ESA Climate Change Initiative – Fire_cci
D3.1 System Specification Document (SSD)

Project Name	ECV Fire Disturbance: Fire_cci
Contract Nº	4000126706/19/I-NB
Issue Date	22/04/2020
Version	2.1
Authors	Thomas Storm, Martin Böttcher, Grit Kirches
Document Ref.	Fire_cci_D3.1_SSD_v2.1
Document type	Public

To be cited as: T. Storm, M. Boettcher, G. Kirches (2020) ESA CCI ECV Fire Disturbance: D 3.1 System Specification Document, version 2.1.

Available at: http://www.esa-fire-cci.org/documents

Project Partners

Prime Contractor/ Scientific Lead & Project Management	UAH – University of Alcala (Spain)
Earth Observation Team	UAH – University of Alcala (Spain) UPM – Universidad Politécnica de Madrid (Spain) CNR-IREA - National Research Council of Italy – Institute for Electromagnetic Sensing of the Environment (Italy)
System Engineering	BC – Brockmann Consult (Germany)
Climate Modelling Group	MPIM – Max Planck Institute for Meteorology (Germany) CNRS - National Centre for Scientific Research (France)

Distribution

Affiliation	Name	Address	Copies
ESA	Simon Pinnock (ESA) Clément Albergel (ESA)	simon.pinnock@esa.int clement.albergel@esa.int	electronic copy
Project Team	Emilio Chuvieco (UAH) M. Lucrecia Pettinari (UAH) Joshua Lizundia (UAH) Gonzalo Otón (UAH) Mihai Tanase (UAH) Miguel Ángel Belenguer (UAH) Consuelo Gonzalo (UPM) Dionisio Rodríguez Esparragón (UPM) Ángel García Pedrero (UPM) Daniela Stroppiana (CNR) Mirco Boschetti (CNR) Thomas Storm (BC) Martin Böttcher (BC) Grit Kirches (BC) Angelika Heil (MPIM) Idir Bouarar (MPIM) Florent Mouillot (CNRS) Philippe Ciais (CNRS)	emilio.chuvieco@uah.es mlucrecia.pettinari@uah.es joshua.lizundia@uah.es gonzalo.oton@uah.es mihai.tanase@uah.es miguel.belenguer@uah.es consuelo.gonzalo@upm.es dionisio.rodriguez@ulpgc.es angelmario.garcia@upm.es stroppiana.d@irea.cnr.it boschetti.m@irea.cnr.it thomas.storm@brockmann-consult.de martin.boettcher@brockmann-cons... grit.kirches@brockmann-consult.de angelika.heil@mpimet.mpg.de idir.bouarar@mpimet.mpg.de florent.mouillot@cefe.cnrs.fr philippe.ciais@lsce.ipsl.fr	electronic copy

Summary

This document is the version 2.1 of the System Specification Document for the Fire_cci project. It introduces the structure, workflows and mode of operation of the processing system used in the Fire_cci project.

	Affiliation/Function	Name	Date
Prepared	BC	Thomas Storm Martin Böttcher Grit Kirches	22/04/2020
Reviewed	UAH – Project Manager	M. Lucrecia Pettinari	22/04/2020
Authorized	UAH - Science Leader	Emilio Chuvieco	22/04/2020
Accepted	ESA - Technical Officer	Clément Albergel	23/04/2020

This document is not signed. It is provided as an electronic copy.

Document Status Sheet

Issue	Date	Details
1.0	24/02/2016	First document for Phase 2 of Fire_cci
1.1	16/03/2016	Addressing ESA comments according to CCI_FIRE_EOPS_MM_16_0034.pdf
1.2	06/09/2016	Updated release of the document with minor corrections
1.3	25/10/2016	Addressing ESA comments according to CCI_FIRE_EOPS_MM_16_0109.pdf
1.4	30/09/2017	Update for MODIS / SFD processing
1.5	30/01/2018	Addressing ESA comments according to CCI-EOPS-FIRE-MM-17-0098.pdf
2.0	27/03/2020	First document for Phase 1 of CCI+
2.1	22/04/2020	Addressing ESA comments according to Fire_cci+:SSD_v2.0_RID,doc

Document Change Record

Issue	Date	Request	Location	Details
1.1	16/03/2016	ESA	Version number Section 1 Section 2.1 Section 2.2 Section 2.3 Section 3.1 Figure 3.1 Section 3.2 Section 3.4 Table 3.3 Section 5.1 Section 5.3 Annex	Change of previous version number from 2.0 to 1.0 to address this document being independent from the one in Phase 1. Minor changes in the text. Re-ordering of sentences. Minor changes in the text. Addition of new reference documents. Minor changes in the text. Updated Minor changes in the text. Inclusion of text detailing the SoW requirements. Changes in the requirements. Minor changes in the text. Minor changes in the text. Inclusion of additional information in the list of steps. Addition of new acronyms.
1.2	06/09/2016	BC	Section 1 Section 2.1 Section 2.2 Section 2.3 Section 2.4	Updated. Paragraph removed. Minor changes in the text. Last paragraph shortened. Reference documents updated. Minor changes in the text.

Issue	Date	Request	Location	Details
			Section 3.1 Section 3.3 Section 3.4 Section 4 Section 4.2.3 Section 5.1 Section 5.2	Figure 3.1 updated. Minor changes in the text. Added information about Calvalus MapReduce Requirements text resumed. Reference to the MERIS ATBD updated. Removed sub-section "Docker", as it is not applicable anymore. The rest of the sub-sections were re-numbered. Figure 4.4 updated. Added information about MapReduce and formatting tool implementation, eliminated information about Docker container. Figure 5.1 updated. Fully re-written according to system updates.
1.3	25/10/2016	ESA	Naming convention All document Section 3.1 Section 3.2 Section 5.1 Section 5.2	Reference to the year of the project added to the name of the document. The reference to the MERIS data as FSG was changed to FRS. Figure 3.1 updated. New entity in the text added. Figure 3.2 updated. Small changes in the text. Added information on versions of auxiliary data.
1.4	30/09/2017	BC	Whole document	Added system overview, technical methods and workflows for SFD and MODIS processing
1.5	30/01/2018	ESA	Sections 2.1, 3, 3.2, 3.3, 5.2, 5.2.3 Section 3.1 Section 4 Section 6	Small changes in the text Figure 3.1 updated and text expanded Section restructured, including previous sub-section 3.4 New section added
2.0	27/03/2020	BC	Sections 1, 2.1, 2.2, 2.3, 3.3, 4.1 Sections 3.1, 4.2, 4.2.1, 4.3 Section 5.1	Sections updated Small changes in the text Section referring to MERIS deleted, and now refers to the MODIS processing. Section and sub-sections updated.
2.1	22/04/2020	ESA	Section 3.1 Section 3.3.	Figure 3.1 updated Table 3.1: added definition to acronym

Table of Contents

List of Tables

List of Figures

1. Executive Summary

This document is the Fire_cci System Specification Document (SSD). The Fire_cci system is employed in order to generate the Fire_cci dataset, which will comprise several global and regional sub-datasets (see below).

This document describes the Fire_cci system, the workflows it defines and uses, and the environment in which it exists.

In the present version, the SSD describes the processing system which has been used for computing the MODIS global time series (FireCCI51), and which will be used to produce the small fires database (FireCCISFD20) based on Sentinel-2 MSI data. Later versions will also describe the processing of:

- the Sentinel-3-based global dataset (FireCCIS310 + FireCCIS311)
- the AVHRR-based global dataset (FireCCILT20) (formatting only)
- the merged-reflectance global datasets (FireCCIMR10 + FireCCI60)

2. Introduction

This document describes the system developed for the ESA Fire_cci project and its elements used in the current phase.

2.1. Background

The ESA Climate Change Initiative (ESA CCI) stresses the importance of providing a higher scientific visibility to data acquired by ESA sensors, especially in the context of the IPCC reports. This implies to produce consistent time series of accurate Essential Climate Variables (ECV) products that can be used by the climate (primarily) and other scientists for their modelling efforts. Long-term observations and the international links with other agencies currently generating ECV data are keys to this activity.

The fire disturbance ECV includes Burned Area (BA) as the primary variable. The project includes the development of algorithms for the BA retrieval of MODIS, Sentinel-2 MSI, AVHRR, and Sentinel-3 OLCI/SLSTR data.

For all of the input datasets, the satellite data must be acquired, algorithms and processing workflows must be defined and implemented, and the data processing must be performed.

2.2. Purpose and scope

The purpose of the Fire_cci system is the generation of the Burned Area product, the exchange of the results with project partners and delivery of the final products to the end users. There are various large satellite input datasets to be processed in a performant way in the course of the project. The algorithms used for processing will be continuously improved and adapted to new sensors with the need of several test and improvement cycles before full-scale processing. This shall also be supported by the processing system.

This SSD exists in the context of other Fire_cci documents: The Fire_cci ADP [RD-1] describes the content and format of the BA products to be generated, which are complementary to the PDS [RD-2] specifications, and the ATBD [RD-3] defines the algorithms that are used for BA processing. This SSD describes the design of the system that generates these BA products, including the integration of the processors that implement the algorithms defined in the ATBDs, the handling of the input datasets, and the control of the production workflow for the Fire_cci processing chains. The Fire_cci processing system is based on several layers of generic components of the BC Calvalus processing system and its deployment on various shared infrastructures, and on the STFC

JASMIN super cluster [RD-4][RD-5]. The main concepts of the elements used by Fire_cci and the Fire_cci-specific configuration and extension are also in the scope of this document.

This document currently covers the workflows of Fire_cci used to produce the datasets FireCCI51 and FireCCISFD20. It will be extended in the course of the project for the specifics of each sensor that will be included.

2.3. Applicable and reference documents

The following documents are applicable to this document:

[AD-1]	Climate Change Initiative Extension (CCI+) Phase 1 - Statement of Work, prepared by ESA Climate Office, Reference ESA-CCI-PRGM-EOPS-SOW-18-0118, Issue 1.0, date of issue 31 May 2018
[AD-2]	ESA Climate Change Initiative – CCI Project Guidelines. Ref. EOP-DTEX-EOPS-SW-10-0002, Issue 1.0, date of issue 05 November 2010, available at http://cci.esa.int/filedepot_download/40/4
[AD-3]	Data Standards Requirements for CCI Data Producers, CCI-PRGM-EOPS-TN-13-0009, Issue 2.0, date of issue 17/09/2018

The following documents are further referenced in this document:

[RD-1]	Pettinari M.L., Chuvieco E., Lizundia-Loiola J., Tanase M. (2019) ESA CCI ECV Fire Disturbance: D1.2 Algorithm Development Plan, version 1.1. Available at: https://www.esa-fire-cci.org/documents
[RD-2]	E. Chuvieco, M.L. Pettinari, A. Heil and T. Storm (2017) ESA CCI ECV Fire Disturbance: D1.2 Product Specification Document, version 6.3. Available at: https://www.esa-fire-cci.org/documents
[RD-3]	Lizundia-Loiola J., Pettinari M.L., Chuvieco E., Storm T., Gómez-Dans J. (2018) ESA CCI ECV Fire Disturbance: D2.1.3 Algorithm Theoretical Basis Document-MODIS, version 2.0. Available at: https://www.esa-fire-cci.org/documents
[RD-4]	CEDA archive, services and JASMIN help docs, available at https://help.jasmin.ac.uk/
[RD-5]	IBM Platform LSF documentation, available at https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_welcome.html
[RD-6]	MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat, 2004
[RD-7]	Bastarrika A., Roteta E. (2018) ESA CCI ECV Fire Disturbance: D2.1.2 Algorithm Theoretical Basis Document-SFD, version 1.0. Available at: https://www.esa-fire-cci.org/documents

3. System overview

This section provides an overview of the Fire_cci system with its system context, its main function and processing chain, and its architecture. Also, a set of system requirements are derived from the requirements in [AD-1] as starting point of the system design and later verification.

3.1. Fire_cci system context

The Fire_cci system's two main counterparts are the Fire_cci team, and on the other side data providers. There are a few more entities the Fire_cci system interacts with as shown in Figure 3.1.

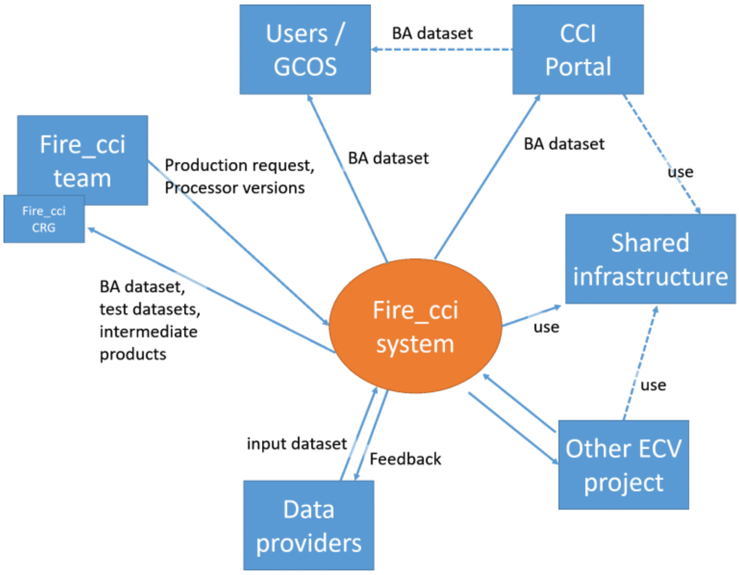

Figure 3.1: System context of the Fire_cci system with team, data providers, and other entities

The entities of the context are:

- The Fire_cci team as provider of processors, high level requests to produce certain results and feedback, and as first consumer of results, among them test datasets and intermediate results for assessment. Note that for this version of the system it is assumed that the Fire_cci team provides the validated products to users (not the processing system). The Fire_cci CRG, as part of the Fire_cci team, contributes to product quality control and assessment by intercomparisons.
- Data providers of the required satellite input data, in particular ESA.
- Other ECV projects as both providers of certain auxiliary datasets, among them the CCI Land Cover product (LC_cci) for a background land classification map, and also LC_cci as consumer of the BA product.
- The CCI portal as a distinct receiver of the validated BA product.
- Other data users, such as GCOS or ECMWF, especially in the context of the Copernicus Climate Change Service.
- The common part of the processing infrastructure (i.e. the Calvalus system of Brockmann Consult, and JASMIN).

The interfaces towards these entities partially determine the Fire_cci system in addition to its main function that is subject of the next section.

3.2. Main function and processing chain

The main function of the Fire_cci system is the repeated generation of the BA products with different input datasets and algorithms which undergo phased updating.

Functions of data management as well as sharing help to cope with the large amount of input data. Functions for processor integration, configuration control and versioning help to support the continuous development of the Fire_cci project.

3.3. System requirements

System requirements are usually derived from use cases, user requirements, and other inputs on the basis of a first decomposition of the system into elements. In the CCI projects, the focus is mainly on the products to be generated, rather than on the system, which is only the means to produce the product but not a deliverable itself. Therefore, the main requirement for the system is to generate the product according to the ATBD and the ADP.

This leads to a first set of system requirements, as detailed in Table 3.1.

Table 3.1: System requirements in the SoW

ID	Title	Requirement
010	Reprocessing capability	The Fire_cci system shall be able to (re)process complete missions of Earth Observation (EO) data.
020	Processor improvement cycle	The Fire_cci system shall support the improvement cycle of processors and provide configuration control for processor versions
030	Sentinel data handling	The Fire_cci system shall ingest and process Sentinel data
040	Initial capacity and resources	The Fire_cci system shall provide sufficient space for the input and output data, and provide sufficient 200 compute cores for concurrent processing.
050	System scalability	The Fire_cci system shall be scalable by adding new hardware, without change in the architecture.
060	Additional missions	The Fire_cci system shall be extendable for additional missions by adding the corresponding processors

The ADP [RD-1] and PSD [RD-2] identify and specify two user products to be generated:

- The "Pixel BA product"
- The "Grid BA product"

The products shall further be compliant with the CCI data guidelines [AD-3] regarding format (NetCDF-CF), naming, and metadata. This leads to another set of system requirements, detailed in Table 3.2.

Table 3.2: System requirements specified in the ADP/PSD and CCI data guidelines

ID	Title	Requirement
110	Pixel BA product	The Fire_cci system shall generate the Pixel BA product according to the specification in the ADP/PSD.
120	Grid BA product	The Fire_cci system shall generate the Grid BA product according to the specification in the ADP/PSD.
130	CCI data guideline compliance	The Fire_cci system shall generate NetCDF products compliant in naming, metadata and format to the CCI data producer recommendations and to the CF convention.

The ATBD [RD-3] identifies algorithms to be integrated as processors into the Fire_cci processing system:

- Pre-processing processors
- BA processors

This leads to another set of system requirements, detailed in Table 3.3.

Table 3.3: System requirements specified in the ATBDs

ID	Title	Requirement	Source
210	Pre-processing processors	If necessary, the Fire_cci system shall integrate processors for pre-processing to generate surface reflectance (SR) products from Level 1 inputs according to the ATBD.	[RD-3]
220	BA processors	The Fire_cci system shall integrate BA processors to generate BA products from SR products according to the ATBD.	[RD-3]

4. Fire_cci system architecture

This section describes the Fire_cci system architecture. It first provides an overview over the high-level decomposition (Section 4.1) and provides a justification why the decomposition is needed. In the later sub-sections of this section, the components are described in-depth: section 4.2 explains the Calvalus sub-system, and section 4.3 explains the JASMIN sub-system.

4.1. High-level system decomposition

In the previous contract, the BC Calvalus system has been used to fully produce the MERIS-based BA dataset (FireCCI41) and the Sentinel-2 based BA dataset (FireCCISFD11), in both cases starting from the L1 input data, generating the final user products. Also, because the MapReduce approach (see [RD-6]) employed by Calvalus is an excellent fit to the formatting of the BA data to the final user products, this last step has been performed on Calvalus for all of the Fire_cci datasets processed by BC, which are FireCCI41, FireCCI50, FireCCI51, FireCCISFD11, and FireCCILT10. In order to produce the BA dataset based on MODIS input data, the STFC JASMIN system[1] has been used, due its larger bandwidth, which allowed to download the large data amounts through the internet. Thus, the overall Fire_cci system is composed from both the Calvalus system and JASMIN.

The following sections address the architectures and technical concepts and methods used for each sub-system.

4.2. Calvalus system

The Fire_cci system is to a large extent based on Calvalus and the underlying infrastructure, with certain elements specifically configured and extended for Fire_cci (Figure 4.1). In addition to the software stack with many functions of the Fire_cci system provided by Calvalus, Hadoop or other frameworks, there is a shared cluster infrastructure that runs the corresponding services.

[1] http://www.jasmin.ac.uk/

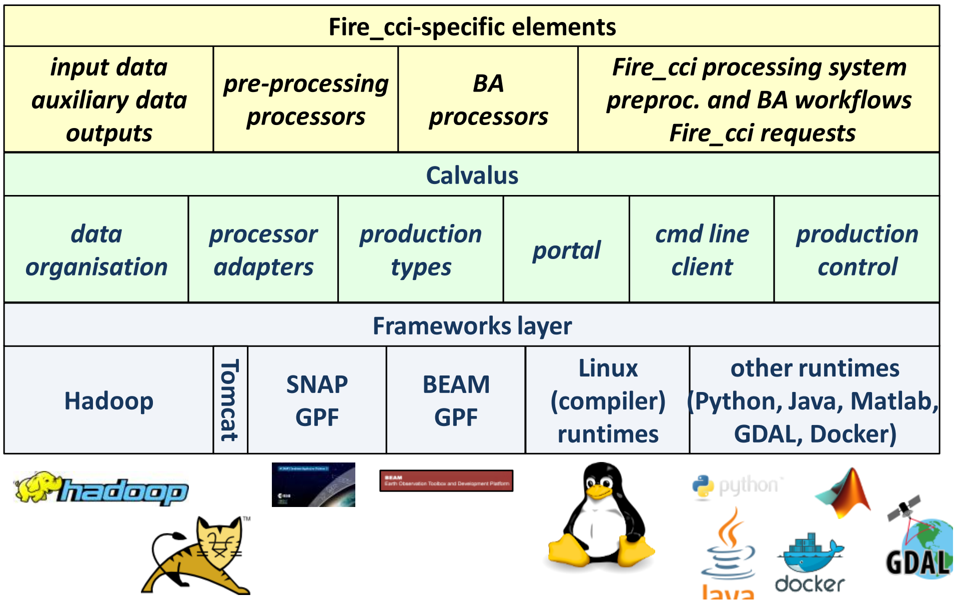

Figure 4.1: Fire_cci system architecture layers with specific elements, Calvalus, and Hadoop

The main elements that are specific to Fire_cci are:

- Access to the shared input data, space for auxiliary data and Fire_cci project intermediates and outputs.
- Processors or Fire_cci-specific processor configurations of existing processors for pre-processing.
- The BA processors and their wrappers for integration into Calvalus.
- The Fire_cci processing system instance on a virtual machine for the control of all Fire_cci workflows, with rules and request templates for pre-processing, BA processing, and formatting and all the respective dependencies.

In addition to this there are several Fire_cci-specific configurations of shared elements, in particular:

- A “fire” queue in the Calvalus Hadoop scheduler for the adequate and fair sharing of the cluster processing resources.
- space on the shared Brockmann Consult’s (BC) FTP server

The main shared elements and functions used from Calvalus and underlying frameworks are:

- The Calvalus data organisation for EO data, auxiliary data, and project-specific data, and the corresponding functions of the Hadoop Distributed File System (HDFS) for *data management*.
- The Calvalus production control tool *pmonitor* for the control of processing chains and bulk production, in combination with the Calvalus production types for massive-parallel processing, and Hadoop HDFS and YARN for fair scheduling, load balancing, and robust failure handling for different layers of *production management*.
- The Calvalus processor adapters to wrap the Fire_cci processors, in particular SNAP for the processors for pre-processing, and the corresponding automated deployment and runtime provisioning for *processor integration*.

- The Calvalus framework for MapReduce, which employs the Hadoop MapReduce framework, and provides the basis for generic MapReduce algorithms. This is used for the final *formatting* step.

How these elements and functions are used is described in more detail in Section 4.2.1.

4.2.1. Technical methods and concepts used in Calvalus

Calvalus is proprietary software developed by Brockmann Consult GmbH since 2010. It employs the Apache Hadoop software and adapts it to the processing of Earth Observation data. It provides the possibility to perform full mission EO data processing, data aggregation, validation, and value-adding with many frequently updated algorithms and data processors on standard hardware scalable for the amount of data of Sentinel 1-2-3.

HDFS

HDFS, the Apache Hadoop distributed file system, is a distributed and highly scalable file system. A typical cluster running HDFS has a single master plus multiple datanodes. Each datanode stores blocks of data, and serves them over the network. This file system is based upon the Google File System (GFS), introduced by Google employees in 2003.

HDFS is designed to store large files, typically in the range of gigabytes to terabytes across multiple machines. This data is replicated across multiple nodes in order to ensure data availability. Data nodes communicate in order to rebalance data, to move copies around, and to keep the replication of data.

The main advantage of HDFS is that it allows for data-local processing (see Figure 4.2 and Figure 4.3). This means that the scheduler distributes processing jobs to nodes with an awareness of the data location. For example: if node A contains data X and node B contains data Y, the job tracker schedules node B to perform processing tasks on Y, and node A scheduled to perform processing tasks on X. This considerably reduces the amount of traffic that goes over the network.

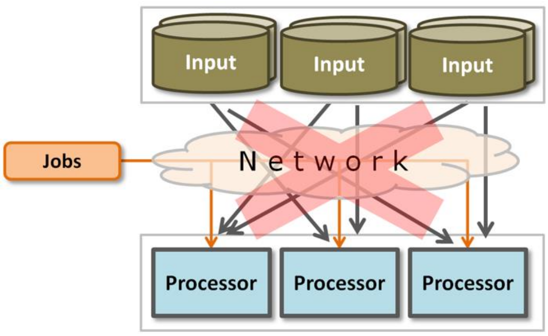

Figure 4.2: Archive-centric

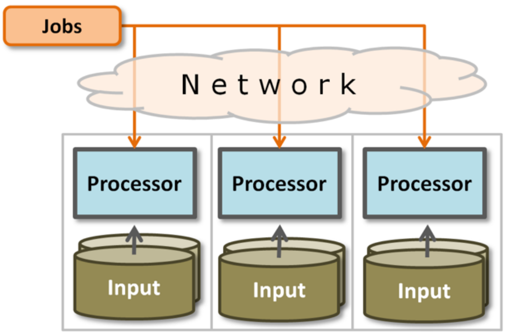

Figure 4.3: Data-local

Software bundles

A software bundle is a Calvalus concept that allows system operators to deploy any runnable software to the system. The processors used in Fire_cci, which are provided by project partners, are integrated into software bundles, and as such integrated into the system.

Hardware Infrastructure

The infrastructure used for Fire_cci processing is a Calvalus/Hadoop cluster consisting of computing hardware, storage, input/output elements, and network. This cluster is shared with other projects that each provide parts of the hardware and get in turn a corresponding share of the resources of the cluster. The current cluster has 113 processing nodes and a storage capacity of about 2.4 PB.

Figure 4.4 shows the layout of the Calvalus/Hadoop cluster with master node, computing and storage nodes, the feeder as input/output element, and an optional test server for services and development. The computing and storage nodes are simple computers with local disks. These disks together are the distributed storage of the cluster managed by the Hadoop Distributed File System.

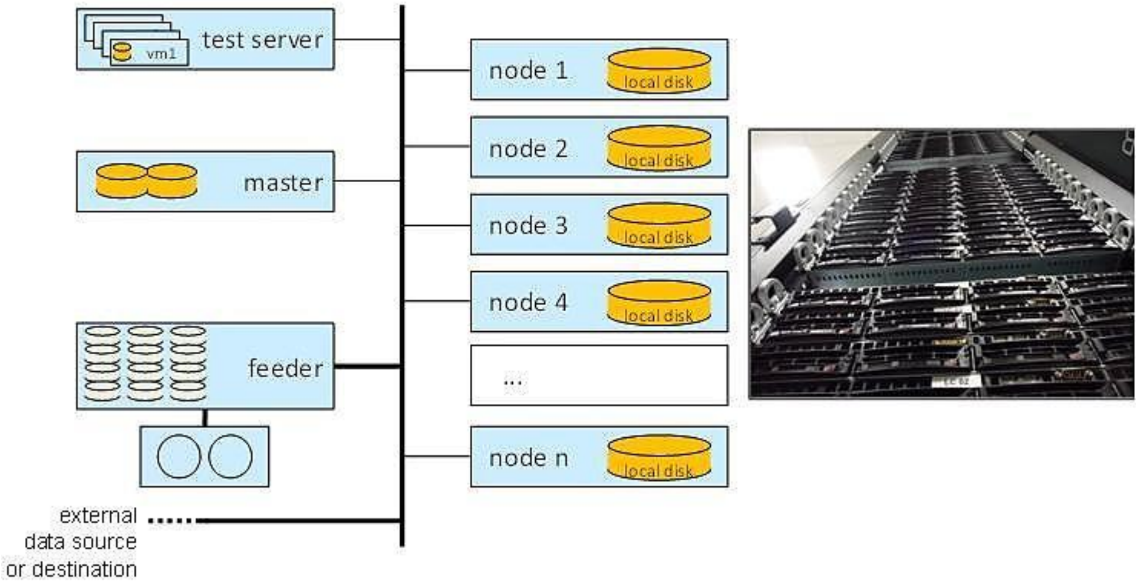

Figure 4.4: Calvalus architecture

This approach of Hadoop as middleware and with computing nodes that are at the same time storage nodes has been selected for Fire_cci because the knowledge of the location of the respective data allows for data-local processing with minimal use of the network, which makes the approach suitable for massive parallel processing of the large data volumes of pre-processing. The approach is scalable to several petabytes which is an advantage regarding the considerable data volumes needed for Fire_cci processing.

Software infrastructure

The software system for Fire_cci processing deployed on the infrastructure above is shown in Figure 4.5. The different layers of the software system start from the operating system up to the individual processors.

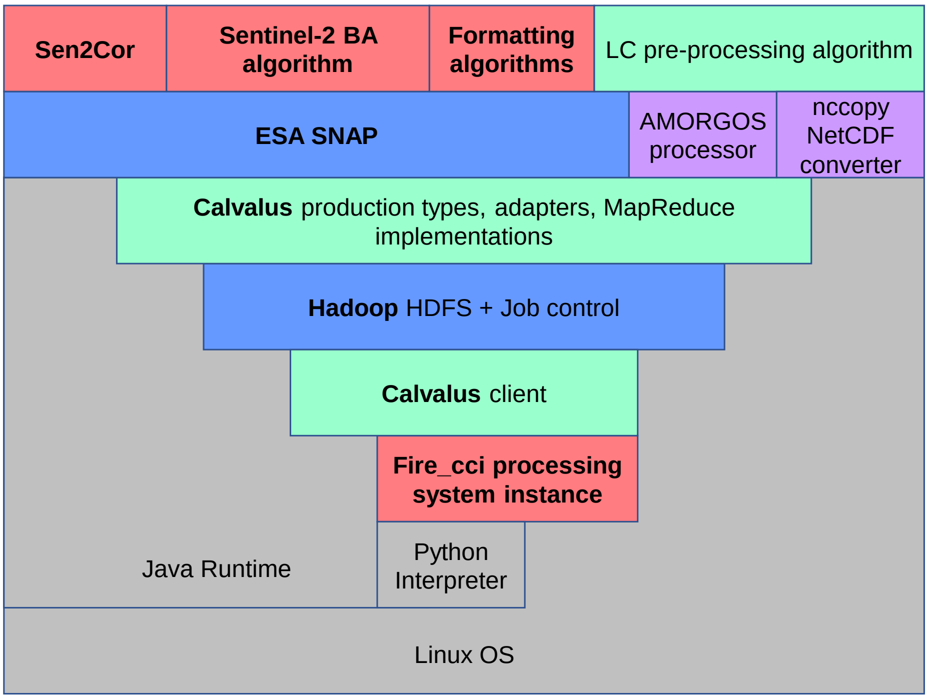

Figure 4.5: Software architecture

The layers are:

- Basic software (shown in grey) comprises the Linux operating system, the Java runtime, and the Python interpreter.
- Calvalus (shown in green) provides several layers, with a client layer that uses Hadoop and a layer plugged into the Hadoop framework with production types and adapters, both together implementing the Earth Observation functions not available in bare Hadoop. This part also contains the formatting tool MapReduce implementation.
- Apache Hadoop with its distributed file system and its job scheduling functions and cluster tools as well as ESA SNAP with its graph processing framework are layers used by Calvalus for cluster processing. They are shown in blue.
- Third-party data processors used within the Fire_cci processing are AMORGOS and a NetCDF tool shown in pink. They are integrated as Unix executables into Calvalus for parallel execution.
- Specific elements developed or assembled for Fire_cci (specifically for FireCCISFD11) are Sen2Cor (used for Sentinel-2 pre-processing), the Sentinel-2 BA algorithm, the formatting software, and the Fire_cci processing system instance. They are shown in red. They will get updates to cover additional sensors.

Like the hardware, the software is also for the most part shared with other projects. But this needs more precise consideration. It is not always the same version of a software item that is used by different projects. Therefore, the versioning approach has a key role in the software deployment and runtime use for Fire_cci in the Calvalus environment:

- Several versions of software items can be deployed in this environment at the same time. The actual requests generated by the processing system instance determine which combinations of versions are actually used. This is the case for processors (upmost layer) but also for ESA SNAP and Calvalus and the processing system, optionally also for Java and Python. Only the operating system and the Hadoop version being used is a single one at a time.

- The software items are versioned and the versions are managed in version control systems. For software items developed by Brockmann Consult the version control system is Git. The software repositories are hosted on GitHub, a cloud service. Some of the repositories are public, e.g. the one for SNAP which is open source.
- Other software items are version-controlled externally. Examples are Apache Hadoop maintained as open source by Apache Foundation, the programming language runtimes, or the AMORGOS processor provided by ESA and maintained by the company ACRI-ST.

Table 4.1 lists the software items with their role in the Fire_cci processing subsystem and its configuration control.

Table 4.1: Software items for Fire_cci processing

Software item	Purpose, use	Configuration control
Java runtime environment	Basic software used for Hadoop, Calvalus, ESA SNAP, some processors	Oracle SDK, Version 8 (1.8.0_40-b25), available from http://www.oracle.com/technetwork/java/index.html[2]
Python interpreter	Basic software used for pmonitor	Version 2.7.6, available from www.python.org
Apache Hadoop	Cluster software with data management HDFS, job scheduling, command line tools and Web operating interface, several APIs: client side for job submission and monitoring, server side for plug-in of map and reduce tasks.	Version 3.2.1, versions maintained by Apache, available from hadoop.apache.org[2] (open source)
Calvalus	Earth Observation application layer on top of Hadoop with HDFS archive directory structure with archiving rules, client tool for request submission, software deployment, data ingestion processor adapters for various programming languages (among them for BEAM/SNAP operators processor deployment as versioned software bundles). User portal for on-demand processing. Processing system instances for controlled bulk production.	Version 2.19, version control on GitHub in private repository of Brockmann Consult GmbH
ESA SNAP	Framework for processor development and execution, concepts for product, reader, writer, operator, operator chaining, tile cache and many more, basic software for LC pre-processor, Level 3 aggregator, and overlap determination.	Version 7.0, provided by ESA, maintained and further developed by Brockmann Consult GmbH, version control in public repository of Brockmann Consult GmbH (open source).
Daily quicklook aggregation	Generates quicklooks with global extent of the input products of a day, uses L3	Maintained as part of Calvalus.

Software item	Purpose, use	Configuration control
	aggregation with sub-sampling, and final JPEG formatting. Used for input screening for quality checks.	
AMORGOS	Improved geocoding of MERIS products based on DEM, attitude and orbit files and knowledge on acquisition features, adds corrected Lat and Lon bands to MERIS products. It may be used for the Sentinel-3 products	Version 4.0p1, provided by ESA, maintained by ACRI-ST.
Fire Level 3 aggregator (production type "FireL3ProductionType")	Aggregation of surface directional reflectance (SDR) values, reprojection, application of a temporal cloud filter, tiling and NetCDF-formatting of the result.	Maintained as part of Calvalus.
Fire SR product writer	This SNAP product writer implementation is a modification to the LC_cci product writer, which in turn extends the NetCDF 4 CF writer to set band names and attributes, global attributes, and the file name according to the LC_cci product specification. It is used in the generic Calvalus product formatting step.	Maintained as part of Calvalus.
Fire Formatting Tool	Uses the burned area data in order to create ADP/PSD-compliant, formatted final user data sets.	Maintained as part of Calvalus.
nccopy NetCDF conversion tool	Used to convert SNAP-generated NetCDF 4 files into classic format as required by CCI product convention	NetCDF library version 4.1.3, available for Ubuntu via apt-get.
Quicklook generator	Uses the generic L3 workflow to mosaic the tiles into one quicklook JPEG image.	Maintained as part of Calvalus.
fire-inst	Processing system instance with workflow control and processing progress and status.	Automated daily backup, no version control, instead re-configuration for each production run.
fire-1.x	Software bundle containing the BA processor and the environment it uses.	Versioned; each update gets a new version, while old versions are kept and can be re-used.

4.3. JASMIN system

JASMIN is described in-depth in [RD-4]. It employs the LSF software for scheduling and running processing jobs, which is described in [RD-5]. From a Fire_cci perspective, it is vital that the system is able to ingest and handle MODIS SR product data (MOD09GQ, MOD09GA), to perform the BA retrieval on these inputs, and to allow to transfer the results to Calvalus, where the final user products are generated.

4.3.1. Technical methods and concepts used in JASMIN

This section describes the environment on JASMIN from a client's perspective only, as this is the perspective we used in the Fire_cci project.

Hardware environment

As described in [RD-4], the cluster consists of 218 processing hosts with 4764 CPUs and > 50 TB physical RAM.

Job submission and processing environment

JASMIN offers to submit commands to the LOTUS cluster, which are scheduled and executed by that engine. The commands that are submitted may be any executable which is able to run within the prepared environment. Hence, in order to run a specific task (such as the burned-area-retrieval for MODIS), it is necessary that the executable is prepared in the user's home directory, and it is ensured that all shared libraries are available. If these prerequisites are fulfilled, the execution command may be submitted to the LOTUS cluster.

5. Workflows

This section describes the processing workflows in detail, the data they use, the techniques employed, and the outputs they create.

5.1. MODIS workflow

There is a sub-section for each processing step: Section 5.1.1 deals with the data acquisition, Section 5.1.2 explains the MODIS BA retrieval, Section 5.1.3 describes the data repackaging and transfer, and Section 5.1.4 describes the final step, the formatting.

The basis for the processing is already pre-processed data, which means that no pre-processing is done within the context of Fire_cci. Second, the input dataset has not been available on the system at the start of the processing, meaning that it had to be acquired first. Two datasets needed to be acquired to the system for processing: MOD09GQ (daily surface reflectance data, global, 250m, 1200*1200 km tiles), and MOD09GA (daily surface reflectance data, global, 1km and 500m SIN Grid).

A general issue was the limited space of the platform. As the whole input data amount sums up to about 200 TB data, while only 84 TB are available on the platform, and ~10 TB of space had to be left for temporary files and results, the processing had to be done in cycles. So, we have split the processing into continent datasets (Africa, Asia, Australia + Europe, North America, South America), and used the following pattern:

1) process tiles of continent A

2) in parallel, download tiles of continent B

After 1) and 2) are done:

3) remove tiles of continent A

4) process tiles of continent B

5) in parallel, download tiles of continent C

… and so on, until all continents are processed.

Figure 5.1 shows the logical ordering of the workflow; the single steps are explained in the upcoming subsections.

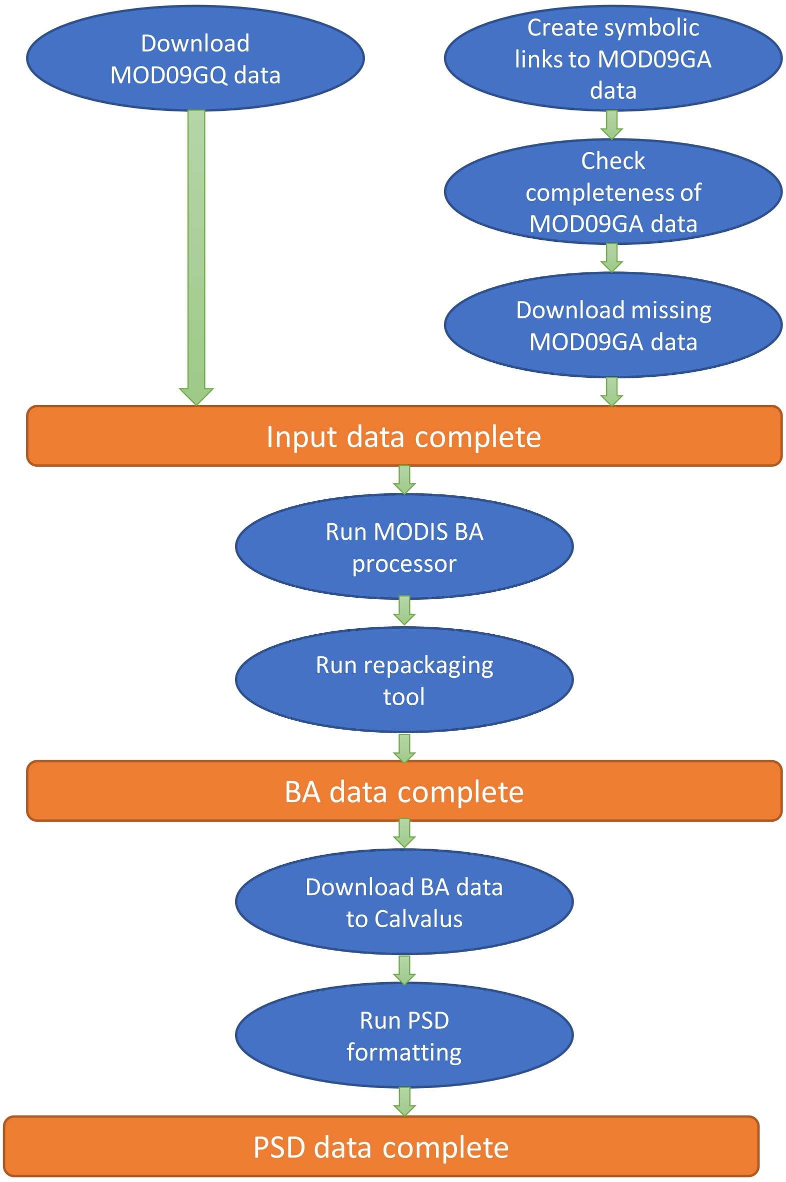

Figure 5.1: Sequence of MODIS processing actions

5.1.1. Data acquisition

As stated before, there are two datasets which needed to be acquired to the system for processing: MOD09GQ (daily surface reflectance data, global, 250m, 1200*1200 km tiles), and MOD09GA (daily surface reflectance data, global, 1km and 500m SIN Grid).

In order to reliably download the MOD09GQ dataset, a python script has been set up, which allows to start multiple (typically ~40) download jobs in parallel, tracks successfully downloaded files, compares checksums of the downloaded files with the checksums provided by the download site, and reports failing download attempts. Thus, it is able to handle broken connections, incomplete downloads, and the regular and irregular system downtimes. This script has been run on a dedicated transfer node of the JASMIN system, which allowed for high download rates.

Large parts of the MOD09GA dataset had already been acquired by another working group and could be re-used. Hence, in order to acquire the MOD09GA dataset, three steps were necessary:

1) Create symbolic links in the filesystem, so that the existing MOD09GA files are visible to the BA retrieval process. In order to do that, a simple bash-script has been created and run once.
2) Identify the missing MOD09GA files, as the archive turned out to have small gaps, and was also missing all data from 2016 on completely. This could also be done with a short bash-script, which checked the existence of input files against the expected list of input files.
3) Download the missing MOD09GA files. In order to do that, a slightly adapted version of the MOD09GQ download script has been used.

5.1.2. Data processing

The MODIS BA processor has been described in [RD-3].

In order to run the MODIS BA processor on JASMIN, the environment had to be prepared accordingly. The MODIS BA processor has been written in Python2, and has a set of dependencies which are not all fulfilled by the Python version already installed on JASMIN, such as `gdal`, `ogr`, `scipy.spatial`, or `cv2`. Thus, another version of Python2 has been prepared, which contains all the needed dependencies. Prior to the processing execution, a dedicated activate-script has to be called, which sets up the environment and ensures that the correct python version is being used.

The processing has been split up into MODIS tiles and into years, which leads to a large number of processing tasks. In principle, the LOTUS system accepts the submission of a large number of processing tasks at once, but in order to keep the Calvalus way of monitoring, it was decided that the submission of jobs was done by a python script using the same monitoring approach as the MERIS processing. This required the implementation of three dedicated scripts, ordered bottom-up:

1) **run.sh**. This bash script is the wrapper around the MODIS BA processor. It sets the environment (`source mypython/bin/activate`), runs the python executable with the main function of the BA processor for the given tile and the given year (`mypython/bin/python2.7 modis-proc/MAIN.py $tile $year`), and returns the exit code of the BA processor to the start.sh script.
2) **start.sh**. This bash script does the actual submission of a processing job to the LOTUS cluster by submitting the run.sh script with the necessary parameters (`bsub -o $logfile -W 24:00 -M 20971520 ./run.sh $tile $year`). Furthermore, it checks once every five minutes if the job is finished; if it is finished, it checks if the job has finished successfully. If so, it exits with exit code 0, signaling success, otherwise it archives the log file so that it may be analysed, and exits with exit code 1, signaling failure.
3) **modis.py**. This script controls, starts and monitors the execution of the start.sh script. According to its exit code, it marks processing attempts of tile and year as successful or as failure.

5.1.3. Data repackaging and transfer

Since the ADP/PSD formatting is done using the Calvalus system instead of JASMIN, the data needs to be downloaded to Calvalus. The data produced by the MODIS BA processor contains a high number of image files, which are not used for ADP/PSD product creation, but serve as temporary files or are used for further analysis. In order to facilitate the download to Calvalus, it is repackaged; this means that the image files relevant for ADP/PSD-compliant product creation (= burned area, uncertainty, and fraction of observed area) are put together into an internally compressed NetCDF-file for each month and tile. In order to achieve this repackaging, a Java tool has been written, based on the

SNAP library. It accepts as input the result of the BA processor for a given tile, year, and month, and writes as output the internally compressed NetCDF-file. Also, a bash script has been developed, which acts as wrapper around that Java tool and ensures that all results are considered, and that the NetCDF files are put into the download area.

On the Calvalus side, a script similar to the script described in Section 5.1.1 has been put into place, which is responsible for controlling the download of the data.

5.1.4. ADP/PSD product production

The ADP/PSD-compliant formatting of the BA data has been implemented in order to exploit the massive parallelity of the Calvalus system. This implementation has been employed to generate the user data compliant to the ADP/PSD. The formatting step works on single months, and takes as input all global tiles for each month.

Pixel processing

Due to the higher resolution of the resulting products, the pixel processing has higher demands on memory consumption. So, the process has been split up twice: the first step computes the output variables separately in a MapReduce step, and the second step puts them together (Figure 5.2). The process produces data for a single month and one of the six target areas defined in the ADP/PSD.

In the first step, a mapper implementation takes each tile and creates the respective output for one of the three output variables day of year, confidence level, and LandCover class. So, it is run three times for each tile in parallel. The pixel output is generally equivalent to the input.

The PixelReducer runs once for each output of the mapper. It creates actual files in the file system (as opposed to the PixelMapper, which keeps the data in memory). Finally, the PixelMergeMapper collects these data, and creates the result product and the respective metadata.

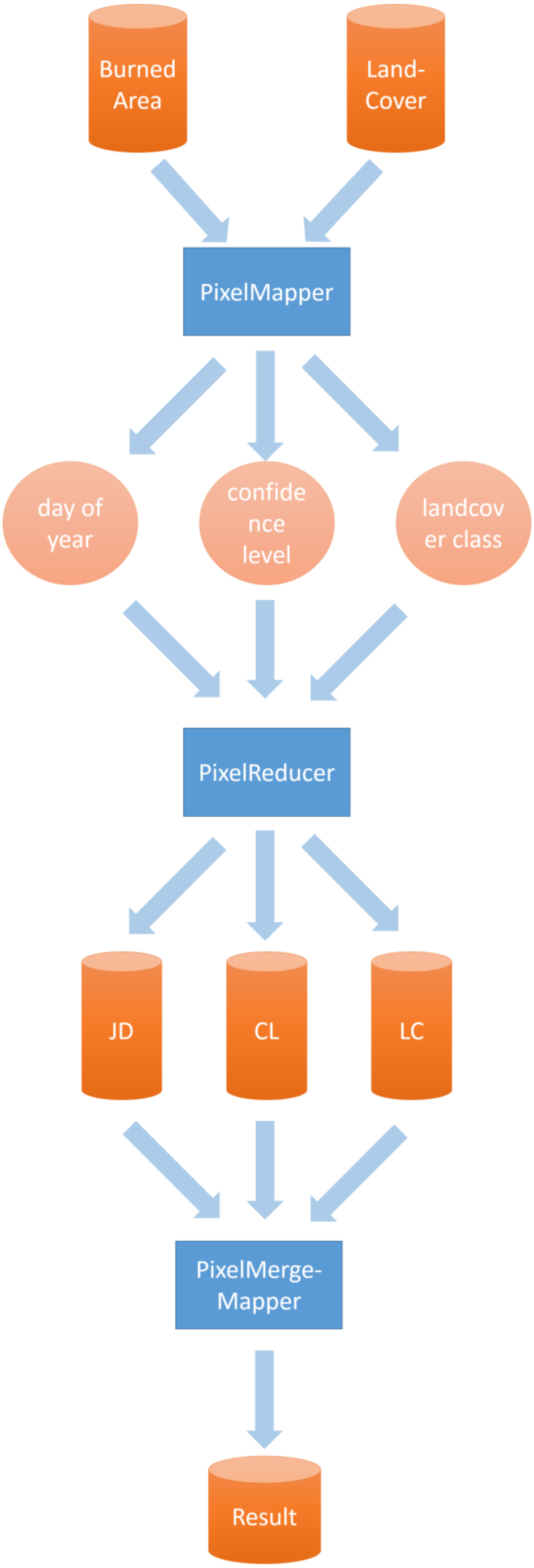

Figure 5.2: Formatting

Error handling in the grid processing is done using the Calvalus system's error handling protocol. Generally, no errors are accepted, as the process is expected to successfully use all available burned area data. Consequently, the overall failure rate of the process is 0%.

Grid processing

The grid processing basically consists of both a mapper and a reducer implementation. The mapper implementation works in parallel on each available tile; it does the aggregation of the burned area and computes the errors as well as the patch numbers, the fractions of observed area, and the burned area per Land Cover class. Since there are two grid products for each month, the mapper aggregates the data accordingly.

The reducer implementation retrieves the output of all the mappers, creates the two result files (one for each half of the respective month), fills them with metadata, and writes the mapper's output into them. These result files are archived afterwards, ready to be delivered to the users. This approach allows for a high degree of parallelisation.

Error handling in the grid processing is done using the Calvalus system's error handling protocol. Generally, no errors are accepted, as the process is expected to successfully use all available burned area data. Consequently, the overall failure rate of the process is 0%.

5.2. SFD workflow

The workflow for producing the Small Fire Database (FireCCISFD20) will be run on the CreoDIAS[2] platform, as this platform already contains all the needed Sentinel-2 A and B L2A input data, and offers the processing capabilities to compute the whole dataset. The platform also offers a cloud infrastructure; the software part of the BC Calvalus processing system has been deployed on this infrastructure, so from a high-level perspective, the processing system is identical to the Calvalus processing system described in Section 4.2.

The workflow is depicted in Figure 5.3. The single steps are discussed in the upcoming sections.

The FireCCISFD20 production is controlled by the Fire_cci processing system instance. A control script is configured for the pre-processing as well as BA retrieval and aggregation. It can be parameterised for the time period (full time series, or only some months), and optionally the steps to be performed; e.g. only pre-processing may be done on demand. The production may be interrupted and resumed, and steps may be repeated if required.

---

[2] https://creodias.eu/

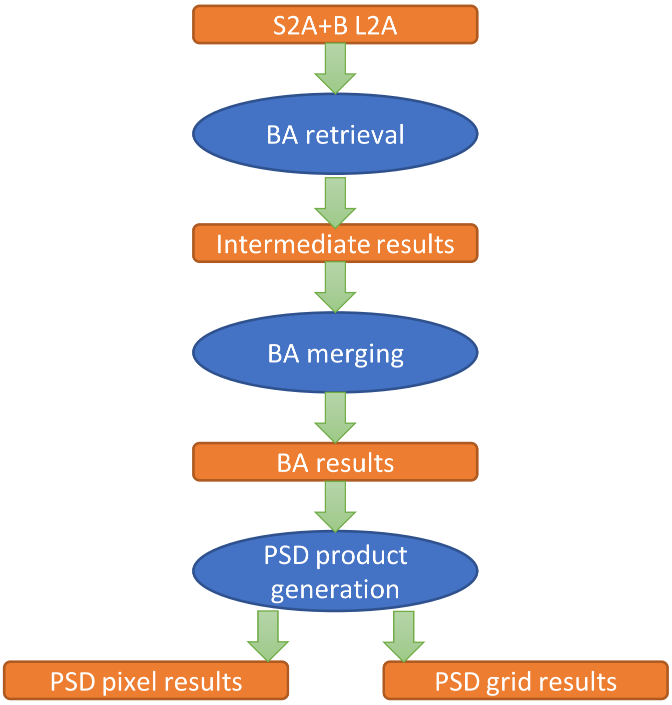

Figure 5.3: Sentinel-2 workflow

5.2.1. BA retrieval

The Burned Area retrieval for Sentinel-2 is done using the Sentinel-2 BA algorithm developed by EHU (see [RD-7]), which has originally been written in Python. In order to facilitate running the processor on Calvalus, it has been re-written in Java. It uses some geographical algorithms, which are not available by common Java libraries, so these had to be implemented as well.

The algorithm itself runs for a single-tile of input data produced by the pre-processing. First, it creates an intermediate NetCDF file containing the BA retrieval of a single Sentinel-2A/B L2A input tile. The latest four outputs for each tile are then merged afterwards, in order to give consistent images.

Figure 5.4 displays the software actors in the Sentinel-2 BA retrieval: the core software uses the Java Runtime libraries. Some generic libraries had to be written as well:

- the connected component library, which finds patches of equal pixels
- the ConvolveOperator library, which allows to run an image convolution with an arbitrary kernel
- the DilateOperator library, which is used to buffer the active fires to 500m
- the HoleRemover library, which allows to remove cloudy areas of an arbitrary maximum size

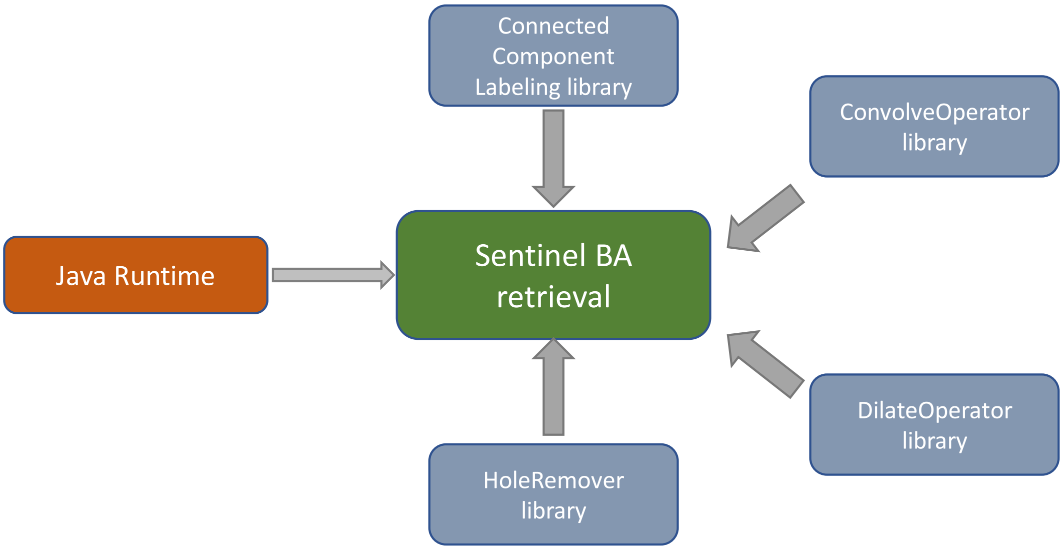

Figure 5.4: MERIS BA retrieval

In order to set up the environment, a Calvalus Production Type (Fire-S2-BA) se been created. The Calvalus Production Type predominantly takes care of providing the input data in the expected structure, and collecting the output data after the processing has been done. If errors occur, the process fails, so it can be re-done, and it is configured to automatically try four times before ultimately failing.

The production is controlled by the Fire_cci processing system instance. A dedicated control script is configured for the burned area retrieval, which can be parameterised for the time period (complete mission or only some months). The production can be interrupted and resumed, and steps can be repeated if required.

The error handling is established by the same control script. If the error message is memory-related, the process has failed due to the processing environment on the specific node. The reaction is to consider the attempt a failure, and re-try the same processing step on a different node.

5.2.2. Grid/pixel formatting

For the creation of the ADP/PSD-compliant products the same approach is used for Sentinel-2 as it is for MODIS data, which is described in section 5.1.4.

6. System outputs

The system, in the version described in this document, produces the datasets stated in Table 6.1.

Table 6.1: Output products

	MODIS		Sentinel 2	
	Pixel product	**Grid product**	**Pixel product**	**Grid product**
Geographic extent	Continental tiles divided in 6 zones: Africa, Asia, Australia, North+Central America, South America and Europe	Global	Sub-Saharan African area corresponding to Zone 5 of the continental tiles, provided in 2°x2° tiles.	Global, but data only provided over the Sub-Saharan African landmass
Temporal extent	2001-2019		2019	

Annex: Acronyms and abbreviations

AD	Applicable Document
ADP	Algorithm Development Plan
AMORGOS	Accurate MERIS Ortho-Rectified Geo-location Operational Software
API	Application Programming Interface
ATBD	Algorithm Theoretical Basis Document
AVHRR	Advanced Very High Resolution Radiometer
BA	Burned Area
BC	Brockmann Consult GmBH
BEAM	Basic ERS & Envisat (A) ATSR and Meris Toolbox
CCI	Climate Change Initiative
CEDA	Centre for Environmental Data Analysis
CF	Climate Forecast
CPU	Central Processing Unit
CRG	Climate Research Group
DEM	Digital Elevation Model
ECMWF	European Centre for Medium-range Weather Forecast
EHU	University of the Basque Country
EO	Earth Observation
ESA	European Space Agency
ECV	Essential Climate Variables
FTP	File Transfer Protocol
GCOS	Global Climate Observing System
GDAL	Geospatial Data Abstraction Library
GFS	Google File System
HDFS	Hadoop Distributed File System
IPCC	Intergovernmental Panel on Climate Change

JPEG	Joint Photographic Experts Group
LC	Land Cover
LC_cci	Land Cover CCI project
Lat	Latitude
Lon	Longitude
LSF	Load Sharing Facility
MERIS	Medium Resolution Imaging Spectrometer
MODIS	Moderate Resolution Imaging Spectroradiometer
MSI	MultiSpectral Instrument
NetCDF	NETwork Common Data Format
OLCI	Ocean and Land Colour Instrument
PB	Peta Byte
PSD	Product Specification Document
SDK	Software Development Kit
SDR	Surface Directional Reflectance
SFD	Small Fire Database
SIN	Sinusoidal
SLSTR	Sea and Land Surface Temperature Radiometer
SNAP	Sentinel Application Platform
SoW	Statement of Work
SR	Surface Reflectance
SSD	System Specification Document
STFC	Science and Technology Facilities Council
TB	Tera Byte
UAH	University of Alcala
YARN	Yet Another Resource Negotiator