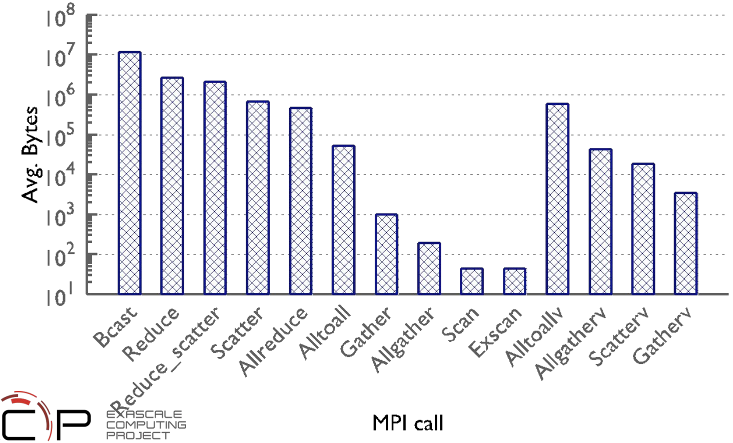
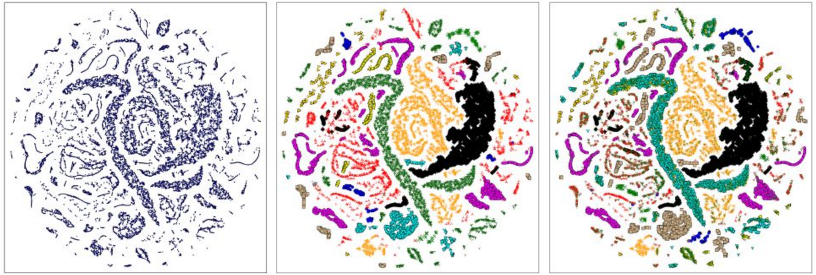
Autoperf

AutoPerf 1.0

DARSHAN
HPC I/O Characterization Tool

- 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
    - https://dl.acm.org/doi/abs/10.1145/3392717.3392774





(a) Unlabeled 2D t-SNE embedding.

(b) Labeled by executable name.

(c) Labeled by username.

Figure 10: Two-dimensional t-SNE embedding of task representation. Dots (tasks) with the same color share the same information: executable name in (b) and username in (c).

EXASCALE COMPUTING PROJECT

Argonne NATIONAL LABORATORY 75 1946–2021

AutoPerf 2.0

- Decided to rebuild Autoperf as module of Darshan
  - Reuse existing data capture and analysis frameworks
  - Focus on adding value with 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

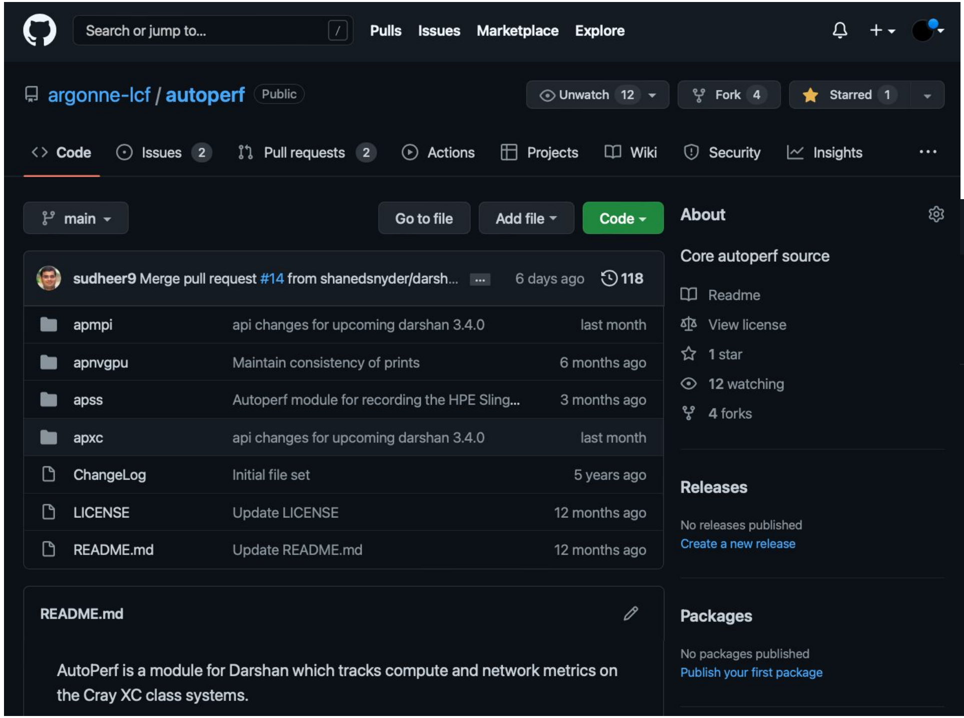
- AutoPerf becomes a submodule within the Darshan library
  - Reduce redundant work by leveraging existing logging/reporting framework
  - Compiler/linker integration, log structure, testing, deployments

- 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



argonne-lcf / autoperf Public

sudheer9 Merge pull request #14 from shanedsnyder/darsh... 6 days ago 118

apmpi	api changes for upcoming darshan 3.4.0	last month
apnvgpu	Maintain consistency of prints	6 months ago
apss	Autoperf module for recording the HPE Sling...	3 months ago
apxc	api changes for upcoming darshan 3.4.0	last month
ChangeLog	Initial file set	5 years ago
LICENSE	Update LICENSE	12 months ago
README.md	Update README.md	12 months ago

README.md

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

About

Core autoperf source

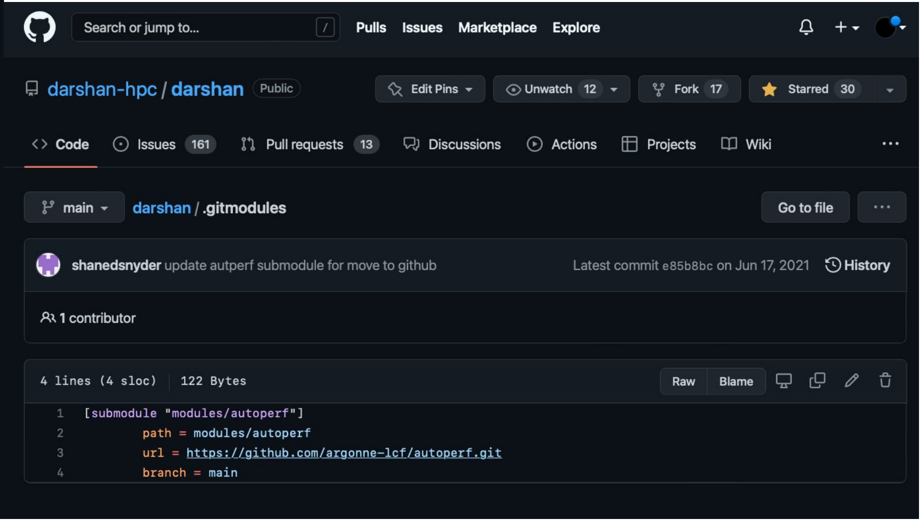
- Readme
- View license
- 1 star
- 12 watching
- 4 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package



darshan-hpc / darshan Public

darshan / .gitmodules

shanedsnyder update autperf submodule for move to github   Latest commit e85b8bc on Jun 17, 2021   History

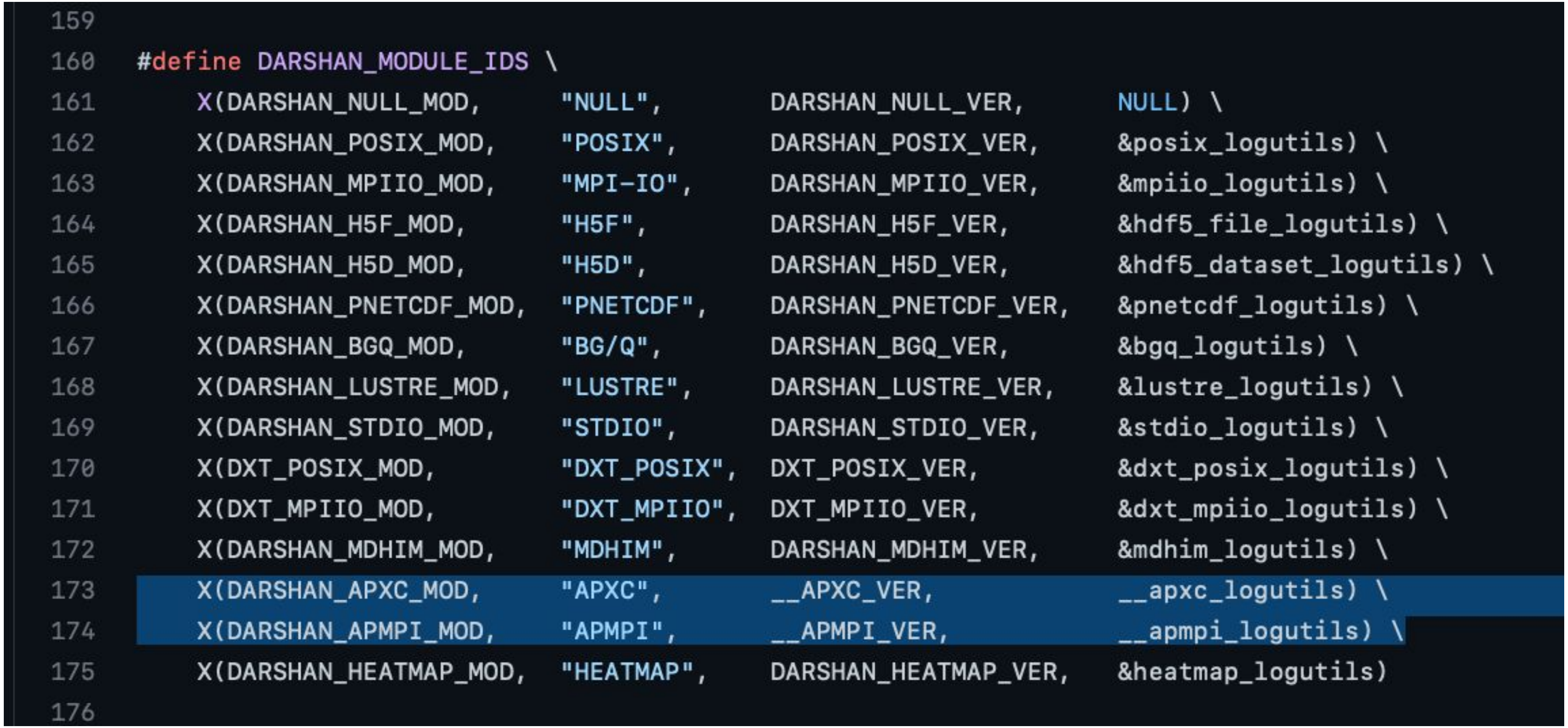
1 contributor

4 lines (4 sloc) 122 Bytes

```
[submodule "modules/autoperf"]
	path = modules/autoperf
	url = https://github.com/argonne-lcf/autoperf.git
	branch = main
```

darshan-log-format.h

DARSHAN
HPC I/O Characterization Tool



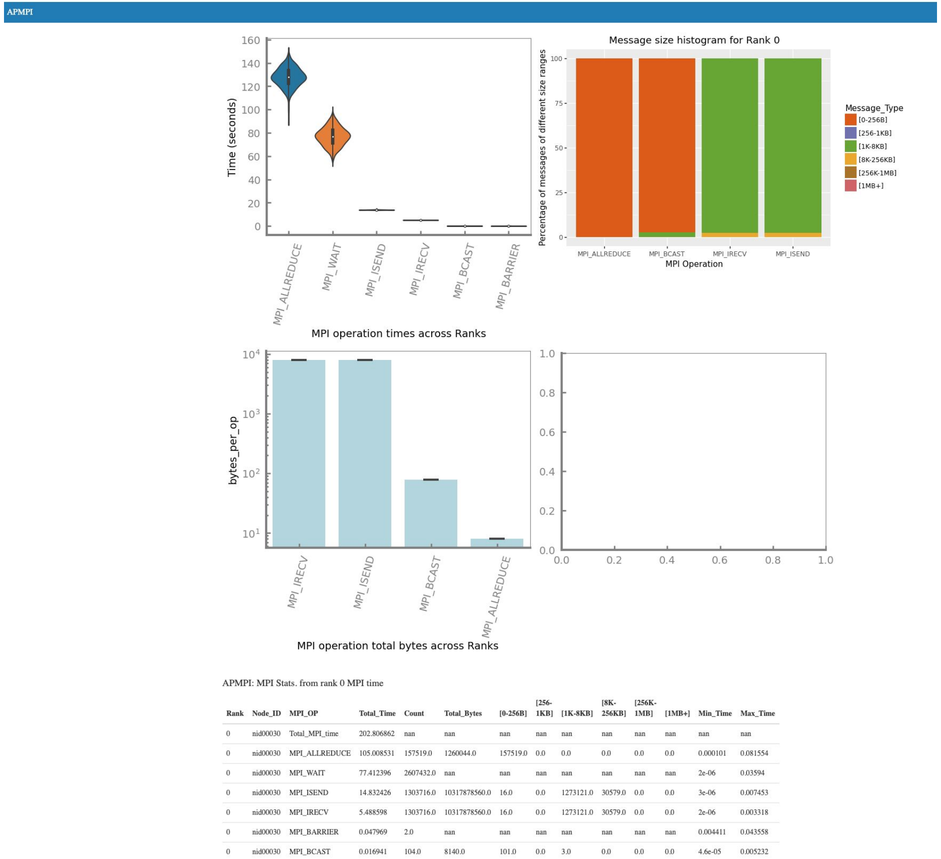
```
#define DARSHAN_MODULE_IDS \
    X(DARSHAN_NULL_MOD,      "NULL",       DARSHAN_NULL_VER,       NULL) \
    X(DARSHAN_POSIX_MOD,     "POSIX",      DARSHAN_POSIX_VER,      &posix_logutils) \
    X(DARSHAN_MPIIO_MOD,     "MPI-IO",     DARSHAN_MPIIO_VER,      &mpiio_logutils) \
    X(DARSHAN_H5F_MOD,       "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
- https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-runtime.html#_using_autoperf_instrumentation_modules

Analysis

DARSHAN
HPC I/O Characterization Tool



MPI operation times across Ranks

MPI operation total bytes across Ranks

APMPI: MPI Stats. from rank 0 MPI time

Rank	Node_ID	MPI_OP	Total_Time	Count	Total_Bytes	[0-256B]	[256-1KB]	[1K-8KB]	[8K-256KB]	[256K-1MB]	[1MB+]	Min_Time	Max_Time
0	nid00030	Total_MPI_time	202.806862	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
0	nid00030	MPI_ALLREDUCE	105.008531	157519.0	1260044.0	157519.0	0.0	0.0	0.0	0.0	0.0	0.000101	0.081554
0	nid00030	MPI_WAIT	77.412396	2607432.0	nan	nan	nan	nan	nan	nan	nan	2e-06	0.03594
0	nid00030	MPI_ISEND	14.832426	1303716.0	10317878560.0	16.0	0.0	1273121.0	30579.0	0.0	0.0	3e-06	0.007453
0	nid00030	MPI_IRECV	5.488598	1303716.0	10317878560.0	16.0	0.0	1273121.0	30579.0	0.0	0.0	2e-06	0.003318
0	nid00030	MPI_BARRIER	0.047969	2.0	nan	nan	nan	nan	nan	nan	nan	0.004411	0.043558
0	nid00030	MPI_BCAST	0.016941	104.0	8140.0	101.0	0.0	3.0	0.0	0.0	0.0	4.6e-05	0.005232

- 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.