CorrelX:
A Cloud-Based Software Correlator for Very Long Baseline Interferometry (VLBI)

User and Developer Guide [1.0]

Project leads: Victor Pankratius, Pedro Elosegui
Project developer: A.J. Vazquez Alvarez

MIT Haystack Observatory

2017-03-10

Contents

1 Introduction

CorrelX is a Very Long Baseline Interferometry (VLBI) correlator [1]. It is a multi-purpose offline correlator that can be adapted to different application scenarios in astronomy, geodesy, and signal processing. The main objectives of this project are scalability, flexibility, and simplicity. The software platform is based on Python and Apache Hadoop to enable quick prototyping for algorithm research and execution in cloud environments with real-world data sets. Users should be familiar with VLBI correlation [1], Linux bash scripting, and Python.

We would like to thank the MIT Haystack Observatory staff, in particular A. Rogers and R. Capallo for invaluable help during developing and testing, and J. Barrett, G. Crew, M. Gowanlock, J. Li, G. Rongier, C. Rude, and M. Titus for generous feedback. We acknowledge partial support from the NASA Space Geodesy Program and the NSF for supporting the Massachusetts Green High Performance Computing Center for our experiments in Holyoke, Massachusetts.

This documentation is organized as follows:

- *Design*: Describes the architecture, interfaces, and configuration of the system.
- *Operation*: Describes the installation and usage of the system, along with the provided tools.
- *Development*: Describes aspects related to profiling and code management.

Part I
Design

2 System Overview

The system is composed of:

- **CorrelX**: The core correlator generates visibilities and metadata using data files and configuration files defining parameters of the experiment.
- **Configuration converter**: Maintains compatibility with existing systems, such as DiFX. Converts DiFX configuration files into CorrelX configuration files.
- **Output converter**: Converts CorrelX output to DiFX output. Maintains compatibility with current post-correlation processing toolchains.

CorrelX maintains compatibility with the DiFX format and third-party tools that are currently used in the VLBI community, including:

- Experiment definition: VEX file [9] (DiFX tool vex2difx [10]).
- Delay model polynomials: NASA's CALC [11] (DiFX tool calcif2 [10]).
- Post-processing: MIT Haystack's HOPS [12] (DiFX tool difx2mark4 [10]).

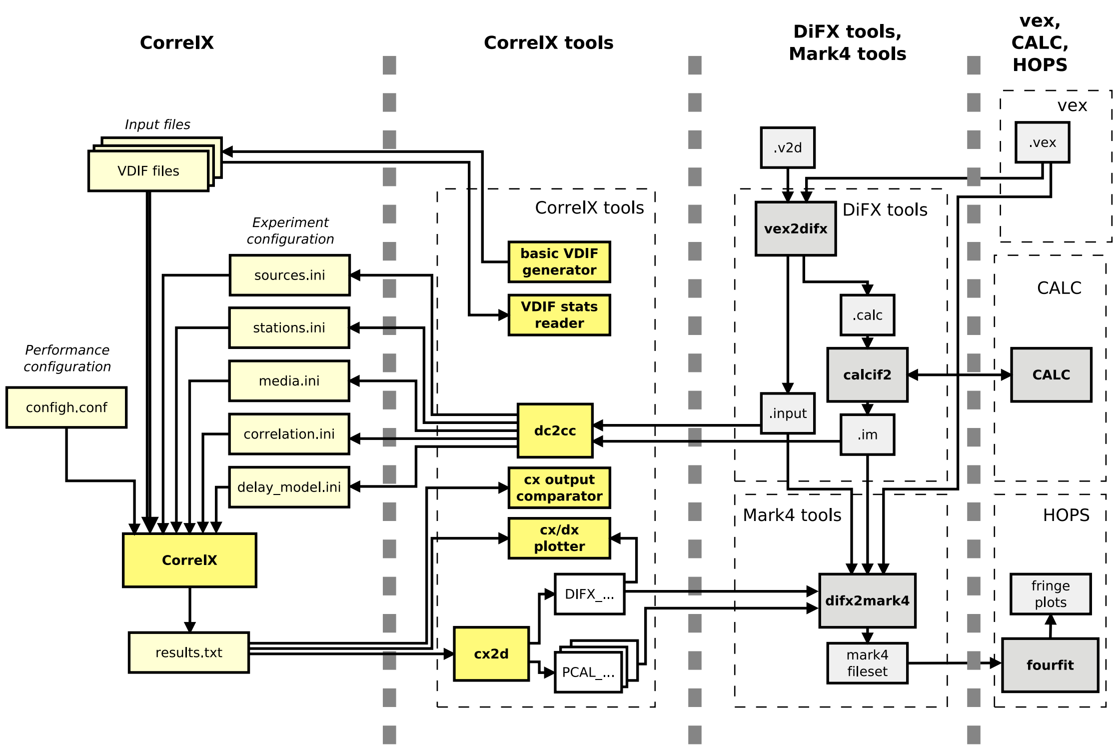


Figure 1: CorrelX integration into current VLBI data processing chains.

Figure 1 illustrates the integration of CorrelX with existing processing chains [13]. Yellow blocks denote CorrelX functionality; gray blocks depict other tools used in the VLBI community.

3 Architecture

CorrelX has a layered architecture to allow customization at different abstraction levels, as shown in Fig. 2. This architecture facilitates the separation of parallelization strategies from scientific data processing.

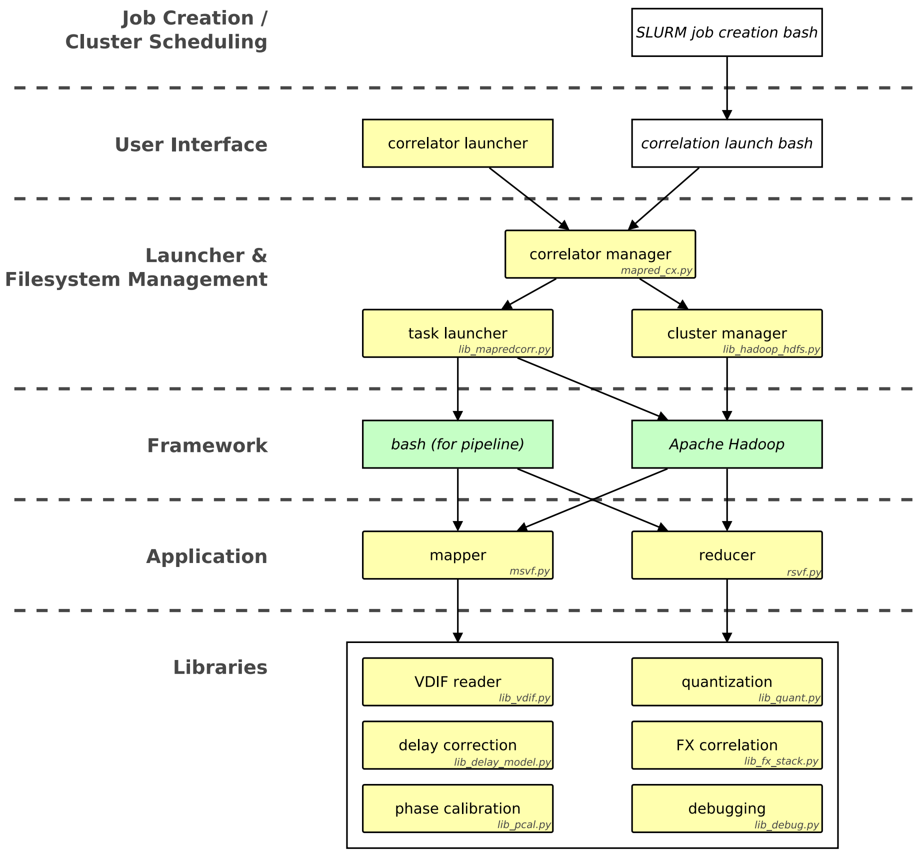


Figure 2: CorrelX architecture layers. Yellow blocks represent CorrelX sources; green blocks execution frameworks (e.g., for parallelization); white blocks user-developed scripts.

- **Job Creation/Cluster Scheduling**: Handles job configuration and requests in the cloud environment. Includes scripts required to set up correlation commands to be executed on cloud nodes.
- **User Interface**: Includes scripts to initialize allocated cloud nodes, set up login permissions, and launch the correlator's main script.
- **Launcher & Filesystem Management**:
  - *Correlator Manager*: Includes main CorrelX script and configuration libraries. Processes configuration files, experiment files, and computations on the delays to be applied during

correlation. Sets up the Hadoop cluster to run the correlation jobs. Includes tools to generate network statistics.

  - *Cluster Management*: Setup and deployment of the Hadoop cluster. This block includes the functions required to start and stop the Hadoop cluster, distribute configuration files among the nodes of the cluster, and prepare the media to be processed.
  - *Task Launcher*: Starts a correlation job. Includes functions to launch a correlation job in sequential and parallel modes and retrieve the file with the correlation results.

- **Framework**: Third-party software for pipelining or parallelization. Offers functionality for correlation without parallelization (i.e., without Hadoop) and cloud parallelization with Hadoop. The framework can be extended to include other ways of parallelization in the future.

- **Application**: Implements the mapper and reducer programs for Hadoop MapReduce. These programs modularize the key correlation processing.

- **Libraries**: Implements reusable functionality, such as computations associated with FX correlation, quantization, delay correction, phase calibration, debugging, profiling, and so on. CorrelX enables a plugin architecture for such functionality and easy extensions in the future. All new plugins should be created as part of these libraries.

4 Parallelization Framework

4.1 Overview of Apache Hadoop

Apache Hadoop [2] is an "*open-source software for reliable, scalable, distributed computing*" based on MapReduce [3]. It is maintained by the high-performance computing community, and it can scale to thousands of nodes [4].

There are a number of distributions [5] that provide additional functionality to simplify Hadoop's operation. CorrelX operates on Hadoop's base distribution to allow flexibility.

The version of Apache Hadoop used in this release is 2.7.3. Upgrading Hadoop is described in §8.2.4 along with the required modifications to CorrelX.

4.1.1 Network Architecture and Distributed Filesystem

The network architecture of a Hadoop cluster is described in [6]. From a high-level perspective, we differentiate between two kinds of nodes:

- **Master**: this node runs the main Hadoop entities and the main CorrelX script.
- **Workers**: these nodes run the MapReduce correlation tasks.

The Hadoop entities related to the work distribution include:

- *Resourcemanager*: interface for users launching MapReduce jobs; coordinates the MapReduce workflow.
- *Nodemanager*: launches containers to run mappers or reducers, one per container. The configuration of the node manager includes the available number of virtual CPU cores, available memory, number of CPU cores per container, etc.

The Hadoop entities regarding the Hadoop Distributed File System (HDFS) are:

- *Namenode*: keeps track of the location of the files in the distributed filesystem.
- *Datanode*: provides the storage capability in the distributed filesystem.

The HDFS filesystem stores data redundantly, broken down into blocks of a certain size. Both redundancy and block size are configurable. Typically the master node will run the namenode and the worker nodes run the datanodes. Hadoop provides a plugin to use Lustre as the distributed filesystem, which is the default choice for CorrelX (for simplicity and performance reasons). The configuration of other distributed filesystems for CorrelX is described in §8.2.3.

4.1.2 The MapReduce Approach

The MapReduce processing is performed in three stages:

- **Map stage**: all the worker nodes run mappers. The output from this stage are files containing (key,value) pairs or MapReduce records.
- **Shuffle and sort**: the output from the mappers is distributed to the required workers based on the key and sorting configuration.

- **Reduce stage**: all the worker nodes run reducers.

Each key is sub-divided into two sub-keys. One key is for partitioning so that different reducers can be launched for partitions (i.e., groups of pairs with the same partitioning key). The second key is required for sorting (so that the data order can be preserved and restored when processed by reducers). Example diagrams of these steps in the MapReduce correlation process are shown in [7].

Even though it is possible in theory to overlap the map and the sort executions, in the Hadoop implementation the workers do not start the reduce phase until the first two stages are completed. Therefore, performance tuning of respective parameters may be required:

Each worker node allows for a configuration of its capacity:

- Number of virtual CPU cores available.
- Memory available.

Each worker node runs the mappers or reducers in containers, constrained by configuration parameters, such as:

- Number of virtual CPU cores per container.
- Memory per container.

For more information, please consult [8].

4.2 Execution Modes—From Sequential to Parallel

4.2.1 Sequential Mode

MapReduce applications can be run without the parallelization framework, which is useful for debugging. In that case, overhead of the parallelization framework is avoided as well. CorrelX can be executed without Hadoop in an environment as simple as a single-core machine.

4.2.2 Pipeline Mode

Pipeline mode implements the MapReduce process with basic Linux commands. It involves pipelining the files into the mappers one by one, then concatenating the results, sorting them, and pipelining the resulting file into one reducer.

CorrelX provides the option to run in pipeline mode without revealing this execution mode to the user. This mode is intended to provide developers interested in operation or research with a quick setup of a development environment. Developers of the MapReduce logic can benefit from this mode for quick testing but should eventually test their changes in Hadoop, too, especially if these changes have implications on the sorting or key configuration.

4.2.3 Parallel Mode

Parallel mode fully uses Hadoop for correlation processing. It is the standard mode of operation.

4.3 Load Balancing

Load balancing controls how resources are distributed among map tasks and reduce tasks, e.g., how many mappers and reducers are launched.

4.3.1 Load Balancing in the Map Phase

CorrelX is configured to launch one mapper for each data block (in the distributed filesystem) to be processed. The block size can be configured as a multiplier of the size of the VDIF frames in the media files, driven by a parameter controlling the number of frames per block.

4.3.2 Load Balancing in the Reduce Phase

The reduce stage is generally the most computation-intensive part of the processing. Ideally, the number of reducers should be equal to the number of partitions (§4.1.2). A partition is a set of records (key,value) with the same key.

A Hadoop partitioner guarantees that all records that have the same partitioning key arrive at the same reducer and in the defined sorting order.

CorrelX incorporates a custom Hadoop partitioner that bypasses a hashing step of the default Hadoop partitioner, and launches one reducer per partition. It thus facilitates a balanced load at the reducer stage.

Additional details on the partitioner and the load balancing are discussed in §8.2.3 and §15.5, respectively.

5 Configuration

This section describes the configuration of the correlator and the definition of the experiment.

5.1 CorrelX Configuration

Configuration of the CorrelX correlator is done through a `.ini` file. It is also possible to override many of its parameters through the command-line interface for benchmarking or testing. See §9.1 and §9.2 for details.

Besides the CorrelX configuration, this file also groups all the parameters to be written into the Hadoop `.xml` configuration files. All of these parameters are written into the Hadoop configuration files and distributed to all nodes of the Hadoop cluster, thus avoiding the need to configure the nodes of the cluster individually.

5.1.1 CorrelX Configuration File

The CorrelX configuration file includes three variables that are replaced at runtime:

- `localpath` → to be replaced by the absolute path to the main script (same for all nodes). This is replaced by the CorrelX configuration libraries during initialization.
- `localuser` → to be replaced by the current user (same for all nodes). This is replaced by the CorrelX configuration libraries during initialization.
- `localhost` → to be replaced by the master host name (same for all nodes). This is replaced by the CorrelX configuration libraries during initialization.
- `${host.name}` → to be replaced by every host name in each of the nodes (different for every node). This is replaced by the Hadoop initialization scripts; that is, it is not written to the file, but the script creates an environment variable with the hostname at each of the workers (see details about `yarn-env.sh` in §8.2.2).

These variables are used to allow multiple instances of the correlator to be run and to avoid conflicts on the local configurations of the workers on a cluster with NFS. We provide an example at the end of this subsection. Variables defined as "bool" correspond to a string with either "yes" or "no". The strings `localpath`, `localuser`, and `localhost` are reserved, but they can be changed in `const_config.py`.

[General] section This section includes some basic configuration parameters: pipeline/parallel mode, type of deployment, etc.

Field	**Type**	**Value**
`Log file`	str	Path to the main log file.
`Run pipeline`	bool	See §4.2.2.
`Run hadoop`	bool	See §4.2.3.
`Sort output`	bool	Sort lines in the correlation results file. Deactivation may be useful for debugging so that the output records are displayed in the same order as produced by the reducer(s).

`Over SLURM`	bool	Enable if running on a cluster where the local filesystem synchronizes the home folder among all nodes (e.g., SLURM cluster with NFS), and disable if running on a cluster of independent machines (e.g., a test environment with virtual machines).
`Use NoHash partitioner`	bool	Enable to activate the custom NoHash partitioner, and disable to use Hadoop's default partitioner. Using the NoHash partitioner is recommended. See §8.2.3 for more information.
`Use Lustre plugin`	bool	Enable to activate the Seagate Lustre plugin, and disable to use the Hadoop's default HDFS filesystem. Using Lustre is recommended. See §8.2.3 for more information.
`Lustre prefix`	str	Path to Lustre base folder used by Lustre plugin. This is the base folder where the output files are placed.
`Lustre user folder`	str	Path to Lustre folder used in the MapReduce job. This is the base folder for placing the working directories for the nodes, so that mappers' and reducers' output will be placed in this folder.

Table 1: CorrelX configuration file, [General] section.

[Profiling] section This section includes parameters related to the profiling of the mapper and reducer.

Field	**Type**	**Value**
`Profile mapper (pipeline)`	bool	In effect only in pipeline mode. Enables the generation of pycallgraph plots (or cProfile text files) for all the mappers. Note that this will increase the overall execution times, so this option should be off unless needed.
`Profile reducer (pipeline)`	bool	In effect only in pipeline mode. Enables the generation of pycallgraph plots (or cProfile text files) for the reducer. Note that this will increase the overall execution times, should this option should be off unless needed.
`Use PyCallGraph`	bool	Select 0 to use cProfile profiler, and 1 to use PyCallGraph profiler.

Table 2: CorrelX configuration file, [Profiling] section.

[Benchmarking] section This section includes parameters related to the benchmarking of the application.

Field	Type	Value
`Avoid copying input files (lustre)`	bool	Enable to avoid copying files into the Lustre distributed filesystem if the folder has been created previously. May be useful if the same dataset needs to be processed many times.
`Delete output files (lustre)`	bool	Enable to delete the output results after correlation (will still keep metadata and a sample of the output data). May be useful in benchmarking scenarios where the output is not required.

Table 3: CorrelX configuration file, [Benchmarking] section.

[Files] section This section includes parameters related to the location of the scripts, configuration file templates, Hadoop installation path, etc.

Field	Type	Value
`Mapper`	str	Path to Python mapper script filename.
`Reducer`	str	Path to Python reducer script filename.
`Dependencies`	str	Comma-separated list with filenames for all libraries used by the mapper and the reducer.
`Mapper bash`	str	Filename of the bash script to write the full mapper command (all the mappers are called with the same arguments independently of the data that they will be processing).
`Reducer bash`	str	Filename of the bash script to write the full reducer command (all the reducers are called with the same arguments independently of the data that they will be processing).
`Job bash`	str	Filename of the bash script to write the full mapreduce job command, regardless of the operation mode (pipeline or parallel).
`Python executable`	str	Path to python executable that will be used for calling the mapper and the reducer (e.g., `python` or `/usr/bin/python`).
`Nodes`	str	Filename of the file to write the list of allocated nodes for the cluster deployment.
`Src directory`	str	Path to the folder containing the sources for the mapper, the reducer, and their dependencies (listed in `Dependencies`).

`App directory`	str	Path to the folder to place the folders associated with each master with the application, that is, mapper, reducer, and their caller scripts (`Mapper bash` and `Reducer bash`) and dependencies (listed in `Dependencies`).
`Conf directory`	str	Path to the configuration folder. All the configuration files defined in this section will be placed on the configuration folder, under a folder with the same name as the master node.
`Conf templates`	str	Path to the folder containing the templates with the Hadoop configuration files that will be used in the deployment. These templates will be copied into the configuration folder (of the master node), modified based on the parameters defined in the Hadoop parameters sections, and distributed to the worker nodes.
`Hadoop directory`	str	Path to the base folder with the Hadoop installation.
`Temporary data directory`	str	Path where the data to be processed will be split before being moved into the distributed filesystem.
`Temp directory`	str	Path to folder for Hadoop temporary files.
`Temp log`	str	Path to log for intermediate file to store interactions with Linux.
`Output directory`	str	Path for placing new folders with correlation results.
`Prefix for output`	str	Prefix for the result files.
`Username machines`	str	Username to be used to ssh into the machines during the cluster deployment.

Table 4: CorrelX configuration file, [Files] section.

[HDFS] section This section includes parameters related to the distributed filesystem.

Field	**Type**	**Value**
`Packets per HDFS block`	int	Number of VDIF frames per block. See §4.3.1.
`Input data directory`	str	Prefix of the path (before "Input directory suffix"), relative to the distributed filesystem base path, where the input files (blocks) will be placed.
`Input directory suffix`	str	Suffix of the path (after "Input data directory"), relative to the distributed filesystem base path, where the input files (blocks) will be placed.
`Checksum size`	int	Number of bytes for the checksum for each file. Used only in HDFS, not in Lustre.

Table 5: CorrelX configuration file, [HDFS] section.

[Experiment] section This section includes parameters related to the naming convention of the experiment definition files.

Field	Type	Value
`Experiment folder`	str	Path to the folder containing the experiment definition files. Typically overridden through the command-line interface. See §9.1.
`Stations file`	str	Name of the stations .ini file.
`Delay model file`	str	Name of the delay model .ini file.
`Delays file`	str	Name of the delays .ini file.
`Media file`	str	Name of the media .ini file.
`Correlation file`	str	Name of the correlation .ini file.
`Media sub-folder`	str	Name of the folder containing the symbolic links to the media (or the actual media files).
`Output sub-folder prefix`	str	Prefix name (the suffix will be a timestamp) of the folder to place the correlation results.

Table 6: CorrelX configuration file, [Experiment] section.

[Hadoop-master] [Hadoop-slave] sections All sections starting with "Hadoop-" (except the section [Hadoop-other]) correspond to configurations related to Hadoop files [18].

Field	Type	Value
`Configuration file`	str	Filename of the masters file used by Hadoop.
`Master is slave`	bool	Enable to make the master node run mapreduce containers. Should generally be disabled, unless working with a small dataset or running on a cluster with very few nodes.

Table 7: CorrelX configuration file, [Hadoop-master] section.

Field	Type	Value
`Configuration file`	str	Filename of the slaves file used by Hadoop.
`Max number of slaves`	int	Maximum number of slaves to deploy Hadoop. Unless only a subset of nodes in the nodes file is to be used, this field should be set to -1, so that all nodes in the nodes file are used.

Table 8: CorrelX configuration file, [Hadoop-slaves] section.

[Hadoop-yarn] [Hadoop-mapred] [Hadoop-core] [Hadoop-hdfs] The sections beginning with [Hadoop-*] have two kinds of parameters:

- An initial field with the file name (`Configuration file`, same as in [Hadoop-master]).
- As many fields as required that are simply Hadoop configuration parameters used in the files yarn-site.xml, mapred-site.xml, core-site.xml and hdfs-site.xml, respectively. Comprehensive lists of these parameters, their default values, and explanations can be found in [14], [15], [16], and [17], respectively. These parameters will overwrite the existing ones in the respective templates, if they exist, or added as new ones otherwise.

For example, in a cluster with a master `node025` and a slave `node026` in which the [Hadoop-yarn] section of the CorrelX configuration file is:

```
[Hadoop-yarn]
Configuration file:                       yarn-site.xml
yarn.resourcemanager.hostname:            localhost
yarn.nodemanager.aux-services:            mapreduce_shuffle
yarn.nodemanager.localizer.address:       ${host.name}:20016
```

and the content of the yarn-site.xml template located in the templates folder is:

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>10.0.2.4</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

During setup, the configuration file will be replaced by

```
[Hadoop-yarn]
Configuration file:                       yarn-site.xml
yarn.resourcemanager.hostname:            node026
yarn.nodemanager.aux-services:            mapreduce_shuffle
yarn.nodemanager.localizer.address:       ${host.name}:20016
```

Then the new yarn-site.xml file will be:

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>node065</value>
```

```
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.localizer.address</name>
    <value>${host.name}:20016</value>
  </property>
```

The workers will initialize the Hadoop entities based on these files, so that each worker will have its associated host name.

5.1.2 Hadoop Configuration Files

The Hadoop nodes are configured by four main configuration files, mentioned in the previous section:

- `yarn-site.xml`: configuration of the of the master, constraints on jobs [14].
- `mapred-site.xml`: configuration of the workers [15].
- `core-site.xml`: configuration of the temporary storage and distributed fielsystem [16].
- `hdfs-site.xml`: configuration of the distributed filesystem [17].

These .xml files follow a common structure, with as many properties as needed:

```
<configuration>
  <property>
    <name>parameter_i</name>
    <value>value_i</value>
  </property>
  ...
</configuration>
```

These files are filled in the required format by CorrelX during configuration. For more details on the specific library, please see §7.1. For more information on the configuration files themselves, see [18].

5.2 Experiment Definition

Each experiment requires a folder containing the following structure:

- `sources.ini`
- `station.ini`
- `correlation.ini`
- `delay_model.ini`

- media.ini
- media
  - media_0.vdif
  - media_1.vdif
  - [..]
  - media_n.vdif

Note that these names can be changed in the CorrelX configuration file (`[Experiment]` section).

The experiment definition files were designed to be human-readable and as simple as possible. No specific ordering of the fields within a section is required. Due to the interrelation between these files, newly defined parameters will be accompanied by an id (a unique integer).

5.2.1 Sources File

The file `sources.ini` provides one section for each source, with each section being the name of the source. Each section includes the field `id`, a unique integer for each section. For example:

```
[3C454.3]
id = 0
```

5.2.2 Stations File

The file `stations.ini` provides one section for each station, with each section being the two-character name of the station. Each section includes three fields: `id` is a unique integer for each section, `clock_ref` is the MJD epoch for the station clock, and `clock_poly_us` has the zero and first-order coefficients of the polynomial for the station clock separated by a colon.

For all polynomials, coefficient x is in [μs/s^x]; i.e., coefficient zero (leftmost) is in microseconds and coefficient one in microseconds per second.

For example, for two stations BR and LA:

```
[BR]
id = 0
clock_ref = 57059.4977777778
clock_poly_us = -5.617731720000001e-01:-7.681715779999999e-08

[LA]
id = 1
clock_ref = 56933.4977777778
clock_poly_us = -6.392623500000000e-01:-4.814717640000000e-08
```

5.2.3 Correlation File

The file `correlation.ini` first provides one section (`elements`) to configure the number of stations that will take part in the experiment (so that only stations with an id lower than this value will participate), and whether autocorrelations and cross-polarization correlations will be computed;

another section (`computation`) to define the number of coefficients in the visibilities, the duration of the accumulation period in seconds, the type of window to be used during correlation, and whether phase calibration tones will be extracted; and a section (`times`) to define the start and duration of the scan. For example:

```
[elements]
stations = 2
autocorr_station = yes
cross_polarization = yes

[computation]
FFT = 40960
accumulation = 0.32
window = square
phase_calibration = no

[times]
mjd_start = 57235
seconds_start = 30000
seconds_duration = 300
```

5.2.4 Delay Model File

The file `delay_model.ini` first provides one section with the delay polynomials for a given time period for each possible combination of source–station formatted as (`[MJD-START-END-soS-stT]`, where `MJD` is the MJD for the polynomial, `START` and `END` are the start and end seconds within that MJD for which the polynomial is valid, `S` is the source id from the sources.ini file, and `T` is the station id from the stations.ini file. The parameter `delay_us` is a vector with the coefficients of the polynomial modeling the total delay (station–Earth center) for that pair station–source, where the leftmost element is the zero-order coefficient, the next element is the first-order coefficient, etc. The parameters `dry_us` and `wet_us` follow the same format and model the components of the total delay corresponding to the "dry and wet atmosphere" respectively.

For all polynomials, the coefficient x is in [μs/s^x], i.e., coefficient zero (leftmost) is in microseconds, coefficient one in microseconds per second, coefficient two in microseconds per square second, and so on. For example:

```
```

5.2.5 Media File

The file `media.ini` contains multiple sections that can be divided into three groups:

- *Definitions* for the channels and polarizations used later in the description of the media. Includes the following sections:
  - `channels`, with a list of pairs with channel name and a unique id for this channel starting at 0;

  - `frequencies`, with the same list of channels and their assignment of lower edge frequencies for each band [Hz];
  - `bandwidths`, with again the same list of channels and their associated bandwidth [Hz];
  - `polarizations`, with a list of characters identifying the polarizations and their association to unique ids.

  Additionally, it is possible to configure zoom bands to be computed after correlation (see section §9.7) through the sections:

  - `zoom_freq` provides a list of zoom band identifiers and their associated lower edge frequencies [Hz];
  - `zoom_bw` provides the same list but with the bandwidths of each zoom band [Hz].

- *Files* with a section `files` and a parameter `list` with a comma-separated list with the symbolic links to the media files, all of them located in the folder `media` described at the beginning of this subsection.

- *File descriptions*, with one section for each of the files in the previous comma-separated list (identified by its associated header), where each section includes a field with the station name (from stations.ini), a colon-separated list of channels (as defined in the corresponding definitions section), a colon-separated list of polarizations associated with each of these channels, a forced frame length (leave as 0 to let CorrelX read this frame length from the frames), the sampling frequency, the format (configured in two fields, currently always VDIF and custom), the frequency and separation of the phase calibration tones (zero if none), and a colon-separated list with the sideband associated with each band (`L`, lower; `U`, upper).

The colon-separated lists included in the file description map directly to the VDIF frames contained in that file. Based on the normal usage scenarios considered in the VDIF specification (either "a single Data Thread carrying multi-channel Data Frames" or "multiple single-channel Data Threads"), CorrelX currently considers only these two cases and automatically detects the case of operation based on the frames read from the file, so for multi-thread VDIF each element of these vectors represents one thread, and for multi-channel VDIF each element represents one channel. The channel/thread ids corresponds to the keys of the `channels`, so it is recommended to define these channels, typically as `CH0 = 0`, `CH1 = 1`, and so on. This is the convention taken by the configuration converter (§9.6), but this convention is not enforced. For example:

```
[channels]
CH0 = 0
CH1 = 1

[frequencies]
CH0 = 86140.0e6
CH1 = 86268.0e6

[bandwidths]
CH0 = 128.0e6
CH1 = 128.0e6

[polarizations]
L = 0
R = 1
```

```
[zoom_freq]
ZF0 = 86330.290625e6
ZF1 = 86271.696875e6
ZF2 = 86213.103125e6
ZF3 = 86154.509375e6

[zoom_bw]
ZB0 = 51.2e6
ZB1 = 51.2e6
ZB2 = 51.2e6
ZB3 = 51.2e6

[zoom_post]
zoom_freq = ZF0:ZF1:ZF2:ZF3
zoom_bw = ZB0:ZB1:ZB2:ZB3

[files]
list = BM434A-BR-No0017.vdif,BM434A-LA-No0017.vdif

[BM434A-BR-No0017.vdif]
station = BR
channels = CH0:CH0:CH1:CH1
polarizations = R:L:R:L
framebytes = 0
f_sample = 256.0e6
format = VDIF
version = custom
f_pcal = 0.0e6
o_pcal = 0e6
frequencies = CH0:CH0:CH1:CH1
sidebands = U:U:U:U

[BM434A-LA-No0017.vdif]
station = LA
channels = CH0:CH0:CH1:CH1
polarizations = R:L:R:L
framebytes = 0
f_sample = 256.0e6
format = VDIF
version = custom
f_pcal = 0.0e6
o_pcal = 0e6
frequencies = CH0:CH0:CH1:CH1
sidebands = U:U:U:U
```

This configuration file corresponds to the VDIF file in which statistics are displayed in the "frame-by-frame information display" described in §9.8.1. Since this is a threaded VDIF file, the vectors `channels`, `polarizations`, and `sidebands` are indexed by the thread id of the frame; if it were a single-thread VDIF with multiple bands, these vectors would be indexed by the channel id. Thus, frames for thread 0 correspond to channel and polarization `CH0 R`, respectively, thread 1 to `CH0 L`, thread 2 to `CH1 R` and thread 3 to `CH1 L`. Note that the VDIF reader supports single-thread multi-channel and multi-thread single-channel files seamlessly. Multi-thread multi-band files are not currently supported.

The ids (the values) in the sections `[channels]` and `[polarizations]` of the media configuration file are not related to the media file, but are simply ids used internally in CorrelX. The parameters `CH0, CH1, ...` could be changed, for example, to `bandA,bandB,...` as long they are used consistently in the media file.

6 Interfaces

In this section, we define interfaces and indicate the library and function where each interface is located in the source code, as well as associated tools for input (configuration) and output conversion. The source code includes extensive additional documentation for each interface.

6.1 External Interfaces

This section lists the input and output external interfaces of the CorrelX system (the three main blocks presented in §2), classified into data and control in Tables 9 and 10, respectively.

Block	**Input Interface**	**Output Interface**
CorrelX	Media files [19]	Visibilties (see Table 11, reduce out)
dc2cc	*.input* [20] *.im* [21]	Experiment definition files §5.2
cx2d	CorrelX output (see Table 11, reduce out)	*DIFX_** [22] *PCAL_** [10] (§6.8.2 Pulse cal data)

Table 9: CorrelX data external interfaces.

Block	**Input Interface**	**Output Interface**
CorrelX	Configuration file §5.1.1 Experiment definition files §5.2 User interface (Table 12, corr. manager)	CorrelX logs §13.5 Hadoop logs §13.5
dc2cc	User interface (see Table 13, reduce out)	
cx2d	User interface (see Table 13, cx2d)	

Table 10: CorrelX control external interfaces.

Note that for the configuration converter block, the configuration files are considered to be part of the data flow, even though they are technically part of the control chain, because in this case they are treated as data.

6.2 Internal Interfaces

Here we again differentiate between data and control interfaces, although the data flow includes metadata that could be considered part of the control chain. We consider it to be metadata associated with this data because it will be ignored by the MapReduce framework.

For the data interfaces (Table 11), we consider the MapReduce chain or, equivalently, the data flow in the components of the **application** layer in the architecture (Fig. 2):

Media files → Map → Reduce → *Visibilities*

From → to	Common	Output (from)	Input (to)
Media files → map			`msvf.read_frame()` `lib_vdif.read_vdif_frame()`
Map → reduce	`const_mapred`	`msvf.get_pair_str()`	`rsvf.split_input_line()` `rsvf.extract_params_split()`
Reduce → *visibilities*		`rsvf.get_lines_out_for_all()` `rsvf.get_lines_stats()`	

Table 11: CorrelX data internal interfaces.

For the control interfaces (Table 12), we consider all the components of the two layers: **application** and **launcher & filesystem management**.

Block	Constants	Interface	Comments

correlation manager	
	`const_config`
		`mapred_cx.main.cparser`
		`lib_config.get_configuration()`
		`lib_ini_exper.process_ini_files()`
			User arguments
			Configuration parsing
			Experiment parsing

task launcher		`lib_mapredcorr.pipeline_app()` `lib_mapredcorr.run_mapreduce_sh()`	Pipeline mode task Parallel mode task

cluster manager	`const_hadoop`
	`const_hadoop`
	`const_hadoop`
		`lib_hadoop_hdfs.cluster_start()`
		`lib_hadoop_hdfs.cluster_stop()`
		`lib_hadoop_hdfs.copy_files_to_hdfs()`
			Start cluster
			Stop cluster
			Distribute files

map	`const_mapred`	`lib_mapredcorr.get_mapper_params_str()`	Mapper control
reduce	`const_mapred`	`lib_mapredcorr.get_reducer_params_str()`	Reducer control

Table 12: CorrelX control internal interfaces.

The control interfaces for the conversion tools are listed in Table 13.

Block	Constants	Interface	Comments
dc2cc	`cx2d_lib`	`convert_im_cx`	Configuration conversion
cx2d	`cx2d_lib` `cx2d_lib`	`process_zoom` `convert_cxd`	Zoom band processing Output conversion

Table 13: CorrelX conversion tools control internal interfaces.

7 Functional Description

In this section, we provide a high-level description of the processing at each layer of the architecture. For call-graphs generated by the CorrelX profiling modules, see §15.

7.1 Job Launch and Cluster Deployment

Correlation manager The control flow of the correlation manager (`mapred_cx`) is as follows:

- Process arguments.
- Process correlator configuration.
- Distribute cluster configuration files.
- Process experiment definition.
- Launch pipeline mode task (if required).
- Parallel mode task (if required):
  - Prepare and distribute Hadoop configuration files.
  - Prepare map and reduce launcher scripts.
  - Distribute additional configuration files and application files.
  - Initialize Hadoop cluster.
  - Move media files into distributed filesystem.
  - Launch correlation (Hadoop job).
  - Shut down Hadoop cluster.
  - Report statistics.

7.2 Application

Mapper The control flow of the mapper (`msvf`) is as follows:

- Process arguments.
- Process experiment definition files.
- Loop for every read VDIF frame:
  - Determine accumulation period for this frame.
  - Compute initial delay.
  - Align first sample of the first frame (only for first frame).
  - Corner-turning.
  - Apply offsets to access samples in the frame (only for first frame), and determine location of samples in the whole stream.
  - Loop for every band in the frame:
    - Encode samples.
    - Prepare output line.
    - Write output line.

Reducer The control flow of the reducer (`rsvf`) is as follows:

- Process arguments.
- Loop for every read line:
  - Separate line into key and value.
  - Separate value into metadata and data.
  - Decode and de-quantize samples.
  - Process metadata.
  - If new accumulation period:
    * Compute FX correlation of stored samples.
    * Compute phase calibration of stored samples.
    * Prepare and write output lines with visibilities and phase calibration tones for this accumulation period.
    * Prepare and write output lines with statistics.
    * Restart data structures.
    * Store new samples (reducer structures).
  - Otherwise, if same accumulation period:
    * If first round of data (no more data for this station-polarization), store data (reducer structures) and continue.
    * Otherwise, if existing data, append new samples (into FX structures).

Note that there are two kinds of data structures in the reducer: the first one is an intermediate buffer for storing the samples as they come from the read lines; the second one includes additional processing for FX correlation (such as integer sample realignment, etc.).

We provide a high-level description of the control flow for mapper and reducer in Fig. 3. Note that this diagram is highly simplified, providing just an overview of the MapReduce processing.

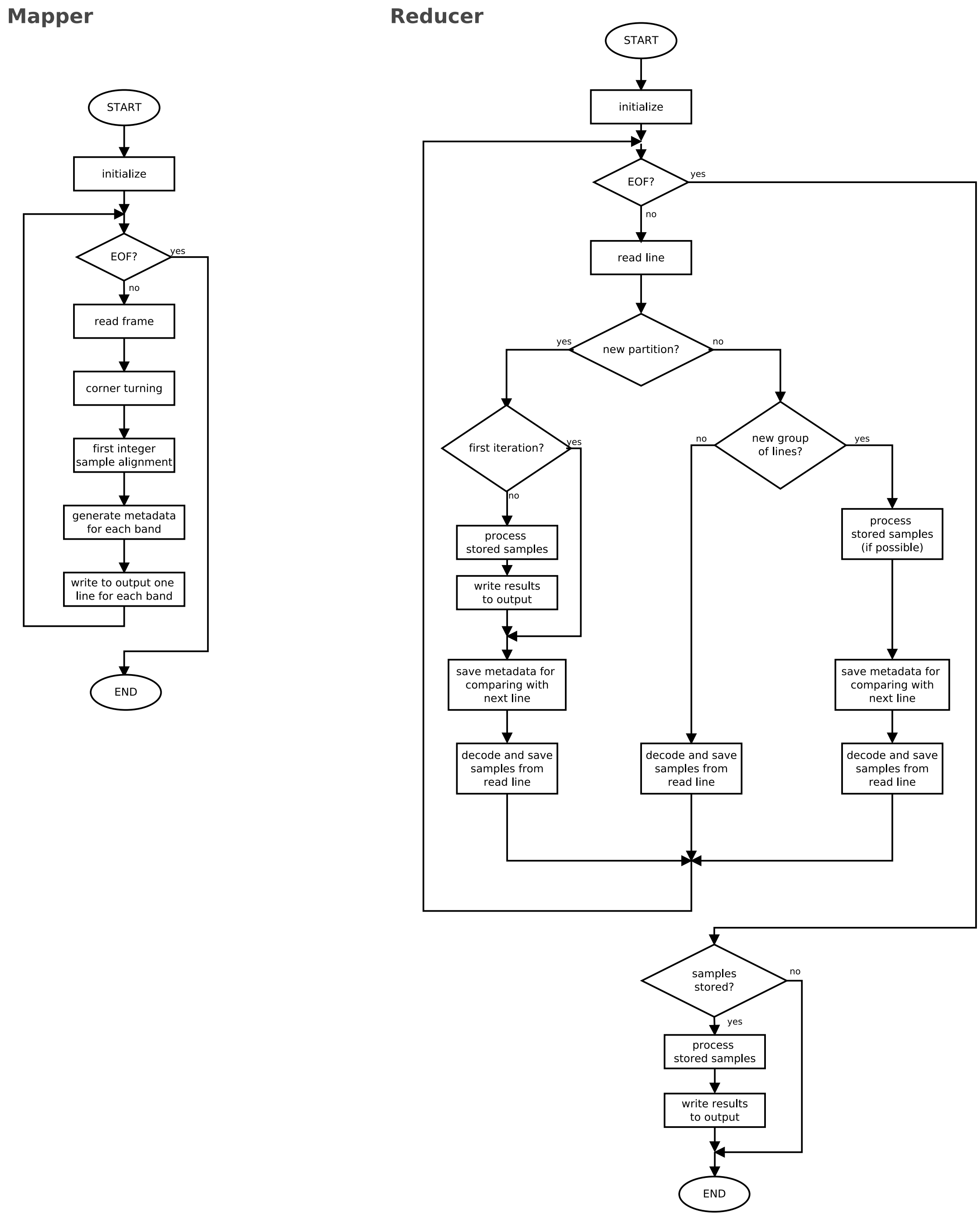


Figure 3: High-level description of the control flow for the mapper and the reducer.

7.3 Libraries

In this section, we describe the main functionality of each of the modules in the **libraries** layer of the architecture and of the main functions in each library. The sources are extensively documented; these sections are intended as a guide to direct the reader to the appropriate source files and functions.

We describe the main functionality of the libraries and whether they are used by the mapper (M) and/or the reducer (R) in Table 14.

Library	Constants	M	R	Description
`lib_ini_files`	`const_ini_files`	x		Functions to access data structures containing information on the experiment definition files.
`lib_vdif`		x		Functions to read VDIF frames.
`lib_acc_comp`		x		Computations related to accumulation periods, relative positions of frames in the acc. period, etc.
`lib_debug`	`const_debug`	x		Functions for displaying tabulated debugging information for mapper and reducer. See §13.1 and §13.2 for more details.
`lib_delay_model`		x	x	Computations related to delays.
`lib_pcal`		x	x	Phase calibration computations.
`lib_fx_stack`			x	Computations related to samples stacking, integer sample re-alignment and FX correlation.
`lib_quant`	`const_quant`		x	De-quantization functions.

Table 14: CorrelX libraries.

7.4 Tools

dc2cc The configuration converter is coded into a user interface script (`convert_im_cx`), with all the functionality in the conversion toolkit library (`cx2d_lib`). The control flow of the configuration converter is as follows:

- Parse *.im* file and generate `delay_model.ini`.
- Parse *.input* file and generate `stations.ini`.
- Parse *.im* file and generate `sources.ini`.
- Parse *.input* file and generate `correlation.ini`.
- Parse *.input* file and generate `media.ini`.

cx2d The output converter is coded into `cx2d_lib.convert_cx2dpc()`, with an interfacing script `convert_cx2d`. Its associated control flow is as follows:

- Convert visibilities:
  - Process experiment definition files.
  - Read all visibilities from CorrelX output file.
  - For each set of visibilities:
    - Create binary headers.
    - Convert the visibilities to the required binary format.
  - Sort binary records as required.
  - Write binary records into DiFX output file.
- Convert phase calibration tones:
  - Process experiment definition files.
  - Read all phase calibration results from CorrelX output file.
    - Calculate phase calibration tone positions.
    - Extract tones and generate "pulse calibration" records.

Additionally, the zoom band processing (called prior to the previous script) provides its interface in `process_zoom` and its functionality in `cx2d_lib.process_zoom_band()`. The associated control flow is as follows:

- Process experiment definition files.
- Generate list of zoom bands, along with their associated metadata.
- For every line containing visibilities from the CorrelX output file:
  - Apply zoom band.
  - Average channels if required.
  - Modify band identifier in the header.
  - Append to new CorrelX output file.

Part II
Operation

8 Installation

There are three types of deployments (summarized in Table 15):

- **Minimal deployment**: pipeline mode only. This deployment does not require Hadoop or any additional module—only a machine with Linux and Python (Python2, NumPy, SciPy, and PyCallGraph).
- **Cluster deployment**: capability to both pipeline and parallel mode, generally applicable to a set of independent machines (e.g., a test environment composed of virtual machines). This deployment also requires Apache Hadoop (and optionally a custom partitioner module for better load balancing).
- **Cloud deployment**: same capabilities as the local cluster but higher performance, generally applicable to a cluster with a shared filesystem and a job scheduler (e.g., SLURM), and typically a Lustre filesystem. This deployment requires Hadoop, the custom partitioner module, and the Seagate Hadoop Lustre plugin.

Deployment Type	**Linux**	**Python**	**Hadoop**	**Custom Partitioner**	**Lustre Plugin**	**Installation**
Minimal	x	x				§8.1
Cluster	x	x	x	(x)	(x)	§8.1, §8.2
Cloud	x	x	x	x	x	§8.1, §8.2

Table 15: CorrelX requirements.

The versions of the software used so far are listed in Table 16.

Software	**Version**
Linux	Debian 8.6 Ubuntu Server 14.04.2 LTS Ubuntu 14.04.3 LTS Red Hat Enterprise Linux Server 6.6 CentOS release 6.8 (Final)
Hadoop	Apache Hadoop 2.7.3
Python2	Python 2.7.6, 2.7.8
NumPy	1.8.2, 1.9.2
SciPy	0.13.3, 0.15.1
PyCallGraph	1.0.1
Lustre	2.5.3
Seagate Lustre plugin	0.9.1

Table 16: Known compatibility.

8.1 CorrelX Minimal Deployment

We recommend that you run CorrelX as a user with no `sudo` privileges. The following example steps were tested on Ubuntu, so exact commands may differ for other Linux distributions:

```
sudo adduser cxuser
```

1. Install dependencies:

```
sudo apt-get install python python-numpy python-scipy python-pycallgraph
```

2. Download the latest version of CorrelX from [23] and extract it to the Hadoop installation folder:

```
ssh cxuser@localhost
tar -xvf correlx-alpha0.tar
```

3. Test dependencies. The following command should return no errors:

```
cd correlx/src
python -c "import msvf,rsvf,mapred_cx,cx2d_lib"
cd ~
```

The basic deployment is ready; see §9.2 for operation.

8.2 CorrelX Cluster and Cloud Deployment

A cluster or cloud installation requires a basic deployment on one of the nodes, and Hadoop and the applicable additional plugins installed on all nodes.

8.2.1 CorrelX

Master node Perform a basic deployment of CorrelX as in §8.1, steps 1–3.

Other nodes Install dependencies (Python and libraries) as in §8.1, step 1.

8.2.2 Hadoop 2.7.3

This procedure is divided in two parts: the first part applies to all the nodes of the cluster, and the second part to only the master node (where CorrelX will be launched). If the cluster has a shared filesystem, then the first part can be done at one of the nodes.

All nodes Again, we recommend that you run CorrelX with a user with no `sudo` privileges, and this user should be present on all the machines. For example:

```
sudo adduser cxuser
```

1. Hadoop requires Java installed; see [24] for a list of supported Java versions. Install the required tools for scp/ssh on multiple nodes. Note that this is required only if running on a local cluster (Table 1 `Over SLURM: no`), but not for a cloud cluster (Table 1 `Over SLURM: yes`); this is required only on a cluster without NFS (for example a test environment with multiple virtual machines, or a set of independent machines). For example:

```
sudo apt-get install openjdk-8-jdk pdsh     # openjdk-7-jdk in Debian
```

2. Switch user to `cxuser`, download the latest version of Hadoop from [25] (2.7.3 binary [26]), and extract it to the Hadoop installation folder. For example:

```
ssh cxuser@localhost
wget <<<hadoop-2.7.3 link>>>
mkdir hadoop
tar -xvzf hadoop-2.7.3.tar.gz -C hadoop
```

3. Run the provided configuration script:

```
./correlx/sh/configure_hadoop_cx.sh
```

The script will search and ask for confirmation on the CorrelX, Hadoop, and Java paths found and will perform the required configurations on the Hadoop scripts:

```
Hadoop for CorrelX: configuration

CorrelX folder found: /home/cxuser/correlx
Enter to confirm, new path to override: [Enter]
Using Correlx dir: /home/cxuser/correlx

Hadoop folder found: /home/cxuser/hadoop/hadoop-2.7.3
Enter to confirm, new path to override: [Enter]
Using Hadoop dir: /home/cxuser/hadoop/hadoop-2.7.3

Java folder found: /usr/lib/jvm/java-8-openjdk-amd64
Enter to confirm, new path to override: [Enter]
Using Java dir: /usr/lib/jvm/java-8-openjdk-amd64

Configuring environment scripts...
Configuring initialization scripts...
Copying Hadoop configuration files to templates folder:
    correlx/templates/hadoop_config_files
Done
```

For Hadoop versions after 2.7.3, please see §8.2.4, and also revise the results of the configuration script.

4. This step is required in clusters of two nodes or more if the nodes cannot resolve the names of other nodes. In a cluster with a single node, this step may be skipped. That is, it is necessary for the master node to be able to resolve the host names of the worker nodes, and for the worker nodes to resolve the name of the master, so the following step may be necessary.

Replace the first part of the hosts file with all the addresses and names of the nodes in the cluster. For the worker nodes, add the IP and host name of the master node.

```
  sudo vi /etc/hosts
    127.0.0.1       localhost
    <<ip0>>         <<master_hostname>>
    <<ip1>>         <<worker0_hostname>>
    <<ip2>>         <<worker1_hostname>>
    [...]
```

The following example is for two virtual machines:

```
  sudo vi /etc/hosts
    127.0.0.1       localhost
    10.0.2.9        ubuntutd
    10.0.2.10       ubuntutd2
```

Now the nodes should resolve names. Testing with `ping`, from the master:

```
  ping ubuntutd2
```

And from the workers:

```
  ping ubuntutd
```

Master node The last steps are basically setting up ssh access from the master to the workers.

5. Set up password-less ssh connections from master to all worker nodes. This is required to allow the master node to launch Hadoop entities on the worker nodes, and to distribute configuration and application files. This step is required also in a cluster a single node.

In the master node, go to the home folder and generate the key with no password [29]:

```
cd ~
ssh-keygen -t rsa
```

The next step is copying this key into all nodes. For the master node:

```
ssh cxuser@‘hostname‘ mkdir -p .ssh
cat .ssh/id_rsa.pub | ssh cxuser@‘hostname‘ ’cat >> .ssh/authorized_keys’
```

Test passwordless ssh. The following command should not ask for the password again:

```
ssh cxuser@`hostname` hostname
ssh cxuser@0.0.0.0 hostname
```

For the rest of the nodes (where `workernode` is the host name of the worker):

```
ssh cxuser@workernode mkdir -p .ssh
cat .ssh/id_rsa.pub | ssh cxuser@workernode 'cat >> .ssh/authorized_keys'
```

Test passwordless ssh as shown previously:

```
ssh cxuser@workernode hostname
```

Alternatively, a script is provided for NFS/SLURM environments (see §10.1 for details). To avoid manual confirmation of the authenticity of each worker in the first connection, ssh key checking may be disabled. For example:

```
vi /etc/ssh/ssh_config
   Host *
   StrictHostKeyChecking no
```

The cluster deployment is ready; see the following subsection for additional plugins or §9.3 for operation. For more information and tutorials about setting up a Hadoop cluster, refer to [30] (single node cluster) and [31] (multiple node cluster).

8.2.3 Extra Modules

In this subsection, we describe how to build and configure the custom partitioner and the Lustre plugins from source.

Custom partitioner As introduced in §4.3.2, the custom partitioner bypasses the hashing of the key. The default partitioner is generally useful in other Hadoop applications (e.g., text processing, where there may be many different keys), but is problematic with CorrelX (where the number of keys is known a priori), and it is better to have full control on the number of reducers and the load per reducer, as opposed to randomizing this choice. In this section, we describe how to modify and compile this partitioner from the Hadoop sources.

1. Create folders:

```
mkdir -p correlx/partitioner/mapreduce
mkdir -p correlx/partitioner/mapred
```

2. Save the contents of the source [33] into:

```
correlx/partitioner/mapreduce/KeyFieldBasedPartitionerNH.java
```

3. Edit text and add:

```
import java.util.Arrays;
import java.nio.charset.StandardCharsets;
```

Replace:

```
KeyFieldBasedPartitioner
```

with:

```
KeyFieldBasedPartitionerNH
```

And replace:

```
currentHash = 31*currentHash + b[i];
```

with:

```
currentHash = 10*currentHash + (b[i]-48);
```

4. Save the contents of the source [34] into:

```
correlx/partitioner/mapred/KeyFieldBasedPartitionerNH.java
```

5. Edit text and replace:

```
KeyFieldBasedPartitioner
```

with:

```
KeyFieldBasedPartitionerNH
```

6. Compile mapreduce-folder file and move into Hadoop installation folder (it will be required by mapred-folder file):

```
cd correlx/partitioner/mapreduce
javac -cp "/home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/common/*:/home/cxuser/
  hadoop/hadoop-2.7.3/share/hadoop/common/lib/*:/home/cxuser/hadoop/hadoop-2.7.3/
  share/hadoop/mapreduce/*" KeyFieldBasedPartitionerNH.java
mkdir -p org/apache/hadoop/mapreduce/lib/partition
cp KeyFieldBasedPartitionerNH.class org/apache/hadoop/mapreduce/lib/partition
jar cvf partitionernohashlib.jar org
cp partitionernohashlib.jar ~/hadoop/hadoop-2.7.3/share/hadoop/mapreduce/
```

In a system with no shared filesystem—for example, for nodes `ubuntutd` and ubuntutd2:

```
pdcp -d -R ssh -l cxuser -w ubuntutd,ubuntutd2 partitionernohashlib.jar
   /home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/mapreduce
```

7. Repeat the process for the mapred-folder file (note the changes in the filename and paths):

```
cd ../mapred
javac -cp "/home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/common/*:/home/cxuser/
  hadoop/hadoop-2.7.3/share/hadoop/common/lib/*:/home/cxuser/hadoop/hadoop-2.7.3/
  share/hadoop/mapreduce/*" KeyFieldBasedPartitionerNH.java
mkdir -p org/apache/hadoop/mapred/lib
cp KeyFieldBasedPartitionerNH.class org/apache/hadoop/mapred/lib/
jar cvf partitionernohash.jar org
cp partitionernohash.jar ~/hadoop/hadoop-2.7.3/share/hadoop/mapreduce/
```

In a system with no shared filesystem—for example, for nodes `ubuntutd` and `ubuntutd2`:

```
pdcp -d -R ssh -l cxuser -w ubuntutd,ubuntutd2 partitionernohash.jar
   /home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/mapreduce
```

8. Enable the new partitioner in the configuration file (Table 1):

```
vi ./correlx/conf/correlx.ini
  [General]
  Use NoHash partitioner: yes
```

The new partitioner is ready. The next submitted job should show the following extract in the log:

```
... -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitionerNH
```

Lustre filesystem plugin This plugin allows Hadoop to use Lustre as the distributed filesystem, avoiding the overhead of the HDFS filesystem and increasing performance [35]. The Seagate [36] Lustre Hadoop plugin is used, for which we summarize the installation instructions provided in detail in [36].

1. Install Maven:

```
sudo apt-get install git maven
```

2. Clone the Lustre plugin repository:

```
mkdir correlx/lustre
cd correlx/lustre
git clone https://github.com/Seagate/lustrefs
```

3. Build jar:

```
cd lustrefs
mvn package
```

4. Copy the plugin into the Hadoop installation folder into all nodes in the cluster. In a system with a shared filesystem:

```
cp target/lustrefs-hadoop-0.9.1.jar \
  /home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/common/lib/
```

In a system with no shared filesystem—for example, for nodes **ubuntutd** and ubuntutd2:

```
pdcp -d -R ssh -l cxuser -w ubuntutd,ubuntutd2 target/lustrefs-hadoop-0.9.1.jar
   /home/cxuser/hadoop/hadoop-2.7.3/share/hadoop/common/lib/
```

5. Configure the Lustre paths in the configuration file (Table 1):

```
vi ./correlx/conf/correlx.ini
  [General]
  Use Lustre plugin:       yes
  Lustre prefix:           <path to lustre folder to place input data folders>
  Lustre user folder:      <path to lustre folder to place output data folders>

  [Files]
  Temporary data directory: <path for splitting input files>
  Temp directory:           <path to lustre folder with filesystem descriptors>

  [HDFS]
  Input data directory:    <prefix of the path relative to ‘‘Lustre prefix’’
                              for input data>
  Input directory suffix:  <suffix for the previous path, specific per
              experiment, typically specified through the command line>

  [Hadoop-yarn]
  yarn.app.mapreduce.am.staging-dir:   <path relative to ‘‘Lustre prefix’’ for
                                          staging>
  yarn.nodemanager.local-dirs:         <comma separated list of folders for
                                          worker nodes>
```

For the nodemanager local directories, it is recommended to include the hostname keyword in folders in Lustre to keep one folder per node, and the machine's temporary folder can be added as backup. For example:

```
yarn.nodemanager.local-dirs:  /tmp/cxuser,/lustre/cxuser/tmp-worker-${host.name}
```

This is applicable also to other shared filesystems, so this practice is encouraged to minimize potential issues. In this example, the Lustre filesystem is mounted in `/lustre/cxuser`:

```
vi ./correlx/conf/correlx.ini
  [General]
  Use Lustre plugin:           yes
  Lustre prefix:               /lustre/cxuser/hadoop
  Lustre user folder:          /lustre/cxuser/hadoop/out

  [Files]
  Temporary data directory:    /lustre/cxuser/tmp-in
  Temp directory:              /lustre/cxuser/hadoop/tmp/fs-info-localhost

  [HDFS]
  Input data directory:        /input
  Input directory suffix:      exp01

  [Hadoop-yarn]
  yarn.app.mapreduce.am.staging-dir: /master-st/staging-localhost
  yarn.nodemanager.local-dirs: /lustre/cxuser/nm-work/tmp-${host.name}
```

8.2.4 Upgrading Hadoop

The CorrelX's cluster manager and task launcher modules interface directly to Apache Hadoop, and thus some modifications may be required if the version of Hadoop is other than that implemented in CorrelX (currently Hadoop 2.7.3).

These changes are required only if Hadoop configuration files and interfaces change with newer versions. We detail the changes that would be required in that case in the following steps.

1. Download and install Hadoop's alternative version as in §8.2.2.
2. In the configuration file, change the path to the Hadoop installation:

```
[Files]
Hadoop directory:         /home/cxuser/hadoop/hadoop-2.7.3
```

3. In `const_hadoop.py`, change the following line as required [27] (check equivalent reference for the required version):

```
HADOOP_STREAMING_JAR =    "hadoop-streaming-2.7.3.jar"
```

4. In `const_hadoop.py`, change the parameter names if they change [14], [16], [15], and [17] (check equivalent references for the required version). For example:

```
C_H_YARN_MAX_VCORES =     "yarn.scheduler.maximum-allocation-vcores"
```

5. Additionally, the following files need to be checked:
   - `lib_mapredcorr`: it includes the function to launch the Hadoop job and retrieve the resulting file (`run_mapreduce_sh()`) [28] (check equivalent reference for the required version).
   - `lib_hadoop_hdfs`: it includes the functions to start the cluster(`cluster_start()`), stop the cluster (`cluster_stop()`), moving media to the HDFS filesystem (`copy_files_to_hdfs()`), create the Hadoop configuration files (`update_hcparam()`).
   - `const_mapred`: it includes the environment variable (constant `MAP_INPUT_FILE`) read by the mapper to retrieve the filename of the block currently being processed [37].
   - `configure_hadoop_cx.sh`: the configuration script was developed for the 2.7.3 version; the folder structure and environment initialization scripts should be checked for changes.

8.2.5 Maintenance

Because CorrelX is still a prototype version, there are currently two potential storage issues:

- Hadoop's logs are not removed automatically from the system.
- CorrelX output is text-based, and does not apply averaging during correlation.

These features are useful for debugging, but storage may become an issue, due not only to storage size but also to the number of files generated. Run the following maintenance commands periodically:

```
du -sh ~/correlx/*
rm -rf ~/correlx/logs
rm -rf ~/hadoop/hadoop-2.7.3/logs
```

If running on Lustre, periodically check the `Lustre user folder`.

9 Operation

In this section, we show how to get help on the user interface and then explain how to run CorrelX. For troubleshooting during operation, please refer to §10.

9.1 User Interface Help

For command-line help:

```
python correlx/src/mapred_cx.py -h
```

which will return the following:

```
usage: mapred_cx.py [-h] [-c CONFIGURATION_FILE] [-n NODES_LIST]
                    [-s OUTPUT_LOG_FOLDER] [-f FORCED_PARAMS]
                    [--help-parameters]

CorrelX optional arguments:
  -h, --help            show this help message and exit
  -c CONFIGURATION_FILE Specify a configuration file.
  -n NODES_LIST         Specify a comma-separated list of nodes.
  -s OUTPUT_LOG_FOLDER  Specify a folder to store the output log files.
  -f FORCED_PARAMS      Specify a comma-separated list of parameter=value to
                        override the configuration file(see --help-
                        parameters).
  --help-parameters     Show all parameters for option -f.
```

For help on the forced parameters:

```
python correlx/src/mapred_cx.py --help-parameters
```

which will return the following:

```
Showing available configuration parameters:

[Argmument]        # [Comments]                               [Type]  [Example]
ppb                # Number of frames per split               int     5000
slowstart          # Initialize reduce after map completion   float   0.95
replication        # Number of copies of input splits in HDFS int     2
fftm               # FFT in mapper                            int     0
fftr               # FFT in reducer                           int     1
adjm               # Adjust number of mappers                 float   1
adjr               # Adjust number of reducers                float   1
vcores             # Number of cores per node                 int     14
mapspernode        # Simultaneous maps (?)                    int     8
reducespernode     # Simultaneous reduces (?)                 int     8
vcorespermap       # Number of virtual cores per map task     int     1
vcoresperred       # Number of virtual cores per reduce task  int     1
```

```
nodemem            # Memory per node [MB]                       int     59000
containermemmap    # Memory per container [MB]                  int     2048
containerheapmap   # Memory heap per container (map) [MB]       int     1800
containermemred    # Memory per container (map) [MB]            int     4096
containerheapred   # Memory heap per container (reducer) [MB]   int     3800
containermemam     # Memory per container (app manager) [MB]    int     2048
containerheapam    # Memory heap per container (app mgr) [MB]   int     1800
sortmem            # Sort memory for shuffle [MB]               int     800
blocksize          # Block size in distributed filesystem [MB]  int     1640000000
masterworks        # Master also doing computations             int     1
tasksperjvm        # Tasks per JVM before reinit                int     -1
avoidcopy          # Avoid copying input files if existing      int     1
deleteoutput       # Delete output file (benchmarking only)     int     0
scalest            # Linear scaling stations                    (N/A)
mediasuffix        # Suffix for media folder                    str     _16st
singleprecision    # Single precision in computations           int     0
exper              # Experiment folder                          str     ./ini_vgos_4st
out                # Output folder                              str     ./cx_out
app                # Application sources folder                 str     ./correlx/src
serial             # Run pipeline mode                          int     0
parallel           # Run Hadoop                                 int     1
nmlocport          # Nodemanager localizer port                 int     20000
nmwebport          # Nodemanager web port                       int     20001
shuffleport        # Nodemanager shuffle port                   int     20002

Example: python mapred_cx.py -f ppb=5000,slowstart=0.95
```

This help on the forced commands relies on the source code in `const_config.py`; see this file for more information. For instructions on how to add new parameters, refer to §12.1.

9.2 The Pipeline Mode in a Single Machine

Requirements Let us consider a system with at least a CorrelX minimal deployment (§8.1), and a test dataset as follows:

```
ls -R test_dataset/
  test_dataset/:
  correlation.ini  delay_model.ini  media  media.ini  sources.ini stations.ini

  test_dataset/media:
  BM434A-BR-No0017-Aa3m.vdif  BM434A-LA-No0017_3m.vdif
```

Procedure Run the following command:

```
python correlx/src/mapred_cx.py \
  -c correlx/conf/correlx.ini \
  -f exper=test_dataset,serial=1,parallel=0
```

This will invoke CorrelX on the current host, with the configuration file defined after `-c`, and overriding the parameters listed after `-f`.

Results The last lines of the output will report the execution times:

```
...
Execution times
 Type                Exec. time [s]
 Pipeline            5.17947888374
```

A new folder will be created in the test dataset folder with a symbolic link to the output file.

```
ls test_dataset
  correlation.ini     delay_model.ini  delays.ini_debug  media.ini  stations.ini
  cx_20161123_144546  delays.ini       media             sources.ini
```

Note that CorrelX generates two intermediate configuration files: `delays.ini`, with the delay information that will be passed to the mappers, and `delays.ini_debug`, which is a debugging file (see §13.4 for more details).

All intermediate and final files can be found in the newest folder created in the configured output folder (`out=...`) from the previous command (or `correlx/output` by default). Note that this is the folder that contains the output file, not the symbolic link (from the dataset folder). For example:

```
ls ~/correlx/output
  e20161123_144546

ls ~/correlx/output/e20161123_144546
  OUT_s0_v0.out  OUT_s0_v0.outpart1  OUT_s0_v0.outpart2  OUT_s0_v0.out_tmp
```

where:

- `OUT_s0_v0.outpart*` are the files with the mapper outputs: basically, the key, the metadata, and the samples encoded as base64 (please refer to Table 11 for more details, and §13.3 for a list of the parameters in the metadata). For example:

  ```
  vi correlx/output/e20161207_104603/OUT_s0_v0.outpart1
    px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.1-    0.1 789185 0.151631360059
    0.016383533664 0.00308275760408 -1.595329085e-07 -1.07192608994e-11
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 10815 256000000 2 4983372 r 64 0 0
    86140000000.0 0.32 no U MPDI3Ob/HTrMcw1AM8wy7Mocc4D6k/zDiH8McO
    HzzMMF+tA0zAPE7nDRVgdGPcDZDXHPwhgL9zPPsIc4cLB/DI8gtx8kgx+zjOD
    DwsyzfBwzPzz3sx9fDMPBPwBEPIQz8zDCMMzxpvDy8TjfNIMXP8NPA4gQ5Pw
    Xy4MLw/OCPOP0wcc4PyzNSwPnBzIA+jA8w4TwwwiXw73QDzOmLCDPA8XU/[...]
    [...]
  ```

  The string `A.A-A.A` is used to indicate that the mode "all-baselines-per-task" is used (see §16.1.1 for details). Note that the order of the headers follow the order in which they were read from the VDIF file. For the first file:

```
head -n10 correlx/output/e20161207_104603/OUT_s0_v0.outpart1|cut -c1-45
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.1-
  px-A.A-A.A-a-1-0-1-f0.00000004983372.1-s0.1-
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.0-
  px-A.A-A.A-a-1-0-1-f0.00000004983372.1-s0.0-
  px-A.A-A.A-a-0-0-0-f0.00000004994187.0-s0.1-
  px-A.A-A.A-a-1-0-1-f0.00000004994187.1-s0.1-
  px-A.A-A.A-a-0-0-0-f0.00000004994187.0-s0.0-
  px-A.A-A.A-a-1-0-1-f0.00000004994187.1-s0.0-
  px-A.A-A.A-a-0-0-0-f0.00000005014187.0-s0.1-
  px-A.A-A.A-a-1-0-1-f0.00000005014187.1-s0.1-
```

For the second file:

```
head -n10 correlx/output/e20161207_104603/OUT_s0_v0.outpart2|cut -c1-45
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s1.1-
  px-A.A-A.A-a-1-0-1-f0.00000004983372.1-s1.1-
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s1.0-
  px-A.A-A.A-a-1-0-1-f0.00000004983372.1-s1.0-
  px-A.A-A.A-a-0-0-0-f0.00000005003372.0-s1.1-
  px-A.A-A.A-a-1-0-1-f0.00000005003372.1-s1.1-
  px-A.A-A.A-a-0-0-0-f0.00000005003372.0-s1.0-
  px-A.A-A.A-a-1-0-1-f0.00000005003372.1-s1.0-
  px-A.A-A.A-a-0-0-0-f0.00000005023372.0-s1.1-
  px-A.A-A.A-a-1-0-1-f0.00000005023372.1-s1.1-
```

- `OUT_s0_v0.out_tmp` is the file with all the mapper output sorted (that is, in the order that they will be read by the reducer). Note that in pipeline mode there is only one reducer.

```
head -n10 correlx/output/e20161207_104603/OUT_s0_v0.out_tmp|cut -c1-45
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.0-
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.1-
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s1.0-
  px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s1.1-
  px-A.A-A.A-a-0-0-0-f0.00000004994187.0-s0.0-
  px-A.A-A.A-a-0-0-0-f0.00000004994187.0-s0.1-
  px-A.A-A.A-a-0-0-0-f0.00000005003372.0-s1.0-
  px-A.A-A.A-a-0-0-0-f0.00000005003372.0-s1.1-
  px-A.A-A.A-a-0-0-0-f0.00000005014187.0-s0.0-
  px-A.A-A.A-a-0-0-0-f0.00000005014187.0-s0.1-
```

- `OUT_s0_v0.out` is the file with the visibilities. Each line is composed by a key (before `\t`), some meatadata, and the visibilities. The header should follow the same format as in the map-reduce interface, but replacing the baseline by the actual baseline (`px-<st0>.<pol0>-<st1>.<pol1>`). The metadata is kept for debugging, but it should be removed. For more details please refer to Table 11).

```
vi test_dataset/cx_20161207_104603/OUT_s0_v0.out
  px-0.0-0.0-a.0.0.0-sxa28        1.1 0 0.0 0.0194662912681 -0.0 -0.0 -0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 20000 256000000 2 8103372 r 0 0 0
  86140000000.0 0.32 no 4983372.0 (1.3452471729e-05+0j) (8.3260625233e-06+0j)
  (9.55185368682e-06+0j) (1.07734919678e-05+0j) (1.31659563953e-05+0j)
  (1.2814532666e-05+0j) (1.20450905059e-05+0j) (1.32270510592e-05+0j)
  (8.48127965319e-06+0j) (1.03547918437e-05+0j) (1.04767386383e-05+0j) [...]
  [...]
```

The first keys would be as follows:

```
head -n10 correlx/output/e20161207_104603/OUT_s0_v0.out|cut -c1-25
  px-0.0-0.0-a.0.0.0-sxa28
  px-0.0-0.1-a.0.0.0-sxa28
  px-0.0-1.0-a.0.0.0-sxa28
  px-0.0-1.1-a.0.0.0-sxa28
  px-0.1-0.1-a.0.0.0-sxa28
  px-0.1-1.0-a.0.0.0-sxa28
  px-0.1-1.1-a.0.0.0-sxa28
  px-1.0-1.0-a.0.0.0-sxa28
  px-1.0-1.1-a.0.0.0-sxa28
  px-1.1-1.1-a.0.0.0-sxa28
```

If the results include phase calibration information, the same format is followed (the full accumulation window in the Fourier domain is provided):

```
  vi correlx/output/e20161207_144446/OUT_s0_v0.out
    pcal-1.0-1.0-a.0.0.0-sxa374     0.1 0 0.0 0.0155949369383 -0.0 -0.0 -0.0
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2048 32000000 2 1190586 c 160
    5000000 0 3480400000.0 1.0 no 998074.0 (0.157935828877-0.362545454545j)
    (0.123346675256+0.00569898088228j) (-0.210728794277-0.0564855844591j)
    (-0.320702738751-0.351978281952j) (-0.366532691707+0.140074840036j)
    (-0.0607365741594-0.376711394726j) (0.369518306128-0.0331197836515j) [...]
    [...]
```

And the first keys would be:

```
  cat correlx/output/e20161207_144446/OUT_s0_v0.out|cut -c1-29|grep pcal|
      head -n10
    pcal-0.0-0.0-a.0.0.0-sxa374     0
    pcal-0.1-0.1-a.0.0.0-sxa374     0
    pcal-1.0-1.0-a.0.0.0-sxa374     0
    pcal-1.1-1.1-a.0.0.0-sxa374     0
    pcal-0.0-0.0-a.10.0.10-sxa374
    pcal-0.1-0.1-a.10.0.10-sxa374
```

```
    pcal-1.0-1.0-a.10.0.10-sxa374
    pcal-1.1-1.1-a.10.0.10-sxa374
    pcal-0.0-0.0-a.1.0.1-sxa374     0
    pcal-0.1-0.1-a.1.0.1-sxa374     0
```

The prefix `OUT` can be configured (Table 4), and `s<S>_v<V>` indicates the number of workers (slaves) (`<S>`) and virtual CPU cores per node (`<V>`) used in Hadoop, which for the pipeline mode are zero by convention. Note however that it is possible to do multicore processing in pipeline mode (see 16.4 for details).

More details on how to use the provided tools to convert the output files into SWIN format and plot the visibilities in the two formats are provided in §9.7, §9.8.4, and §9.8.5, respectively.

9.3 The Parallel Mode in a Local Cluster with One Node

Requirements It is assumed that the steps presented in §9.2 were followed successfully, and also that a correct installation has been performed following the steps presented in §8.2.

Procedure Run the following command, now specifying parallel processing and the number of CPU cores per node and available memory:

```
python -u correlx/src/mapred_cx.py \
  -c correlx/conf/correlx.ini \
  -f exper=test_dataset,serial=0,parallel=1,vcores=2,nodemem=5000,adjr=-1 \
  -s correlx/logs/job_1 | \
  tee correlx/logs/job_1.txt
```

This will invoke CorrelX (on the current host `‘hostname‘`, since no list of nodes is provided with `-n`), with the configuration file defined after `-c`, placing the results on a folder defined after `-s` and overriding the parameters listed after `-f`. Note that number of reducers is forced to 1 (with `adjr=-1`). For more details on the specified arguments, see §9.1. Note also that we add the echoing of the output through `tee` (in combination with `python -u`) for convenience.

Results The main log will show the progress of the complete process, including the Hadoop initialization:

```
[...]
 Launched processes:
  6114 Jps
  5094 NameNode
  6071 JobHistoryServer
  5607 ResourceManager
  5741 NodeManager
  5247 DataNode
  5439 SecondaryNameNode
 List of nodes:
```

```
16/11/30 15:27:03 INFO client.RMProxy: Connecting to ResourceManager at
        ubuntutd/127.0.1.1:8032
   Total Nodes:1
   Node-Id          Node-State Node-Http-Address   Number-of-Running-Containers
   ubuntutd.mit.edu:44514    RUNNING ubuntutd.mit.edu:8188                0
Nodes off: []
[...]
```

A similar list with the Hadoop processes launched should be displayed, and also the list of initialized nodes. For troubleshooting please refer to §10.1. After the initialization has been completed, the progress of the MapReduce will be displayed:

```
[...]
16/11/30 15:27:13 INFO mapreduce.Job: Job job_1480537559896_0001
                                          running in uber mode : false
16/11/30 15:27:13 INFO mapreduce.Job:  map 0% reduce 0%
16/11/30 15:27:19 INFO mapreduce.Job:  map 50% reduce 0%
16/11/30 15:27:22 INFO mapreduce.Job:  map 100% reduce 0%
16/11/30 15:27:33 INFO mapreduce.Job:  map 100% reduce 84%
16/11/30 15:27:39 INFO mapreduce.Job:  map 100% reduce 100%
16/11/30 15:27:47 INFO mapreduce.Job: Job job_1480537559896_0001 completed
                                          successfully
[...]
```

The last lines of the output in the log in `correlx/output/logs/job_1/log...txt` report the execution times of the transfer of files into the distributed system and also of the processing:

```
[...]
File IO approximate times
 Type                File IO. time [s]
 HDFS-put 1s-2v      21.8214468956
 HDFS-get 1s-2v      1.01278018951
 File-sort 1s-2v     0.0125930309296

Execution times
 Type                Exec. time [s]
 Pipeline            0
 Hadoop 1s-2v        57.2714061737
```

That is, moving the input data into HDFS took 22 s, getting the output file from the HDFS back to the local filesystem took 1 s, and doing the MapReduce based correlation took 57 s. Note that these execution times are higher than those in the pipeline example (§9.2); this is due to the use of a small dataset, since in this case the overhead of running Hadoop exceeds the actual computation time.

Similarly to the pipeline mode (§9.2), a new folder with a symbolic link to the output file with the visibilities will be placed in the dataset folder.

9.4 The Parallel Mode in a Local Cluster with Multiple Nodes

Requirements It is assumed that the steps presented in §9.3 were followed successfully. Make sure that Hadoop and its plugins have been properly installed on all nodes and that the master node can login into the worker nodes (§8.2).

We consider a scenario with two machines: a master `ubuntutd` and a worker `ubuntutd2` as in §8.2.2, step 8. More specifically, the procedure presented below has been tested on two virtual machines on VirtualBox [38], connected through a NAT network, and running Ubuntu 16.04.1 Server [39] (with standard system utilities and OpenSSH server). If using this test environment, make sure to run `sudo apt-get update && sudo apt-get upgrade` after the installation.

Procedure Launch CorrelX, specifying the list of available nodes. With many nodes, the master node should not run containers (`masterworks=0`) to avoid overloading this node. In the following example, however, we select both nodes as workers:

```
python -u correlx/src/mapred_cx.py \
  -n ubuntutd,ubuntutd2 \
  -c correlx/conf/correlx.ini \
  -f exper=test_dataset,serial=0,parallel=1,vcores=2,nodemem=5000,adjm=-2, \
     adjr=-2,masterworks=1 \
  -s correlx/logs/job_3 | \
  tee correlx/logs/job_3.txt
```

Note the addition of the argument `-n` with the list of nodes, where the first item in the list must be the master node.

Results Along the processing, both nodes should be listed:

```
[...]
 List of nodes:
16/12/01 14:38:03 INFO client.RMProxy: Connecting to ResourceManager at
                                                        ubuntutd/10.0.2.9:8032
   Total Nodes:2
   Node-Id          Node-State Node-Http-Address   Number-of-Running-Containers
   ubuntutd:33978      RUNNING     ubuntutd:8188                              0
   ubuntutd2:43993     RUNNING    ubuntutd2:8188                              0
Nodes off: []
[...]
```

If the second node is not running, refer to §10.

```
[...]
Execution times
 Type                 Exec. time [s]
 Pipeline             0
 Hadoop 2s-2v         44.5479369164
```

In general, the master node should not be acting as a worker as well.

9.5 The Parallel Mode in the Cloud

The differences with the cluster deployment are basically the addition of the scheduling layer (SLURM) and the use of the Lustre plugin. It is therefore recommended to perfom a cluster deployment (§9.4) prior to following the procedure presented in this section.

Requirements Two additional scripts are required compared to the cluster deployment:

- Job request script: this script contains the `sbatch` call to the launcher script. For example:

```
[...]
sbatch  --job-name=$JOBNAME
        --ntasks=$NTASKS
        --ntasks-per-node=$NTASKS_PER_NODE
        --time=$EST_TIME_EXP
        --partition=$PARTITION
        --output=$LOG_FILE
        $SCRIPT $CONFIG_FILE $ID_JOB $FORCED_PARAMS $NNODES  [...]
[...]
```

  where: the parameter `JOBNAME` is an identifier for the job, `NTASKS` is the total number of cores, `NTASKS_PER_NODE` is the number of cores per node, `EST_TIME_EXP` is the time limit for the job, `PARTITION` is the subset of nodes in the cluster the job will be requested, `LOG_FILE` is the path to the log file for this job, and finally the the launcher script (to be called once the job request is allocated) and all its arguments.

- Launcher script: this script is initiated once the job request has been granted, and it contains the necessary steps to get the cluster ready (ssh configuration, removing temporary files, etc.) and the call to `python mapred_cx.py` with its associated arguments. Note that CorrelX is launched only in one node. For example:

```
# Arguments
CONFIG_FILE=$1                                                    # Configuration file
SUFFIX_FOLDER=$2                                                  # Job id
FORCED_PARAMS=$3                                                  # Number of nodes
NNODES=$4                                                         # Forced parameters
LOG_FILE=$5                                                       # Log file
CONF_FOLDER=$6                                                    # Configuration folder
[...]

# Delete temporary folders, find unused ports, SSH configuration
[...]

# CorrelX
srun -N 1 -n 1 $PARAMS_NODES python --version
srun -N 1 -n 1 $PARAMS_NODES python mapred_cx.py -c ${CONFIG_FILE} -s ${EXP}
   -f ${FORCED_PARAMS} -n `cat $CONF_FOLDER/$FIRST_NODE/hosts_$FIRST_NODE`
```

We provide some support scripts in `correlx/sh` for the ssh configuration and for obtaining a list of open ports (provided to solve a potential issue described in §10.1), although these may vary in the target cloud.

Procedure Configure the variables/arguments for your job request script and launch it.

Results Results and logging should be similar as for the cluster deployment (§9.4).

If running on Lustre, Hadoop output folders will be placed in the path specified in `Lustre user folder` in the configuration file `correlx.ini`. The output folder will display one file per reducer (`part-00000`, `part-00001`, ...) along with an empty file indicating the result of the correlation (_SUCCESS).

The merged output file will be placed in the path specified in the configuration file `correlx.ini` in `Output directory`, and the symbolic link to this file will be in a folder in the dataset directory as for the previous cases.

9.6 Conversion of Input Configuration

Requirements Let us assume a directory structure with a `.input` and a `.im` files. Additionally, there will be a folder with the media referenced by the `.input` file (not necessarily in this same folder). For example:

```
ls -R test_config
  test_config:
  bm434aYs20_20.im  bm434aYs20_20.input  vdif

  test_config/vdif:
  BM434A-BR-No0017-3m.vdif  BM434A-LA-No0017_3m.vdif
```

and where the `.input` file points at the files in `test_config/vdif`:

```
tail -n6 test_config/bm434aYs20_20.input
  # DATA TABLE #######!
  D/STREAM 0 FILES:   1
  FILE 0/0:           /home/cxuser/test_config/vdif/BM434A-BR-No0017-3m.vdif
  D/STREAM 1 FILES:   1
  FILE 1/0:           /home/cxuser/test_config/vdif/BM434A-LA-No0017-3m.vdif
```

These paths to the files are the only ones that are read from the `.input` file.

Procedure Run the conversion script, specifying the target folder and the name of the input files (no extension):

```
python correlx/src/convert_im_cx.py test_config bm434aYs20_20
```

which will return:

```
Generating delay_model.ini...
 Filtering sources: 0
 Processing test_config/bm434aYs20_20.im ...
 Writing results to test_config/delay_model.ini ...
 Writing summary to test_config/delay_model.ini_report ...
Generating stations.ini...
 Processing test_config/bm434aYs20_20.input ...
 Writing results to test_config/stations.ini ...
Generating media.ini...
 Processing test_config/bm434aYs20_20.im ...
 Writing results to test_config/sources.ini ...
Generating correlation.ini...
 Processing test_config/bm434aYs20_20.input ...
 Writing results to test_config/correlation.ini ...
Generating media.ini...
 Processing test_config/bm434aYs20_20.input ...
Creating symbolic links for media in test_config/media
 Link                     Source
 BM434A-BR-No0017-3m.vdif /home/cxuser/test_config/vdif/BM434A-BR-No0017-3m.vdif
 BM434A-LA-No0017-3m.vdif /home/cxuser/test_config/vdif/BM434A-LA-No0017-3m.vdif
unlink test_config/media/BM434A-BR-No0017-3m.vdif
ln -s /home/cxuser/test_config/vdif/BM434A-BR-No0017-3m.vdif test_config/media/\
     BM434A-BR-No0017-3m.vdif
unlink test_config/media/BM434A-LA-No0017-3m.vdif
ln -s /home/cxuser/test_config/vdif/BM434A-LA-No0017-3m.vdif test_config/media/\
     BM434A-LA-No0017-3m.vdif
(!) If moving ini folder do: cp -r --preserve=links test_config
      dir_ini_destination
 Writing results to test_config/media.ini ...
Done.
```

The last part reports the creation of symbolic links to the VDIF files in the folder media, and instructions are provided on how to copy that folder (if necessary).

Results The resulting folder structure is as follows:

```
ls -R test_config
  test_config:
  bm434aYs20_20.im correlation.ini delay_model.ini_report media.ini stations.ini
  bm434aYs20_20.input delay_model.ini media               sources.ini vdif

  test_config/media:
  BM434A-BR-No0017-3m.vdif  BM434A-LA-No0017-3m.vdif

  test_config/vdif:
  BM434A-BR-No0017-3m.vdif  BM434A-LA-No0017_3m.vdif
```

Note that the files in `test_config/media` correspond to symbolic links for the file paths reported in the `.input` file. The created `.ini` files correspond to those described in §5.2. For the delay model conversion, a summary file is provided (only for reporting, not used by CorrelX) with the path of the processed file, the ids for the sources and stations, and information about the epoch of the polynomials:

```
vi test_config/delay_model.ini_report
  Summary:
   Input:    test_config/bm434aYs20_20.im
   Sources:  0
   Stations: 1, 0
   Interval: 120
   MJDs:     57235
   seconds:  30000, 30120, 30240, 30360
```

9.7 Conversion of Output

Requirements Assume a directory structure as follows:

```
ls -R test_output
  test_output/:
  correlation.ini    delay_model.ini delays.ini_debug  media.ini   stations.ini
  cx_20161115_101731 delays.ini      delays.ini_serial  sources.ini

  test_output/cx_20161115_101731:
  OUT_s4_v14_20161115_101731_node091.out
```

Run the conversion script specifying the target folder and the name of the CorrelX output file, averaging groups of 32 coefficients of the visibilities (averaging currently only for zoom band processing):

```
correlx/sh/cx2d.sh ~/correlx/src test_output/cx_20161115_101731
    OUT_s4_v14_20161115_101731_node091.out 32
```

Currently, the path to the CorrelX output file currently must be separated into path and file, and the averaging specification must be indicated in the interface (instead of the configuration files); this should be simplified in the future.

This command returns a log composed of 4 parts:

1. Determination of zoom bands from experiment definition files:

```
Zoom bands
OUT_s4_v14_20161115_101731_node091.out
OUT_s4_v14_20161115_101731_node091.out_zoom
32
[]
Processing metadata for zoom bands...
```

```
Band Zoom Band BW   Zoom BW  Band fft Zoom fft Zoom avg fft Band freq   Zoom freq
1    2    128.0 MHz 51.2 MHz 40960    16384    512          86268.0 MHz 86330.290625 MHz
1    3    128.0 MHz 51.2 MHz 40960    16384    512          86268.0 MHz 86271.696875 MHz
0    4    128.0 MHz 51.2 MHz 40960    16384    512          86140.0 MHz 86213.103125 MHz
0    5    128.0 MHz 51.2 MHz 40960    16384    512          86140.0 MHz 86154.509375 MHz
```

2. Zooming of output file, no change in format:

```
Processing zoom bands...
id                                   read       fft       z_i       z_e
px-0.0-0.0-a.0.0.4-sxa999           40960       512     23393     39777
px-0.0-0.0-a.0.0.5-sxa999           40960       512      4643     21027
px-0.0-0.1-a.0.0.4-sxa999           40960       512     23393     39777
px-0.0-0.1-a.0.0.5-sxa999           40960       512      4643     21027
[...]
px-1.1-1.1-a.7.3.2-sxa125           40960       512     19933     36317
px-1.1-1.1-a.7.3.3-sxa125           40960       512      1183     17567
40960
```

3. Reading of zoomed output file:

```
Convert format
Manually configured values:
 pcal_scaling=0.0027027027027

Converting cx2d...
px-0.0-0.0-a.0.0.4-sxa999       1.1 0 0.0 0.0194662912681 -0.0 -0.0 -0.0 0.0 0.0 0.0 0
px-0.0-0.0-a.0.0.5-sxa999       1.1 0 0.0 0.0194662912681 -0.0 -0.0 -0.0 0.0 0.0 0.0 0
px-0.0-0.1-a.0.0.4-sxa999       1.1 0 0.0 0.0194662912681 -0.0 -0.0 -0.0 0.0 0.0 0.0 0
px-0.0-0.1-a.0.0.5-sxa999       1.1 0 0.0 0.0194662912681 -0.0 -0.0 -0.0 0.0 0.0 0.0 0
[...]
px-1.1-1.1-a.7.3.2-sxa125       1.1 0 0.0 0.0194665978812 -0.0 -0.0 -0.0 0.96 0.0 0.0
px-1.1-1.1-a.7.3.3-sxa125       1.1 0 0.0 0.0194665978812 -0.0 -0.0 -0.0 0.96 0.0 0.0
```

4. Conversion of read records into SWIN format:

```
ac_id      ac_s      ap   chan    pol
0      30000.16   0.32      4    LL
0      30000.16   0.32      5    LL
0      30000.16   0.32      4    LR
0      30000.16   0.32      5    LR
[...]
3      30001.12   0.32      2    RR
3      30001.12   0.32      3    RR
No phase calibration results found.

Output files:
 OUT_s4_v14_20161115_101731_node091.out_zoom.out2swin2scaled
```

Results A SWIN file (and PCAL files if applicable) will be generated:

```
ls test_output/cx_20161115_101731/*DIFX*
  test_output/cx_20161115_101731/[...]DIFX_57235_30000.s0000.b0000
```

The generated files can now be converted into a Mark4 fileset using the DiFX tool `difx2mark4` (see [40] for installation and [41] for usage) and processed using the HOPS tool `fourfit` (see [12] for installation and usage). For example, let `~/difx_bm` be the folder with the DiFX configuration files:

```
mv cx_20161115_101731/OUT_s31_v14_20161115_101731_node068.out_zoom.\
    out2swin2scaledDIFX_57235_30000.s0000.b0000 cx_20161115_101731/\
    DIFX_57235_30000.s0000.b0000
mkdir ~/bm_convert
mkdir ~/bm_convert/bm434aYs20_20.difx
cp cx_20161115_101731/DIFX_57235_30000.s0000.b0000 ~/bm_convert/bm434aYs20_20.difx/
cd ~/bm_convert
cp ~/difx/difx_bm/bm434aYs20_20.{input,im,calc} .
cp ~/difx/difx_bm/bm434aY.vex.obs .
```

Now run the provided script to update the paths in the `.calc` and `.input` files:

```
correlx/sh/rel_paths.sh bm434aYs20_20
```

which will display the found paths, ask for confirmation, and update the paths:

```
Paths found:

bm434aYs20_20.calc
 VEX FILE:           /home/ajva/difx/bm434/bm434aY.vex.obs
 IM FILENAME:        /home/ajva/difx/bm434/bm434aYs20_20.im
 FLAG FILENAME:      /home/ajva/difx/bm434/bm434aYs20_20.flag
bm434aYs20_20.input
 CALC FILENAME:      /home/ajva/difx/bm434/bm434aYs20_20.calc
 CORE CONF FILENAME: /home/ajva/difx/bm434/bm434aYs20_20.threads
 OUTPUT FILENAME:    /home/ajva/difx/bm434/bm434aYs20_20.difx

New path:./
Press any key to continue... (Ctrl+C to cancel)

Updating bm434aYs20_20.calc
 VEX FILE:           ./bm434aY.vex.obs
 IM FILENAME:        ./bm434aYs20_20.im
 FLAG FILENAME:      ./bm434aYs20_20.flag
Updating bm434aYs20_20.input
 CALC FILENAME:      ./bm434aYs20_20.calc
 CORE CONF FILENAME: ./bm434aYs20_20.threads
 OUTPUT FILENAME:    ./bm434aYs20_20.difx
```

```
Unchanged paths: in bm434aYs20_20.input
FILE 0/0:          /nobackup1/ajva/BM434A-BR-No0017_Aa75g.vdif
FILE 1/0:          /nobackup1/ajva/BM434A-BR-No0017_Ab75g.vdif

Done.
```

Finally, run `difx2mark4`:

```
~/difx/difx/bin/difx2mark4 -e 1001 bm434aYs20_20 -b X 0 1e14
  Warning: env. var. DIFX_VERSION is not set

  Processing job 0/1
    Processing scan 0/1: No0017
      Generating root file
        Opening vex file <./bm434aY.vex.obs>
        output rootfile: 1001/No0017/3C454_3.zeyxrk
        number of stations: 2
      Generating Type 1s
        opened input file ./bm434aYs20_20.difx/DIFX_57235_30000.s0000.b0000
        created type 1 output file 1001/No0017/bl..zeyxrk
        created type 1 output file 1001/No0017/bb..zeyxrk
        created type 1 output file 1001/No0017/ll..zeyxrk
        DiFX visibility records read in scan      160
        DiFX visibility records discarded           0
      Generating Type 3s
        created type 3 output file 1001/No0017/b..zeyxrk
        No input phase cal file ./bm434aYs20_20.difx/PCAL_*BR for antenna BR
        created type 3 output file 1001/No0017/l..zeyxrk
        No input phase cal file ./bm434aYs20_20.difx/PCAL_*LA for antenna LA
  1 of 1 scans converted!
  Warning: env. var. DIFX_VERSION is not set
  1 of 1 DiFX filesets converted to 1 Mark4 filesets
```

The Mark4 dataset is ready to be processed in HOPS. For more information, see [12]. For example:

```
fourfit -b ab -p 1001/No0017/3C454_3.ywkmsq
```

Press ‘s’ on the required plots to save them into a `.ps` file. The following scripts are provided to replace the header and convert the files into a single pdf:

```
cd 1001
~/correlx/sh/change_header_plots_correlx.sh .
~/correlx/sh/all_ps_to_pdf.sh .
```

The resulting file will be like the one displayed in Fig. 4.

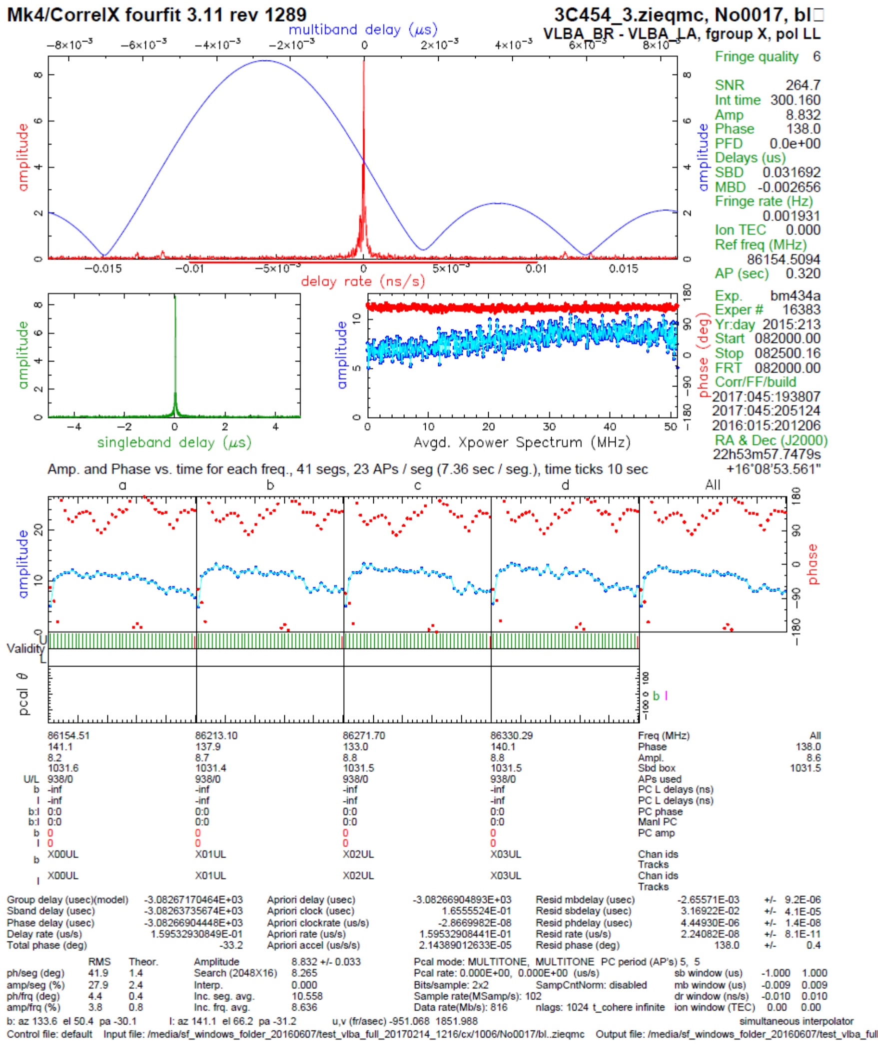


Figure 4: Fourfit plot from visibilities generated with CorrelX.

9.8 Additional Tools

In this section, we show some examples on the operation of additional tools provided with CorrelX. The presented examples were run on an IPython notebook [45]; we provide some of the command-line functions for convenience.

9.8.1 VDIF Statistics

In this section, we show some examples of how to display information from VDIF files.

Frame-by-frame information display To display tabulated information from the headers of all the frames on a given range:

```
python ~/vdif_info.py -h
  usage: vdif_info.py [-h] [-n LIMIT_FRAMES] [-s SKIP_FRAMES] [-b] [--summary]
                      [file_vdif]

  VDIF info

  positional arguments:
    file_vdif        VDIF file.

  optional arguments:
    -h, --help       show this help message and exit
    -n LIMIT_FRAMES  Maximum number of frames.
    -s SKIP_FRAMES   Number of frames to skip.
    -b               Brief output (no legend for columns).
    --summary        Display only a summary for the whole file.
```

For example, to skip the first 1000 frames and showing the header information for the 10 following frames:

```
python correlx/src/vdif_info.py bm434/BM434A-BR-No0017.vdif -n10 -s1000
```

which will return:

```
VDIF file stats:
count:     Counter (based on skip and limit)
st:        Station ID numeric str
I:         Invalid
leg:       Legacy
vv:        VDIF version
epoch:     Ref. epoch:
[s]:       Seconds frame
num:       Frame number
thread:    Thread ID
log2C:     Log2 of the number of channels per frame
len[B]:    Frame Length in bytes
len[kB]:   Frame Length in kilobytes
R/C:       Data type
bpsamp:    Bits per sample

count    st  I  leg  vv  epoch    [s]   num  thread   log2C  len[B] len[kB] R/C  bpsamp
    0 65528  0    0   1  57235   30000  250       0       0    5032       4   R       2
    1 65528  0    0   1  57235   30000  250       2       0    5032       4   R       2
    2 65528  0    0   1  57235   30000  250       1       0    5032       4   R       2
    3 65528  0    0   1  57235   30000  250       3       0    5032       4   R       2
    4 65528  0    0   1  57235   30000  251       0       0    5032       4   R       2
```

```
   5 65528  0    0   1  57235    30000   251        2        0    5032         4   R        2
   6 65528  0    0   1  57235    30000   251        1        0    5032         4   R        2
   7 65528  0    0   1  57235    30000   251        3        0    5032         4   R        2
   8 65528  0    0   1  57235    30000   252        0        0    5032         4   R        2
   9 65528  0    0   1  57235    30000   252        2        0    5032         4   R        2
```

It is easy to see that this example corresponds to a threaded VDIF file (4 threads), with a single band per frame with 2-bit real samples.

The same output can be generated from an IPython notebook:

```
import lib_vdif

lib_vdif.show_headers_vdif("bm434/BM434A-BR-No0017.vdif",\
                         limit_frames=10,skip_frames=1000)
```

General information display The previous script can also be used to display a summary of all the information from the headers of all the frames in a file:

```
python vdif_info.py bm434/BM434A-BR-No0017_300m.vdif --summary -b
```

Note the option `-b` to list the number of frames as ranges (if there are missing frame numbers, then multiple ranges will be displayed), otherwise all the frame numbers found will be displayed. The processing may take a while depending on the size of the VDIF file, so that a percentage on the overall progress is displayed along this processing. That will show:

```
Reading VDIF file...   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
File information:
 File name:         basic_files_data/ini_files_eht_two/media/BM434A-BR-No0017-Aa300m.vdif
 Data size:         314575480 B (~300 MB)
 Stations:          [65528L]
 Epochs:            57235
 Seconds:           30000,30001
 Frames:            [[0:12799]]
 Sizes:             5032
 Data type:         0
 Num.channels:      1
 Num.threads:       0,2,1,3
 Bits per sample:   2
```

Note that the provided lists are sets (`unique`) and not sorted. This utility can also be called from an IPython notebook:

```
import lib_vdif
lib_vdif.get_vdif_stats("bm434/BM434A-BR-No0017_300m.vdif")
```

9.8.2 VDIF Generator

A VDIF file generator is provided to help during debugging. It is important to note that **this is not a VLBI signal simulator**. The provided script's current implementation provides the functionality to generate a signal composed by a set of sines with certain amplitudes and frequencies and a certain signal-to-noise ratio, linearly quantized. The initial purpose of this tool was to debug data distribution in CorrelX by tracking these tones along the system. This tool can generate multithread or multichannel VDIF files.

Even though this tool is not well tested, it is provided for further extension, because it should be easy to modify to add a custom signal generator and quantizer (see §12.7 for more details). The current implementation generates the defined signal composed by sines and noise, then divides the spectrum in the specified number of bands, and finally writes these bands into either threads or channels in the VDIF file. This script requires the installation of the module `bitarray` [46] and activation of the flag `lib_vdif.USE_BITARRAY`.

We provide the following usage example:

```
import vdif_generator

year   = 2016
month  = 4
day    = 7
hour   = 6
minute = 11
second = 0

vdif_generator.generate_vdif(\
            tot_stations           = 2,\
            bw_in                  = 500e3,\
            bytes_payload_per_frame= 1024,\
            bits_quant             = 4,\
            snr_in                 = 1e2,\
            sines_f_in             = [10e3, 290e3, 400e3],\
            sines_amp_in           = [0.3, 0.2, 0.5],\
            prefix                 = "test_s",\
            signal_limits          = [-2, +2],\
            log_2_channels         = 0,\
            num_threads            = 4,\
            threaded_channels      = 1,\
            num_taps_filterbank    = 256,\
            date_vector            = [year, month, day, hour, minute, second],\
            seconds_duration       = 1)
```

This example will generate the same data for two stations: a signal of a bandwidth of 500 kHz composed by three sines at 10 KHz, 290 kHz and 400 kHz with amplitudes 0.3, 0.2, and 0.5 relative to the specified signal-to-noise ratio (100). The signal is divided into 4 bands (processed by a FIR filter with 256 taps) that are mapped into four threads, and linearly quantized between -2 and $+2$ with 4 bits per sample. The approximate size of each VDIF frame is specified to be 1024 bytes (which will be divided by the number of bands if each band corresponds to one thread, as in this case). The timestamp for the first sample is 2016/04/07 06:11:00, and the signal duration is 1 second.

The output will display the following summary:

```
VDIF frames generator:

 Data type: 4-bit real (linearly quantized)
 Total BW            = 500000.0 Hz
 SNR                 = 20.0 dB
 Tones:
  0    f = 10000.0 Hz,   a = 3.0
  1    f = 290000.0 Hz,   a = 2.0
  2    f = 400000.0 Hz,   a = 5.0
 Data channelization:
  Number of bands:     4
  Band BW:             125000.0 Hz
  Filterbank FIR taps: 256
 VDIF channels/threads:
  Number of channels:  1
  Number of threads:   4 (one band per thread)
 Signal time info:
  Date:            2016-04-07 06:11:00
   MJD:            57485
   Seconds:        22260
  Signal duration: 1 s
Output:
 Output folder:       ./
 VDIF file(s):        test_s-0.vt,test_s-1.vt
 Samples per frame:   2048
 Frames per second:   489
 Estimated file size: 489 fps * 1088.0 B/f * 1 s = 532032.0 B (519.5625 kB)
```

To review the statistics of one of the generated files:

```
import vdif_generator

filename="test_s-0.vt"
vdif_generator.get_vdif_stats(filename,short_output=1)
vdif_generator.show_headers_vdif(filename,limit_frames=5,brief=1)
```

which will show:

```
Reading VDIF file...   0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
File information:
 File name:         test_s-0.vt
```

```
  Data size:         563328 B (~0.5372314453125 MB)
  Stations:          [0]
  Epochs:            57485
  Seconds:           22260
  Frames:            [[0:488]]
  Sizes:             288
  Data type:         0
  Num.channels:      1
  Num.threads:       0,1,2,3
  Bits per sample:   4
 VDIF file stats:

  count  st  I  leg   vv   epoch     [s]  num  thread  log2C  len[B] len[kB] R/C  bpsamp
      0   0  0    0    7   57485   22260    0       0      0     288       0   R       4
      1   0  0    0    7   57485   22260    0       1      0     288       0   R       4
      2   0  0    0    7   57485   22260    0       2      0     288       0   R       4
      3   0  0    0    7   57485   22260    0       3      0     288       0   R       4
      4   0  0    0    7   57485   22260    1       0      0     288       0   R       4
```

An additional debugging tool is provided to display the spectra of the different bands available in the VDIF file. The following code will generate the plots in Fig. 5:

```
import vdif_generator

filename="test_s-0.vt"
bw=250e3
vdif_generator.plot_signal_from_packets(filename,bw,only_station=0,\
  packet_limit=4,same_figure=0)
```

It is also possible to plot all the bands concatenated on a common axis on their corresponding part of the full spectrum. The following code will generate the plots in Fig. 6:

```
import vdif_generator

filename="test_s-0.vt"
bw=250e3
vdif_generator.plot_signal_from_packets(filename,bw,only_station=0,\
  packet_limit=4,same_figure=2)
```

9.8.3 CorrelX Output Comparator

The following example shows how to use a CorrelX visibilities comparator. The script simply compares the headers and the number of lines and number of coefficients, and returns the sum of the L2-norm of all pairs of visibilities from two files. This tool can be used when testing changes in the correlator that may involve slight changes in the output such as multi-threading modules, etc.

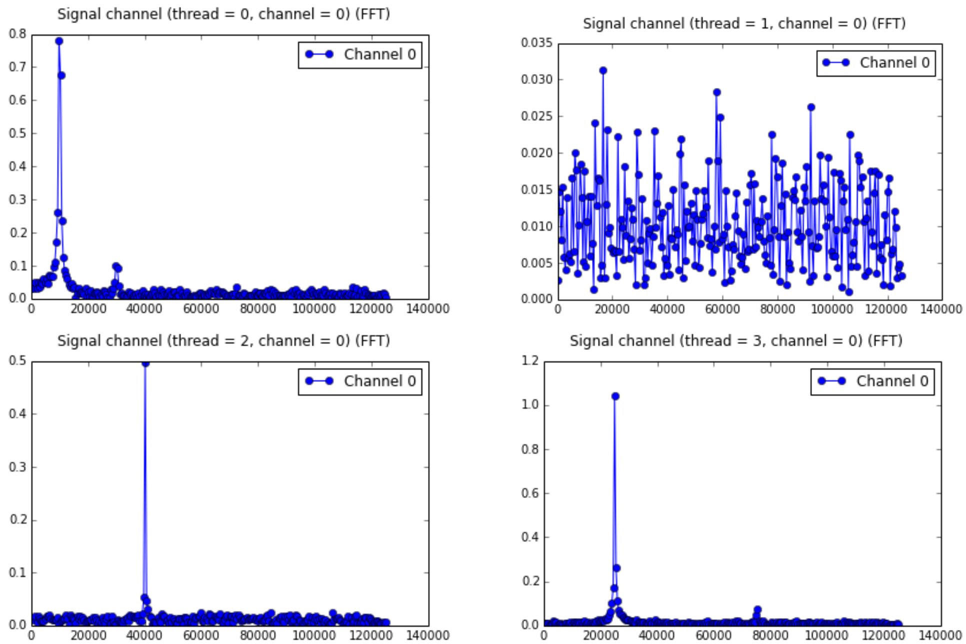


Figure 5: VDIF test file spectra for the four threads, separate bandwidths.

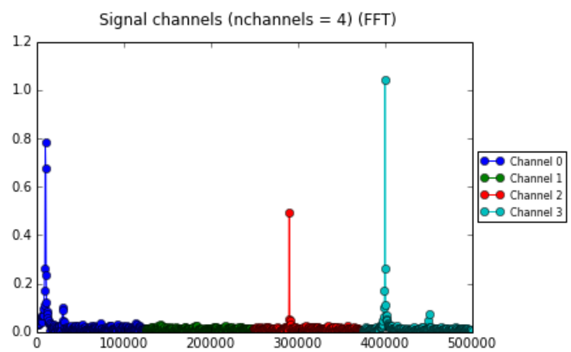


Figure 6: VDIF test file spectra (DFT coefficients) for the four threads, grouped bandwidth.

```
python correlx/src/vis_compare.py -h
  usage: vis_compare.py [-h] [-f] file_1 file_2

  CorrelX visibilities comparator: computes the sum of the L2-norm for all
  visibilities from two files. It will stop if headers or number of visibilities
  differ. Use only for debugging.

  positional arguments:
```

```
    file_1      Reference file with visibilities.
    file_2      Test file with visibilities.

  optional arguments:
    -h, --help  show this help message and exit
    -f          Force execution even if headers or number of visibilities
                differ.
```

For example:

```
python correlx/src/vis_compare.py correlx/output/OUT_ref.out correlx/output/OUT_test.out
  File 1:                        correlx/output/OUT_ref.out
  File 2:                        correlx/output/OUT_test.out
  Visibilities compared:         20
  Num. coefficients per vis.:    40960
  Total error:                   1.51822966843e-31
```

9.8.4 CorrelX Output Plotter

The following example shows how to plot the visibilities of a CorrelX output file:

```
import cx2d_lib
cx_file="bm434/cx_20161115_101731/OUT_s4_v14_20161115_101731_node091.out"
cx2d_lib.plot_vis_cx(cx_file,'Example 1',max_lines=2)
cx2d_lib.plot_vis_cx(cx_file,'Example 2',max_lines=1,\
            interval_start=18000,interval_end=20000)
```

This example will display all the coefficients of the first two lines of the file, then the range of coefficients [18000,19999] for the visibilities of the first line, as in Fig. 7.

Note that, based on the displayed headers, the first line corresponds to the autocorrelation of station 0 polarization 0 (`px-0.0-0.0` i.e., st0.pol0-st0.pol0), and the second line to the correlation of station 0, polarizations 0 and 1 (`px-0.0-0.1` i.e., st0.pol0-st0.pol1); both to the accumulation period 0 and band 0 (`a.0.0.0` i.e., generalkey0.accuperiod0.band0).

9.8.5 SWIN Output Plotter

The following example shows how to plot the visibilities of a DiFX output file (Fig. 8):

```
import cx2d_lib
cx_file="v15328/h_1000.difx/DIFX_57350_068255.s0000.b0000"
filter_pols=["XX"]        # Show only records for polarizations X and X
filter_bands=[0]          # Show only records for band 0
complex_vector_length=128 # Number of coefficients in SWIN file
filter_seconds=[68255.5]  # Show only records for acc. period centered at 68255.5
vis_limit=4               # Show only the first four records that match criteria
```

```
cx2d_lib.read_doutput(cx_file,complex_vector_length,vis_limit,filter_bands,\
    filter_pols,filter_seconds,v=0)
```

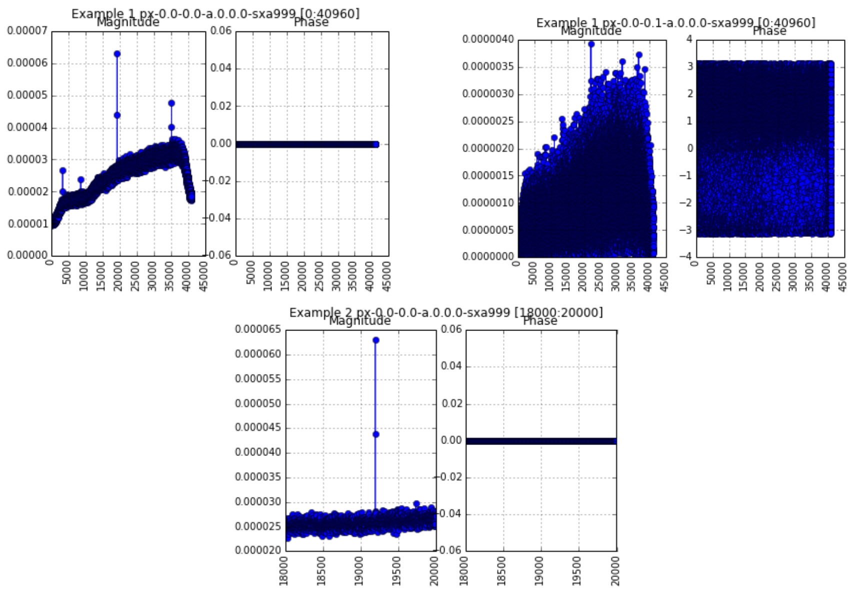


Figure 7: CorrelX output display example.

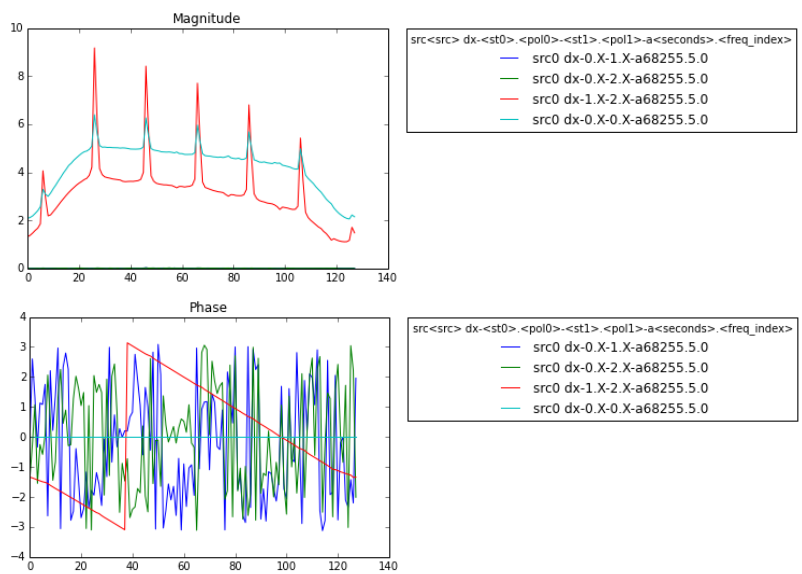


Figure 8: SWIN file display example.

10 Troubleshooting

In this section, we provide some examples of typical debugging procedures.

10.1 Node Issues during Initialization

JAVA_HOME folder not found The following error may be displayed during initialization:

```
Error: JAVA_HOME is not set and could not be found.
```

The first step is to check that the path is actually set:

```
cat hadoop/hadoop-2.7.3/etc/hadoop/hadoop-env.sh|grep -e JAVA_HOME|grep -v \#
```

If the setup script was run properly, the Java home folder should be set, and the previous command should return:

```
  export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

If this folder is not set, run the setup script as in §8.2.2, step 3. If the error persists, the issue may be due to an ssh configuration issue that avoids the Hadoop configuration files in `correlx/templates/hadoop_config_files` from being copied into the master's folder in `correlx/conf/<master_hostname>/etc_hadoop_<master_hostname>`. Make sure that the authenticity of the host has been confirmed manually through ssh, or that this check is disabled as shown in §8.2.2, step 5. If the error persists, check `pdsh` manually. Also note that for clusters with a single node, or with multiple nodes and a shared filesystem, the use of `pdsh` can be disabled by setting `Over SLURM: yes` in the configuration file.

Nodes unavailable to Hadoop Some worker nodes may fail to initiate the Hadoop entities. The CorrelX nodemanager log provides a line reporting the list of nodes that were unable to join the Hadoop cluster:

```
  Total Nodes:0
  Node-Id          Node-State Node-Http-Address       Number-of-Running-Containers
Nodes off: ['node335', 'node336']
Hadoop initialization failed!
```

This error is typically caused by:

- The ports being requested by the Hadoop entities being already held.
- Insufficient storage available at the workers.
- Some issue with the `yarn-env.sh` script where the variable `host.name` is set.

If the issue is for used ports, use the provided scripts `get_used_ports.sh` and `get_params_ports.py` to get free ports for the command-line variables `nmlocport`, `nmwebport`, and `shuffleport` (see `const_config.py` for more information). These scripts can be run as follows (before launching CorrelX):

```
#NNODES: number of nodes in the deployment
NOW=$(date +%Y%m%d_%H%M%S)

srun -N $NNODES -n $NNODES ~/correlx/sh/get_used_ports.sh $NOW ~/correlx/other/used_ports
cat ${USED_PORTS_FOLDER}/used_ports_${NOW}_* > ~/correlx/other/used_ports/up_$NOW.txt
FORCED_PARAMS=${FORCED_PARAMS}`python get_params_ports.py ~/correlx/other/used_ports/up_$NOW.txt`
```

SSH configuration issues An ssh issue may be reported, or CorrelX may stop during the following setup stage:

```
Copying Conf - first node files to nodes...
 Nodes:                  debiantd
 Username:               cxuser
 Conf - first node source dir:  /home/cxuser/correlx/conf/debiantd/
 Conf - first node files:
            core-site.xml hdfs-site.xml mapred-site.xml yarn-site.xml masters
 Conf - first node destination dir:
            /home/cxuser/correlx/conf/debiantd/etc_hadoop_debiantd/
 Fixed .py files:        0
 Execution permission:  N/A
```

If so, or if manual confirmation is requested for every node in the list, refer to §8.2.2, step 4.

The script `key_mgmt.sh` is provided as a template to set up ssh connection; it may need to be modified, depending on your cluster configuration:

```
FIRST_NODE=`scontrol show hostname $LIST_OF_NODES|head -1`
OTHER_NODES=`scontrol show hostname $LIST_OF_NODES|tail -n+2|paste -d, -s`
ALL_NODES=`scontrol show hostname $LIST_OF_NODES|paste -d, -s`
if [ -z "$OTHER_NODES" ] ; then
   PARAMS_NODES="-w $FIRST_NODE"
else
   PARAMS_NODES="-w $FIRST_NODE -x $OTHER_NODES"
fi

mkdir ${CONF_FOLDER}/${FIRST_NODE}
echo ${ALL_NODES} > ${CONF_FOLDER}/${FIRST_NODE}/hosts_${FIRST_NODE}
srun -n 1 ${PARAMS_NODES} ~/correlx/sh/key_mgmt.sh `cat $CONF_FOLDER/$FIRST_NODE/\
     hosts_$FIRST_NODE`~/.ssh/known_hosts
```

10.2 Node Issues During Correlation

Issues with the worker nodes may cause their containers to fail. Hadoop will seamlessly restart the failed containers on other nodes, but in the long term it may be useful to track down the issue to detect potential problems in the nodes.

Some errors may also stop the whole processing. We provide the following two examples related to memory and CPU capacity.

Memory Issue Here is an example of debugging an error, such as the following extract from the main log:

```
Exception from container-launch.
Container id: container_1480457061096_0001_01_000003
Exit code: 1
Stack trace: ExitCodeException exitCode=1:
      at org.apache.hadoop.util.Shell.runCommand(Shell.java:582)
      ...
```

Check the logs for `container_1480457061096_0001_01_000003`.

```
ls correlx/logs/h_userlogs
  application_1480457061096_0001

ls correlx/logs/h_userlogs/application_1480457061096_0001/container_1480457061096_0001_01_000003
  stderr  stdout  syslog

vi correlx/logs/h_userlogs/application_1480457061096_0001/container_1480457061096_0001_01_000003/stderr
  Java HotSpot(TM) 64-Bit Server VM warning: INFO: os::commit_memory(0x00000000bcf71000, 524292096, 0) \
  failed; error='Cannot allocate memory' (errno=12)
```

This error was caused by more memory being configured than is actually available in the system.

CPU Issue This issue may be experienced if running CorrelX in cluster mode on a single machine with one CPU virtual core, and may cause the running processes to terminate and the current ssh/login session to close:

```
Connection closed by remote host
```

If running on a virtual machine, increase the number of CPU cores assigned to it and restart.

10.3 Hadoop Issues

Issues related to Hadoop are also typically reported in the main log, and are typically associated with resource allocation issues.

Insufficient Memory Insufficient memory issues may manifest as in the following example in the main log:

```
[...]
16/11/06 01:04:33 INFO mapreduce.Job:  map 100% reduce 43%
16/11/06 01:06:11 INFO mapreduce.Job:  map 100% reduce 42%
16/11/06 01:08:52 INFO mapreduce.Job:  map 100% reduce 43%
[...]
16/11/06 01:10:36 INFO mapreduce.Job: Task Id : attempt_1478406884001_0001_r_000074_0, Status : FAILED
Container [pid=18081,containerID=container_1478406884001_0001_01_047550] is running beyond physical
       memory limits. Current usage: 4.5 GB of 4 GB physical memory used; 6.7 GB of 10 GB virtual
       memory used. Killing container.
[...]
```

This error corresponds to a resource allocation issue in which the reducer container runs out of memory. The solution is to increase the container memory in the associated parameters in the CorrelX configuration file (forced parameters `container*`; see `const_config` for more details).

The Hadoop nodemanager logs provide periodic feedback on the running containers and their associated resources. For example:

```
[...]
2016-11-19 14:10:45,498 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
    ContainersMonitorImpl: Memory usage of ProcessTree 6075 for container-id container_1479582225084_
    0001_01_000223: 1.2 GB of 4 GB physical memory used; 6.1 GB of 10 GB virtual memory used
2016-11-19 14:10:45,527 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
    ContainersMonitorImpl: Memory usage of ProcessTree 5928 for container-id container_1479582225084_
    0001_01_000063: 1.2 GB of 4 GB physical memory used; 6.1 GB of 10 GB virtual memory used
2016-11-19 14:10:45,578 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
    ContainersMonitorImpl: Memory usage of ProcessTree 6452 for container-id container_1479582225084_
    0001_01_000343: 1.2 GB of 4 GB physical memory used; 6.1 GB of 10 GB virtual memory used
[...]
```

Insufficient Storage Storage issues may manifest as in the following example from the nodemanager logs:

```
[...]
2016-11-02 13:08:09,136 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
     ContainersMonitorImpl: Memory usage of ProcessTree 24517 for container-id container_1478106118687_
     0001_01_000186: -1B of 4 GB physical memory used; -1B of 10 GB virtual memory used
2016-11-02 13:08:09,164 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
     ContainersMonitorImpl: Memory usage of ProcessTree 24309 for container-id container_1478106118687_
     0001_01_000146: -1B of 4 GB physical memory used; -1B of 10 GB virtual memory used
2016-11-02 13:08:12,208 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.
     ContainersMonitorImpl: Memory usage of ProcessTree 24315 for container-id container_1478106118687_
     0001_01_000104: -1B of 4 GB physical memory used; -1B of 10 GB virtual memory used
[...]
```

Make sure that there is enough storage available at the nodes; see §8.2.5 for more details.

10.4 Application Issues

Issues related to the application (map/reduce) typically generate the following error in the main log:

```
[...]
16/11/04 14:02:46 INFO impl.YarnClientImpl: Submitted application application_1478282497325_0001
16/11/04 14:02:46 INFO mapreduce.Job: The url to track the job:
                   http://node118:8088/proxy/application_1478282497325_0001/
16/11/04 14:02:46 INFO mapreduce.Job: Running job: job_1478282497325_0001
16/11/04 14:03:00 INFO mapreduce.Job: Job job_1478282497325_0001 running in uber mode : false
16/11/04 14:03:00 INFO mapreduce.Job:  map 0% reduce 0%
16/11/04 14:03:06 INFO mapreduce.Job: Task Id : attempt_1478282497325_0001_m_000000_0, Status : FAILED
Error: java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1
```

```
      at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:322)
      at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:535)
      at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:130)
      at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:61)
      at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:34)
      at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:453)
      at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
      at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
      at java.security.AccessController.doPrivileged(Native Method)
      at javax.security.auth.Subject.doAs(Subject.java:415)
      at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1698)
      at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)
[...]
```

Note that the application id for this job is `application_1478282497325_0001`, as reported previously. Based on the location specified in the configuration for the application logs (§13.5):

```
[Hadoop-yarn]
yarn.nodemanager.log-dirs:            /nobackup1b/users/ajva/h_userlogs
...
```

All errors can be listed as follows:

```
cat /nobackup1b/users/ajva/h_userlogs/application_1478282497325_0001/\
   container_*/stderr|grep -v bash|grep -v appender|grep -v log4j
```

The output will include the Python trace, for example:

```
[...]
Traceback (most recent call last):
  File "/home/ajva/hadoop/hadoop-2.7.3/msvf.py", line 1691, in <module>
    main()
  File "/home/ajva/hadoop/hadoop-2.7.3/msvf.py", line 948, in main
    current_file_name = get_current_filename()
  File "/home/ajva/hadoop/hadoop-2.7.3/msvf.py", line 193, in get_current_filename
    file_name = str(os.environ.get(MAP_INPUT_FILE))
NameError: global name 'MAP_INPUT_FILE' is not defined
[...]
```

Part III
Development

11 Coding Style

This section contains coding style guidelines.

11.1 File Naming

A library for the mapper/reducer and its associated constants file should differ only by a prefix: `lib_` for the library, and `const_` for the constants.

11.2 Standards

The documentation should follow the format specified in the Sphinx's [47] numpydoc [48] extension. A simplistic approach has been selected where three sections should be provided: `Parameters` (with the arguments of the function), `Returns` (with the output of the function), and `Notes` (with extra information). For more details, see §11.4.

11.3 File Headers

File headers are in the following format (copy license from §A):

```
#
# < COPY LICENSE HERE ! >
#
#
#------------------------------
#------------------------------
#Project: CorrelX.
#File: <filename>.
#Author: <Name Surname> (<email>), <Name Surname> (<email>), ...
#Description:
'''
 <Short description of the functionality provided in the library.>
'''
```

Inclusion of the sections `Assumptions`, `Approximations`, `Configuration`, `Debugging`, `Known Issues`, and `TO DO` is encouraged (§11.4).

11.4 Docstrings

Docstrings for the prototype version generally provide extensive information on functions. This approach is encouraged, as is keeping track of changes in the sources into the docstrings. Docstrings follow this format:

```
def example_function(var_1,var_2):
    '''
    Short explanation on what the function does.

    Parameters
    ----------
```

```
    var_1 : type_1
       description for variable 1.
    var_2 : type_2
       description for variable 2.

   Returns
   -------
    out : type_out
         description for the returned variable.

   Notes
   -----
   |
   | **Procedure:**
   |
   |  Description of the main steps done in the function.
   |
   |
   | **Assumptions:**
   |
   |  List of all assumptions: covered cases, expected range of data, etc.
   |
   |
   | **Approximations:**
   |
   |  List of all approximations: precision reductions, disabled checks, etc.
   |
   |
   | **Configuration:**
   |
   |  Description of all constants related to operation linked in the function.
   |
   |
   | **Debugging:**
   |
   |  Description of all constants related to debugging linked in the function.
   |
   |
   | **Known Issues:**
   |
   |  List of all issues.
   |
   |
   | **TO DO:**
   |
   |  List of all pending work for the function.
   |
   ‘’‘’‘’
```

Some of the provided development tools rely on these conventions to manage the source files. For more information, see §16.2 (assumptions), §16.3 (approximations), §18.1 (known issues), and §19.1 (to-do's). Also note the vertical bar `|` for keeping the line breaks in the generated output.

11.5 Generation of Source Documentation

This section is a short guide on how to generate Sphinx/numpydoc documentation.

Requirements

1. Install the required modules:

```
sudo apt-get install sphinx-common
sudo apt-get install python-pip
sudo pip install numpydoc
```

Procedure

1. Create the required folder structure:

```
mkdir correlx/docs
cd correlx/docs
sphinx-quickstart -p p -a a -v v -q
```

2. Use the provided script to create the CorrelX configuration files for Sphinx, then overwrite the default configuration files. You may need to replace `/home/cxuser/correlx` with the path to your CorrelX installation location:

```
python /home/cxuser/correlx/sh/gen_doc_conf.py /home/cxuser/correlx/src
cp index.rst_auto index.rst
cp conf.py_auto conf.py
```

3. Build the documentation:

```
sphinx-build -b html . build && make html
```

Results

The generated documentation can be accessed by opening `correlx/docs/build/index.html` in a web brower, as shown in Figures 9 and 10.

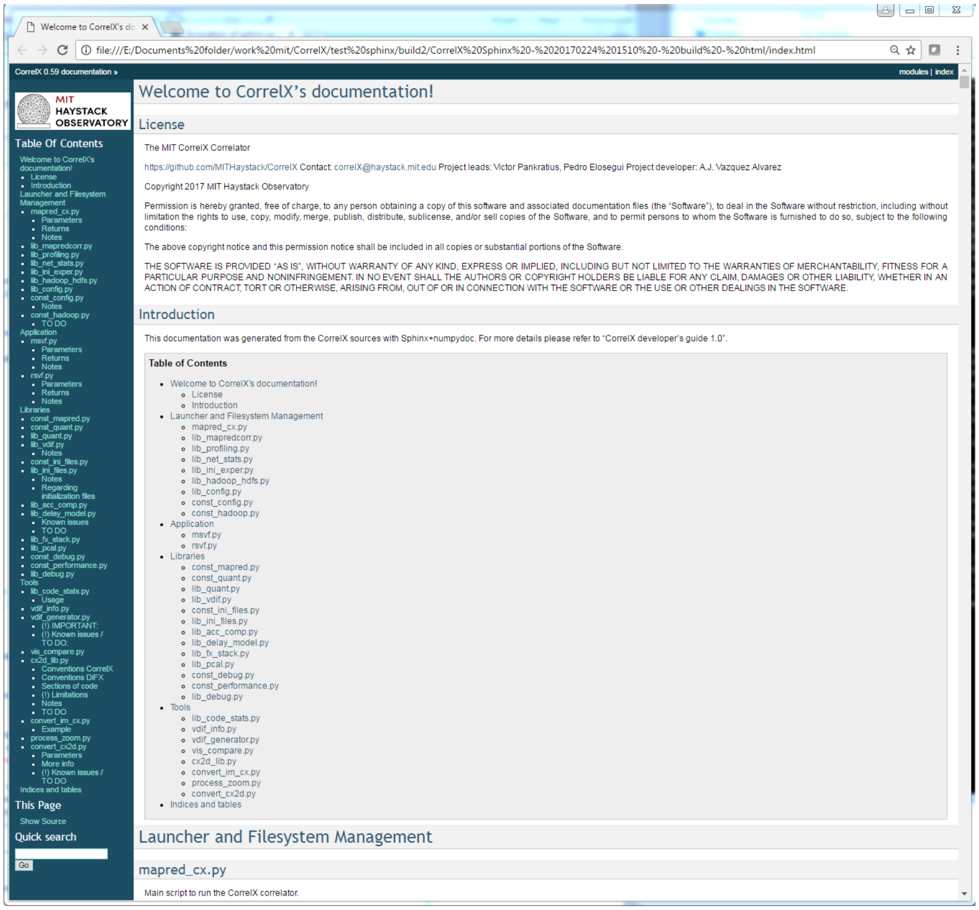


Figure 9: CorrelX Sphinx/numpydoc documentation header.

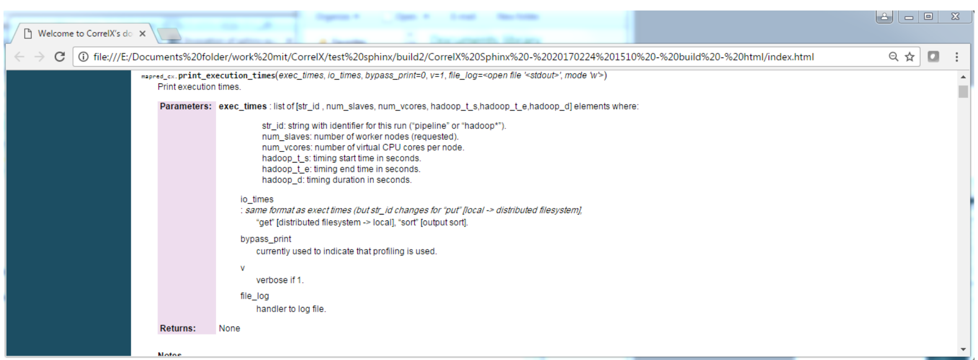


Figure 10: CorrelX Sphinx/numpydoc documentation example.

12 Basic Development

This section contains guidelines for the occasional developer.

12.1 Customizing the User Interfaces

In some scenarios, it is useful to override certain parameters in the CorrelX configuration file via the command-line interface. For details on this process, see Table 12 in section 6.2.

All the currently available parameters are listed in `const_config`, overridden in `lib_config.override_configuration_parameters()` and read in `lib_config.get_configuration()`. This list is displayed to the user after `python mapred_cx --help-parameters` (see `mapred_cx.args.help_parameters`). A comma-separated list of these parameters is passed as the `-f` argument (e.g., `adjm=1,adjr=1`). The following examples demonstrate adding a new parameter.

Example: Adding a Hadoop Override Parameter

1. Add new constant in `const_config` (list of `C_ARG_*`).
2. Check/add constants for hadoop configuration files in `const_hadoop.py` (if required).
3. Add parameter reading in `lib_config.get_configuration()` (if required).
4. Add option in if-structure in `lib_config.override_configuration_parameters()`.

Example: Adding a CorrelX Configuration Parameter

1. Add new constant in `const_config` (list of `C_CONF_*`).
2. Add parameter reading in `lib_config.get_configuration()`.
3. Add parameter in output interface in `lib_config.get_configuration()` and `mapred_cx` call.

12.2 Modifying the Map-Reduce Interface

Modification of the MapReduce logic is discouraged, especially the management of the key in the lines/records. But because modifying the metadata may be required in some applications, here we describe the procedure.

Example: Adding Metadata Fields For adding a parameter in the metadata of the MapReduce records:

1. Increase `const_mapred.META_LEN`.
2. Add the new parameter in the list of medata in `const_mapred` (after `META_LEN`).
3. Add parameter in mapper (`msvf.get_pair_str()`).
4. Add parameter in reducer (`rsvf.extract_params_split()`).

Important! The format conversion toolkit (`cx2d_lib`) reads the `const_hadoop.META_LEN` parameter. If backward compatibility is required, consider keeping a copy of the legacy `const_hadoop` and referencing it properly in `cx2d_lib.CX_IMPORT_CONST_MAPRED` or `cx2d_lib.CX_OVERRIDE_META_LEN`.

12.3 Writing a Custom Partitioner

We have shown how to compile a NoHash partitioner (in §8.2.3). Adding additional partitioners would require new references in `const_hadoop.C_H_INLINE_DEFAULT_PARITIONER` and propagating the changes into `lib_mapredcorr`, `const_config`, and `lib_config`.

12.4 Writing a Plugin for the Mapper

For using custom functions from a library `lib_custom.py`:

1. Copy the custom library to the path specified in `[Files] Src directory` field in the CorrelX configuration file (see Table 4).
2. Add `lib_custom.py` to the comma-separated list in the `[Files] Dependencies` field in the CorrelX configuration file (see Table 4).
3. Add `import` for the custom library in `msvf`.
4. Replace the call to an existing function with a call to the custom function (almost every existing function provides detailed interfacing documentation).

Example: Custom Frame Reader Here we provide a summary of the required steps to add a custom frame reader into the mapper. For more details, refer to the documentation provided in `msvf.read_frame()`, and for an example, see `lib_vdif.read_vdif_frame()`.

1. Add custom library in the CorrelX configuration file as described previously.
2. Add new format and version in `const_ini_media`.
3. Add `import` for the custom library in `msvf`.
4. Add call in the if-structure in `msvf.read_frame()`.

12.5 Writing a Plugin for the Reducer

Follow the same steps as for the mapper, described in §12.4.

Example: Custom DFT Implementation Here we provide a quick example on how to modify the FX library to use a custom DFT library (instead of the one used by default `scipy.fftpack`). For details on the current implementation, conventions, etc. of the FX correlation library please refer to `lib_fx_stack.compute_fx_for_all()`.

1. Add custom library in the CorrelX configuration file as described above.
2. Add `import` for the custom library in `lib_fx_stack`.
3. Replace call to `lib_fx_stack.window_and_fft()` for custom function. Documentation on the interface is provided in the source.

Alternatively, it is possible to add the calls to the DFT functions in `lib_fx_stack.window_and_fft()`. In fact, an implementation for an alternative implementation is available for pyFFTW that may serve as an example; refer to the source for details. Details on the configuration for this module are provided in §16.4.1.

12.6 Modifying the Format Converter

In case of changes in the format of DiFX configuration/output files, update the constants sections (`## DiFX/version`) in the format conversion toolkit library (`cx2d_lib`).

12.7 Modifying the VDIF Generator

Example: Custom Signal The VDIF file generator introduced in §9.8.2 can easily be modified to generate a custom signal:

1. Add `import` for the custom library in `vdif_generator`.
2. Uncomment control if-structure and add call to custom signal generator near to `ymulti = generate_multi_sine_wave()` call in `generate_vdif()`.
3. Replace calls to `simple_quantizer()` with those of the custom quantizer as required.

13 Debugging Tools

In this section, we provide some guidelines and tools to help the developer with the debugging. An incremental testing approach is encouraged: when doing modifications on the source files, first test in pipeline mode with a small dataset, and then in parallel mode (cluster/cloud).

13.1 Debugging the Mapper

A set of tools is provided to debug the mapper in `const_debug` (constants) and `lib_debug` (functions). Use the constants file to change the configuration. These tools generate tabulated lines starting with the string `zM` to indicate a logging line from the mapper.

Sample Alignment Activate the `const_debug.DEBUG_ALIGN` and `const_debug.SILENT_OUTPUT` flags to enable the mapper debugging mode. The first flag generates additional information for each processed frame, and the second flag suppresses the typical output to simplify debugging. A typical mapper debugging output (after `DEBUG_ALIGN=1` and `SILENT_OUTPUT=1`) looks like the following example:

(next page)

```
Debugging alignment computations msvf.
 st:    Station
 id:    Station id (stations.ini)
 tsf:   Total sample components per channel in frame)
 t:     Data type: 0 for real, 1 for complex
 tsff:  Total samples per channel in frame
 s_fr:  Seconds of frame in VDIF header
 n_fr:  Number of frame in VDIF header
 s_adj: Seconds corresponding to the first sample in this frame (based on VDIF header info)
 r:     Delay corresponding to this frame
 s_adj_r: Seconds corresponding to the first sample in this frame adjusted with the delay
 rel_pol: Relative position for this frame into the accumulation period
 ai:    Accumulation block index (i_front)
 acc:   Accumulation block index (accu_block)
 n_fr_ne: Number of frame adjusted with offset due to delay (may be negative)
 f_fr_adj: Number of frame adjusted with offset due to delay (can not be negative)
 sh:    Integer shift to be applied in this accumulation period
 sh_fr: Integer shift to be applied to the first frame of the accumulation period
 [frac|4]: Fractional sample correction for this accumulation block (showing only 4 decimal digits)
 pf:    Process frame, 1 if frame will be processed
 af:    Aligned frame, 0 if it corresponds to previous accumulation block (which has different delay parameters)
 pp:    Previously printed, number lines printed in previous round
 [N]:   Number of sample components to be potentially printed in this round
 [s]:   Accumulation period corresponding to the generated line
 [ofi]: Offset iterator, offset to the sample components to be taken from the frame
 [ofs]: Offset signal, offset to the first sample number to be displayed in the geenrated line
 [cs]:  Chunk size (by default same as N)
 [acc]: Accumulation period (-1 if samples will be discarded)
 [sf]: Superframe mode (0 deafult, 1 if grouping samples for multiple frames into one line
 [sf_id]: Superframe id +1 (incremented before reporting)
zM a   st   id    tsf  t   tsff      s_fr  n_fr        s_adj                 r            s_adj_r  rel_pos   ai      acc  n_fr_ne  n_fr_adj       sh sh_fr  [fra|4] pf af pp       [N]     [s]   [ofi]    [ofs]     [cs]   [acc]  [sf][sf_id]
zM a   BR    0  20000  0  20000     30000     0  30000.0000000   0.00308275449856   29999.9969172      -1   -1       -1      -39     12761   789185  9185   0.1516  1  0 -1   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     0  30000.0000000   0.00308275449856   29999.9969172      -1   -1       -1      -39     12761   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     0  30000.0000000   0.00308275449856   29999.9969172      -1   -1       -1      -39     12761   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     0  30000.0000000   0.00308275449856   29999.9969172      -1   -1       -1      -39     12761   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     1  30000.0000781   0.00308275449856   29999.9969954      -1   -1       -1      -38     12762   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     1  30000.0000781   0.00308275449856   29999.9969954      -1   -1       -1      -38     12762   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     1  30000.0000781   0.00308275449856   29999.9969954      -1   -1       -1      -38     12762   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     1  30000.0000781   0.00308275449856   29999.9969954      -1   -1       -1      -38     12762   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     2  30000.0001563   0.00308275449856   29999.9970735      -1   -1       -1      -37     12763   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000     2  30000.0001563   0.00308275449856   29999.9970735      -1   -1       -1      -37     12763   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
...
zM a   BR    0  20000  0  20000     30000    37  30000.0028906   0.00308275449856   29999.9998079      -1   -1       -1       -2     12798   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    37  30000.0028906   0.00308275449856   29999.9998079      -1   -1       -1       -2     12798   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    37  30000.0028906   0.00308275449856   29999.9998079      -1   -1       -1       -2     12798   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    37  30000.0028906   0.00308275449856   29999.9998079      -1   -1       -1       -2     12798   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    38  30000.0029687   0.00308275449856   29999.9998860      -1   -1       -1       -1     12799   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    38  30000.0029687   0.00308275449856   29999.9998860      -1   -1       -1       -1     12799   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    38  30000.0029687   0.00308275449856   29999.9998860      -1   -1       -1       -1     12799   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    38  30000.0029687   0.00308275449856   29999.9998860      -1   -1       -1       -1     12799   789185  9185   0.1516  1  0  0   [20000L]   [-1]     [0][4954187L][20000L]    [-1]    0     0
zM a   BR    0  20000  0  20000     30000    39  30000.0030469   0.00308275449856   29999.9999641       0    0        0        0         0   789185  9185   0.1516  1  1  0   [10815L]    [0] [9185L][4983372][10815L]     [0]    0    0
zM a   BR    0  20000  0  20000     30000    39  30000.0030469   0.00308275449856   29999.9999641       0    0        0        0         0   789185  9185   0.1516  1  1  1   [10815L]    [0] [9185L][4983372][10815L]     [0]    0    0
zM a   BR    0  20000  0  20000     30000    39  30000.0030469   0.00308275449856   29999.9999641       0    0        0        0         0   789185  9185   0.1516  1  1  1   [10815L]    [0] [9185L][4983372][10815L]     [0]    0    0
zM a   BR    0  20000  0  20000     30000    39  30000.0030469   0.00308275449856   29999.9999641       0    0        0        0         0   789185  9185   0.1516  1  1  1   [10815L]    [0] [9185L][4983372][10815L]     [0]    0    0
zM a   BR    0  20000  0  20000     30000    40  30000.0031250   0.00308275449856   30000.0000422       1    0        0        1         1   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][4994187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    40  30000.0031250   0.00308275449856   30000.0000422       1    0        0        1         1   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][4994187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    40  30000.0031250   0.00308275449856   30000.0000422       1    0        0        1         1   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][4994187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    40  30000.0031250   0.00308275449856   30000.0000422       1    0        0        1         1   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][4994187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    41  30000.0032031   0.00308275449856   30000.0001204       2    0        0        2         2   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][5014187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    41  30000.0032031   0.00308275449856   30000.0001204       2    0        0        2         2   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][5014187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    41  30000.0032031   0.00308275449856   30000.0001204       2    0        0        2         2   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][5014187L][20000L]     [0]    0     0
zM a   BR    0  20000  0  20000     30000    41  30000.0032031   0.00308275449856   30000.0001204       2    0        0        2         2   789185  9185   0.1516  1  1  1   [20000L]    [0]      [0][5014187L][20000L]     [0]    0     0
...
```

The meaning of each field is described in the header of the output. Basically, this output provides one line per frame read, showing multiple fields, including the name of the station `st`, its initial timestamp `s_adj`, its actual timestamp (taking into account the delay) `s_adj_r`, the relative position of this frame in the whole stream (`n_fr_adj`), the offset for the first sample to be read (`ofi`), the number of generated lines for the previous frame [i.e., the number of channels per thread] (`pp`), the number of sample components read (`N`), its associated accumulation period (`acc`), etc.

The previous example corresponds to the media file presented in §5.2.5 (VDIF file with four threads, real samples).

The following is an example of the output for a multi-channel single thread VDIF file with complex samples (note the change in `pp`).

```
Debugging alignment computations msvf.
 st:    Station
 id:    Station id (stations.ini)
 tsf:   Total sample components per channel in frame)
 t:     Data type: 0 for real, 1 for complex
 tsff:  Total samples per channel in frame
 s_fr:  Seconds of frame in VDIF header
 n_fr:  Number of frame in VDIF header
 s_adj: Seconds corresponding to the first sample in this frame (based on VDIF header info)
 r:     Delay corresponding to this frame
 s_adj_r: Seconds corresponding to the first sample in this frame adjusted with the delay
 rel_pol: Relative position for this frame into the accumulation period
 ai:    Accumulation block index (i_front)
 acc:   Accumulation block index (accu_block)
 n_fr_ne: Number of frame adjusted with offset due to delay (may be negative)
 f_fr_adj: Number of frame adjusted with offset due to delay (can not be negative)
 sh:    Integer shift to be applied in this accumulation period
 sh_fr: Integer shift to be applied to the first frame of the accumulation period
 [frac|4]: Fractional sample correction for this accumulation block (showing only 4 decimal digits)
 pf:    Process frame, 1 if frame will be processed
 af:    Aligned frame, 0 if it corresponds to previous accumulation block (which has different delay parameters)
 pp:    Previously printed, number lines printed in previous round
 [N]:   Number of sample components to be potentially printed in this round
 [s]:   Accumulation period corresponding to the generated line
 [ofi]: Offset iterator, offset to the sample components to be taken from the frame
 [ofs]: Offset signal, offset to the first sample number to be displayed in the geenrated line
 [cs]:  Chunk size (by default same as N)
 [acc]: Accumulation period (-1 if samples will be discarded)
 [sf]: Superframe mode (0 deafult, 1 if grouping samples for multiple frames into one line
 [sf_id]: Superframe id +1 (incremented before reporting)
zM a   st   id   tsf  t  tsff    s_fr  n_fr       s_adj                 r          s_adj_r  rel_pos  ai      acc  n_fr_ne  n_fr_adj      sh  sh_fr  [fra|4] pf af pp      [N]     [s]  [ofi]   [ofs]    [cs]   [acc] [sf][sf_id]
zM a   K2    1  2048  1  1024   68255     0  68255.0000000  0.00116935370784  68254.9988306      -1  -1       -1      -36     31214   74838  1110   0.3187  1  0 -1   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     1  68255.0000320  0.00116935370784  68254.9988626      -1  -1       -1      -35     31215   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     2  68255.0000640  0.00116935370784  68254.9988946      -1  -1       -1      -34     31216   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     3  68255.0000960  0.00116935370784  68254.9989266      -1  -1       -1      -33     31217   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     4  68255.0001280  0.00116935370784  68254.9989586      -1  -1       -1      -32     31218   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     5  68255.0001600  0.00116935370784  68254.9989906      -1  -1       -1      -31     31219   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     6  68255.0001920  0.00116935370784  68254.9990226      -1  -1       -1      -30     31220   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     7  68255.0002240  0.00116935370784  68254.9990546      -1  -1       -1      -29     31221   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     8  68255.0002560  0.00116935370784  68254.9990866      -1  -1       -1      -28     31222   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255     9  68255.0002880  0.00116935370784  68254.9991186      -1  -1       -1      -27     31223   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    10  68255.0003200  0.00116935370784  68254.9991506      -1  -1       -1      -26     31224   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    11  68255.0003520  0.00116935370784  68254.9991826      -1  -1       -1      -25     31225   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    12  68255.0003840  0.00116935370784  68254.9992146      -1  -1       -1      -24     31226   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    13  68255.0004160  0.00116935370784  68254.9992466      -1  -1       -1      -23     31227   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    14  68255.0004480  0.00116935370784  68254.9992786      -1  -1       -1      -22     31228   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    15  68255.0004800  0.00116935370784  68254.9993106      -1  -1       -1      -21     31229   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    16  68255.0005120  0.00116935370784  68254.9993426      -1  -1       -1      -20     31230   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    17  68255.0005440  0.00116935370784  68254.9993746      -1  -1       -1      -19     31231   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    18  68255.0005760  0.00116935370784  68254.9994066      -1  -1       -1      -18     31232   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    19  68255.0006080  0.00116935370784  68254.9994386      -1  -1       -1      -17     31233   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    20  68255.0006400  0.00116935370784  68254.9994706      -1  -1       -1      -16     31234   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    21  68255.0006720  0.00116935370784  68254.9995026      -1  -1       -1      -15     31235   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    22  68255.0007040  0.00116935370784  68254.9995346      -1  -1       -1      -14     31236   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    23  68255.0007360  0.00116935370784  68254.9995666      -1  -1       -1      -13     31237   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    24  68255.0007680  0.00116935370784  68254.9995986      -1  -1       -1      -12     31238   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    25  68255.0008000  0.00116935370784  68254.9996306      -1  -1       -1      -11     31239   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    26  68255.0008320  0.00116935370784  68254.9996626      -1  -1       -1      -10     31240   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    27  68255.0008640  0.00116935370784  68254.9996946      -1  -1       -1       -9     31241   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    28  68255.0008960  0.00116935370784  68254.9997266      -1  -1       -1       -8     31242   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    29  68255.0009280  0.00116935370784  68254.9997586      -1  -1       -1       -7     31243   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    30  68255.0009600  0.00116935370784  68254.9997906      -1  -1       -1       -6     31244   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    31  68255.0009920  0.00116935370784  68254.9998226      -1  -1       -1       -5     31245   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    32  68255.0010240  0.00116935370784  68254.9998546      -1  -1       -1       -4     31246   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    33  68255.0010560  0.00116935370784  68254.9998866      -1  -1       -1       -3     31247   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    34  68255.0010880  0.00116935370784  68254.9999186      -1  -1       -1       -2     31248   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    35  68255.0011200  0.00116935370784  68254.9999506      -1  -1       -1       -1     31249   74838  1110   0.3187  1  0  0   [2048L]   [-1]     [0][994916L] [2048L]    [-1]    0     0
zM a   K2    1  2048  1  1024   68255    36  68255.0011520  0.00116935370784  68254.9999826       0   0        0        0         0   74838  1110   0.3187  1  1  0    [938L]    [0] [1110L][998074]  [938L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    37  68255.0011840  0.00116935370784  68255.0000146       1   0        0        1         1   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][999012L] [2048L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    38  68255.0012160  0.00116935370784  68255.0000466       2   0        0        2         2   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][1001060L] [2048L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    39  68255.0012480  0.00116935370784  68255.0000786       3   0        0        3         3   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][1003108L] [2048L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    40  68255.0012800  0.00116935370784  68255.0001106       4   0        0        4         4   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][1005156L] [2048L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    41  68255.0013120  0.00116935370784  68255.0001426       5   0        0        5         5   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][1007204L] [2048L]    [0]    0     0
zM a   K2    1  2048  1  1024   68255    42  68255.0013440  0.00116935370784  68255.0001746       6   0        0        6         6   74838  1110   0.3187  1  1 64   [2048L]    [0]     [0][1009252L] [2048L]    [0]    0     0
...
```

Frame Reading Frame reading should be debugged independently (see for example §9.8.1), but additional flags (`const_debug. VERBOSE_MAPPER_IO`) and `const_debug.SHOW_ERRORS`) are provided for debugging during processing.

Bypassing the Reducer Although running in pipeline mode provides access to all intermediate files (i.e., map ouput, sorted merged file) during the processing, this is not the case in Hadoop. The option to bypassing all reducer processing is available by activating `const_debug.BYPASS_REDUCER`. See §13.5 for more details.

13.2 Debugging the Reducer

A set of tools is provided to debug the reducer in `const_debug` and `lib_debug`. These tools generate tabulated lines starting with the string `zR` to indicate a logging line from the reducer. Note that `const_debug.SILENT_OUTPUT` should be set to 0.

Delay Computations In order to show tabulated information regarding the delays computed for each group of samples processed, enable `const_debug.DEBUG_DELAYS`. We show an example of the corresponding output (again for the dataset from §5.2.5) on page 83.

Samples Stacking As described in §15.3, once the reducer has read the lines for a set of station-polarizations, it stacks these samples with the previously stored samples. Enable `const_debug.DEBUG_DELAYS` to show debugging information for sample stacking. We show an example of the corresponding output (again for the dataset from §5.2.5) on page 84.

Fractional Sample Overflow Due to the delay rate, the fractional sample delay will change along the stream. Once this fractional delay passes the threshold to be considered an integer sample change, this needs to be updated. The mapper computes the delay for the first sample of the stream, and the reducer checks for "overflows" in the fractional sample correction, adding or removing one sample from the stream as required. This is checked at `lib_fx_stack.get_frac_over_ind()` and updated at `lib_fx_stack.fix_frac_over()`. For activating its associated debugging, enable `const_debug.DEBUG_FRAC_OVER`.

(next page)

Delay computation debugging example:

```
 Debugging delay computations rsvf.
 d/f:     Fringe rotation / fractional sample correction
 i:       Index to F1 (new samples)
 ref:     Index to F1_partial (stored samples)
 sp:      Station.polarization
 s_id:    First sample number
 N:       Number of samples
 n:       Length of timescale for calling compute_delay
 0:       First element of the timescale for calling compute_delay [s]
 -1:      First element of the timescale for calling compute_delay [s]
 s_off:   Time corresponding to the first sample [s]
 r[0][s]: Delay for first sample
 r[-1][s]:Delay for last sample
 delta_r: Rate for this interval (linear)
 fra_ini: Fractional sample correction (first one should match that in delays.ini)
 fra_now_full: Full fractional sample correction (including shift if applicable)
 fra_now: Fractional sample correction (between -0.5 and +0.5)
 diff:    Differential delay between acc periods
 C:       Computed (will be zero if taken rotation vectors from previous iteration)
 R:       Rotated (will be zero if no rotation has been applied)
zR d/f    i     ref      sp    s_id         N         n              0        -1              s_off            r[0] [s]          r[-1] [s]            delta_r             fra_ini        fra_now_full              fra_now           diff  C  R
zR
zR >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
zR          px-A.A-A.A-a-0-0-0-
zR >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
zR
zR pre-hsp: stpols [] with 10815,10815,20000,20000 samples, delta_n is  9185
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 10815,10815,20000,20000 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 30815,30815,40000,40000 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 50815,50815,60000,60000 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 70815,70815,80000,80000 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 90815,90815,100000,100000 samples, delta_n is  9185
zR d      0       0     0.0   4983372     81920         2              0.0  0.00032      0.019466296875    0.00308275449856    0.00308275444751  -1.59533328117e-07      0.151631360059                                       0.0  1  1
zR d      1       1     0.1   4983372     81920         2              0.0  0.00032      0.019466296875    0.00308275449856    0.00308275444751  -1.59533328117e-07      0.151631360059                                       0.0  0  1
zR d      2       2     1.0   4983372     81920         2              0.0  0.00032      0.019466296875                 0.0                 0.0                 0.0                 0.0                                       0.0  1  0
zR d      3       3     1.1   4983372     81920         2              0.0  0.00032      0.019466296875                 0.0                 0.0                 0.0                 0.0                                       0.0  0  0
zR f      0       0     0.0   4983372     81920         1  0.019466296875      0.0                  0    0.00308275449856                 [0]                 [0]      0.151631360059       789185.151632      0.151631610817           0
zR f      0       0     0.0   4983372     81920 C,R >>>>                                                                                                                                                                      1  1
zR f      1       1     0.1   4983372     81920 C,R >>>>                                                                                                                                                                      0  1
zR f      2       2     1.0   4983372     81920         1  0.019466296875      0.0                  0                 0.0                 [0]                 [0]                 0.0                 0.0                 0.0           0
zR f      2       2     1.0   4983372     81920 C,R >>>>                                                                                                                                                                      1  0
zR f      3       3     1.1   4983372     81920 C,R >>>>                                                                                                                                                                      0  0
zR oj                 0       mon
zR ---------------
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 28895,28895,38080,38080 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 48895,48895,58080,58080 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 68895,68895,78080,78080 samples, delta_n is  9185
zR pre-hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 20000,20000,20000,20000 samples, delta_n is  0
zR hsp: stpols ['0.0', '0.1', '1.0', '1.1'] with 88895,88895,98080,98080 samples, delta_n is  9185
zR d      0       0     0.0   5065292     81920         2              0.0  0.00032      0.019786296875    0.00308275444751    0.00308275439646  -1.59533336248e-07      0.151631360059                                       0.0  1  1
zR d      1       1     0.1   5065292     81920         2              0.0  0.00032      0.019786296875    0.00308275444751    0.00308275439646  -1.59533336248e-07      0.151631360059                                       0.0  0  1
zR d      2       2     1.0   5065292     81920         2              0.0  0.00032      0.019786296875                 0.0                 0.0                 0.0                 0.0                                       0.0  1  0
zR d      3       3     1.1   5065292     81920         2              0.0  0.00032      0.019786296875                 0.0                 0.0                 0.0                 0.0                                       0.0  0  0
zR f      0       0     0.0   5065292     81920         1  0.019786296875      0.0                  0    0.00308275444751                 [0]                 [0]      0.151631360059       789185.138563      0.138562640641           0
zR f      0       0     0.0   5065292     81920 C,R >>>>                                                                                                                                                                      1  1
zR f      1       1     0.1   5065292     81920 C,R >>>>                                                                                                                                                                      0  1
zR f      2       2     1.0   5065292     81920         1  0.019786296875      0.0                  0                 0.0                 [0]                 [0]                 0.0                 0.0                 0.0           0
zR f      2       2     1.0   5065292     81920 C,R >>>>                                                                                                                                                                      1  0
zR f      3       3     1.1   5065292     81920 C,R >>>>                                                                                                                                                                      0  0
zR oj                 0       mon
zR ---------------
...
```

The lines starting by `pre-hsp` and `hsp` correspond to reports from the main processing function (`lib_fx_stack.compute_fx_for_all()`) at its start and end respectively, reporting the number of samples available for each station-polarization in the reducer structures and in the processing structures).

For the rest of the lines, `d` indicates delay computations during fringe rotation, and `f` fractional sample correction. Note that computations are done only once per station (which can be changed by disabling `const_mapred.SAVE_TIME_ROTATIONS`), so computations are not repeated for alternative polarizations of the same station. This is reflected in the column `C`. If no rotation is done (zero delay), this is reflected in the column `R`.

Samples stacking debugging example:

```
 Debugging hstack rsvf.
 pc/f:         Phase calibration / FX processing
 ip:           Index to F1_partial (stored samples)
 Fip[ip]:      Station.polarization in F1_partial, input of the function
 Fipo[ip]:     Station.polarization in F1_partial, output of the function
 i:            Index to F_ind
 Fi[i]:        Station.polarization in F_ind
 hstack line:  number of samples in F1_partial joined to number of samples in F1, totals and stats where
                F_lti: list with last sample (l), total number of samples processed (t), invalid samples (i), and
                adjuted samples for each stream.
zR pc/f         ip   Fip[ip]  Fipo[ip]        i     Fi[i]
zR
zR >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
zR         px-A.A-A.A-a-0-0-0-
zR >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
zR
zR f             0       N/A      0.0        0       0.0
zR f             1       N/A      0.1        1       0.1
zR f             2       N/A      1.0        2       1.0
zR f             3       N/A      1.1        3       1.1
zR hstack:  F1p () U F1 (10815,10815,20000,20000) -> 10815,10815,20000,20000 with F_lti=[[4994187, 10815, 0, 0], [4994187, 10815, 0, 0], [5003372, 20000, 0, 0], [5003372, 20000, 0, 0]]
zR ------------------------------
zR f             0       0.0      0.0        0       0.0
zR f             1       0.1      0.1        1       0.1
zR f             2       1.0      1.0        2       1.0
zR f             3       1.1      1.1        3       1.1
zR hstack:  F1p (10815,10815,20000,20000) U F1 (20000,20000,20000,20000) -> 30815,30815,40000,40000 with F_lti=[[5014187, 30815, 0, 0], [5014187, 30815, 0, 0], [5023372, 40000, 0, 0], [5023372, 40000, 0, 0]]
zR ------------------------------
zR f             0       0.0      0.0        0       0.0
zR f             1       0.1      0.1        1       0.1
zR f             2       1.0      1.0        2       1.0
zR f             3       1.1      1.1        3       1.1
zR hstack:  F1p (30815,30815,40000,40000) U F1 (20000,20000,20000,20000) -> 50815,50815,60000,60000 with F_lti=[[5034187, 50815, 0, 0], [5034187, 50815, 0, 0], [5043372, 60000, 0, 0], [5043372, 60000, 0, 0]]
zR ------------------------------
zR f             0       0.0      0.0        0       0.0
zR f             1       0.1      0.1        1       0.1
zR f             2       1.0      1.0        2       1.0
zR f             3       1.1      1.1        3       1.1
zR hstack:  F1p (50815,50815,60000,60000) U F1 (20000,20000,20000,20000) -> 70815,70815,80000,80000 with F_lti=[[5054187, 70815, 0, 0], [5054187, 70815, 0, 0], [5063372, 80000, 0, 0], [5063372, 80000, 0, 0]]
zR ------------------------------
zR f             0       0.0      0.0        0       0.0
zR f             1       0.1      0.1        1       0.1
zR f             2       1.0      1.0        2       1.0
zR f             3       1.1      1.1        3       1.1
zR hstack:  F1p (70815,70815,80000,80000) U F1 (20000,20000,20000,20000) -> 90815,90815,100000,100000 with F_lti=[[5074187, 90815, 0, 0], [5074187, 90815, 0, 0], [5083372, 100000, 0, 0], [5083372, 100000, 0, 0]]
```

The column `pc/f` indicates whether these computations correspond to the samples for phase calibration or FX computations (note that the phase calibration accumulation window length will in general be different from the FX window length). These lines report the contents of the data structures to make sure that samples are stacked with the previous samples corresponding to this station (note that due to the integer sample delay correction done at the mapper, the relative order between stations [lines read by the reducer are sorted by the index of the first sample in that line] may change, so this should be taken into account when stacking samples).

The lines starting with `hstack` provide a summary of the number of samples currently stored for each station-polarization (`F1p`), those from the last set of read lines (`F1`), and those once the stacking has been completed (after `->`). Additional statistics are reported; see `lib_fx_stack.hstack_new_samples()` for more details.

13.3 Debugging the Map-Reduce Interface

A simple tool is provided for showing the all the fields in the header of a map-reduce line with labels for debugging. The following example was run on a IPython notebook [45]:

```
import cx2d_lib

cx_file="correlx/output/OUT_s0_v0.outpart1"
cx2d_lib.show_line_cx(cx_file,line_start=0,line_count=1)
```

which will return:

```
icx2d configuration:
-------------------
CX_IMPORT_CONST_MAPRED:  latest
CX_OVERRIDE_META_LEN:    -1
META_LEN:                28
Metadata fields from:    const_mapred.py

Lines:
------
Line id                  0
Key:                     px-A.A-A.A-a-0-0-0-f0.00000004983372.0-s0.1-
ST_POL:                  0.1
SHIFT_DELAY:             789185
FRAC_DELAY:              0.151631360059
ABS_DELAY:               0.016383533664
RATE_DELAY_0:            0.00308275760408
RATE_DELAY_1:            -1.595329085e-07
RATE_DELAY_2:            -1.07192608994e-11
RATE_DELAY_REF:          0.0
RATE_CLOCK_0:            0.0
RATE_CLOCK_1:            0.0
RATE_ZC_0:               0.0
RATE_ZC_1:               0.0
RATE_CLOCK_REF:          0.0
RATE_M_ONLY:             0.0
RATE_C_ONLY:             0.0
RATE_DIFF_FRAC:          0.0
NUM_SAMPLES:             10815
FS:                      256000000
BITS_PER_SAMPLE:         2
FIRST_SAMPLE:            4983372
DATA_TYPE:               r
NBINS_PCAL:              64
PCAL_FREQ:               0
CHANNEL_INDEX:           0
CHANNEL_FREQ:            86140000000.0
ACC_TIME:                0.32
ENCODING:                no
SIDEBAND:                U
Num. visibilities:       1
```

Note that the field labels are read directly from the source file `const_mapred.py`, and thus if the interface is changed, previous versions of this file should be kept to process these headers. This can be configured through `CX_IMPORT_CONST_MAPRED` in the format conversion toolkit library (`cx2d_lib`).

13.4 Debugging the Delay Model Library

The library that performs the delay computations provides some summary information for debugging. This information is written to a file `delays.ini_debug` in the same folder as the experiment definition files are read from. The file displays on a baseline basis (for every station w.r.t. the reference station), the delays and rates (following the same format as in `fourfit`). Note that the fourfit reference time should be updated to match the start of the accumulation period currently being checked. We show an example in the following lines:

```
D  st         t0 [s]     total delay [us]   clock [us]   clock rate [us/s]   total rate [us/s]   total accel [us/s/s]
D 30000.0     30000.0    -3082.66904893     0.166044333118   -2.86699814e-08    0.159532908381     2.14380785716e-05
D 30000.32    30000.32   -3082.6179973      0.166044323944   -2.86699814e-08    0.159539768566     2.14380785716e-05
D 30000.64    30000.64   -3082.56694347     0.166044314769   -2.86699814e-08    0.159546628751     2.14380785716e-05
D 30000.96    30000.96   -3082.51588745     0.166044305595   -2.86699814e-08    0.159553488936     2.14380785716e-05
[...]
D 30299.84  30299.84  -3033.87482948  0.166035736711 -2.86699814e-08  0.165922506274    2.11819951801e-05
```

13.5 Debugging a Hadoop Job

Hadoop provides different logs, that we classify considering the scope of the possible issues:

- **Main Hadoop log (standard output log)**: This log provides the highest level feedback to the developer, providing information for every step during the setup along with feedback during the MapReduce processing. This log is located at the path defined in the CorrelX configuration file in `[General] Log file`, which is generally the standard output. This is the first log to look at to detect possible issues.
- **Application logs**: A folder structure is generated by the worker nodes at the path defined in the CorrelX configuration file in `[Hadoop-yarn] yarn.nodemanager.log-dirs`. This folder structure contains as many folders as Hadoop jobs were launched, and for each job (folder), as many subfolders as containers were run. For each of this subfolders there are the files `stderr`, `stdout` and `syslog` with logging information. These are the logs to look at for issues related to the mapper and the reducer.
- **Hadoop entities logs**: The Hadoop entities introduced in §4.1.1 generate their own logs. These logs are in `<hadoop_path>/logs/*nodemanager*` (for the workers) and `<hadoop_path>/logs/*resourcemanager*` (for the master). It is recommended to copy the logs associated with each job with the rest of the logs. By default Hadoop keeps common logs for the nodes, independent of the jobs, but the CorrelX cloud scripts copy these logs and restart them for the next job. These logs are useful to detect issues at specific nodes and also issues related to resource allocation.

14 Preliminary Testing Results

The datasets currently being used for testing are described in Table 17. Note that all the test results provided herein were generated using CorrelX configuration files generated with the method described in §9.6, using `.im` files with intervals of 1 second. The same experiments were processed with intervals of 120 seconds with similar results (residuals within the displayed tolerances).

Table 17: Datasets currently being used for testing.

Dataset	Details	Data	bits	SSB	Additional	Results
ALMA	e16n07prepass (Az,Lm)	real	2	LSB	Zoom USB	§14.1
VGOS	v15327 (Gs,K2)	complex	2	LSB	Pcal tones	§14.2
VLBA	bm434aYs20_20 (Br,La)	real	2	USB	Zoom USB	§14.3
N/A	N/A	complex	2	USB	-	-

14.1 ALMA Dataset

We provide a comparison between the a priori delays computed in CorrelX (file `delays.ini_debug`; see §9.2), and those displayed by fourfit (for the corresponding values of `fourfit_reference_time` or FRT) in Table 18, for which we display only the values for the first and last accumulation period for brevity, as well as a comparison between some of the parameters displayed by fourfit for CorrelX and DiFX in Table 19.

Table 18: ALMA a priori delays comparison.

SW	Time	Apriori delay (usec)	Apriori clock (usec)	Apriori clockrate (us/s)	Apriori rate (us/s)	Apriori accel (us/s/s)
cx	22260.0	-3287.59569552	-3.57126438	-1.217e-06	0.273221157973	1.49821327492e-05
ff	22260.0	-3287.5956955	-3.571264	-1.217e-06	0.273221158022	1.49839468392e-05
cx	22269.0	-3285.13609842	-3.571275333	-1.217e-06	0.273355955511	1.49622110257e-05
ff	22269.0	-3285.13609842	-3.5712754	-1.217e-06	0.273355955511	1.49622110257e-05

Table 19: ALMA results summary (frequency group a:D, polarization pair LL).

Parameter	CorrelX	DiFX
Fringe quality	9	9
Error code	-	-
SNR	44.0	44.1
Int time	10.000	9.981
SBD	-0.010727	-0.010765
MBD	0.005124	0.005123
Fringe rate (Hz)	-0.011712	-0.011668
Resid mbdelay (usec)	5.12423E-03 +/- 6.5E-06	5.12308E-03 +/- 6.5E-06
Resid sbdelay (usec)	-1.07273E-02 +/- 2.0E-04	-1.07647E-02 +/- 2.0E-04
Resid phdelay (usec)	-1.60776E-06 +/- 3.2E-08	-1.61168E-06 +/- 3.2E-08
Resid rate (us/s)	-5.22479E-08 +/- 5.6E-09	-5.20516E-08 +/- 5.6E-09
Resid phase (deg)	-129.7 +/- 2.6	-130.1 +/- 2.6

For the list of forced parameters, the following configuration has been used:

```
ppb=5000,slowstart=1,vcores=14,containermemmap=4096,containerheapmap=3800,\
containermemred=4096,containerheapred=3800,containermemam=4096,containerheapam=3800,\
mapspernode=1,reducespernode=1,tasksperjvm=1,blocksize=1640000000,sortmem=800
```

14.2 VGOS dataset

We provide a comparison between the a priori delays computed in CorrelX (file `delays.ini_debug`; see §9.2), and those displayed by fourfit (for the corresponding values of `fourfit_reference_time` or FRT) in Table 20, for which we display only the values for the first and last accumulation period for brevity, as well as a comparison between some of the parameters displayed by fourfit for CorrelX and DiFX in Table 21. The control file `cf3528I_K.txt` was used in fourfit for both the CorrelX and DiFX results.

Table 20: VGOS a priori delays comparison.

SW	**Time**	**Apriori delay (usec)**	**Apriori clock (usec)**	**Apriori clockrate (us/s)**	**Apriori rate (us/s)**	**Apriori accel (us/s/s)**
cx	68255.0	1169.35220773	48.980059395	1.697e-06	0.100001042109	-5.60692114959e-06
ff	68255.0	1169.35220773	48.980059	1.697e-06	0.100001041706	-5.59933857797e-06
cx	68284.0	1172.24988126	48.980108608	1.697e-06	0.0998384373073	-5.61406447256e-06
ff	68284.0	1172.24988126	48.980110	1.697e-06	0.0998384374119	-5.6147692498e-06

Table 21: VGOS results summary (frequency group a:F, polarization pair XX).

Parameter	**CorrelX**	**DiFX**
Fringe quality	6	6
Error code	G	G
SNR	185.3	185.7
Int time	30.000	29.984
SBD	0.000891	0.000919
MBD	0.001154	0.001153
Fringe rate (Hz)	0.000834	0.000851
Ion TEC	-1.209	-1.226
Resid mbdelay (usec)	1.15393E-03 +/- 2.5E-06	1.15327E-03 +/- 2.5E-06
Resid sbdelay (usec)	8.91500E-04 +/- 9.3E-05	9.18500E-04 +/- 9.3E-05
Resid phdelay (usec)	-4.61278E-05 +/- 2.9E-07	-4.54463E-05 +/- 2.9E-07
Resid rate (us/s)	1.39000E-07 +/- 1.7E-08	1.41833E-07 +/- 1.6E-08
Resid phase (deg)	-99.6 +/- 3.2	-98.2 +/- 3.2

For the list of forced parameters, the following configuration has been used:

```
ppb=5000,slowstart=1,vcores=14,containermemmap=4096,containerheapmap=3800,\
containermemred=4096,containerheapred=3800,containermemam=4096,containerheapam=3800,\
mapspernode=1,reducespernode=1,tasksperjvm=1,blocksize=1640000000,sortmem=800
```

14.3 VLBA dataset

We provide a comparison between the a priori delays computed in CorrelX (file `delays.ini_debug`, see §9.2), and those displayed by fourfit (for the corresponding values of `fourfit_reference_time` or FRT) in Table 22, for which we display only the values for the first and last accumulation period (with integer times) for brevity, as well as a comparison between some of the parameters displayed by fourfit for CorrelX and DiFX in Table 23.

Table 22: VLBA a priori delays comparison.

SW	**Time**	**Apriori delay (usec)**	**Apriori clock (usec)**	**Apriori clockrate (us/s)**	**Apriori rate (us/s)**	**Apriori accel (us/s/s)**
cx	30000.0	-3082.66904893	0.166044333118	-2.86699814e-08	0.159532908381	2.14380785716e-05
ff	30000.0	-3082.66904893	0.16555524	-2.8669982e-08	0.159532908441	2.144389012633e-05
cx	30296.0	-3034.51181574	0.166035846804	-2.86699814e-08	0.16584116975	2.11742644751e-05
ff	30296.0	-3034.51181574	0.16554666	-2.8669982e-08	0.165841169210	2.11828273607e-05

Table 23: VLBA results summary (frequency group a:d, polarization pair LL).

Parameter	**CorrelX**	**DiFX**
Fringe quality	6	6
Error code	-	-
SNR	264.7	278.8
Int time	300.160	299.980
SBD	0.031692	0.031673
MBD	-0.002656	-0.002661
Fringe rate (Hz)	0.001931	0.001914
Resid mbdelay (usec)	-2.65571E-03 +/- 9.2e-06	-2.66138E-03 +/- 8.7e-06
Resid sbdelay (usec)	3.16922E-02 +/- 4.1e-05	3.16725E-02 +/- 3.9e-05
Resid phdelay (usec)	-4.44930E-06 +/- 1.4e-08	4.44526E-06 +/- 1.3e-08
Resid rate (us/s)	2.24082E-08 +/- 8.1e-11	2.22212E-08 +/- 7.7e-11
Resid phase (deg)	138.0 +/- 0.4	137.9 +/- 0.4

For the list of forced parameters, the following configuration has been used:

```
ppb=5000,slowstart=1,vcores=12,containermemmap=5128,containerheapmap=4900,\
containermemred=5128,containerheapred=4900,containermemam=5128,containerheapam=4900,\
mapspernode=1,reducespernode=1,taskspernjvm=1,blocksize=1640000000,sortmem=800
```

Note that this configuration differs slightly from the one used for both the VGOS and ALMA datasets.

15 Profiling and Call Graph Generation

In this section, we describe the tools provided in CorrelX for profiling the application modules and describe the procedures for the profiling of the complete system. The provided library `lib_profiling` is based on PyCallGraph [49].

15.1 Profiling the Mapper

To profile the mapper, activate the required option in the CorrelX configuration file:

```
[Profiling]
Profile mapper (pipeline):        yes
Use PyCallGraph:                  yes
...
```

One pycallgraph diagram will be generated for each mapper and placed in the configured output folder. An example is provided in Fig. 11.

To instead use `cProfile`, set the following option to “no”:

```
Use PyCallGraph:                  no
```

That setting generates two files—`.cprof` and `.txt`—with the profiling results. The text file looks like this example:

```
vi correlx/output/OUT_s0_v0.outpart1_BM434A-BR-No0017-Aa3m.vdif_map_prof_20161215_112849.txt

  Thu Dec 15 11:28:56 2016    correlx/output/OUT_s0_v0.\
           outpart1_BM434A-BR-No0017-Aa3m.vdif_map_prof_20161215_112849.cprof

         673953 function calls (673484 primitive calls) in 2.719 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.002    0.002    2.722    2.722 msvf.py:37(<module>)
        1    0.057    0.057    2.545    2.545 msvf.py:981(main)
        3    0.035    0.012    1.740    0.580 lib_ini_files.py:64(serialize_config)
        3    0.000    0.000    1.342    0.447 ConfigParser.py:285(read)
        3    0.622    0.207    1.342    0.447 ConfigParser.py:464(_read)
       14    0.053    0.004    0.362    0.026 ConfigParser.py:625(items)
      469    0.004    0.000    0.259    0.001 msvf.py:884(pack_and_encode_samples)
  [...]
```

15.2 Profiling the Reducer

To profile the reducer, activate the required option in the CorrelX configuration file:

```
[Profiling]
Profile reducer (pipeline):       yes
Use PyCallGraph:                  yes
...
```

One pycallgraph diagram will be generated and placed in the configured output folder. An example is provided in Fig. 12.

To instead use `cProfile`, set the following option to "no":

```
Use PyCallGraph:                  no
```

This setting will generate two kind of files—`.cprof` and `.txt`—with the profiling results. The text file looks like this example:

```
vi correlx/output/OUT_s0_v0.outpart2_red_prof_20161215_112849.txt

  Thu Dec 15 11:29:07 2016    correlx/output/OUT_s0_v0.outpart2_red_prof_20161215_112849.cprof

         128457 function calls (127983 primitive calls) in 8.526 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.001    0.001    8.528    8.528 rsvf.py:33(<module>)
        1    0.104    0.104    8.394    8.394 rsvf.py:541(main)
      313    0.059    0.000    5.784    0.018 lib_fx_stack.py:1493(compute_fx_for_all)
      448    2.803    0.006    2.803    0.006 lib_fx_stack.py:516(get_exp)
       56    0.033    0.001    2.345    0.042 lib_fx_stack.py:938(fringe_rotation)
      224    0.165    0.001    2.310    0.010 lib_fx_stack.py:879(fringe_rotation_work)
       56    0.069    0.001    1.983    0.035 lib_fx_stack.py:1119(compute_f_all)
        2    0.001    0.000    1.920    0.960 rsvf.py:324(get_lines_out_for_all)
       20    0.020    0.001    1.919    0.096 rsvf.py:257(get_str_r_out)

  [...]
```

15.3 Profiling the Complete Application

For profiling the complete application, run CorrelX on pycallgraph with the list of functions as follows:

```
import os
import lib_profiling
```

```
libs_include = lib_profiling.get_include_functions("mapred_cx.py")

os.system("pycallgraph -e \"*.<module>\" "+libs_include+" graphviz --"+\
   "./mapred_cx.py -n 10.0.2.4 -c conf/correlx.ini")
```

A pycallgraph will be generated. We provide an example in Fig. 13 for the pipeline mode.

15.4 Profiling the Conversion Tools

Here is an example showing how to profile the configuration converter tool:

```
import os
import lib_profiling

libs_include = lib_profiling.get_include_functions("convert_cx2d_man.py",avoid_v=\
   ["main"],plus_v=["lib_*","cx2d*"])

os.system("pycallgraph -e \"*.<module>\" "+libs_include+" graphviz -- "+\
   "convert_cx2d_man.py exp_folder/out .. OUT_s5_v14.out")
```

Here is an example showing how to profile the output converter tool:

```
import os
import lib_profiling

inout_folder="exp_folder"
file_in="bm434aYs20_20"
forced_files="BM434A-BR-No0017-Aa3m.vdif,BM434A-LA-No0017_3m.vdif"

libs_include = lib_profiling.get_include_functions("convert_im_cx.py",\
                           plus_v=["lib_*","cx2d*"])

os.system("pycallgraph -e \"*.<module>\" "+libs_include+" graphviz -- "+\
   "convert_im_cx.py "+inout_folder+" "+file_in+" "+forced_files)
```

Examples for the generated Pycallgraphs are displayed in Figs 14-17. Figs. 14 and 16 are for an experiment with zoom bands (computed during post-processing), and Figs. 15 and 17 for an experiment with phase calibration.

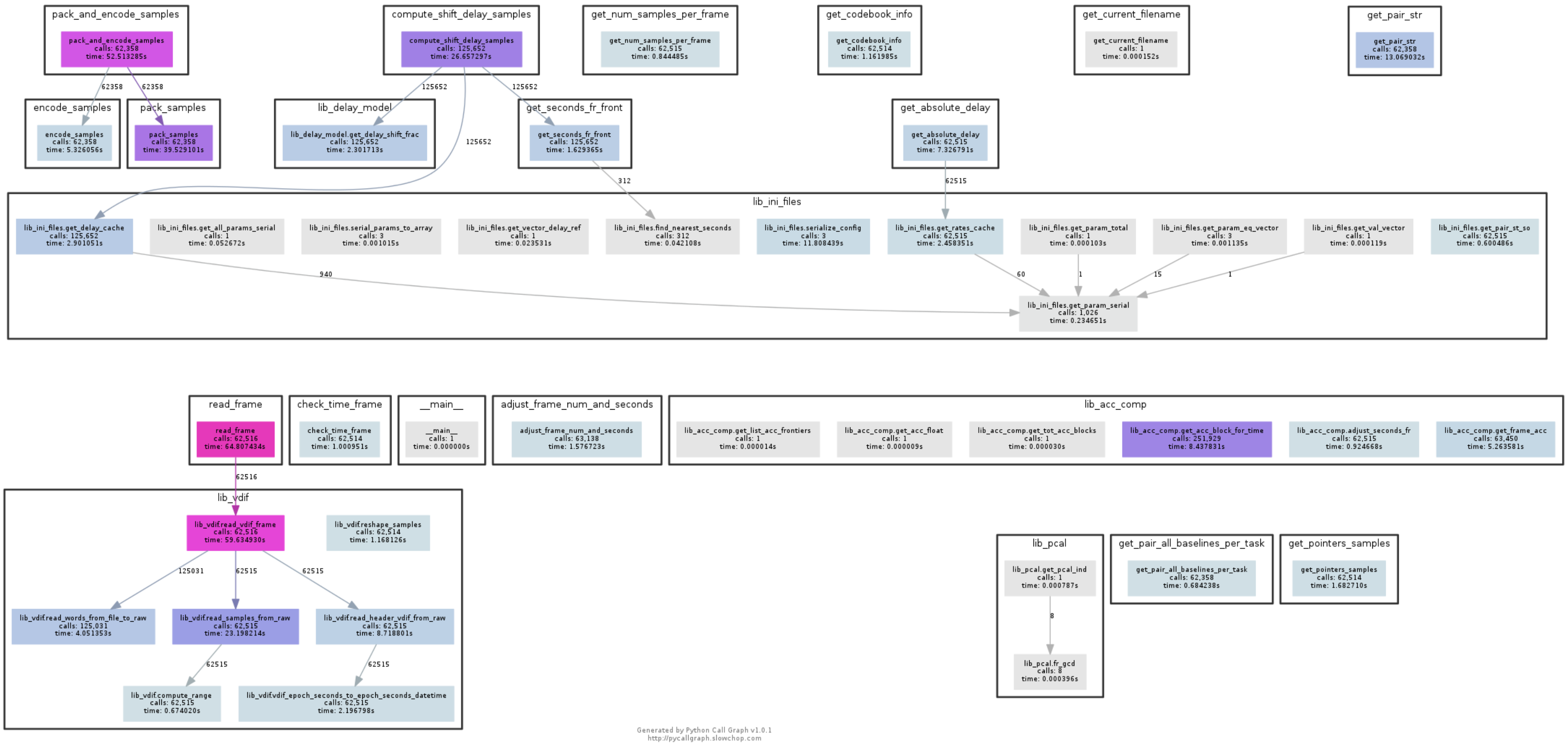


Figure 11: Mapper profiling pycallgraph diagram.

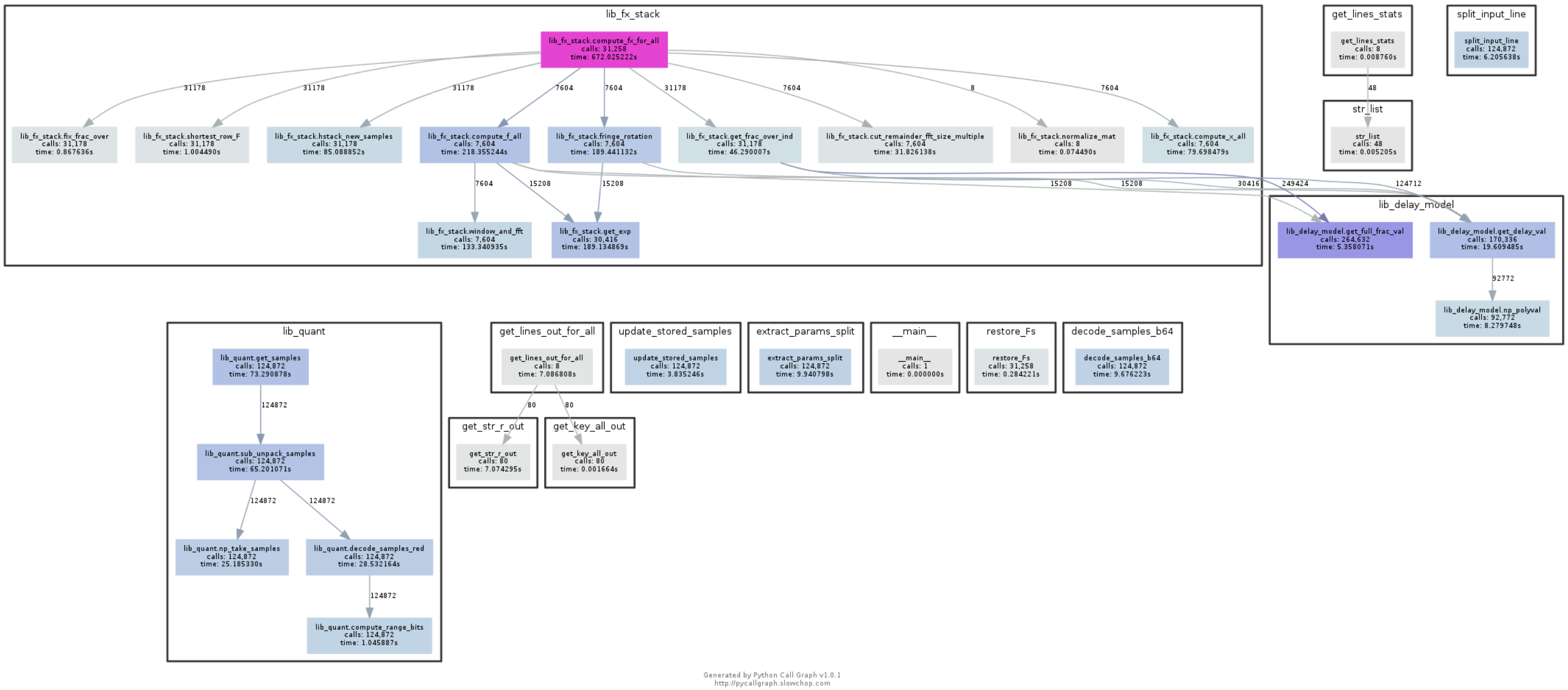


Figure 12: Reducer profiling pycallgraph diagram.

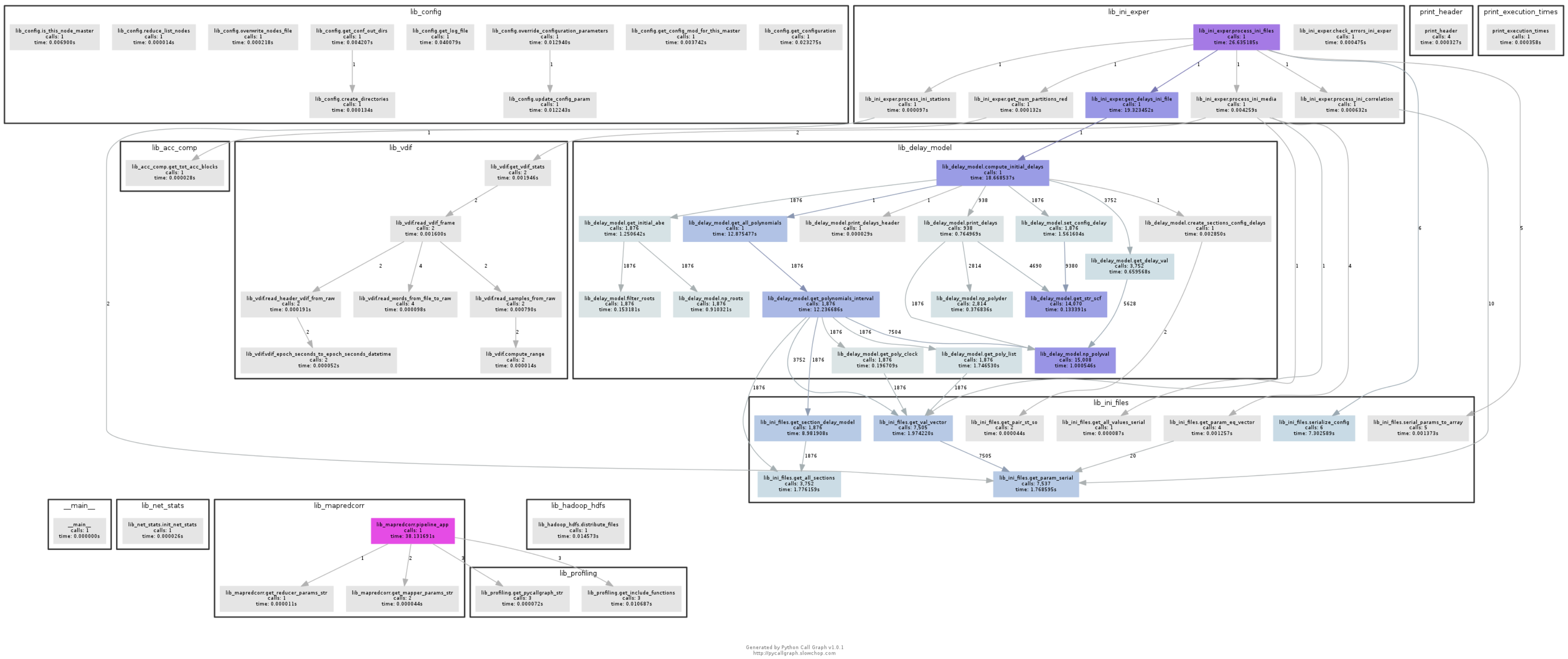


Figure 13: Pipeline mode profiling pycallgraph diagram.

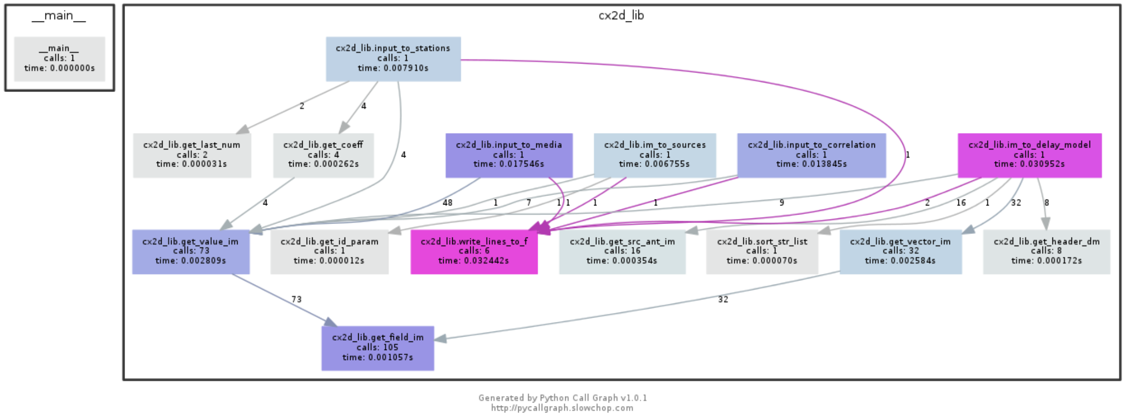


Figure 14: Configuration converter pycallgraph diagram—dataset with zoom bands.

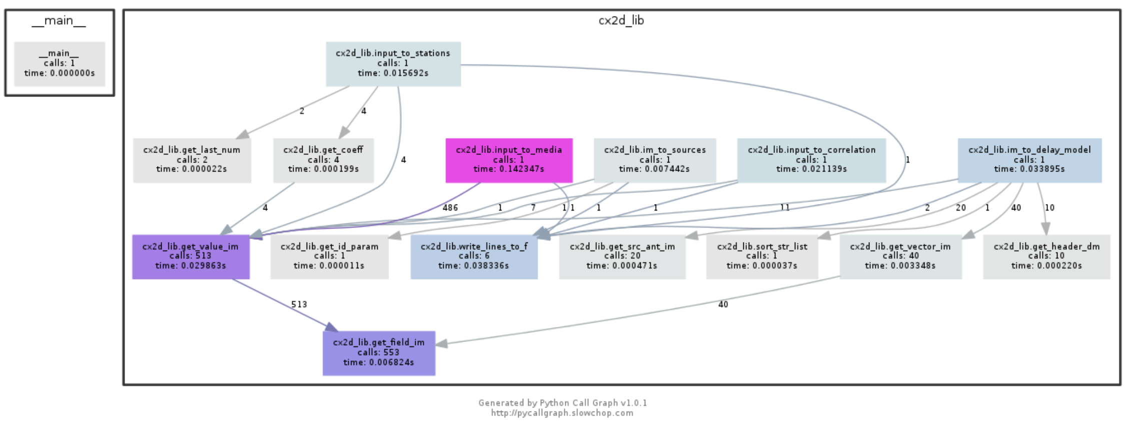


Figure 15: Configuration converter pycallgraph diagram—dataset with phase calibration.

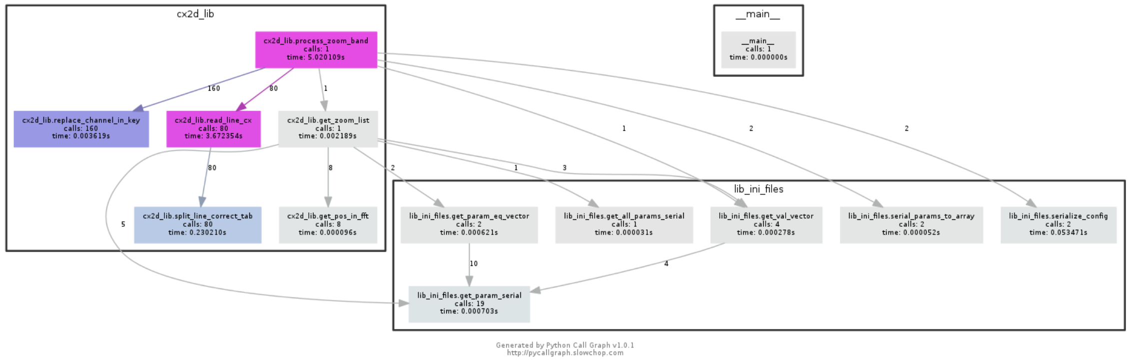


Figure 16: Output converter pycallgraph diagram—dataset with zoom bands.

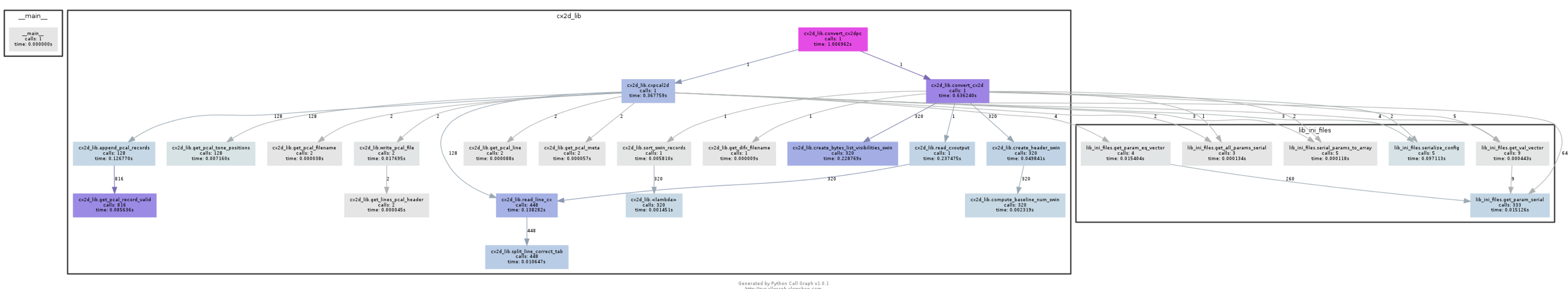


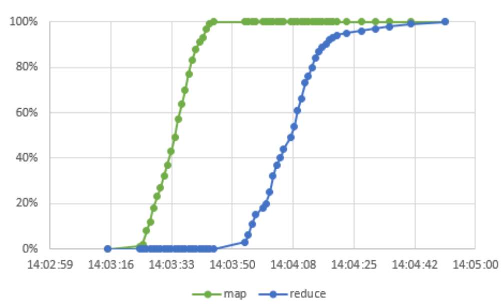
Figure 17: Output converter pycallgraph diagram—dataset with phase calibration.

15.5 Profiling the MapReduce: Load Balancing

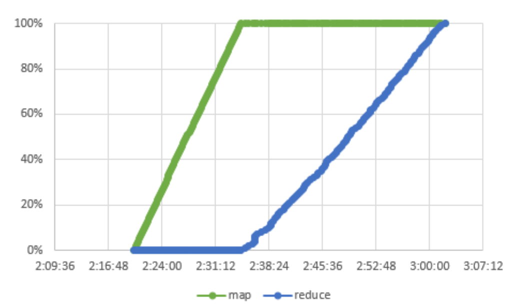
The Hadoop main log provides feedback with the progress on the Map and Reduce stages that can be easily plotted to detect possible issues. The progress reported in the log follows this format:

```
[...]
16/11/17 02:20:03 INFO mapreduce.Job: Running job: job_1479367052430_0001
16/11/17 02:20:12 INFO mapreduce.Job: Job job_1479367052430_0001 running in uber mode : false
16/11/17 02:20:12 INFO mapreduce.Job:  map 0% reduce 0%
16/11/17 02:20:27 INFO mapreduce.Job:  map 1% reduce 0%
16/11/17 02:20:34 INFO mapreduce.Job:  map 2% reduce 0%
[...]
16/11/17 03:00:52 INFO mapreduce.Job:  map 100% reduce 97%
16/11/17 03:01:13 INFO mapreduce.Job:  map 100% reduce 98%
16/11/17 03:01:29 INFO mapreduce.Job:  map 100% reduce 99%
16/11/17 03:02:07 INFO mapreduce.Job:  map 100% reduce 100%
[...]
```

As an illustrative example, we show the results of processing these logs quickly into a spreadsheet in Fig. 18, one of them corresponding to bad load balancing (using the default partitioner) and the other one using the custom partitioner provided in CorrelX. Note that the two graphs correspond to different datasets and different cluster deployments, but they serve to illustrate the issue of some reducers taking longer to finish than other (due to differences in the number of keys that each reducer processes with the default partitioner).



(a)



(b)

Figure 18: MapReduce profiling showing poor (a) and good (b) load balancing in the reduce phase. Note that long times spent in the last part of the reduce phase may indicate load balancing issues.

Considering load balancing, the CorrelX/Hadoop logs provide information on the number of mappers and reducers launched, which should be as described in §4.3. That is, the number of mappers and reducers executed as reported in the Hadoop log:

```
  [...]
  16/11/17 03:02:07 INFO mapreduce.Job:  map 100% reduce 100%
  [...]
Job Counters
```

```
  [...]
Launched map tasks=12289
Launched reduce tasks=1878
Failed map tasks=1
Killed map tasks=1
Killed reduce tasks=2
  [...]
```

should match the reported number of mappers and reducers in the CorrelX log:

```
  [...]
  Forcing mappers: 12287
  Forcing reducers: 1876
  [...]
```

It is possible to monitor the number of containers per machine during correlation by running:

```
  ./hadoop/hadoop-2.7.3/bin/yarn --config correlx/conf/<hostname>/etc_hadoop_<hostname>\
     node -list|grep --line-buffered -e "cluster" -e "Node-Id" -e "Total"
```

where `<hostname>` should be replaced by the host name of the master node. This command reports a list of nodes and containers per node, as in the following example:

```
  ./hadoop/hadoop-2.7.3/bin/yarn --config correlx/conf/node362/etc_hadoop_node362\
     node -list|grep --line-buffered -e "cluster" -e "Node-Id" -e "Total"

   List of nodes:
    Total Nodes:16
   Node-Id      Node-State Node-Http-Address Number-of-Running-Containers
   node362.cm.cluster:37933        RUNNING node362.cm.cluster:20029                    10
   node086.cm.cluster:55461        RUNNING node086.cm.cluster:20029                    11
   node360.cm.cluster:38482        RUNNING node360.cm.cluster:20029                    11
   node105.cm.cluster:55750        RUNNING node105.cm.cluster:20029                    11
   node104.cm.cluster:54377        RUNNING node104.cm.cluster:20029                    11
   node118.cm.cluster:53267        RUNNING node118.cm.cluster:20029                    11
   node361.cm.cluster:60986        RUNNING node361.cm.cluster:20029                    11
   node139.cm.cluster:49581        RUNNING node139.cm.cluster:20029                    11
   node085.cm.cluster:34592        RUNNING node085.cm.cluster:20029                    11
   node141.cm.cluster:34263        RUNNING node141.cm.cluster:20029                    10
   node140.cm.cluster:45405        RUNNING node140.cm.cluster:20029                    11
   node117.cm.cluster:53716        RUNNING node117.cm.cluster:20029                    11
   node083.cm.cluster:56409        RUNNING node083.cm.cluster:20029                    11
   node084.cm.cluster:55874        RUNNING node084.cm.cluster:20029                    10
   node082.cm.cluster:33283        RUNNING node082.cm.cluster:20029                    11
   node116.cm.cluster:38292        RUNNING node116.cm.cluster:20029                    11
```

16 Optimizations, Approximations, and Multi-threading

In this section, we provide some guidelines for performance tuning. Note that **this section and the associated CorrelX source code is work in progress**.

16.1 Optimizations

Optimizations correspond to features that provide performance-tuning capabilities without affecting the precision of the results.

16.1.1 Parallelization Modes

CorrelX provides three parallelization modes; that is, there are three ways of distributing computations among the reducers:

- All-baselines-per-task: **this is the default mode of operation, and also the main branch of development**. Each task (computations corresponding to one partition) encompasses the correlations for all the baselines corresponding to the accumulation period.
  - *Enable*: enabled by default (`One baseline per task:  no` and `Task scaling stations: no` in `correlx/conf/correlx.ini`).
  - *Current status*: main branch of development.
  - *Key*: `px-A.A-A.A`.
- One-baseline-per-task: This corresponds to the initial implementation, where each partition would correspond to only one baseline. This implies that data needs to be duplicated at the output of the mapper, but provides increased scalability in certain scenarios (e.g., few accumulation periods and many stations).
  - *Enable*: Settings are `One baseline per task:  no` and `Task scaling stations:  yes` in `correlx/conf/correlx.ini`.
  - *Current status*: discontinued.
  - *Key*: `py-s0.p0-s1.p1` where `s` and `p` are for station and polarization, respectively.
- Linear-scaling-with-stations: This corresponds to an intermediate case between the previous two, aiming for a linear increase in the data duplication (instead of quadratic as in the previous case) and in computations (as in the first case).
  - *Enable*: Settings are `One baseline per task:  yes` in `correlx/conf/correlx.ini`.
  - *Current status*: discontinued, although sub-case of the "all-baselines-per-task" mode.
  - *Key*: `pr-s0.p0-A.A`. Note that the partition includes only certain baselines for `s0.p0`, not all of them, in order to achieve linear scaling. The matrix with the baselines that correspond to each partition is generated in `msvf.get_alloc_tasks_linear_scaling()`. **Important:** this mode generates balanced load to all partitions, as opposed to simpler methods such as partitioning the correlation matrix into rows, or into sub-matrices as in [42]. For example:

```
import msvf
msvf.get_alloc_tasks_linear_scaling(6)
```

would return:

```
  array([[1, 0, 0, 1, 1, 1],
         [1, 1, 0, 0, 1, 1],
         [1, 1, 1, 0, 0, 1],
         [0, 1, 1, 1, 0, 0],
         [0, 0, 1, 1, 1, 0],
         [0, 0, 0, 1, 1, 1]])
```

In this example with 6 stations, each partition would receive data for 3 or 4 stations, and compute only baselines between the elements in the main diagonal and those in the row that are 1. That is, the partition corresponding to the first row would receive data for stations 0, 3, 4, and 5 and compute correlations for the baselines 0-0, 0-3, 0-4, and 0-5; the partition corresponding to the second row would receive data for stations 0, 1, 4, and 5 and compute correlations for the baselines 0-1, 1-1, 1-4, and 1-5. Note that this matrix is generated from a triangular matrix, so that no computations are duplicated.

The simpler cases previously mentioned consider the correlation matrix:

```
  array([[1, 1, 1, 1, 1, 1],
         [0, 1, 1, 1, 1, 1],
         [0, 0, 1, 1, 1, 1],
         [0, 0, 0, 1, 1, 1],
         [0, 0, 0, 0, 1, 1],
         [0, 0, 0, 0, 0, 1]])
```

In this case, splitting into rows generates unbalanced tasks: the first row duplicates data for all stations and computes baselines 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, etc. Splitting into sub-matrices—for example, of size 2×2 in order to generate the same number of tasks—would generate a number of baselines per task that is still quadratic (number of rows times number of columns), although with less data duplication, and requires a more complicated logic for defining the partitioning strategy (sub-matrix size) (consider an example with 7 stations).

The mapper receives this mode as an argument, and the reducer deduces the operation mode from the key as displayed previously, so this allows for hybrid approaches. Note that the MapReduce framework already provides to each reducer only one partition with its lines sorted.

16.1.2 Mapper's "Superframes"

It is possible to configure the number of frames that go into each output of the mapper in order to reduce the overhead of the metadata in the MapReduce interface.

This feature is disabled by default and should be activated only for development/debugging:

```
vi correlx/src/const_performance.py
  NUM_FRAMES_PER_LINE = -1
```

16.1.3 Reducer's Computation Cycle

It is possible to configure the number of groups of MapReduce lines that are read by the reducer (and which associated data is stored) before calling the computation function from the FX correlation library.

This feature is disabled by default and should be activated only for development/debugging:

```
vi correlx/src/const_performance.py
  COMPUTE_FOR_SUB_ACC_PERIOD = -1
```

16.1.4 FX Library's Rotation for Multiple Polarizations of the Same Station

It is possible to configure the FX correlation library to attempt to avoid recomputing the fringe rotation and fractional sample correction multiplier vectors for multiple polarizations of the same channel and station.

This feature is disabled by default and should be activated only for development/debugging:

```
vi correlx/src/const_performance.py
  SAVE_TIME_ROTATIONS = 0
```

16.2 Assumptions

Assumptions are documented in the docstring of each function. Keeping track of these assumptions (through the convention presented in §11.4) is recommended (note that the following lines correspond to titles of sections of the docstrings):

```
correlx/sh/show_assumptions.sh

  CorrelX - Assumptions
  ---------------------

  /home/cxuser/correlx/src/cx2d_lib.py:1104:    Assumptions:
  /home/cxuser/correlx/src/cx2d_lib.py:1281:    Assumptions:
  /home/cxuser/correlx/src/cx2d_lib.py:2438:    Assumptions:
  /home/cxuser/correlx/src/cx2d_lib.py:2590:    Assumptions:
  /home/cxuser/correlx/src/lib_code_stats.py:88:    Assumptions:
  /home/cxuser/correlx/src/lib_config.py:485:    Assumptions:
  /home/cxuser/correlx/src/lib_delay_model.py:573:    Assumptions:
```

Please refer to the source files for more details.

16.3 Approximations

Approximations correspond to features that **reduce the precision of the results** or **disable some checks** in order to improve performance. All these approximations are documented in the code and can be reported with one of the provided scripts:

```
correlx/sh/show_approximations.sh

  CorrelX - Approximations
  ------------------------

  /home/cxuser/correlx/src/lib_fx_stack.py:96:    Approximations:
  /home/cxuser/correlx/src/lib_fx_stack.py:508:    Approximations:
  /home/cxuser/correlx/src/lib_fx_stack.py:790:    Approximations:
```

A summary for these approximations follows.

Reduced checks for sample concatenation Correctness for sorting is reliant on the MapReduce framework, and it is assumed that no frames are missing; see `lib_fx_stack.hstack_new_samples()` for more information.

Reduced times for trivial cases in exponential The check for the case in which the delays in the vector are equal is simplified; see `lib_fx_stack.get_exp()` for more information.

Interpolation in delay computations The function to compute the delays used for fringe rotation provides three modes of operation:

- *Constant*: evaluate delays for only the first sample of the vector.
- *Full*: evaluate delays for all the samples in the vector.
- *Linear*: interpolate linearly based on delays for first and last samples of the vector.

This can be configured as follows:

```
  vi correlx/src/const_performance.py
    #FULL_TIMESCALE=0  # Evaluate delays only for the first sample
    #FULL_TIMESCALE=1  # Evaluate delays for the full timescale
    FULL_TIMESCALE=2  # Interpolate linearly based on delays for first and last samples
```

It is important to note that this can be used in combination with the reducer computation cycle described in §16.1.3, which defines the length of these vectors of samples.

Single or Double Precision Precision can be configured via the configuration file. The types corresponding to the different precisions are currently hardcoded in `rsvf.py`, with `numpy.complex64` for single and `numpy.complex128` for double. Extended precision is not currently implemented; for more details (`numpy.clongdouble`), refer to [43].

16.4 Multi-threading Support

Multi-threading support in CorrelX is currently a work in progress. This section describes the current status of the implementation and some parameters that can be used for performance tuning, illustrated with examples for a scenario with nodes of 12 CPU cores per node and 4 cores per task.

Support for these modules is still being developed and is not yet complete.

16.4.1 pyFFTW

Support for the pyFFTW [44] wrapper of FFTW3 [50] is provided in `lib_fx_stack.window_and_fft()` and can be enabled in `const_performance`.

Requirements

1. Install the required modules.

```
sudo apt-get install python-fftw
```

2. Activate the pyFFTW module and configure the number of threads. This option should be used only for development/debugging.

```
vi correlx/src/const_performance.py
  USE_FFTW =     1
  THREADS_FFTW = 4
```

Procedure If running in serial mode, launch CorrelX as in §9.2; if running in parallel mode (Hadoop), add `vcores=12,vcoresperred=4` to the forced parameter list via the command line as described in §9.4.

Note that in order to have effectively the three expected reducers running at the same time in the nodes, their requested memory cannot exceed the memory available in the node (see, e.g., [8] for details).

16.4.2 Numexpr

Support for the numexpr [51] module is provided in certain parts of the processing in the FX library (computation of exponentials, fringe rotation).

Requirements

1. Install the required modules.

```
sudo apt-get install python-numexpr
```

2. Activate the numexpr module and configure the number of threads. This option should be used only for development/debugging.

```
vi correlx/src/const_performance.py
  USE_NE =     1
  THREADS_NE = 4
```

It is possible to use this module only on certain parts of the process in the conditional presented after these constants in `const_performance`.

Procedure The procedure is similar to the one presented in §16.4.1.

16.4.3 Multiprocessing Pool

Support for the pool of workers in the Python multiprocessing module [52] is currently provided for fringe rotation. This option should be used only for development/debugging.

Requirements Activate the multiprocessing module and configure the number of threads. This option should be used only for development/debugging.

```
vi correlx/src/const_performance.py
  USE_MP =      1
  MP_THREADS =  4
```

Procedure The procedure is similar to the one presented in §16.4.1.

16.4.4 Numpy's High-Performance Libraries

Most of the computations in CorrelX are performed through the module `numpy` (and FFTs through `scipy`, which also depends on numpy). This module is generally configured to use the reference implementation library. It is possible to activate multi-thread high performance libraries like OpenBLAS (see, e.g., [53]) or Intel MKL (see e.g., [54]).

16.4.5 Numba

Numba has only been tested preliminarily on the FX library (more specifically in `lib_fx_stack.compute_x_all()`). In this section, we provide some usage guidelines.

Requirements Numba can be installed easily through Miniconda. Note that doing so will change the default Python interpreter; the path for this interpreter can be specified as described in Table 4 (`python executable`).

1. Install numba:

```
miniconda
conda update conda
conda install numpy
conda install scipy
conda install numba
```

2. Import module and add annotations before the applicable function definitions:

```
vi correlx/src/lib_fx_stack.py
  import numba
  [...]
  @numba.jit
  def [...]
  [...]
```

Procedure The procedure is similar to the one presented in §16.4.1.

16.4.6 Tools for Development

The CorrelX output comparator can be used to test the new results against the reference results; see §9.8.3 for details.

17 Source Code Management

In this section, we describe the changes for each of the software releases and provide guidelines to generate statistics from the source code.

17.1 Release Versions

The changes corresponding to each pre-alpha release are listed in Table 24.

Table 24: Software versions.

Release	Date	Description	Changes
correlx-alpha0.63.tar	2017.02.27	Public release.	First release.

17.2 Source Code Statistics

The library `lib_code_stats` provides some tools for the generation of statistics from the code. For generating statistics on the number of lines of code and comments/docstrings:

```
import lib_code_stats
lib_code_stats.get_cx_code_stats()
```

which will generate the following statistics:

```
Statistics for CorrelX Python sources on 2017/02/27:
*Summary:
  Lines of code:            7971
  Lines of comments:        8174

*Details:
Launcher and Filesystem Management:
-----------------------------------
 [File]                     [Total]        [Code]    [Doc+Comm]       [Empty]   [Non-empty]
 const_config.py                221           122            60            39           182
 const_hadoop.py                183            50            86            47           136
 lib_config.py                  997           368           435           194           803
 lib_hadoop_hdfs.py            1030           422           401           207           823
 lib_ini_exper.py               670           215           321           134           536
 lib_mapredcorr.py             1017           368           488           161           856
 lib_net_stats.py               218            84           100            34           184
 lib_profiling.py               139            34            74            31           108
 mapred_cx.py                   769           404           166           199           570
 <Totals>                      5244          2067          2131          1046          4198

Application:
------------
 [File]                     [Total]        [Code]    [Doc+Comm]       [Empty]   [Non-empty]
 msvf.py                       1892           695           801           396          1496
 rsvf.py                       1401           615           519           267          1134
 <Totals>                      3293          1310          1320           663          2630

Libraries:
----------
 [File]                     [Total]        [Code]    [Doc+Comm]       [Empty]   [Non-empty]
```

```
 const_debug.py                   96           16           60           20           76
 const_ini_files.py              244           97          118           29          215
 const_mapred.py                 153           49           61           43          110
 const_performance.py            124           21           78           25           99
 const_quant.py                   39            2           32            5           34
 lib_acc_comp.py                 331           92          191           48          283
 lib_debug.py                    281          190           48           43          238
 lib_delay_model.py             1343          438          580          325         1018
 lib_fx_stack.py                1883          720          830          333         1550
 lib_ini_files.py                690          204          365          121          569
 lib_pcal.py                     232           75          116           41          191
 lib_quant.py                    337          100          167           70          267
 lib_vdif.py                    1181          559          368          254          927
 <Totals>                       6934         2563         3014         1357         5577

Tools:
------
 [File]                      [Total]       [Code]   [Doc+Comm]      [Empty]  [Non-empty]
 convert_cx2d.py                 109           38           47           24           85
 convert_im_cx.py                 87           34           34           19           68
 cx2d_lib.py                    3174         1411         1138          625         2549
 lib_code_stats.py               308          142          108           58          250
 process_zoom.py                  77           20           41           16           61
 vdif_generator.py               758          330          279          149          609
 vdif_info.py                     77           32           31           14           63
 vis_compare.py                   67           24           31           12           55
 <Totals>                       4657         2031         1709          917         3740

 <TOTALS>                      20128         7971         8174         3983        16145
```

These statistics are saved to a file `stats_code_<date>`; another file `stats_code_debug_<date>` with all the docstrings is generated for checking the correct behavior of the library.

18 Known Issues

18.1 CorrelX

Known issues are kept in the inline documentation of the sources. For example:

```
correlx/sh/show_known_issues.sh

  CorrelX - Known Issues
  ----------------------

  /home/cxuser/correlx/src/convert_cx2d.py:25:(!) Known issues / TO DO:
  /home/cxuser/correlx/src/lib_delay_model.py:11:Known issues:
  /home/cxuser/correlx/src/lib_fx_stack.py:714:    Known issues:
```

Note that the script simply reports appearances of the string "Known Issues". Lines without the symbol # correspond to sections in the docstrings that include many lines below.

A summary of these known issues follows.

Delay Model

- Delays are all computed with regard to the first station (closest to the source). These should be computed with regard to the center of the Earth instead.
- The precision of the polynomial generation should be checked (high-order terms).

Format Conversion

- The phase calibration scaling needs further work: currently manually scaling and forcing conjugation.

18.2 Hadoop

Text/Binary Reader Currently, the Hadoop TextInputFormat reader is used as a binary reader to read VDIF frames. This requires the configuration of the end-of-line separator `[Hadoop-other] Text delimiter` that is checked by the Hadoop filesystem to split lines. No issues have been experienced yet, but this should be fixed by, for example, using the Hadoop binary reader or a custom reader.

HDFS replication When moving data into the HDFS system, Hadoop may raise the following error [56], even though this does not affect the processing:

```
  16/11/29 17:06:14 INFO hdfs.DFSClient: Exception while adding a block
  org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.hdfs.server.namenode.
        NotReplicatedYetException): Not replicated yet: [...]
        [...]
```

Obtaining output file The following exception may be raised after retrieving the output file, even though this does not affect the processing:

```
16/12/07 14:04:17 INFO jvm.JvmMetrics: Initializing JVM Metrics with
    processName=JobTracker, sessionId=
Exception in thread "main" java.lang.NullPointerException
[...]
```

18.3 Python

Even though most of the code is compatible with both Python 2 and 3, there is an issue with `numpy.fromfile` in Python 3. The following error may be returned when trying to run the mapper on Python 3:

```
Traceback (most recent call last):
  File "/home/cxuser/src/msvf.py", line 1697, in <module>
[..]
  File "/home/cxuser/src/lib_vdif.py", line 346, in read_words_from_file_to_raw
    words_array = np.fromfile(file = f,dtype=TYPE_WORD, count=n_words)
OSError: first argument must be an open file
cat: write error: Broken pipe
```

19 Future Development

This section provides some recommendations on next steps to evolve the current CorrelX prototype.

19.1 Short-Term Development

TODOs are kept in the in-line documentation of the sources. For example (showing only part of the output):

```
correlx/sh/show_todos.sh

  CorrelX - TODOs
  ---------------

  [...]
  /home/cxuser/correlx/src/lib_delay_model.py:577:    TO DO:
  /home/cxuser/correlx/src/lib_delay_model.py:633:    # TO DO: check thresholds
  /home/cxuser/correlx/src/lib_delay_model.py:778:    TO DO:
  [...]
```

Note that the script simply reports appearances of the string "TODO"/"TO DO". Lines without the symbol `#` correspond to TO-DO sections in the docstrings that include many lines below.

Besides addressing the list of known issues, some features are currently being developed:

- Complete implementation for all cases (bits per sample, etc.).
- Python FFTW library (`pyfftw`): included in `lib_fx_stack.py` and operative, but not well tested.
- Rest of multi-threading support (see §16.4 for details).
- Further testing (`unittest...`).
- Add cases for checking configuration errors (see, for example, `lib_ini_exper.check_errors_ini_exper()`).
- The number of stations in the experiment configuration file should be replaced by a list with the names of the stations (§5.2.4).
- The configuration of the Lustre plugin should be made more intuitive (§8.2.3).
- The metadata section should be removed from the reducer output.
- Add the required functionality to write binary output files and introduce averaging too (and add the parameter for the averaging factor in the file `correlation.ini`).
- Allow the zoom-band functionality to be applied during correlation stage instead of during post-processing.

19.2 Long-Term Development

- Explore the extension to other parallelization frameworks (e.g., Spark) for reading directly from playback units instead of disk.
- As already pointed out in §2, the use of the DiFX tools to interface with vex, CALC, and HOPS is due to existing extensive use in the VLBI community. The development of tools to either bypass these DiFX tools or extend the current format conversion toolkit to cover general cases should be considered.

Appendix

A License

The MIT CorrelX Correlator

https://github.com/MITHaystack/CorrelX
Contact: correlX@haystack.mit.edu
Project leads: Victor Pankratius, Pedro Elosegui
Project developer: A.J. Vazquez Alvarez

B Acronym List

Table 25: Acronyms.

Acronym	Description
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FX	Fourier Transform and Multiplication
HOPS	Haystack Observatory Postprocessing System
MJD	Modified Julian Date
N/A	Not applicable
NFS	Network File System
SLURM	Simple Linux Utility for Resource Management
SWIN	Swinburne file format
TBC	To be confirmed
TBD	To be determined/done
VEX	VLBI experiment definition
VLBI	Very Long Baseline Interferometry

C Conventions

Table 26: Conventions.

Item	Description
Date format	YYYY.MM.DD

D Frequently Asked Questions

What are the minimum requirements to run CorrelX? A computer with Linux and Python 2.7 installed. See §8.1 for more information.

Is Hadoop required to run CorrelX? No, but it is recommended for scalable processing. See §8.1 and §9.2 for more information.

Is any knowledge about Hadoop required to run CorrelX? No, the Hadoop cluster is deployed seamlessly to the user.

Are third-party tools required to run CorrelX? For any experiment, only the configuration files described in §5.2 and the media are needed. We provide the conversion tools to generate these files from third-party software (e.g., DiFX, CALC, etc.). See §2 for more information.

What format conversion tools are currently provided with CorrelX? For input data: a VDIF file metadata reader (§9.8.1); for configuration data: a DiFX-to-CorrelX configuration converter (§9.6); for visibilities: a CorrelX-to-DiFX visibilities (and phase calibration results) converter (§9.7), a CorrelX-visibilities comparator (§9.8.3), a CorrelX-visibilities plotter (§9.8.4), and a DiFX-visibilities plotter (§9.8.5).

What coding tools are currently provided with CorrelX? For code statistics: a code statistics generator (§17), and scripts to list TODOs (§19.1) and known issues (§18.1) inline with the code; for profiling: mapper and reducer profiling options, and call-graph generators (§15.1); for debugging: verbose modes with tabulated information for debugging mapper, reducer, delay computations, and map-reduce interface reader (§13).

Is it possible to integrate a custom library into CorrelX easily? Yes, we provide some guidelines and examples in §12.

Why is the application written in Python? CorrelX is written in Python to facilitate quick prototyping. Even though Python might be slower than C++ on a single node, we can scale correlation performance by adding more nodes in the cloud. Our architecture allows for modular replacement of components, e.g., Python-implemented mappers and reducers can be easily replaced by C++ implementations if necessary.

Why Python 2 and not Python 3? Most of the code has been written to be compatible with the two versions, except for a function in the VDIF reader library (see §18.3 for details).

Is Python going to be a limitation in the future? No. There are numerous ways of replacing the libraries by Python-wrapped compiled versions of programs written in other languages.

Can I rewrite the application layer in a compiled language? Yes. The current version corresponds to a reference implementation that captures key VLBI correlation functionality and can serve as a specification for future projects. The Hadoop streaming mode (which is the standard mode of operation in CorrelX) allows users to run applications written in any language. See [27] for more information.

References

[1] Thompson, Moran, Swenson *Interferometry and Synthesis in Radio Astronomy*, 3rd Edition. Springer, 2017. doi: 10.1007/978-3-319-44431-4

[2] Apache, *Hadoop*, last accessed on 2016.11.17, available online: `http://hadoop.apache.org/`.

[3] Apache, *MapReduce Tutorial*, last accessed on 2016.11.17, available online: `https://hadoop.apache.org/docs/r2.7.3/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html`

[4] Yahoo, *Scaling Hadoop to 4000 nodes at Yahoo!*, last accessed on 2016.11.17, available online: `https://developer.yahoo.com/blogs/hadoop/scaling-hadoop-4000-nodes-yahoo-410.html`

[5] Apache, *Hadoop Distributions*, last accessed on 2016.11.17, available online: `https://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support`.

[6] Apache, *Hadoop YARN*, last accessed on 2016.11.17, available online: `https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-site/YARN.html`.

[7] Pankratius et al., *CorrelX: A Cloud-Based VLBI Correlator*, last accessed on 2016.12.08, available online: `http://www.haystack.mit.edu/workshop/ivtw2016/presentations/vpankratius-ivtw2016.pdf`.

[8] Hortonworks, *Managing CPU resources in your Hadoop YARN clusters*, last accessed on 2016.11.17, available online: `http://hortonworks.com/blog/managing-cpu-resources-in-your-hadoop-yarn-clusters/`.

[9] VLBI.org, *VLBI Experiment Definition (VEX)*, last accessed on 2016.12.08, available online: `http://www.vlbi.org/vex/`.

[10] Brisken, *A Guide to the DiFX Software Correlator (Version 2.2)*. 2014.

[11] NASA, *Mark-5 VLBI Analysis Software Calc/Solve*, last accessed 2016.11.22, available online: `http://gemini.gsfc.nasa.gov/solve/`.

[12] MIT Haystack, *Haystack Observatory Postprocessing System (HOPS)*, last accessed 2016.11.22, available online: `http://www.haystack.mit.edu/tech/vlbi/hops.html`.

[13] Cappallo, Mark4 & DiFX Integrated Data Flow, (2011.07.14) last accessed on 2016.11.17, available online: `http://www.atnf.csiro.au/vlbi/dokuwiki/lib/exe/detail.php/difx/difx_mk4_dataflow.png?id=difx%3Ahops`.

[14] Apache, *Hadoop yarn-site.xml*, last accessed on 2016.11.17, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-common/yarn-default.xml.

[15] Apache, *Hadoop mapred-site.xml*, last accessed on 2016.11.17, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml.

[16] Apache, *Hadoop core-site.xml*, last accessed on 2016.11.17, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/core-default.xml.

[17] Apache, *Hadoop hdfs-site.xml*, last accessed on 2016.11.17, available online: http://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.

[18] Apache, *Hadoop Cluster Setup*, last accessed on 2016.11.18, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/ClusterSetup.html.

[19] VLBI.org, *VLBI Data Interchange Format (VDIF) Specification, Release 1.1.1*, 2009.

[20] ATNF-CSIRO, *DiFX wiki – The correlator input file*, last accessed on 2016.11.18, available online: http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/files#the_correlator_input_file.

[21] ATNF-CSIRO, *DiFX wiki – The .im file*, last accessed on 2016.11.18, available online: http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/files#the_im_file.

[22] ATNF-CSIRO, *DiFX wiki – The SWIN output data format*, last accessed on 2016.11.18, available online: http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/files#the_swin_output_data_format.

[23] MIT Haystack, *CorrelX on github*, last accessed on 2016.11.21, available online: https://github.com/MITHaystack/CorrelX.

[24] Apache, *Hadoop Java Versions*, last accessed 2016.11.22, available online: https://wiki.apache.org/hadoop/HadoopJavaVersions.

[25] Apache, *Hadoop Releases*, last accessed on 2016.11.21, available online: http://hadoop.apache.org/releases.html.

[26] Apache, *Hadoop Releases*, last accessed on 2016.11.26, available online: http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz.

[27] Apache, *Hadoop Streaming*, last accessed on 2016.11.21, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-streaming/HadoopStreaming.html#Hadoop_Streaming.

[28] Apache, *Hadoop Streaming Command Options*, last accessed on 2016.11.21, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-streaming/HadoopStreaming.html#Streaming_Command_Options.

[29] Kettner, *SSH login without password*, last accessed on 2016.11.29, available online: http://www.linuxproblem.org/art_9.html.

[30] Apache, *Hadoop: Setting up a Single Node Cluster*, last accessed on 2016.12.01, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/SingleCluster.html.

[31] Apache, *Hadoop Cluster Setup*, last accessed on 2016.12.01, available online: https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/ClusterSetup.html.

[32] Apache, *Hadoop on GitHub*, last accessed on 2016.12.05, available online: https://github.com/apache/hadoop.

[33] Apache, *Hadoop on GitHub, (mapreduce) KeyFieldBasedPartitioner*, last accessed on 2016.12.05, available online: https://github.com/apache/hadoop/blob/trunk/hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/lib/partition/KeyFieldBasedPartitioner.java.

[34] Apache, *Hadoop on GitHub, (mapred) KeyFieldBasedPartitioner*, last accessed on 2016.12.05, available online: https://github.com/apache/hadoop/blob/trunk/hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapred/lib/KeyFieldBasedPartitioner.java.

[35] Kulkarni, *Hadoop MapReduce over Lustre*, (2013.04.16), available online: http://www.opensfs.org/wp-content/uploads/2013/04/LUG2013_Hadoop-Lustre_OmkarKulkarni.pdf.

[36] Seagate, *LustreFS Hadoop Plugin on github*, last accessed on 2016.11.21, available online: https://github.com/Seagate/lustrefs.

[37] Apache, *Hadoop API, Interface Mapper*, last accessed on 2016.11.22, available online: https://hadoop.apache.org/docs/r2.7.3/api/org/apache/hadoop/mapred/Mapper.html.

[38] Oracle, *VirtualBox*, last accessed on 2016.12.01, available online: https://www.virtualbox.org/.

[39] Ubuntu, *Download Ubuntu Server*, last accessed on 2016.12.01, available online: https://www.ubuntu.com/download/server.

[40] ATNF-CSIRO, *DiFX wiki – DiFX Installation*, last accessed on 2017.01.25, available online: http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/installation.

[41] ATNF-CSIRO, *DiFX wiki – difx2mark4*, last accessed on 2016.12.02, available online: http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/difx2mark4.

[42] Harris et al., *GPU accelerated radio astronomy signal convolution.* L. Exp Astron (2008) 22: 129.

[43] Scipy.org, *Data Types*, available online: https://docs.scipy.org/doc/numpy-dev/user/basics.types.html#extended-precision.

[44] PyPI, *pyFFTW*, last accessed on 2016.12.15, available online: https://pypi.python.org/pypi/pyFFTW.

[45] The IPython development team, *IPython Interactive Computing*, last accessed on 2016.12.01, available online: https://ipython.org/notebook.html.

[46] PyPI, *Bitarray*, last accessed on 2016.12.19, available online: https://pypi.python.org/pypi/bitarray.

[47] Sphinx, *Python Documentation Generator*, last accessed on 2017.02.27, available online: http://www.sphinx-doc.org/en/stable/.

[48] Numpy, *A Guide to Numpy/SciPy Documentation*, last accessed on 2017.02.27, available online: https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt.

[49] Slowchop, *Python Call Graph*, last accessed on 2016.11.22, available online: http://pycallgraph.slowchop.com/en/master/.

[50] FFTW.org, *FFTW* last accessed on 2016.12.15, available online: http://www.fftw.org/.

[51] PyPI, *Numexpr*, last accessed on 2016.12.15, available online: https://pypi.python.org/pypi/numexpr. Numexpr

[52] Python documentation, *multiprocessing.pool*, last accessed on 2016.12.15, available online: https://docs.python.org/2/library/multiprocessing.html#using-a-pool-of-workers.

[53] E. Huns, *Installing Numpy and OpenBLAS*, last accessed on 2016.12.15, available online: https://hunseblog.wordpress.com/2014/09/15/installing-numpy-and-openblas/.

[54] Intel Developer Zone, *Speeding up Python* scientific computations*, last accessed on 2016.12.15, available online: https://software.intel.com/en-us/node/696338.

[55] Continuum Analytics, *Numba*, last accessed on 2016.12.15, available online: http://numba.pydata.org/.

[56] M. G. Noll, *Benchmarking and Stress Testing an Hadoop Cluster With TeraSort, TestDFSIO & Co.*, last accessed on 2016.11.22: http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/.