esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Outline

9:00am

- Infrastructure hardware: - 30 minutes -KC
  - Storage devices characteristics
  - Storage devices evolution
  - Importance of software in infrastructure
  - Resulting stack and standardization aspects
  - New applications
- Infrastructure software - 30 minutes - Sai
  - posix
  - mpi-io
  - netcdf
  - object
- Storage trend and possible futures
  - Deep and multi-tier storage hierarchy
  - Technical challenges
    - metadata, data policies, fault tolerance
    - perspective - Storage Class Memory

10:00am KC

- Introduction to Darshan - 30 minutes -
  - Why, Install, HOWTO
  - Darshan DXT

10:30am virtual break

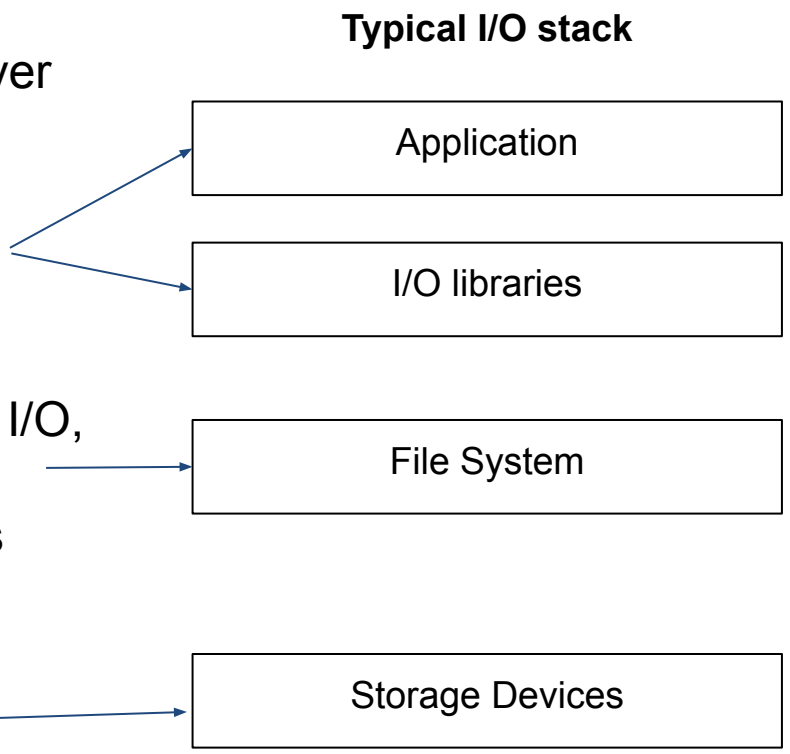
10:45am - KC

- Hands-on session - 1H -
  - 4 differents code to analyse

12:00 wrap-up

I/O tracing and monitor possibilities

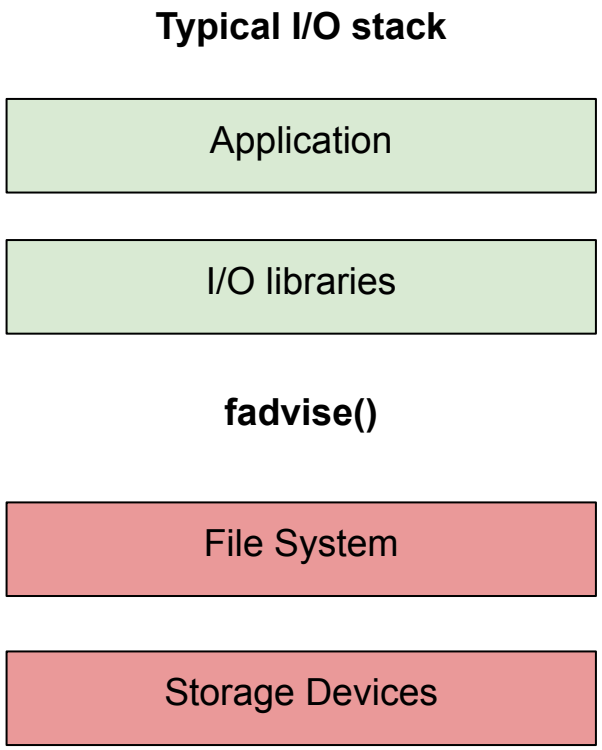
- I/O path consist of several layers
  - Different options to monitor each layer
- Applications level tracing
  - Tools: Darshan, Scalatrace etc.
- File System instrumentation
  - Not always the same as application I/O, libraries may modify the I/O pattern
  - Tools: Lustre and GPFS diagnostics
- Block device instrumentation
  - I/O requests to the actual devices
  - Tools: Linux block tracing, etc.



I/O stack tuning

- I/O the access pattern in all layer impacts performance

- Applications developers
  - Control the I/O request from the Application to the I/O libraries and the file system
  - Can **not** control how file system will internally translate their I/O requests
    - However, the **fadvice()** to pass hints for access patterns
- File System developers / System Admins
  - Control file system request to storage devices



esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Application level I/O monitoring with Darshan

- What is Darshan
  - Name means “sight” or “vision” in Sanskrit
  - Lightweight, scalable I/O characterization tool
  - Transparently captures application I/O access pattern information
  - Open source library and runtime
    - Developed and maintained at Argonne National Laboratory

Key features

- Captures several I/O interfaces
  - POSIX I/O, MPI-IO, and limited HDF5 and PNetCDF
- Instrumentation on compile time or at run time
- Compatible with popular compilers and MPI implementations
- File system agnostic
  - Can be used with any file system
- Does not impact application performance in measurable way
  - Use it on production runs
- No need for applications code modification

Components

- **darshan-runtime**
  - Used to capture I/O statistics while the application is running
  - Installed on an HPC system to instrument MPI applications
    - Installation steps vary depending on the platform
- **darshan-util**
  - Use to annayle Darshan log files
  - Installed on a workstation to analyze Darshan log files
    - (log files themselves are portable)
  - Installation is generic for almost any unix-like platform

Compilation and Installation process

- System-wide (available to all users)
- User's home directory (no root access required)
  - There is no difference in functionality
- Download source code from
  - https://www.mcs.anl.gov/research/projects/darshan/download/
- `tar -zxvf darshan-$version.tar.gz`
- Compile Darshan runtime, use the same compiler as your application
  - `cd darshan-$version/darshan-runtime`
  - `./configure CC=mpicc --prefix=$installation-dir --with-log-path-by-env=DARSHAN_LOGPATH --with-jobid-env=NONE --with-mem-align=128`
  - `make && make install`
- Compile Darshan util
  - `cd darshan-$version/darshan-util`
  - `./configure --prefix=$installation-dir`
  - `make && make install`

How to use it

- The simplest method to use Darshan is to build a dynamic executable that is dynamically linked with the MPI library
  - To determine if your executable is dynamic or not:
    - `ldd a.out`
    - `libmpi.so.1 => /$inst_path/libmpi.so.1 [ ... ]`
- Set log path directory
  - `export DARSHAN_LOGPATH=./`
- Then prefix the MPI execution command with the Darshan library
  - `LD_PRELOAD=$path/libdarshan.so mpirun -np 4 a.out`
- Each job instrumented with Darshan produces a single log file
  - Application must call MPI_Finalize() to generate the log file
- Darshan command line utilities are used to analyze these log files
- Online doc:
  - https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-runtime.html

Log file analysis tools

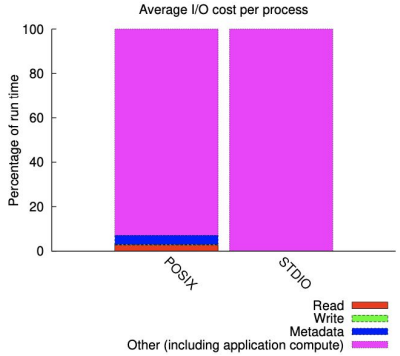
- **darshan-job-summary.pl**
  - creates pdf file with graphs useful for initial analysis
  - packages needed: Perl, pdflatex, epstopdf, and gnuplot
- **darshan-summary-per-file.sh**
  - similar to above, but creates a separate pdf file for each file opened by the application
- **darshan-parser**
  - dumps all information into ascii (text) format

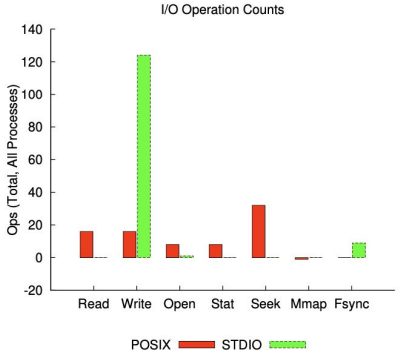
- Online documentation at
  - https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-util.html

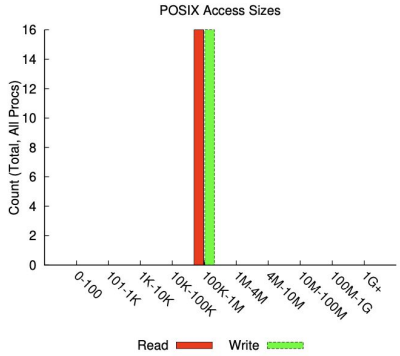
Darshan job summary example

jobid: 202545	uid: 1008	nprocs: 4	runtime: 1 seconds

I/O performance *estimate* (at the POSIX layer): transferred 8.0 MiB at 101.99 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 2.71 MiB/s







esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Darshan job parser example

#<module>	<rank>	<record id>	<counter>	<value>	<file name>	<mount pt>	<fs type>
POSIX	-1	22..36	POSIX_OPENS	8	file	/home	nfs4
POSIX	-1	22..36	POSIX_FILENOS	0	file	/home	nfs4
POSIX	-1	22..36	POSIX_DUPS	0	file	/home	nfs4
POSIX	-1	22..36	POSIX_READS	16	file	/home	nfs4
POSIX	-1	22..36	POSIX_WRITES	16	file	/home	nfs4
POSIX	-1	22..36	POSIX_SEEKS	32	file	/home	nfs4
POSIX	-1	22..36	POSIX_STATS	8	file	/home	nfs4
POSIX	-1	22..36	POSIX_MMAPS	-1	file	/home	nfs4
POSIX	-1	22..36	POSIX_FSYNCS	0	file	/home	nfs4
POSIX	-1	22..36	POSIX_MODE	436	file	/home	nfs4
POSIX	-1	22..36	POSIX_BYTES_READ	4194304	file	/home	nfs4

Darshan eXtended Tracing (DXT) module

- “Advanced” Darshan to report every intercepted call
- Not on by default, to enable
  - export DXT_ENABLE_IO_TRACE=1
- I/O Traces appear as a time series
- Special tool for post process analysis
  - darshan-dxt-parser
- Provide tools for applying different types of analyses to the logs.
- Provides different levels of granularity
  - DXT_TRIGGER_CONF_PATH environment variable to notify DXT of the path of the configuration file
    - file triggers: trace files based on regex matching of file paths
    - rank triggers: trace files based on regex matching of ranks
    - dynamic triggers: trace files based on runtime analysis of I/O characteristics (e.g., frequent small or unaligned I/O accesses

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

darshan-dxt-parser example output

```
# ************************************************
# DXT_POSIX module data
# ************************************************

# DXT, file_id: 16457598720760448348, file_name: /tmp/test/testFile
# DXT, rank: 0, hostname: shane-thinkpad
# DXT, write_count: 4, read_count: 4
# DXT, mnt_pt: /, fs_type: ext4
# Module    Rank  Wt/Rd  Segment        Offset       Length    Start(s)      End(s)
 X_POSIX      0  write       0             0       262144      0.0029      0.0032
 X_POSIX      0  write       1        262144       262144      0.0032      0.0035
 X_POSIX      0  write       2        524288       262144      0.0035      0.0038
 X_POSIX      0  write       3        786432       262144      0.0038      0.0040
 X_POSIX      0   read       0             0       262144      0.0048      0.0048
 X_POSIX      0   read       1        262144       262144      0.0049      0.0049
 X_POSIX      0   read       2        524288       262144      0.0049      0.0050
 X_POSIX      0   read       3        786432       262144      0.0050      0.005
```

Other darshan tools

- **darshan-convert:**
  - converts an existing log file to the newest log format
- **darshan-diff:**
  - provides a text diff of two Darshan log files, comparing both job-level metadata and module data records between the files
- **darshan-analyzer:**
  - walks an entire directory tree of Darshan log files and produces a summary of the types of access methods used in those log files
- **dxt_analyzer**:
  - plots the read or write activity of a job using data obtained from Darshan's DXT modules (if DXT is enabled)

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Hands on tutorial

- Download virtual machine
  - https://rb.gy/n82oex
- Download sample applications
  - https://github.com/kchasapis/esiwace_demo_darshan

Guidelines for optimizing I/O

- Large request size
- Avoid single shared file for parallel file systems
- Sequential I/O performs always better
- For MPI-I/O Collective I/O results in better performance

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

zenodo

OPEN ACCESS

The ESiWACE2 is on Zenodo, the Open Access repository for our results
https://zenodo.org/communities/esiwace

Interested in getting in touch?
Twitter: https://twitter.com/esiwace
Website: www.esiwace.eu

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988