CTA Real-Time Analysis Pipeline Configuration

N.Parmiggiani[(1)], A. Bulgarelli[(1)]

[(1)]INAF/OAS – Bologna, Via P. Gobetti 93/3, 40129 Bologna

Change history:

Version	Date	Notes
1.0	Mar 4th, 2019	
1.1	Sep 10th, 2020	Revised and updated
1.2	Oct 29th, 2020	Referee review

Table of contents

1. Introduction

At OAS-Bologna the Team working on CTA Science Alert Generation work package has developed different software prototypes.

This software can be downloaded with this setup repository https://github.com/cta-rta/rta-sci-cta-Setup

2. Containers: layers

CTA pipelines prototypes are deployed into a Singularity container environment.

To build the containers, download this repository: https://github.com/ASTRO-EDU/CTA-containers

Each container has four files.

- .build is used to build a layer.
- .start file is used to start a container and contains different commands.
- .simg is the container image.
- .recipe contains all commands needed to build the container

The container environment is created with different layers, each with a specific function.

1) Operative system and base libraries
2) Science tools installation
3) System services
4) Dummy
5) Pipeline installation
6) Layer with a specific configuration for a particular machine (IP, port etc)

To build the container, the user can be root inside the machine and can use the script buildall.sh, which contains a list of commands. Each command runs the build of a single layer.

sh cta_layer1_core.build
sh cta_layer2_scitools.build
sh cta_layer3_service.build
sh cta_layer5_instance.build
sh cta_layer6_[machine]_[user]_instance.build

3. CONFIGURE CTA RTA PIPE INSIDE THE CONTAINER FROM SCRATCH - FULL PROCEDURE

3.1. Configure the container environment

```
root@machine# vim /opt/prod/modules/cta_rta_pipe_build[build number]
```

```
source /opt/module/anaconda-3.6
source /opt/module/ctools-1.5.2
export PIPELINE=/opt/prod/CTA_BUILD2/
export PYTHONPATH=/opt/prod/CTA_BUILD2/pipeline_manager:$PIPELINE:$PYTHONPATH
```

Download repository, starting from the container directory

```
rt@machine $ cd tmp_download
rt@machine $ git clone https://github.com/cta-rta/rta-sci-cta-Setup.git CTA_BUILD2
rt@machine $ cd CTA_BUILD2
rt@machine $ git checkout [build]
rt@machine $ cp conf.py.default conf.py

Setup configuration

rt@machine $ vi conf.py

# copy this file to "conf.py" and insert parameters

def get_pipedb_conf():
    host = 'localhost'
    username =
    password =
    database = 'cta_pipe_db'
    port =
    return {'host':host,'username':username,'password':password,'database':database,'port':port}

def get_resultsdb_conf():
    host = 'localhost'
    username =
    password =
    database = 'cta_results_db'
```

```
    port =
    return {'host':host,'username':username,'password':password,'database':database,'port':port}

def get_evtdb_conf():
    host = 'localhost'
    username =
    password =
    database = 'evt_db'
    port =
    return {'host':host,'username':username,'password':password,'database':database,'port':port}

def get_path_conf():
    path_base_fits = '/opt/prod/CTA_BUILD2/CTAGammaPipeCommon/templates/base_empty.fits'

    return {'path_base_fits':path_base_fits}

def get_data_repository(instrumentname):
    if(instrumentname == "ASTRI"):
        datarepositoryid = '4'
    if(instrumentname == "CTA-SOUTH"):
        datarepositoryid = '1'
    return {'datarepositoryid':datarepositoryid}



rt@machine $ ./download.sh [username] [-t to donwload without tags]
rt@machine $ cd ScientificGUI
rt@machine $  cp config.php.default config.php
rt@machine $  vi config.php

Setup configuration

<?php

    // copy this file to config.php and change the configurations without committing the new file with
git.

    function getPipeConfig()
    {
            // db data
        $username = "";
        $password = "";
        $database = "cta_pipe_db";
        $database_results = "cta_results_db";
```

```
      $host = "localhost";
      $tmp_path = "/";
      $port = "";
      $root_results_dir = "/ANALYSIS3/";
      $web_results_dir = "../analysis/";

      $array =
array('database_results'=>$database_results,'root_results_dir'=>$root_results_dir,'web_results_dir'=>$
web_results_dir,'host'=>$host,'username' => $username,'password' => $password,'database' =>
$database,'tmp_path'=>$tmp_path,'port'=>$port);

      return $array;
   }

   function getEvtConfig()
   {
    // db data
      $username = "rt";
      $password = "";
      $database = "";
      $host = "localhost";
      $port = "";

      $array = array('host'=>$host,'username' => $username,'password' => $password,'database' =>
$database,'port'=>$port);

      return $array;
   }

?>
```

Create databases

```
rt@machine $ vi conf/init_db_script

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('[password]');

CREATE USER 'rt'@'localhost' IDENTIFIED BY '[passwprd]';
CREATE USER 'rt'@'%' IDENTIFIED BY '[passwprd]';

CREATE DATABASE IF NOT EXISTS pipe_manager;
GRANT ALL PRIVILEGES ON pipe_manager.* TO 'rt'@'%';
CREATE DATABASE IF NOT EXISTS rt_alert_db;
GRANT ALL PRIVILEGES ON rt_alert_db.* TO 'rt'@'%';
```

```
CREATE DATABASE IF NOT EXISTS rt_alert_results_db;
GRANT ALL PRIVILEGES ON rt_alert_results_db.* TO 'rt'@'%';

CREATE DATABASE IF NOT EXISTS cta_pipe_db;
GRANT ALL PRIVILEGES ON cta_pipe_db.* TO 'rt'@'%';
CREATE DATABASE IF NOT EXISTS cta_results_db;
GRANT ALL PRIVILEGES ON cta_results_db.* TO 'rt'@'%';
CREATE DATABASE IF NOT EXISTS evt_db;
GRANT ALL PRIVILEGES ON evt_db.* TO 'rt'@'%';

CREATE USER 'slurm'@'localhost' IDENTIFIED BY 'my';
CREATE USER 'slurm'@'%' IDENTIFIED BY 'my';

CREATE DATABASE IF NOT EXISTS slurm_acct_db;
GRANT ALL PRIVILEGES ON slurm_acct_db.* TO 'slurm'@'%';
GRANT ALL PRIVILEGES ON slurm_acct_db.* TO rt@'%';

START CONTAINER

rt@machine $ sh start_instance_agilehost2_rt.sh

NOW ENTER INSIDE THE CONTAINER

rt@machine $ sh shell_instance.sh

RTApipe $ cd /opt/start
RTApipe $ sh start_services.sh

Inside the start_services.sh, the following command is executed:
RTApipe $ #mysqld --initialize  --init-file=/opt/conf/init_db_script

Create schemas
RTApipe $  cd /opt/prod/CTA_BUILD2
RTApipe $  sh init_db_content.sh [password]

Setup slurm cluster name in cluster table in cta_pipe_db, usually this is hostname=[clustername] e.g.
agilehost2
check instrument and network “TEST” id = 0

RTApipe $ mysql -urt -p

use cta_pipe_db;
insert into cluster values(0, [clustername]);
update instrument set instrumentid=0 where name='TEST';
```

```
update network set networkid=0 where name='TEST';
```

Setup Slurm

Check with sinfo.

```
Warning: the username should be the same as the machine
Warning: E.g. [clustername]=[machine name]

RTApipe $ sacctmgr add cluster [clustername]
RTApipe $ sacctmgr create user name=rt cluster=[clustername] account=root
```

Create reservation

```
Update info

RTApipe $ cd /opt/prod/CTA_BUILD2/pipeline_manager/slurm
RTApipe $ vim slum_init.sh
RTApipe $ sh /opt/prod/CTA_BUILD2/pipeline_manager/slurm/slum_init.sh

EXAMPLE:

Warning: the username should be the same as the machine
Warning: CoreCnt is the limit of pipeline job number
OR run commands:

RTApipe $ scontrol create reservation Reservation=large_rt  StartTime=now Duration=315360000
Nodes=[machine name] partition=large CoreCnt=10 User=rt flag=overlap
RTApipe $ scontrol create reservation Reservation=fast_rt  StartTime=now Duration=315360000
Nodes=[machine name] partition=fast CoreCnt=10 User=rt flag=overlap

Warning: restart container instance
RTApipe $ exit
agilehost2 $ sh stop_instance.sh
agilehost2 $ sh start_instance_[machine name]_[user name].sh
agilehost2 $ sh shell_instance.sh
RTApipe $ sh /opt/start/start_services.sh
```

3.2. Configure the pipeline manager

pipeline_manager

```
RTApipe $ cd /opt/prod/CTA_BUILD2/pipeline_manager/scripts
RTApipe $ cp conf.py.default conf.py
RTApipe $ vi conf.py
def get_task_manager_db_conf():

    host = 'localhost'
    username = ''
    password = ''
    port = ''
    pipemanager_db = 'pipe_manager'

    return
{'host':host,'username':username,'password':password,'port':port,'pipemanager_db':pipemanager_db}

def get_slurm_sbatch_command():

    sbatch_command = '/usr/bin/sbatch'

    return {'sbatch_command':sbatch_command}

def get_slurm_scancel_command():

    scancel_command = '/usr/bin/scancel'

    return {'scancel_command':scancel_command}
```

Start pipeline manager

```
RTApipe $ sh /opt/start/start_pipes_cta_build2.sh
```

Create a row in the pipe_manager table. Every row represents a pipeline. The daemons of the manager execute a for loop on these rows to submit runs

```
RTApipe $ mysql -urt -p
```

INSERT INTO`pipe_manager`.`pipe_list`(`id_pipe`, `name`, `db_host`, `db_user`, `db_password`, `db_name`, `db_port`, `directory`, `env_file`, `status`, `build_number`, `repository`) VALUES(NULL, '**cta_pipe_db**', '**localhost**', '**[username]**', '**[password]**', '**cta_pipe_db**', [port], '/opt/prod/**CTA_BUILD2**/', '/opt/prod/**CTA_BUILD2**/skeleton.ll', '1', '**2**', '');

4. CONFIGURE AND START A SIMULATION FROM DL3 event list

4.1. Import a photon list

Create an event list (check if it exists)

```
RTApipe $ cd /opt/prod/CTA_BUILD2
RTApipe $ source /opt/module/ctools-1.5.2
RTApipe $ ctobssim ra=83.6331 dec=22.0145 rad=5.0 tmin=MJD58130 tmax=MJD58130.3 emin=0.1
emax=100 inmodel=ctoolsint/examples/crab.xml caldb=prod2 irf=South_0.5h outevents=events_test.fits
```

Insert evt from simulation

```
RTApipe $ cd /opt/prod/CTA_BUILD2/CTAGammaPipeCommon
RTApipe $ source /opt/module/cta_rta_pipe_build2
RTApipe $ python import_observation_fits.py /opt/prod/CTA_BUILD2/test/system/events_test.fits 1 1

Where the parameter of import_observation_fits.py are: file.fits observationid datarepositoryid
```

4.2. Configure a NEW simulation (new observation)

The first step is to create the observation into the cta_pipe_db

After the login into mysql:

```
use cta_pipe_db;
```

4.2.1. Observation

To create an observation, we need to insert a list of parameters:

l,b: galactic coordinates of the observation center

Time interval: tstart and tstop time expressed in TT format

status: status=1 for active observations

timereftypeid,skyreftypeid are fixed to 1,2 respectively

insert into observation (tstartplanned,tendplanned,tstartreal,tendreal,status,fitspath,l,b,timereftypeid,skyreftypeid,name) VALUES (tstartplanned,tendplanned,tstartreal,tendreal,status,fitspath,l,b,1,2,name);

```
insert into observation
(tstartplanned,tendplanned,tstartreal,tendreal,status,fitspath,l,b,timereftypeid,s
kyreftypeid,name) VALUES
(442800000,442820000,442800000,442820000,1,"",184.557449,-5.7,1,2,"test");
```

```
test
```

4.2.2. proposal

Create the proposal

```
insert into proposal (name,description) VALUES ("[nome
proposal]","[descrizione]");
```

```
insert into proposal (name,description) VALUES ("test_name","test_description");
```

```
testproposal
```

4.2.3. Observing mode

Check that an observingmode exists linked to the instrument

```
select * from observingmode where instrumentid = [instrumentid] ;

select * from observingmode where instrumentid = 10;
```

If the observing mode is not present:

Insert just the instrumentid, other fields must remain null

Get the instrumentid from the instrument table, usually it is CTA-SOUTH

Use this query to find the instrumentid for an instrument name

```
select * from instrument

insert into observingmode (instrumentid) VALUES ([instrumentid])

insert into observingmode (instrumentid) VALUES (10)
```

4.2.4. target

Create a target

emin,emax: expressed in TeV,

L,b: galactic coordinates

Substitute the **proposalid** and **observingmodeid** values with values of the entity created before. Use these queries:

```
select * from proposal;

(proposalid=3)

select * from observingmode where instrumentid = [instrumentid];

select * from observingmode where instrumentid = 10;

(observingmodeid=1)
```

The targetcode is an arbitrary code used to distinguish between same proposals with different parameters

Xmlmodel is the model used as input for ctools, it must have the ctools format. It can be the xml used to simulate the data or a specific xml created for the analysis.

```
crab.xml
<?xml version="1.0" standalone="no"?>
<source_library title="source library">
  <source name="Crab" type="PointSource">
        <spectrum type="PowerLaw">
        <parameter name="Prefactor"   scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
        <parameter name="Index"         scale="-1"      value="2.48" min="0.0"   max="+5.0"   free="1"/>
        <parameter name="PivotEnergy" scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
        </spectrum>
        <spatialModel type="PointSource">
        <parameter name="RA"  scale="1.0" value="83.6331" min="-360" max="360" free="0"/>
```

```
      <parameter name="DEC" scale="1.0" value="22.0145" min="-90"  max="90"  free="0"/>
      </spatialModel>
  </source>
  <source name="CTABackgroundModel" type="CTAIrfBackground" instrument="CTA">
      <spectrum type="PowerLaw">
      <parameter name="Prefactor"   scale="1.0"  value="1.0"  min="1e-3" max="1e+3"   free="1"/>
      <parameter name="Index"         scale="1.0"  value="0.0"  min="-5.0" max="+5.0"   free="1"/>
      <parameter name="PivotEnergy" scale="1e6"  value="1.0"  min="0.01" max="1000.0" free="0"/>
      </spectrum>
  </source>
</source_library>
```

Emin,emax,l,b: are used for the analysis with ctools

targetcode and proposalid are a key for this table and cannot be used more than one time

insert into target (targetcode,proposalid,observingmodeid,name,l,b,xmlmodel,emin,emax)
VALUES(targetcode,proposalid,observingmodeid,name,l,b,xmlmodel,emin,emax);

```
insert into target
(targetcode,proposalid,observingmodeid,name,l,b,xmlmodel,emin,emax)
VALUES(10,1,1,"Crab",184.5,-5.7,"test_xml",0.1,100);
```

```
NB: add tscalc="1" to calculate the ts with ctools
```

```
insert into target
(targetcode,proposalid,observingmodeid,name,l,b,xmlmodel,emin,emax)
VALUES(10,3,1,"CrabAB",184.5,-5.7,"<?xml version=\"1.0\" standalone=\"no\"?>
<source_library title=\"source library\">
  <source name=\"Crab\" type=\"PointSource\" tscalc=\"1\">
      <spectrum type=\"PowerLaw\">
      <parameter name=\"Prefactor\"   scale=\"1e-16\" value=\"5.7\"  min=\"1e-07\"
max=\"1000.0\" free=\"1\"/>
      <parameter name=\"Index\"      scale=\"-1\" value=\"2.48\" min=\"0.0\"
max=\"+5.0\"   free=\"1\"/>
      <parameter name=\"PivotEnergy\" scale=\"1e6\"   value=\"0.3\"  min=\"0.01\"
max=\"1000.0\" free=\"0\"/>
      </spectrum>
  </source>
  <source name=\"CTABackgroundModel\" type=\"CTAIrfBackground\"
instrument=\"CTA-SOUTH\">
      <spectrum type=\"PowerLaw\">
      <parameter name=\"Prefactor\"   scale=\"1.0\"  value=\"1.0\"  min=\"1e-3\"
max=\"1e+3\"   free=\"1\"/>
      <parameter name=\"Index\"      scale=\"1.0\"  value=\"0.0\"  min=\"-5.0\"
max=\"+5.0\"   free=\"1\"/>
      <parameter name=\"PivotEnergy\" scale=\"1e6\"  value=\"1.0\"  min=\"0.01\"
max=\"1000.0\" free=\"0\"/>
      </spectrum>
  </source>
```

```
</source_library>
",0.1,100);
```

Observation_target

Create a into the observation_target table the relationship between the observation and the target

```
insert into observation_target (observationid,targetid) values
([observationid,targetid]);
```

```
insert into observation_target (observationid,targetid) values (8,6);
```

4.2.5. observation_to_datarepository

Create row into the observation_to_datarepository

The datarepositoryid should be just one

```
insert into observation_to_datarepository
(datarepositoryid,observationid,tstartdata,tenddata) VALUES (1,[observation
created before],[observation tstart real],[observation tstart real]);
```

```
insert into observation_to_datarepository
(datarepositoryid,observationid,tstartdata,tenddata) VALUES
(1,6,442800000,442800000);
```

```
insert into observation_to_datarepository
(datarepositoryid,observationid,tstartdata,tenddata) VALUES
(1,8,442800000,442800000);
```

Check that the instrument and the datarepository are already inserted into the table analysistrigger and that this row has status=1, otherwise if there is not the trigger for that insturment the analysis can not start

```
select status from analysistrigger where triggersourcedatarepositoryid=1 and
triggersourcedatarepositoryidinstrumentid = [instrumentid];
```

```
select status from analysistrigger where triggersourcedatarepositoryid=1 and
triggersourcedatarepositoryidinstrumentid = 10;
```

Inside the analysissessiontype table it is possible to enable or disabble the analysis with the “runnable” attribute

```
select analysissessiontypeid,shortname from analysissessiontype where runnable=1;
```

Note:

- analysistypeid defines a tool, the commant to be executed and other configurations.
- analysissessiontypeid can be used to change the analysis configurations of the same science tool

4.2.6. NEW ANALYSISSESSIONTYPE

To create a new type of analysis, we can copy an existing row and change the configurations.

There are two main types of analysis with CTA

1) Analysis with bin=step, bins are not overlapped, copy the analysis with ID=1
2) Analysis with bin>step, bins are overlapped, copy the analysis with ID=10

```
insert into analysissessiontype
(aggregation,description,analysistypeid,analysistriggerid,deltatstart,deltatstop,t
imebinsize,minbinsize,maxbinsize,timestep,runnable,queue,reservation,energybingrou
pid,skypositiontype,skyringgroupid,analysisradius,postanalysisid) select
aggregation,description,analysistypeid,analysistriggerid,deltatstart,deltatstop,ti
mebinsize,minbinsize,maxbinsize,timestep,runnable,queue,reservation,energybingroup
id,skypositiontype,skyringgroupid,analysisradius,postanalysisid from
analysissessiontype where analysissessiontypeid = ID;

select analysissessiontypeid from analysissessiontype order by analysissessiontypeid desc
limit 1;

update analysissessiontype set
name="name",shortname="shortname",description="description" where
analysissessiontypeid = [analysissessiontypeid];
```

NB shortname is used to create the analysis directory path

Case 1

Update the times to create the analysis

X = temporale bin = step

```
update analysissessiontype set timebinsize=X,minbinsize=X,timestep=X where
analysissessiontypeid = analysissessiontypeid;
```

Case 2

X = temporal bin

Y = step

```
update analysissessiontype set timebinsize=X,minbinsize=X,timestep=Y where
analysissessiontypeid = [analysissessiontypeid];
```

Activate this analysis and disable the others

```
update analysissessiontype set runnable=1 where analysissessiontypeid =
analysissessiontypeid;

update analysissessiontype set runnable=0 where analysissessiontypeid !=
analysissessiontypeid;
```

If a datarepository is in relationship with an instrument, all the analysissessiontype with runnable=1 are started automatically.

4.3. Configure the simulation for an existing observation

Clean analysis and run directories

1) Remove from directory /ANALYSIS3/RUN/CTA-SOUTH the results of a datarepository
2) Remove from directory /ANALYSIS3/CTA-SOUTH the results of a datarepository

---

Configure database

Setup cluster table in cta_pipe_db

```
RTApipe $ mysql -urt -p
```

Query for the observation_to_datarepository initialization, database: **cta_pipe_db**

1) **use cta_pipe_db;**
2) **update observation_to_datarepository set tstartdata = 442800000, tenddata=442800000 WHERE observationid=1;**
3) **delete from analysissessiontype_observation where observationid=1;**

Change db to **evt_db**

1) **use evt_db;**

Check that there is a row in **stream_event. If there is not**

2) **insert into stream_event (id, status) values (NULL, '0');**

otherwise

2) update stream_event set status = 0;

This is the general state of the simulations. If it is equal to 0 all the simulations are stopped. If it is =1, all simulations with streamstatus = 1 into the stream_data tables are executed.

This command starts a daemon that runs every second and executes the simulations present into the stream_data table.

---

3) **INSERT INTO stream_data (streamdataid,observationid,datarepositoryid,tstart,tstop,streamstatus,twindowstart,twindowstop,pipedbname,timestep,timestepcount,speedfactor) VALUES(NULL, '1', '1', '442800000', '442810000', '1', '0', '0', 'cta_pipe_db', '1', '0', '10');**

This query creates a row to simulate observation 1. The tstart must be equal to the observation:tstartreal. The tstop can be fixed at different intervals. When the tstop reaches the twindowstop the simulation is stopped. Streamstatus must be 1 to enable the streaming. When the stop is reached, it is set to 0. To restart the simulation, this value must be changed to 1. The MySQL daemon increments the timestepcount of 1 every second. When it is equal to the timestep the status of the events is changed. The speedfactor indicates the time window that is changed when the changing status is triggered. Speedfactor 10 means that every second 10 seconds of events are simulated.

Reset the system before new simulations

1) **use evt_db; update stream_event set status = 0;**

1) Delete from directory /ANALYSIS3/RUN/CTA_SOUTH the analysis of a particular datarepository

   Delete from directory /ANALYSIS3/CTA_SOUTH the analysis of a particular datarepository

2) **use cta_pipe_db;**
3) **update observation_to_datarepository set tstartdata = 442800000, tenddata=442800000 WHERE observationid=1;**
4) **delete from analysissessiontype_observation where observationid=1;**
5) **use evt_db; update evt3 set status=0,timerealtt=0,insert_time=NULL where observationid=1 and datarepositoryid=1; UPDATE stream_data SET streamstatus=1,twindowstart=0,twindowstop=0 WHERE observationid= 1 and datarepositoryid=1;**

Restart simulations

use evt_db; update stream_event set status = 1;

Check results:

http://localhost:[port]/scigui/gettable.php?db=cta_results_db&tb=detection

Stop the simulations

use evt_db; update stream_event set status = 0;

5. Conclusions

This document describes the procedures to be used to configure the prototypes of the pipelines for the Science Alert Generation that are being developed at OAS Bologna. The first step is to download the code from Github repositories. Then it is required the building of a Singularity container with all necessary packages installed. The final step is to configure the pipeline software and database. At this point, the user can start a simulation of data processing using the prototype.