Autoperf
AutoPerf 1.0

- AutoPerf 1.0 was a standalone tool for collecting MPI and BG/Q specific data
  - Deployed on ANL Mira system
  - Aspects of the implementation were loosely modeled on Darshan
  - Major findings published in
    - https://dl.acm.org/doi/10.1109/SC.2018.00033
AutoPerf 2.0

• Decided to rebuild Autoperf as module of Darshan
  – Reuse existing data capture and analysis frameworks
  – Focus on adding value with MPI, network and performance counters

• Limitations of 1.0
  – Only data from 4 ranks is logged and thus far, only data from the avg. rank has been used (rank with MPI time close to avg. MPI time)
  – MPI specific issues:
    • Per an MPI operation, only the average time is recorded – distribution is not captured
    • Per an MPI operation, only the average message size used is recorded – distribution is not captured
    • Message sizes for collectives like Alltoallv are not accurate
    • MPI Multi-threading – correctness issues (counters support atomic increments or not)
AutoPerf 2.0 Design

- AutoPerf becomes a submodule within the Darshan library
  - Reduce redundant work by leveraging existing logging/reporting framework
  - Compiler/linker integration, log structure, testing, deployments

- MPI specific
  - Intercept more MPI operations
    - 359 total ops in MPI 3.1 standard
      - 74 prominently used ops are intercepted
        - MPI3 ops such as RMA and non-blocking collectives are also intercepted
        - Add distribution counters for message size (six bins such as [0-256B], [256B-1K] … [1MB+])
    - MPI stats from every rank is logged
    - Reduction and analysis of the log records from all the ranks is by a post-processing tool
    - A python based post-processing (pydarshan) is under development
Autoperf Module

- External to the darshan repo
  - Autoperf has its own git repo: https://github.com/argonne-lcf/autoperf
  - Modified darshan to allow for external modules
    - Still require those modules to be defined with in darshan header
    - Configuration parameters also in darshan repo
  - Currently can only be built and run in the context of Darshan
  - Future – simplified build, interception and log system to facilitate use separate from Darshan

- Designed as multiple modules for different aspects
  - Users can choose what aspects of Autoperf they want to use on their systems
  - apmpi – MPI counters, system agnostic
  - apxc – Cray XC Aries counters
  - apss – HPE Slingshot counters
  - apnvgpu – Nvidia GPU performance data via TAU
GitHub View

AutoPerf is a module for Darshan which tracks compute and network metrics on the Cray XC class systems.

```
1 [submodule "modules/autoperf"]
2 path = modules/autoperf
3 url = https://github.com/argonne-lcf/autoperf.git
4 branch = main
```
#define DARSHAN_MODULE_IDS 
X(DARSHAN_NULL_MOD,  "NULL",          DARSHAN_NULL_VER,    NULL) \ 
X(DARSHAN_POSIX_MOD, "POSIX",         DARSHAN_POSIX_VER,  &posix_logutils) \ 
X(DARSHAN_MPIIO_MOD, "MPI-O",         DARSHAN_MPIIO_VER,  &mpiio_logutils) \ 
X(DARSHAN_H5F_MOD,   "H5F",           DARSHAN_H5F_VER,    &hdf5_file_logutils) \ 
X(DARSHAN_H5D_MOD,   "H5D",           DARSHAN_H5D_VER,    &hdf5_dataset_logutils) \ 
X(DARSHAN_PNETCDF_MOD, "PNETCDF",     DARSHAN_PNETCDF_VER, &pnetcdf_logutils) \ 
X(DARSHAN_BGQ_MOD,   "BG/Q",          DARSHAN_BGQ_VER,    &bgq_logutils) \ 
X(DARSHAN_LUSTRE_MOD, "LUSTRE",       DARSHAN_LUSTRE_VER, &lustre_logutils) \ 
X(DARSHAN_STDIO_MOD, "STDIO",         DARSHAN_STDIO_VER, &stdio_logutils) \ 
X(DXT_POSIX_MOD,     "DXT_POSIX",     DXT_POSIX_VER,    &dxt_posix_logutils) \ 
X(DXT_MPIIO_MOD,     "DXT_MPIIO",     DXT_MPIIO_VER,    &dxt_mpiio_logutils) \ 
X(DARSHAN_MDHIM_MOD, "MDHIM",         DARSHAN_MDHIM_VER, &mdhim_logutils) \ 
X(DARSHAN_APXC_MOD,  "APXC",          __APXC_VER,       __apxc_logutils) \ 
X(DARSHAN_APMPI_MOD, "APMPI",         __APMPI_VER,      __apmpi_logutils) \ 
X(DARSHAN_HEATMAP_MOD, "HEATMAP",     DARSHAN_HEATMAP_VER, &heatmap_logutils)
Build and Use

- git submodule update --init
- Configure with --enable-apmpi-mod and/or --enable-apxc-mod to enable autoperf at configuration time
  - Build and run darshan as normal
  - Darshan logs will contain these modules
    - Data can be viewed with darshan-parser
Analysis

- Initial prototype analysis in python
- Plan to integrate analysis tools into pydarshan work
This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.