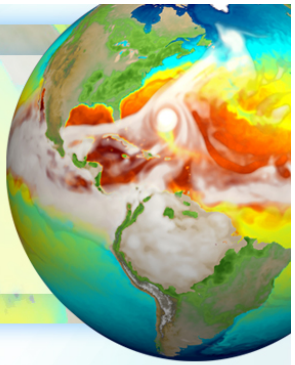

zstash v0.4.2: HPSS long-term archiving tool

Core Development Team: Ryan Forsyth, Chris Golaz, and Zeshawn Shaheen

Lawrence Livermore National Lab

October 2020

E3SM Energy Exascale Earth System Model

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. It is supported by the Energy Exascale Earth System Model (E3SM) project, funded by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research. IM Release LLNL-PRES-815713

U.S. DEPARTMENT OF ENERGY

About zstash

- Zstash is a Python3 utility developed to serve E3SM long-term archiving needs.
- Files are archived into **standard tar** files with a user specified **maximum size**.
- Tar files are created locally, then transferred to HPSS.
- **Checksums (md5)** are computed *on-the-fly* during archiving.
- Metadata stored in a sqlite3 index **database**.
- Database enables **faster retrieval** of individual files by locating in which tar file a specific file is stored, as well as its location (offset) within the tar file.
- **File integrity** is verified by computing checksums *on-the-fly* while extracting files.
- **Parallel extraction and verification** for increased performance.

Terminology

- **zstash archive**: A set of files {index.db, nnnnnn.tar where nnnnnn=000000, 000001, …}.
- **HPSS**: A long-term tape storage system.
- **HPSS archive**: The zstash archive on HPSS.
- **Local archive (or cache)**: The zstash archive on our local file system. The default name is `zstash`.
- **Source directory**: The directory whose contents we are archiving.

Commands

- **Usage documentation:** https://e3sm-project.github.io/zstash/docs/html-v0-4-2/usage.html
- **Create**: create a **new local archive (cache; default name is `zstash`)** in the source directory, create a tar file of the source directory's contents, and if using HPSS, then store the tar file on the HPSS archive.
  - $ zstash **create** --hpss=<**HPSS archive** path> <**source directory** path>
- **Check**: verify integrity of zstash archive (e.g., if using HPSS, check that files were uploaded on HPSS successfully).
  - $ zstash **check** --hpss=<**HPSS archive** path> [--workers=<num of processes>] [--cache=<cache>] [--keep] [-v] [files]
- **Update**: add new or modified files to an existing zstash archive (ignoring unmodified files).
  - $ zstash **update** --hpss=<**HPSS archive** path> [--cache=<cache>] [--dry-run] [--exclude] [--keep] [-v]
- **Extract**: extract files from an existing zstash archive into current directory.
  - $ zstash **extract** --hpss=<**HPSS archive** path> [--workers=<num of processes>] [--cache=<cache>] [--keep] [-v] [files]
- **List (ls)**: view files in an existing zstash archive.
  - $ zstash **ls** --hpss=<**HPSS archive** path> [-l] [--cache=<cache>] [-v] [files]
- **Version**: show version number.

New Features

- `--hpss=none` option (as of v0.4.1)
  - Use this option if you don't want to use HPSS.
  - This is useful if you're using a machine without HPSS (e.g., Compy).
- `--cache` option (as of v0.4.2)
  - Allow users to specify a local archive (cache) named something other than `zstash`.

Example 1: archive or extract simulation data

- https://e3sm-project.github.io/zstash/docs/html/best_practices.html#nersc
- Suppose you have a directory `e3sm_output` of simulation data (e.g., `20180215.DECKv1b_H1.ne30_oEC.edison.cam.h0.1850-01.nc `) to archive.
- We could archive that data with:
  - $ ssh dtn01.nersc.gov
  - $ screen
  - $ bash
  - $ source /global/cfs/cdirs/e3sm/software/anaconda_envs/load_latest_e3sm_unified.sh
  - $ cd $CSCRATCH/example
  - $ zstash create --hpss=path/to/zstash_archive e3sm_output
- Someone else could then extract the data from the HPSS archive with:
  - $ zstash extract --hpss=path/to/zstash_archive

Example 2: transfer to a machine that has HPSS

- https://e3sm-project.github.io/zstash/docs/html/best_practices.html#compy-anvil
- Compy and Anvil do not have HPSS. How can we get that data to HPSS?
- Archive the data locally
  - $ zstash create --hpss=none e3sm_output
- Transfer to NERSC HPSS using Globus
  - Login to Globus web interface at https://www.globus.org/ using your NERSC credentials.
  - Select all files in the cache (default name `zstash`)
  - Check boxes to transfer new/changed files, preserve modification time, verify file integrity
- Check the file integrity on NERSC
  - $ ssh dtn01.nersc.gov
  - $ cd <some scratch directory>
  - $ zstash check --hpss=<HPSS path>

Figures from https://e3sm.org/new-zstash-capabilities

Performance

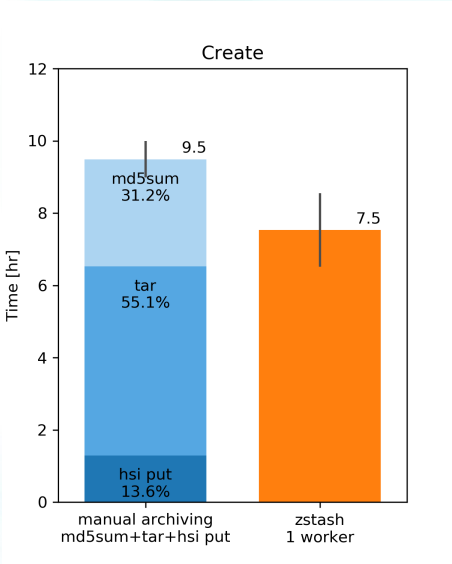

Figure 1: Performance comparison for `zstash create`. Left: manual operations with separate md5sum, tar and hsi put steps. Right: comparable combined operations with zstash. Performance data for a 4 TB archive consisting of more than 13,000 files. Mean and range of three realizations on NERSC's Data Transfer Nodes (dtn).

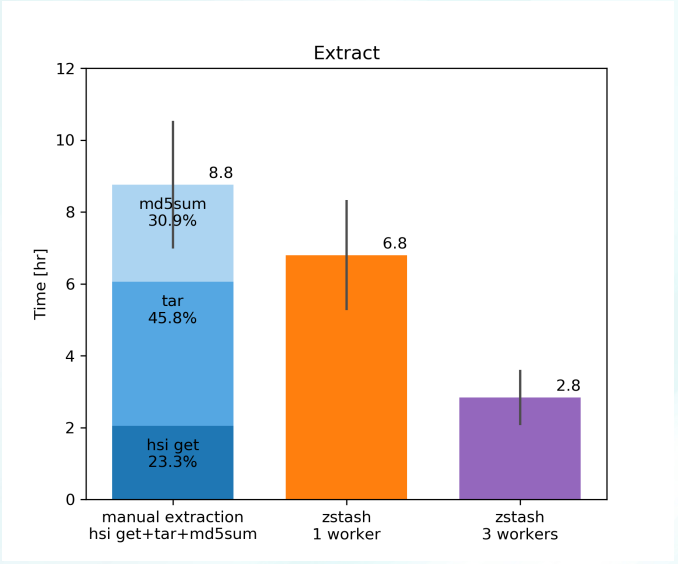

Figure 2: Performance comparison for `zstash extract`. Left: manual operations with separate hsi get, tar and md5sum steps. Middle: comparable combined operations with zstash. Right: zstash parallel with 3 workers. Performance data for a 4 TB archive consisting of more than 13,000 files. Mean and range of three realizations on NERSC's Data Transfer Nodes (dtn).

Getting Started

- On NERSC, load the E3SM unified environment which includes zstash: $ source /global/cfs/cdirs/e3sm/software/anaconda_envs/load_latest_e3sm_unified.sh
- More details: https://e3sm-project.github.io/zstash/docs/html-v0-4-2/getting_started.html

Resources

- Latest documentation (for `master` branch): https://e3sm-project.github.io/zstash/docs/html/index.html
- v0.4.2 documentation: https://e3sm-project.github.io/zstash/docs/html-v0-4-2/index.html
- v0.4.2 tutorial: https://e3sm-project.github.io/zstash/docs/html-v0-4-2/tutorial.html
- v0.4.1 ES3M article: https://e3sm.org/new-zstash-capabilities
- E3SM website: https://e3sm.org/resources/tools/data-management/zstash-hpss-archive/
- Source code: https://github.com/E3SM-Project/zstash
- Contact: Ryan Forsyth (forsyth2@llnl.gov), Chris Golaz (golaz1@llnl.gov)