Dac-Man Documentation

version 1.0.0

July 8, 2019

Contents

1

Introduction

1.1 Overview

Scientific datasets are updated frequently due to changes in instrument configuration, software updates, quality assessments or data cleaning algorithms. However, due to the large size and complex data structures of these datasets, existing tools either do not scale or are unable to generate meaningful change information.

The **Dac-Man** (**DA**ta **C**hange **Man**agement) framework allows users to efficiently and effectively identify, track and manage data change and associated provenance in scientific datasets. There are two main components of Dac-Man:

- Change tracker that keeps track of the changes between different versions of a dataset.
- Query manager that manages data change related queries.

1.2 Features

The key features of Dac-Man include:

- HPC support. Dac-Man provides MPI support for enabling parallel change capture in HPC environments.
- Offline comparison. Datasets can be compared away from the actual location of the data, allowing users to find changes without necessarily moving the datasets to a common location.

- Extendable. Users can plug-in their own scripts to calculate changes.
- Flexible command-line options. Provides different options to configure change detection.
- Detailed output. DAC-MAN outputs contain details on the different types and amount of change.
- Customizable logging. Users can customize where and what to log, including the detailed steps in the change capture process.

1.3 Requirements

DAC-MAN is developed using Python. It requires Python 2.7 or greater. Users need Python setuptools and pip to install DAC-MAN. Fore detailed instructions on the installation, please refer to Chapter 2.

DAC-MAN is known to work on the following operating systems:

- Linux
- Unix-like OSs
- Mac OS

2

Installation

This chapter describes the steps required to install Dac-Man. It requires Python version 3.6 or greater. It is installed as any other Python package and uses Python `setuptools` package. For enabling advanced features in Dac-Man, additional packages may need to be installed. You can install these packages using `pip`.

More general information about installing Python packages can be found in Python's documentation at `http://www.python.org/doc/`.

2.1 Installing Dac-Man

This section describes the steps for installing Dac-Man. Upon successful installation, you can use Dac-Man as a command-line tool.

2.1.1 Getting the Package

You can download the package from the Dac-Man repository (`https://github.com/dghoshal-lbl/dac-man`) as a tarball. Alternatively, you can clone the source tree from the repository:

```
$ git clone git@github.com:dghoshal-lbl/dac-man.git
```

2.1.2 Installation from Source

Once the package is downloaded/cloned, Dac-Man can be installed by running the following commands:

```
$ cd dacman
$ python setup.py install
```

If you are installing to a location that requires special permissions (like /usr/local), you may need to run the last command with **sudo**. Alternatively, you can create and activate a build environment through **virtualenv** or **conda** as described below.

Using virtualenv

You can install virtualenv using pip.

```
$ pip install virtualenv
```

More details on installing and using virtualenv can be found in `https://packaging.python.org/guides/installing-using-pip-and-virtualenv/`.

After installing virtualenv, you need to create and activate the environment, and then install Dac-Man.

```
$ virtualenv venv
$ source venv/bin/activate
(venv)$ cd dacman
(venv)$ python setup.py install
```

Using conda

Conda can be installed using the OS-specific installer that can be downloaded from `https://conda.io/docs/user-guide/install/index.html`. After installing, the Python environment can be created and activated as:

```
$ conda create --name env
$ source activate env
(env)$ cd dacman
(env)$ python setup.py install
```

More information about using conda environments can be found in `https://conda.io/docs/user-guide/tasks/manage-environments.html`.

2.2 Testing the Installation

In order to test the Dac-Man installation, run the following commands:

```
$ cd examples/
$ ./scripts/simple.sh
```

On successful execution, this prints the summary of change and detailed change information between two example directories.

2.3 Dependencies

Dac-Man primarily depends on the following packages:

- scandir
- six
- PyYAML
- straight.plugin

These dependencies are listed in `requirements.txt` file and are automatically installed during the build process.

Additional dependencies for running Dac-Man on HPC environments include:

- numpy : Python library for operations on large, multi-dimensional arrays
- mpi4py : Python MPI bindings

3

Using Dac-Man on Desktops

3.1 Quick Tutorial

To capture changes between two directories `dir1` and `dir2`, run the following command using the Dac-Man command-line:

```
$ dacman diff dir1 dir2
```

The above command identifies the number of files changed between the two directories. In order to retrieve detailed infromation about the changes, you can use the following command:

```
$ dacman diff dir1 dir2 --detailed
```

3.2 Command-line

Dac-Man enables change capture and analysis in four simple steps, which provide flexibility to the users in identifying and capturing changes. Dac-Man provides four command-line options to manage each of these steps separately.

3.2.1 scan

This option scans and saves the directory structure and other metadata related to a data path. You can specify an optional staging directory, where the metadata information will be saved.

```
$ dacman scan <path> [-s STAGINGDIR] [-i [IGNORE [IGNORE ...]]]
[--nonrecursive] [--symlinks]
```

The arguments to the command are:

−s STAGINGDIR : directory where filesystem metadata and indexes are saved.
−i [IGNORE [IGNORE ...]] : list of file types to be ignored.
−−nonrecursive : do not scan the directory contents recursively.
−−symlinks : include symbolic links.

3.2.2 index

This command indexes the files, mapping the files to their contents.

```
$ dacman index <path> [-s STAGINGDIR] [-m python,tigres,mpi]
```

The arguments to the command are:

−s STAGINGDIR : directory where filesystem metadata and indexes are saved.
−m python,tigres,mpi : index manager for parallelizing the index creation. The options are python/mpi/tigres. By default, it uses the Python multiprocessing module (manager=python) that is suitable for parallelizing on one node. For multi-node parallelism, users can select between MPI (manager=mpi) or tigres (manager=tigres).

3.2.3 compare

This command compares two datapaths. It compares and calculates the different types of changes.

```
$ dacman compare <oldpath> <newpath> [-s STAGINGDIR]
```

The arguments to the command are:

−s STAGINGDIR : directory where filesystem metadata and indexes are saved.

3.2.4 diff

This command retrieves changes between two datapaths.

```
$ dacman diff <oldpath> <newpath> [-s STAGINGDIR] [-o OUTDIR]
[-a ANALYZER] [--datachange] [-e default,threaded,mpi,tigres]
```

The arguments to the command are:

−s STAGINGDIR : directory where filesystem metadata and indexes are saved. −o OUTDIR : directory where the summary of changes is saved. −p plugin : user-defined scripts for analyzing data changes. −−detailed : calculate data level changes in addition to file changes. −e default,threaded,tigres,mpi : type of executor (or runtime) for parallel data change capture. The options are default/threaded/mpi/tigres. The default option uses single threaded. The threaded option uses the Python multiprocessing module that is suitable for parallelizing on one node. For multi-node parallelism, users can select between MPI or tigres.

In addition to the four commands, Dac-Man also provides two additional commands for cleanup and metadata management.

3.2.5 clean

This option removes all the indexes and cache information associated with the specified directories.

```
$ dacman clean <path> [path ...]
```

The arguments to the command are:

path : path to data directories.

3.2.6 metadata

This command allows users to add user-defined metadata for a data directory.

```
$ dacman metadata [-m METADATA] [-s STAGINGDIR] insert,retrieve,append
datapath
```

The arguments to the command are:

−s STAGINGDIR : directory where filesystem metadata and indexes are saved. −m METADATA : user-defined metadata information. insert,retrieve,append : options related to user-defined metadata information. datapath : path to the data directory.

3.3 Outputs

Dac-Man prints the summary of changes on standard output. The summary lists the number of changes between two datasets. An example output looks

like below:

```
Added: 1, Deleted: 1, Modified: 1, Metadata-only: 0, Unchanged: 1
```

You can opt to save a more detailed output by specifying the output directory where the detailed change information will be saved:

```
$ dacman diff /old/path /new/path -o output
```

The `output/` directory contains a list of files with detailed information about the changes. It also contains a summary of the change information as:

output/summary

```
counts:
  added: 1
  deleted: 1
  metaonly: 0
  modified: 1
  unchanged: 1
versions:
  base:
    dataset_id: /path/to/old/data
    nfiles: 3
  revision:
    dataset_id: /path/to/new/data
    nfiles: 3
```

4

Using Dac-Man on HPC Clusters

4.1 Using MPI

Dac-Man allows you to parallelize `index` and `diff` steps. To parallelize on HPC clusters, you need to enable the MPI support by using the appropriate flags.

```
$ dacman index ...  -m mpi
$ dacman diff ...  -e mpi
```

In order to distribute the tasks to multiple workers, you need to use `mpirun` or `mpiexec`. For example, running Dac-Man on an HPC cluster with 8 nodes and 32 cores per node, you can do the following:

```
$ mpiexec -n 256 dacman index ...  -m mpi
$ mpiexec -n 256 dacman diff ...  -e mpi
```

4.2 Batch Script

In order to submit a batch job in a cluster, you need to include the Dac-Man command in your job script. The example below shows a batch script (hpcEx.batch) for the Slurm scheduler.

hpcEx.batch

```
#!/bin/bash
#SBATCH -J example
#SBATCH -t 00:30:00
#SBATCH -N 8
#SBATCH -q myqueue

mpiexec -n 256 dacman diff /old/data /new/data -e mpi
```

The script can then be submitted to the batch scheduler as:

```
$ sbatch hpcEx.batch
```

5

Configuration & Customization

5.1 Staging

For every dataset, Dac-Man saves all metadata and index information in a staging area. Each directory in the staging area uniquely identifies the dataset (using a hash representation of the dataset path) indexed by Dac-Man. By default, this staging area for is located in `$HOME/.dacman/data`. However, the staging area can be changed to a custom location through the command-line. You can change the staging area by using the following command:

```
$ dacman index mydir/ -s mystage
```

The command above creates the indexes inside `mystage` directory. You can copy or move these indexes to compare and calculate the changes, without necessarily copying or moving the data. This is specifically useful when the datasets to be compared are located on different systems. The example below shows how can the staged indexes and metadata information be copied and compared for finding changes, without copying the data itself.

```
$ scp -r user:pwd@ /.dacman/data/ /path/to/mystage/
$ dacman diff /path/to/localdir/ /remotedir/ -s /path/to/mystage
```

5.2 Plug-ins

By default, Dac-Man compares the data by transforming file data into Dac-Man records, that uses one-dimensional arrays as their underlying data structure.

You can also use plug-ins by providing external scripts. You can use your own custom scripts as plug-ins by simply providing the path to the script. For example, `myscript` can be used as a plug-in:

```
$ dacman diff /old/path/file1 /new/path/file1 -p myscript
```

The command above uses `myscript` as an external plug-in to compare the contents of files `/old/path/file1` and `/new/path/file1` instead of the default Dac-Man data comparator. If you want to use Unix diff to compare all the modified files in the directories `dir1` and dir2, run the following command:

```
$ dacman diff /path/to/dir1 /path/to/dir2 --detailed -p /usr/bin/diff
```

The `--detailed` option tells Dac-Man to compare the data within the files of the two directories.

Finally, you can build your own plug-ins by extending the ComparatorBase class. Please refer to the document PluginBuilderGuide.pdf for details.

5.3 Logging

Dac-Man uses the standard Python logging for creating execution logs. The default logging configuration is saved in `$HOME/.dacman/config/logging.yaml` file. Dac-Man logs all INFO level messages, and prints messages with levels equal to or over the WARNING level. However, you can configure the logging as per your requirement by modifying the configuration file.

6

API

The top-level dacman module provides methods and classes for doing data comparisons at scale. Data can be compared using dacman.diff().

6.1 dacman.diff

`dacman.diff(new_file, old_file, *argv, comparator_plugin=None)`

Method to perform a single comparison using the specified plugin.

Parameters:
new_file: path to the new file (data to compare)
old_file: path to the old file (data compared against)
*args : optional parameters for specific comparisons
comparator_plugin: plugin to be used for comparing the files

Returns: None

6.2 dacman.DataDiffer

`class dacman.DataDiffer(comparisons, executor=Executor.DEFAULT)`

List of comparisons for a dataset. The executor argument specifies the type of runtime to be used for doing the comparisons.

6.3 dacman.Executor

`class dacman.Executor`

Types of runtime in Dac-Man.

DEFAULT : A single threaded runtime.
THREADED : Runtime based on Python's multiprocessing.
MPI : MPI based.
TIGRES : Tigres workflow management system.

6.4 dacman.plugins

`dacman.plugins.default.DefaultPlugin`

Module listing the system and user-defined plugins.

7

License & Copyright

7.1 License

Dac-Man is licensed under the "new" or "revised" BSD license.

```
Dac-Man (DAta Change Management) Copyright (c) 2018, The Regents of the
University of California, through Lawrence Berkeley National Laboratory
(subject to receipt of any required approvals from the U.S. Dept.  of
Energy).  All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

(3) Neither the name of the University of California, Lawrence Berkeley
National Laboratory, U.S. Dept.  of Energy nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
```

IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7.2 Copyright

The copyright for DAC-MAN is described below.

8

Team

Dac-Man is developed as part of the Deduce project, whose PI is Deborah Agarwal [daagarwal-AT-lbl.gov]. The development of Dac-Man is led by Lavanya Ramakrishnan [lramakrishnan-AT-lbl.gov].

As of now, the following developers have contributed to the development of Dac-Man:

- Devarshi Ghoshal [dghoshal-AT-lbl.gov]. Initial design and development of Dac-Man.
- Drew Paine [pained-AT-lbl.gov]. User interviews and initial evaluation.
- Abdelrahman Elbashandy [aaelbashandy-AT-lbl.gov]. Extending Dac-Man to handle streaming data.