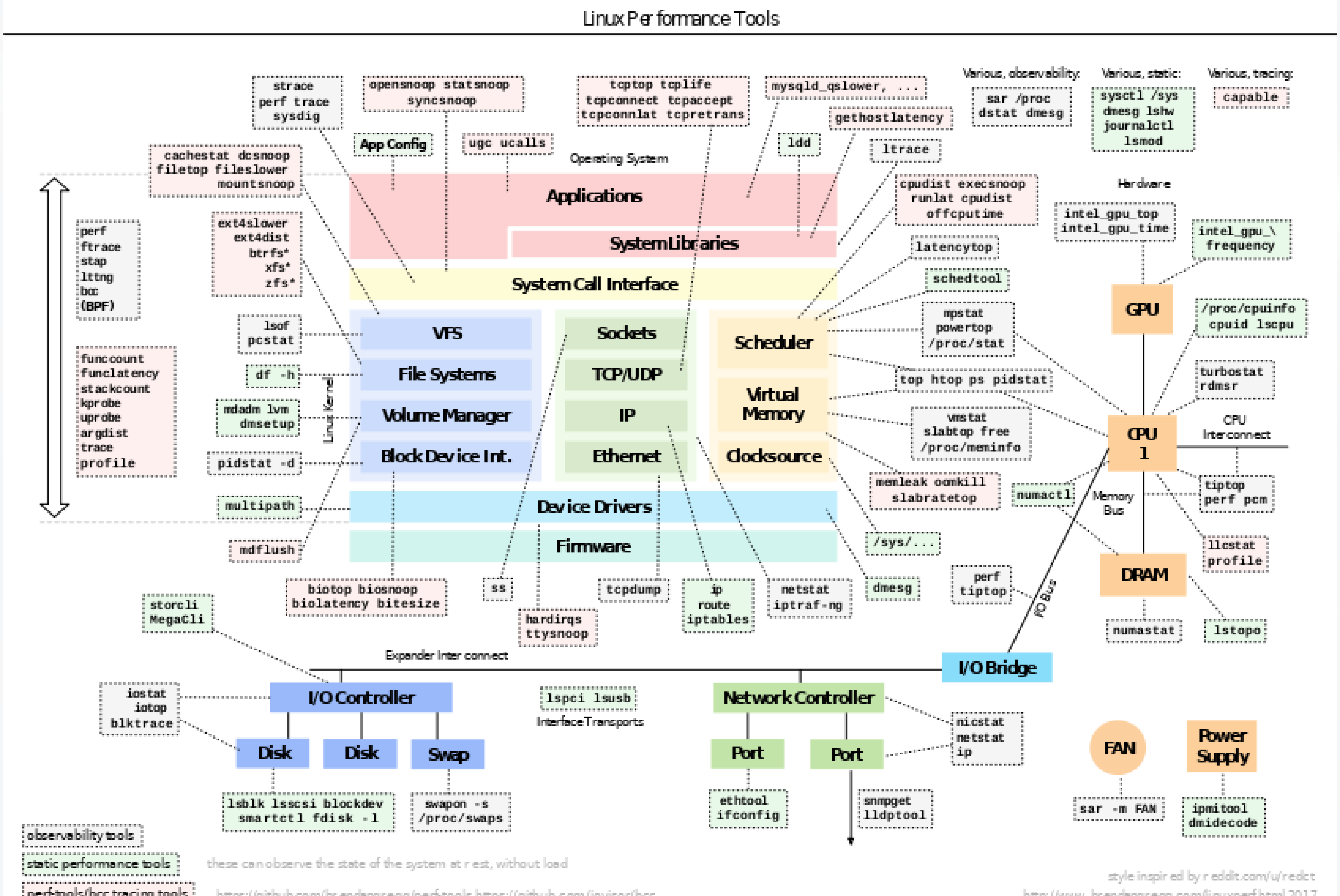
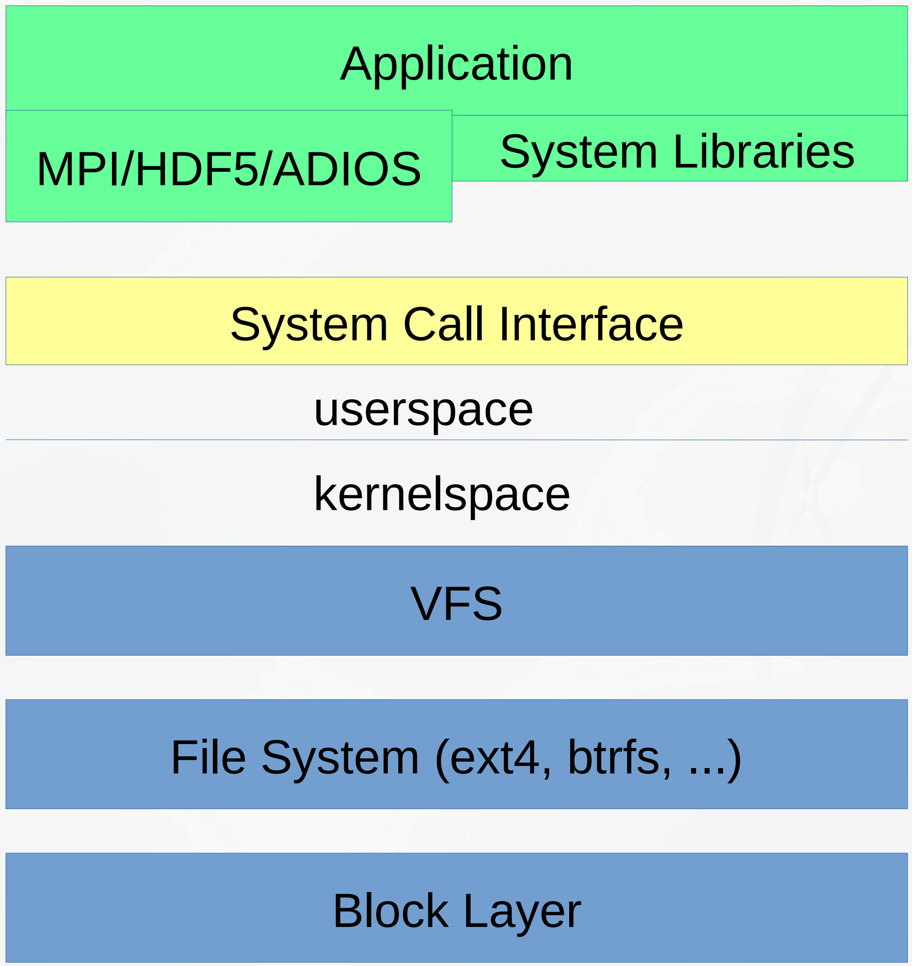


Image: http://www.brendangregg.com/Perf/linux_perf_tools_full.png

Different layer, different tool!



Application & System Libraries:
- gprof – GNU Profiler
- ltrace – trace library calls
- uprobes – dynamic userspace tracepoints

System Call Interface:
- strace – trace syscalls w. ptrace()
- sysdig – needs kernel module
- perf – use Kernel trace events

VFS:
- lsof – list open files
- pcstat

File System:
- perf – as swiss army knife
- Fs specific tools

Block Layer:
- iostat
- iotop
- blktrace

Score-P & Vampir

Score-P

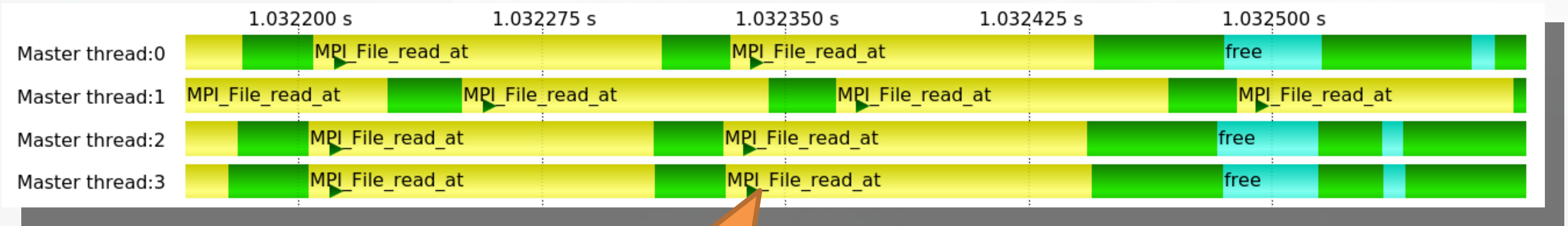
Scalable performance measurement infrastructure for parallel codes

VAMPIR

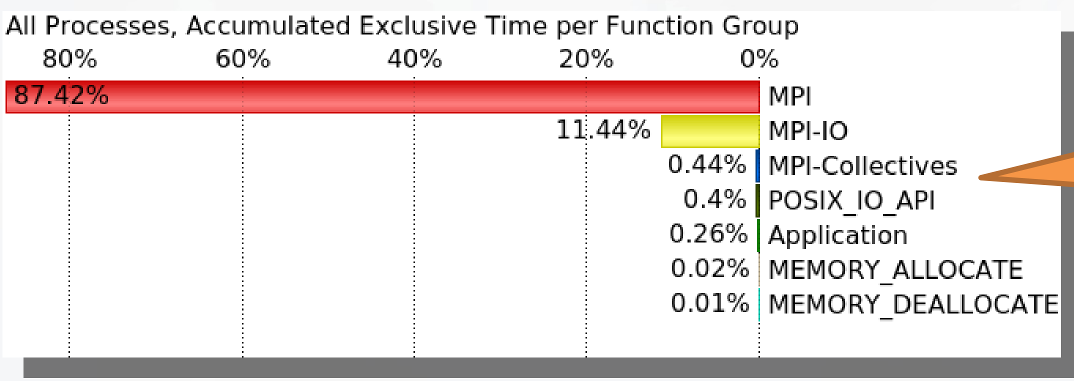
Tapping I/O Layers

- I/O layers
  - Lustre File System
    - Client side
    - Server side
  - Kernel
  - POSIX
  - MPI-I/O
  - HDF5
  - NetCDF
  - PnetCDF
  - ADIOS

I/O operations over time



Individual I/O Operation

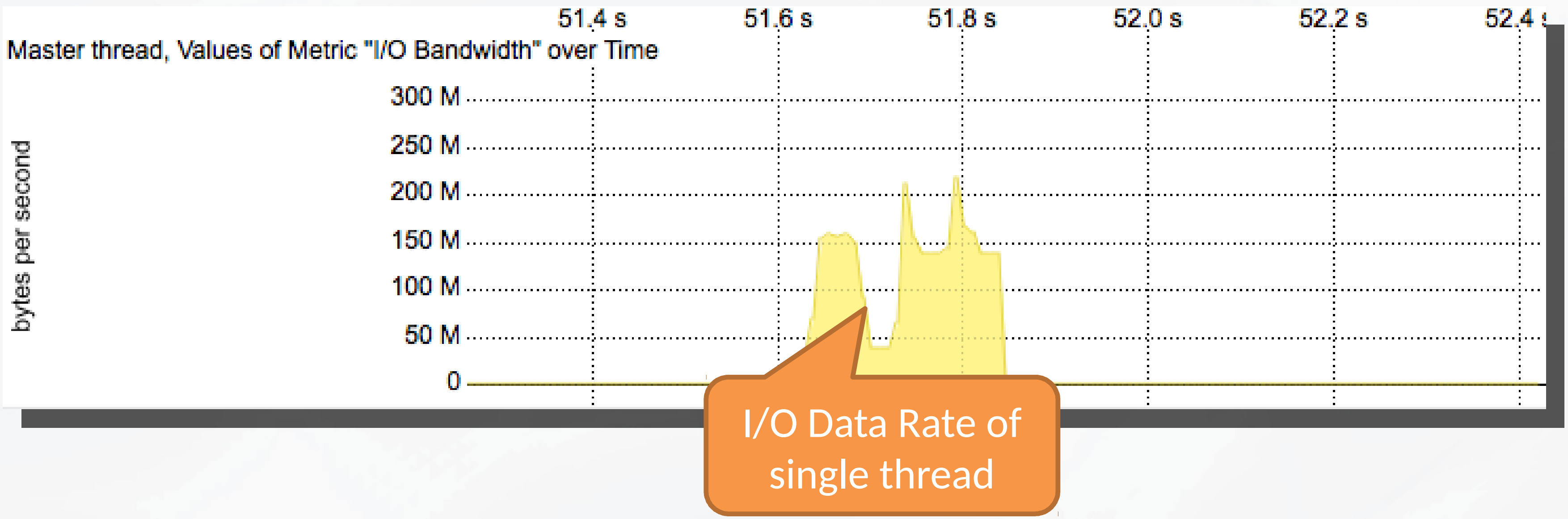


I/O Runtime Contribution

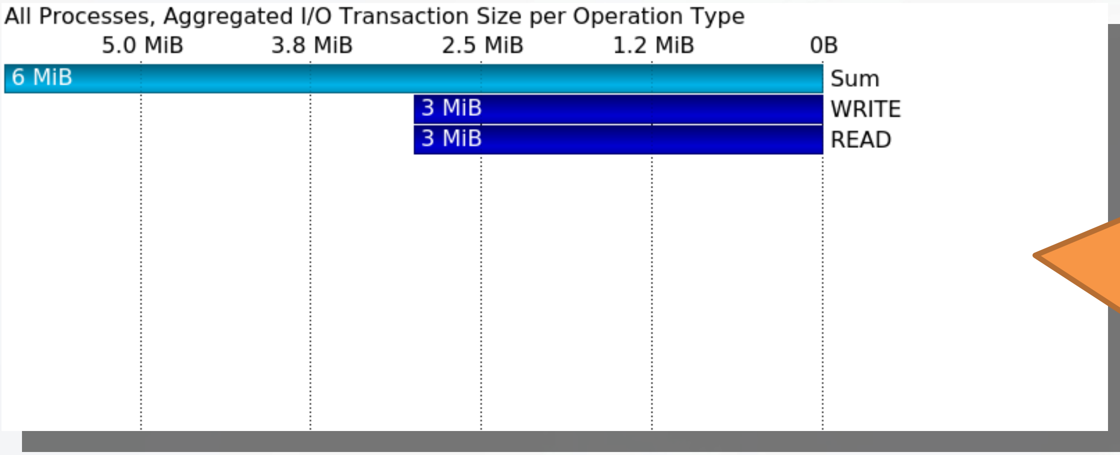
TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

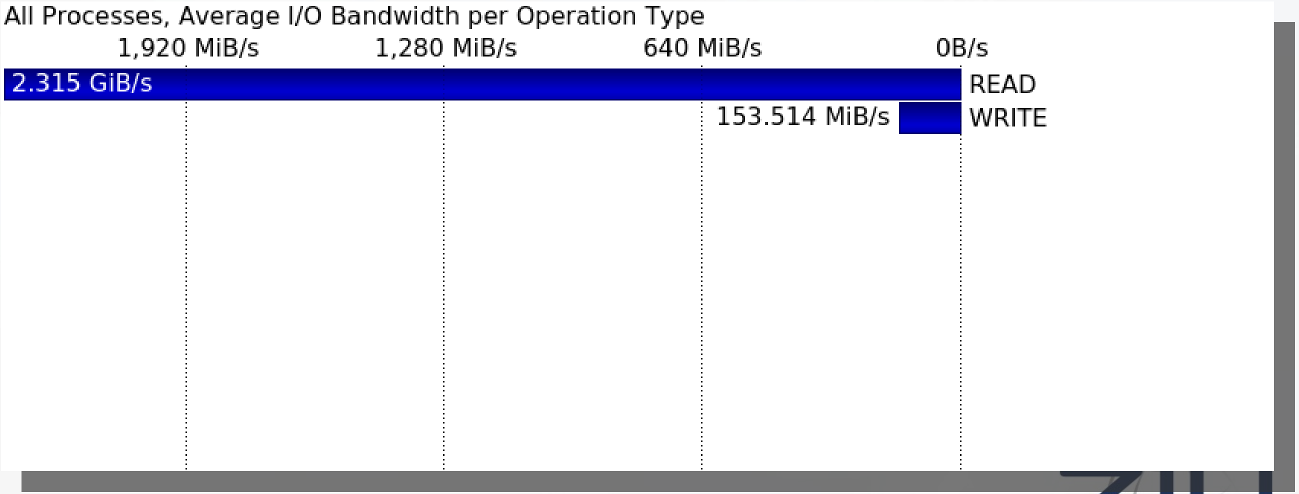
I/O data rates over time



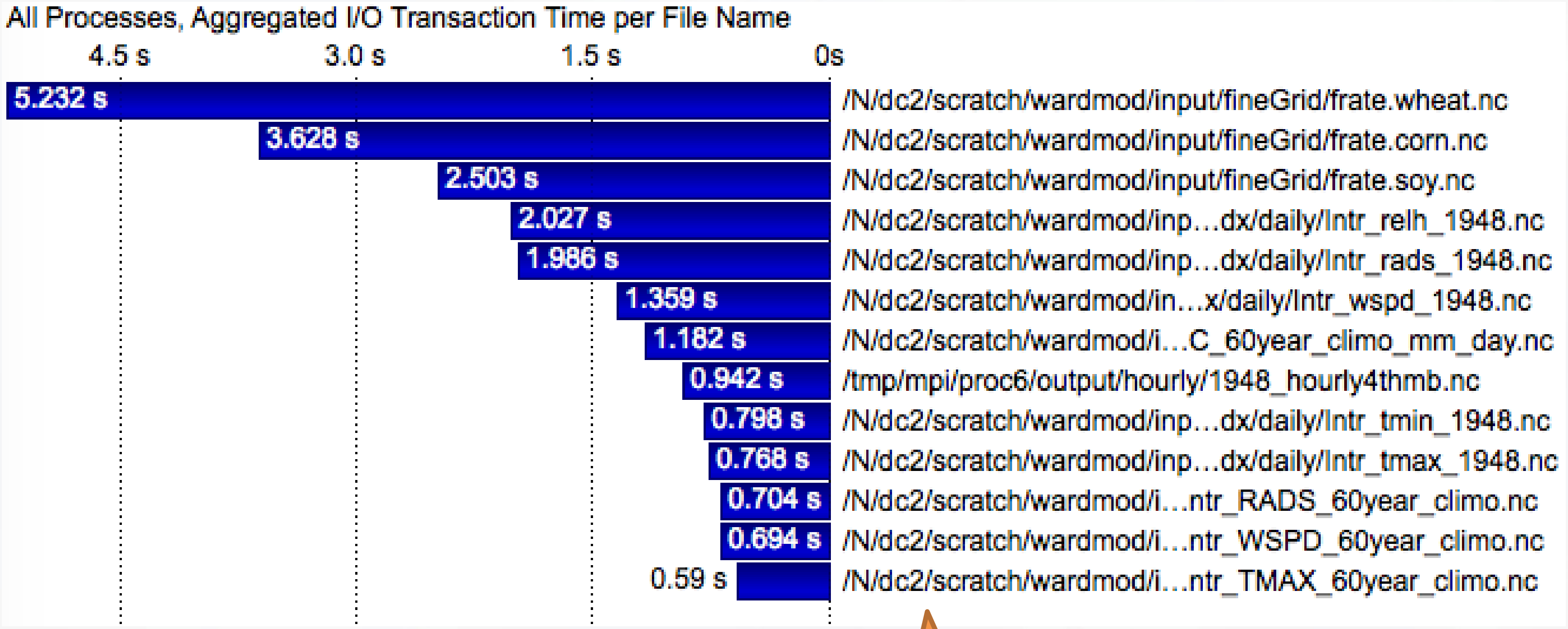
I/O summaries with totals



Other Metrics:
- IOPS
- I/O Time
- I/O Size
- I/O Bandwidth



I/O summaries per file

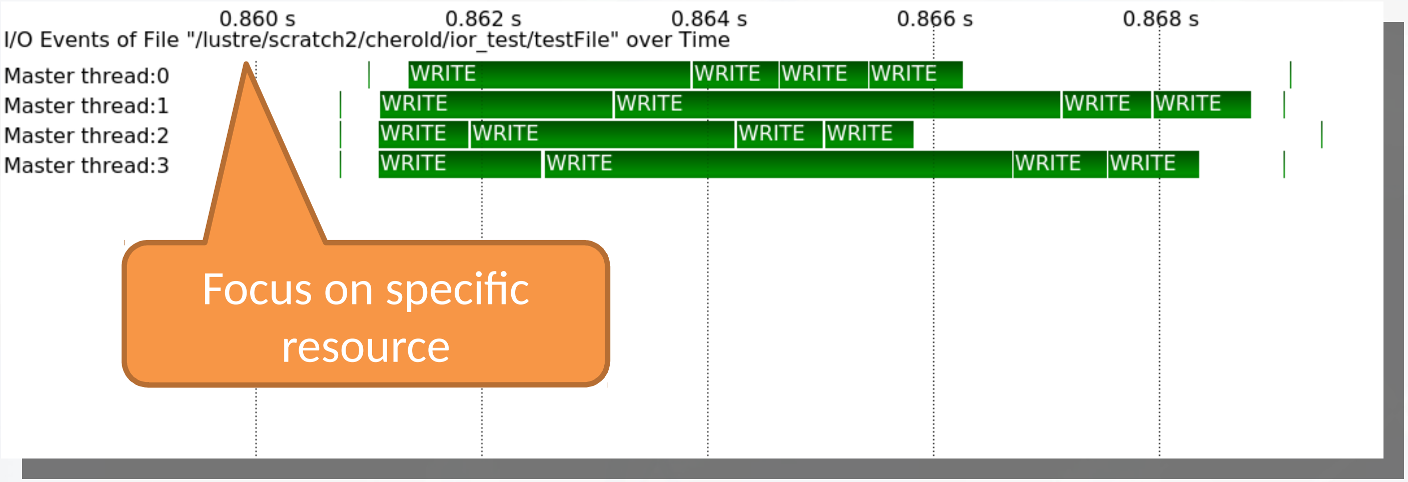


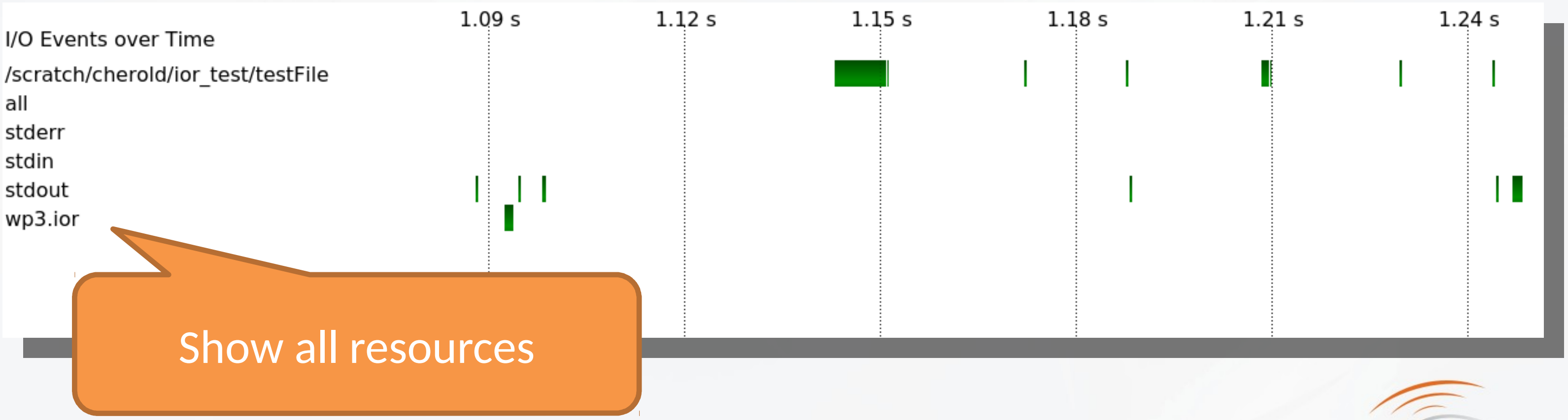
Aggregated data for specific resource

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

I/O operations per file





Tell Score-P to record I/O data

- Score-P does not record I/O data by default
  - Score-P wrapper
    - see option `--io=help`
    - has variants
  - Score-P installation
    - default if I/O libraries are detected correctly

Select I/O layer of interest

- scorep --io=netcdf --io=posix
- --io=
  - mpi
  - none
  - posix
  - netcdf
  - netcdf_par
  - hdf5

Optionally set library wrapping method

- **scorep --io=runtime:netcdf --io=linktime:posix**
- runtime:
  - I/O calls are instrumented during binary loading
  - reveals even internal I/O in libraries,
    - e.g. NetCDF doing POSIX
  - requires **--dynamic** link option in scorep
- linktime: (default)
  - I/O calls are instrumented when linking
  - reveals direct calls to I/O only
    - e.g. your code doing MPI-IO but not the I/O underneath

I/O data recording and static linking

- --static
  - symbols are resolved during compile and link time
  - user calls to I/O libraries are recorded
  - internal I/O in libraries not recorded
    - if library is not compiled with scorep
- --dynamic
  - symbols are resolved loading binary into memory
  - needed for **--io=runtime:posix**

How to use Score-P for your application?

In your makefile:

```
PREP = scorep --dynamic --io=runtime:netcdf --io=runtime:posix
CC = $(PREP) gcc
CFLAGS = -Wall -Wextra

instrumented: foo.c
    $(PREP) $(CC) $(CFLAGS) -o foo foo.c
```

In your batch file:

```
#! /bin/bash
#SBATCH -nodes=256
#SBATCH -ntasks=256
#SBATCH ...

export SCOREP_ENABLE_TRACING=true
export SCOREP_ENABLE_PROFILING=false
export SCOREP_TOTAL_MEMORY=256MB
export SCOREP_METRIC_RUSAGE=ru_stime,ru_inblock,ru_oublock

srun -n 256 ./your-app
```