Сибирский радиогелиограф

Ссылки на ресурсы

Инфраструктура СРГ

- https://badary.iszf.irk.ru - корреляционные кривые, потоки, картинки, quicklook, спектрополяриметры
- http://10.1.2.60/telemetry/ - состояние антенн в реальном времени
- http://10.1.2.60/imaging/ - доступность данных и синтез изображений
- http://10.1.1.35 или http://90.188.35.59:8501 - интерактивные графики СРГ
- http://10.1.2.60/grafana/ - мониторинг инфраструктуры в веб-интерфейсе Grafana
- http://10.1.2.60/api/ или https://api.rao.istp.ac.ru - сервис доступности данных СРГ, база данных сканов изображений в сырых FITS файлах

Сырые данные наблюдений для синтеза, полный архив

- **Иркутск**: http://ftp.rao.istp.ac.ru, доступ по HTTPS и FTP
- **Бадары**:
  - https://nas.rao.istp.ac.ru - основное хранилище, HTTPS
  - 10.1.2.60-10.1.2.66 - доступ по FTP
  - http://10.1.2.60/nas - доступ по HTTP (можно использовать айпишники 40 и 60-66)
  - http://10.1.1.14 - бэкап основного хранилища, доступны только последние данные
  - 10.1.1.28 - дополнительный бэкап, доступ только по SSH. Есть данные, которые отсутствуют на 14 машине.

Софт для синтеза данных

Потребуется: Python 3.10, miniconda или pip, опционально: Docker. Почему желательна именно miniconda, а не, например, mamba, так это потому что в miniconda нормально заводится библиотека для ускорения вычислений MKL.

- https://github.com/maria-globa/srhdata - программа Марии Глобы
- https://git.iszf.irk.ru/anfinogentov/srh48_api - программа Сергея Анфиногентова
- https://git.iszf.irk.ru/fedenev/srhimages/ - графический интерфейс, задействующий оба интерфейса + здесь инструкция по их установке

Изменения в ПО сбора данных

2024-05-04

В заголовки файлов исходных данных диапазонов 6-12добавлено поле IF_1 - первая промежуточная частота в КГц. Рабочая частота, на которой измеряются видности определяется как разность между частотой гетеродина, указанной в таблице SRH_DATA исходного файла и значением поля IF_1.

2024-05-01

В программы `srh*DataAcquire` добавлены функции автоматического завершения наблюдений. Время остановки указывается в конфигурационных файлах `srh*DataAcquire.ini` значениями полей:

```
[receiver]
autoStop=<True> | <False>
autoStopHour=<int>
autoStopMinute=<int>
```

В заголовки файлов исходных данных диапазонов 3-6 и 12-24 добавлено поле IF_1 - первая промежуточная частота в КГц. Рабочая частота, на которой измеряются видности определяется как разность между частотой гетеродина, указанной в таблице SRH_DATA исходного файла и значением поля IF_1.

2024-07-07

Попытка определить источник сбоев синхронизации пакетов коррелятора 0612: сбивается соответствие время-поляризация-частота. Обычно это сопровождается пропаданием субпакетов коррелятора. Вместо 44 субпакетов приходит меньшее количество или даже 0. Сбор 0306, 1224 работает без сбоев. Источником может быть ПО сбора, компьютер сбора, коррелятор. Для того чтобы выяснить источник было сделано следующее:

1. Сетевые конфигурации всех диапазонов сделаны идентичными
2. Заменены кабели сетевая плата-коммутатор и изменены входы коммутатора

3. Изменен PCI разъем сетевой платы в компьютере сбора 0612

Результата нет.

2024-07-09

Сбор 612 был запущен на машине 1224 (10.1.1.64). Сетевой кабель коррелятора 612 был подключен к 1224. В mainwindow.cpp CfgNew.FrqRange была установлена в SyncDriverCfgRange_12_24. Старт в 03:02, сбой в 03:41.

Сбор в обычном режиме запущен в 03:45 1224, 03:47 612.

Замена в парсинге 612 .remove(..., 1) на .remove(..., sizeof(tPkg)).

2024-07-10

Коррелятор 612 подключен к машине 1224 Коррелятор 1224 подключен к машине 612

Сбор до 05:50 без построения изображений Сбор после 05:50 с построением изображений

Сбоев нет.

2024-07-12

Обнаружена ошибка в имени переменной конфигурационных файлов всех решеток. frequencySwitchTime устанавливалась по умолчанию. Значение по умолчанию было таким же, как и в конфиге, поэтому влияния на сбор не оказывалось.

В сбор 612 добавлена функция рестарта по обнаружению числа субпакетов != 44.

2024-07-13

Сбор 612 идет с функцией рестарта.

2024-08-06 06:25

Поиск причины сбоев синхронизации 612 ПО сбора 612 использует выходы синхронизатора 1224 ПО сбора 1224 использует выходы синхронизатора 612

все посыпалось, но удалось заметить, что 612 все равно поймала неверное число субпакетов. скорее всего синхронизатор работает правильно. возвращаем обратно.

В 07:30 вернули все в прежнее состояние.

2024-08-17

Добавил переменную bitWindowPosition в проект srh36DataAcquire. Значение задается в файле конфигурации receiver/bitWindowPosition, значение по умолчанию 10.

2024-08-18

Добавил переменную bitWindowPosition в проекты srh612DataAcquire м srh1224DataAcquire. Значение задается в файле конфигурации receiver/bitWindowPosition, значение по умолчанию 10. Добавил во все проекты usleep(10000) в initDigitalReceivers после каждой pCorrelatorClient->write

Синтез радиоизображений

Инструкции по синтезу радиоизображений, полученных Сибирским Радиодогелиографом с помощью программного обеспечения, разработанного в радиоастрофизическом отдлеле ИСЗФ СО РАН.

Синтез радиоизображений с помощью пакета srh_synth

Установка пакета `srh_synth` на свой компьютер

ПО для синтеза радиоизображений СРГ написано на языке программирования `Python 3`, поэтому прежде чем его использовать, вам необходимо установить интерпретатор этого языка.

Если вы синтезируете изображения на одном из серверов отдела радиоастрофизики ИСЗФ СО РАН, то Продолжите чтение инструкции с раздела Синтез радиоизображений, так как у вас уже установлены все необходимые пакеты.

Установка Python 3 в Linux

В большинстве популярных дистрибутивов Linux Интерпретор `Python 3` обычно установлен по умолчанию. Проверить установлен ли Python на вашей машине можно, запустив следующую команду в терминале:

```
python --version
```

или

```
python3 --version
```

Если `Python` установлен, то вы команда выведет версию установленного Интерпретатора. Мы рекомендуем использовать `Python 3.10` или более свежую версию. Если же в вашей системе Python 3 не установлен, то установите его с помощью менеджера пакетов вашего дистрибутива. Например, в Ubuntu для этого нужно в терминале выполнить следующую команду.

```
sudo apt install python3
```

Установка Python 3 в Windows

Интерпретатор языка `Python` не входит в состав операционной системы Windows, поэтому его предётся устанавливать отдельно. Это можно сделать двумя способами.

1. Установить "чистый" интерпретатор Python запустив инсталятор, скачанный с официального сайта Python .
2. Установить Python в составе менеджера пакетов `Anaconda` и `Miniconda`. В первом случае вместе с интерпретаром языка будут также установлен большой набор пакетов предназначенных для научных вычислений и визуализации данных. В случае же `Miniconda` будет установлен только интерпретатор языка Python и менеджер пакетов `conda`.

Создание виртуального окружения

Мы рекомендуем установливать ПО для синтеза радиоизображений в отдельном виртуальном окружении (virtual environment), чтобы избужать конфликта зависимостей. В случае, если вы планируете использовать Python только для синтеза радиоизображений, этот шаг можно пропустить.

TO BE DONE

Установка пакета `srh_synth`

Пакет `srh_synth` в текущей версии использует часть функций универсального пакета `srhimages` и устанавливается вместе с ним командой:

```
pip install -U "srhimages[anfinogentov] @ git+https://git.iszf.irk.ru/fedenev/srhimages"
```

Альтернативно можно установить пакет `srhimages` с полным набором опций.

```
pip install -U "srhimages[all] @ git+https://git.iszf.irk.ru/fedenev/srhimages"
```

Тогда вместе с пакетом srh_synth будет доступен также и код для синтеза изображений `srhdata`, разрабатываемый Марией Глоба. Если вы используете виртуальное окружение, то перед запуском команд установки его нужно активировать.

Синтез радиоизображений

Синтез радиоизображений состоит из двух этапов:

1. Калибровка коэффициентов усиления антенн - функция `srh_synth.srh_calib`
2. Синтез изображений - функция `srh_synth.srh_synth` В начале работы необходимо импортировать импортировать эти функции.

```
from srh_synth import srh_synth, srh_calib
```

Калибровка коэффициентов усиления антенн

Калибровка выполняется для заданного интервала времени. Приэтом калибруются не все изображения, а токлько некоторые. По умолчанию калибруются изображения с интервалом 5 минут. Затем полученные калибровочные коэффициенты подвергаются постобработке в ходе которой устраняются пространственные сдвиги между отдельными изображениями и выравниваются амплитуды коэффициентов усиления. Полученные калибровочные данные можно затем использовать для построения изображений внутри выбранного интервала на заданных частотах с произвольной скважностью и произвольным временем накопления.

Примеры команд для калибровки:

Калибровка для частот 6 и 6.4 ГГц для одного момента времени 2024-06-01 03:00:

```
calibrations = srh_calib("2024-06-01 03:00", frequency = [6000, 6400])
```

Калибровка наблюдений за весь день для частоты 6 ГГц:

```
calibrations = srh_calib("2024-06-01 00:00","2024-06-01 12:00", frequency = [6000])
```

Калибровка наблюдений за два часа для частоты 16 ГГц:

```
calibrations = srh_calib("2024-06-01 02:00","2024-06-01 04:00", frequency = [16000])
```

Калибровка наблюдений за весь день для частоты 3 ГГц с сохранением калибровок в файл 'calib.json':

```
calibrations = srh_calib("2024-06-01 00:00","2024-06-01 12:00", frequency = [3000], write_to=
'calib.json')
```

Калибровка данных решётки 6-12 ГГЦ для одного момента времени 2024-06-01 03:00:

```
calibrations = srh_calib("2024-06-01 03:00", frequency = 'SRH0612')
```

Полный список параметров функции `srh_calib`:

- `start_time` - начало интервала времени для калибровки, например '2024-05-14 00:30'
- `stop_time` - конец интервала времени для калибровки, например '2024-05-14 03:00'
- `frequency` - список частот, которые надо откалибровать, например [3200, 4000]. Значение по умолчанию 'all' -- калибровать все частоты, 'SRH0306', 'SRH0612', 'SRH1224' -- калибровать все частоты соответвующей решётки.
- `n_proc` - Количество параллельно обрабатываемых калибровок. По умолчанию 16. Если это число больше числа имеющихся в систме процессорных ядер, то этот параметр уменьшается до числа ядер в системе. При назначении числа параллельных процессов, нужно иметь ввиду, что каждый такой процесс потребляет около 1 Гб оперативной памяти.
- `cadence`- скважность калибровок в секундах. По умолчанию калибруются данные с интервалом в 300 с. Менять не рекомендуется.
- `int_time` - Время накопления в секцндах для каждого калибровочного изображения. По умолчанию 30 с. Менять не рекомендуется
- `parallel` - запускать калибровки параллельно в дочерних процессах. По цмолчанию True.В случае установки в False, калибровки будут выполняться последовательно в основном процессе.
- `post_process` - постобработка калибровок с целью устранения 'дрожания' и 'уезжания' изображений. Кроме того выравнивается амплитуда коэффициентов усиления антенн
- `raw_dir` - директория в которой код программа будет искать сырые данные СРГ и куда их будет скачивать при необходимости. Рекомендуется для всех своих задач использовать одну и ту же директорию на достаточно емком диске.
- `db_file` - имя файла с базой данных SQLite для сохранения калибровок. По умолчанию калибровки сохраняются в файл 'calibrations.db' в текущем каталоге. Обратите внимание, что в БД создайтся запись для каждой комбинации даты

наблюдения и частоты наблюдения.
- `write_to` - имя файла для сохранения калибровок в формате JSON. По умолчанию калиброви сохраняются только в локальную базу данных SQLite.
- `recalibrate` - по умолчанию `False`: программа берёт готовые калибровки из локальной БД SQLite, если они уже делались ранее. Если вы хотите пересчитать калибровки, то установите этот параметр в `True`

Возвращаемое значение: Функция `srh_calib` возвращает словарь, в котором находятся результаты калибровки разобранные по частотным каналам и отсорторированные по времени. Данную переменную нужно использовать для синтеза изображений.

Синтез радиозображений

После выполнения калибровки можно приступать непосредственно к построению изображений.

Примеры команд синтеза рад иоизображений

Синтез изображений на двух частототах для одного момента времени с использованием калибровок, полученных функцией `srh_calib`:

```
srh_synth("2024-06-01 03:00", frequency = [6000, 6400], calibrations=calibrations)
```

Синтез серии изображений на одной часоте для заданного интервала времени с временным разрешением 60 секунд, используя калибровки из файла 'calib.json':

```
srh_synth("2024-06-01 02:30", "2024-06-01 03:30", frequency=[3000], cadence=60,
calibrations='calib.json')
```

Синтез изображений для заданного интервала времени с накоплением 30 секунд и временным разрешением 300 секунд. Изображения сохраняться в папку "srh20240601", имена сохранённых файлов будут напечатаны в консоли:

```
files = srh_synth("2024-06-01 03:00", int_time=30, cadence=300, calibrations=calibrations,
out_dir='srh20240601')
print(files)
```

Полный список параметров функции srh_synth:

- `start_time` - начало интервала времени для калибровки, например '2024-05-14 00:30'
- `stop_time` - конец интервала времени для калибровки, например '2024-05-14 03:00'
- int_time - время интегрирования для каждого изображения в секундах. По умолчанию изображения строится из одной наблюдательной записи (скана).

- `cadence` - скважность (временное разрешение) синтезируемых изображений. По умолчанию 0 - изображения строятся с максимально возможной скважностью.
- `frequency` - список частот, на которых надо построить изображения, например [3200, 4000]. Значение по умолчанию 'all' -- строить изображения на всех частотах. Данные на выбранных частотах должны быть предварительно откалиброваны функцией `srh_calib`
- `out_dir` - путить к каталогу, куда будут записаны построенные изображения. Если в данном каталоге уже имеются файлы изображений, они будут перезаписаны при совпадении имён. По умолчанию изображения записываются в текущий каталог
- `raw_dir` - директория в которой код программа будет искать сырые данные СРГ и куда их будет скачивать при необходимости. Рекомендуется для всех своих задач использовать одну и ту же директорию на достаточно емком диске.
- **`calibrations`** - калибровочные данные, возвращаемые функцией `srh_calib`, или путь к JSON файлу с сохранёнными калибровками.
- `naxis` - размер изображения в пикселях (по умолчанию 512)
- `cdelt` - размер одного пикселя в секундах дуги (по умолчанию 5.)
- `save_RL`- сохранять изображения в правой (R) и левой (L) круговой поляризации. По умолчанию False - сохраняется интенсивность (I) и поляризация (V).
- `no_channel_dirs` - не создавать подкаталоги для каждого частотного канала. По умолчанию False - подкаталоги создаются
- `save_psf` - сохранять диаграмму нааправленности для каждого изображения в отдельный FITS файл
- `n_proc` - Количество параллельно синтезируемых изображений. По умолчанию 16. Если это число больше числа имеющихся в систме процессорных ядер, то этот параметр уменьшается до числа ядер в системе.
- `round_beam` - использовать симметричную кргулую "чистую" диаграмму направленности при "чистке" изображений. Данная опция экспериментальная. По умолчанию False. Применения этой опции приводит к уменьшению пространственного разрешения изображения.
- `high_resolution` - Искусственно увеличить пространственое разрешения изображения за счёт уменьшения размера "чистой" диаграммы направленности. Данная опция эксперментальная. Ей использование имеет смысл только для заведомо компактных источников. А интерпретация полученных изображений возможна только с оговорками.
- `compressed` - сохранять изображения в сжатом формате FITS. По умолчанию True.
- `parallel` - запускать синтез изображений параллельно в дочерних процессах. По цмолчанию True.В случае установки в False, изображения будут строится последовательно в основном процессе.
- `clean_weights` - Путь к FITS файлу с радиомзображением Солнца на низкой часте, на которое нет перекрытия порядков или двухмерный numpy массив с весами, которые будут использоваться алгоритмом CLEAN при поиске точки с максимальной интенсивностью. Массив должен иметь те же размеры, что и изображение, то есть `[naxis, naxis]`. В случае, если в этом параметре передается путь к FITS файлу, изображеие из файла будет автоматически преобразовано в веса для CLEAN и отмасштабировано для требуемых значений NAXIS и CDELT.

Возвращаемое значение: Функция `srh_synth` возвращает список имен файлов синтезированных изображений.

Синтез радиоизображений с помощью пакета srhdata

Универсальный пакет для синтеза изображений srhimages

Программа `srhimages` предоставляет функции для синтеза радиоизображений из данных Сибирского Радиогелиографа (SRH).

Установка

Код Сергея Анфиногентова работает на python 3.13, код Марии Глобы - пока только на 3.10! Поэтому желательно устанавливать именно ту версию, которую нужно. Для расчётов на удалённых машинах версия питона не особо важна, но рекомендуется использовать что-то новое.

```
conda create -c conda-forge -n srhsynth python=3.13 uv git
conda activate srhsynth

uv pip uninstall srhimages srh_synth srhdata
uv pip install -U "srhimages[anfinogentov,globa] @
git+https://git.iszf.irk.ru/fedenev/srhimages"
# или [globa], [anfinogentov] вместо [all], чтобы выбрать только конкретный расчётный код
```

Если производится свежая установка с использованием расчётного кода Марии Глобы, то требуется обновить данные CASA:

```
python3 -m casaconfig --update-all
```

Установка из каталога с git-репозиторием (для разработки)

```
# git clone https://git.iszf.irk.ru/fedenev/srhimages
# cd ./srhimages
git pull -a
```

```
uv pip uninstall srhimages srhsynth
uv cache clean
uv pip install -U -e "srhimages[anfinogentov] @ ."
```

Использование уже установленного окружения на сервере ИСЗФ

```
conda activate /opt/miniconda3/envs/srhsynth/
```

Интерфейс командной строки

```
srhimages create_synth_tasks --help
```

Использование в своих скриптах

```
import srhimages


help(srhimages.create_synth_tasks)
help(srhimages.run_computation)
```

Обратите внимание, что если запуск кода происходит через Python скрипт `.py`, то требуется запускать расчёты следующим образом:

```
import srhimages


if __name__ == "__main__":
    # for result in srhimages.run_computation(.....)
    pass
```

Этот костыль связан с особым механизмом работы систем параллельных расчётов в Python. При запуске кода в Jupyter, скорее всего, это не понадобится.

Описание команд и принципы работы

Расчётные задачи

Программа работает в парадигме так называемых расчётных задач ("tasks"). Каждая из задач представляет собой Python-словарь или его JSON-представление и описывает, из каких данных (сырых файлов и сканов внутри них) должно получиться итоговое изображение, куда оно будет сохранено и с какими параметрами синтезировано.

Актуальная спецификация формата расчётной задачи есть в исходном коде в формате `jsonschema`. Спецификация API для работы с амплитудно-фазовыми калибровками антенн находится в репозитории badary-services.

Интерфейс программы:

1. `create_synth_tasks(time1, time2=None, cadence="15min", frequencies="all", resample_from=None, save_to=None, average_width=20, average_unit="scans", average_position="after", average_mode = "visibilities", output_polarizations="IV", naxis=512, cdelt=4.9, clean_disk=True, compressed=True, smooth_gains=False)`
   - **Назначение:** Создает список (некалиброванных) задач по синтезу радиоизображений с телескопа.
   - **Аргументы:**
     - **time1** (str): Время начала наблюдения в формате 'ГГГГ-ММ-ДД ЧЧ:ММ:СС'.
     - **time2** (str или None, опционально): Время окончания в том же формате.
     - **cadence** (str, опционально): Временной интервал между каждым наблюдением в формате "NNmin" или "NNs". По умолчанию "15min".
     - **frequencies** (list(int) или str, опционально): Список частот для наблюдения в МГц или "all" в виде строки, или диапазон вида "3000-5000", или список вида [3000, "6000-8000", "SRH1224"].
     - **resample_from** (list или str): Список задач или путь к файлу, содержащему список задач в формате JSON, откуда брать калибровки. Доступные частоты в списке resample должны совпадать с запрошенными пользователем частотами.
     - **save_to** (str, опционально): Путь к json файлу для сохранения списка задач.
     - **average_width** (int или float, опционально): Количество сканов или секунд для усреднения. По умолчанию 20.
     - **average_unit** (str, опционально): Может быть 'scans' или 'seconds'. По умолчанию 'scans'.
     - **average_position** (str, опционально): Временное окно для усреднения. Может быть 'after', 'before' или 'center'. По умолчанию 'after'.
     - **average_mode** (str, опционально): Режим усреднения. Может быть 'visibilities', 'gridding' или 'images'. По умолчанию 'visibilities'.
     - **output_polarizations** (str, опционально): Может быть 'IV' или 'RL'.
     - **naxis** (int, опционально): Количество пикселей вдоль каждой оси выходного изображения. По умолчанию 512.
     - **cdelt** (float, опционально): Размер каждого пикселя в угловых секундах. По умолчанию 4.9 для СРГ.
     - **clean_disk** (bool, опционально): Флаг для включения/выключения очистки "грязного" изображения. По умолчанию True.
     - **compressed** (bool, опционально): Флаг для включения/выключения сжатия FITS выходного изображения. По умолчанию True.
     - **smooth_gains** (bool, опционально): Предпочтение сплайн-интерполяции калибровок вместо ближайших соседей при передискретизации.

Используйте True для калибровок за весь день.
- **Возвращает:**
  - Список задач синтеза (каждая задача – Python dict), например, для отправки на кластер или для локального вычисления.

2. `run_computation(task_list, algorithm, cache_dir="./images/raw/", out_dir="./images/out/", ftp_server="https://ftp.rao.istp.ac.ru", n_threads=5, calibrate="prefer_server", skip_postprocessing=False, skip_images=False, input_dir=None, cluster_object="local", run_id=None)`

- **task_list** (list или str): Список задач для вычисления или путь к файлу, содержащему список задач в формате JSON.
- **algorithm** (str): "globa" или "anfinogentov".
- **cache_dir** (str): Директория для хранения загруженных сырых файлов СРГ.
- **out_dir** (str): Директория для сохранения синтезированных изображений.
- **ftp_server** (str): Адрес сервера для загрузки файлов. По умолчанию загрузка осуществляется с использованием HTTPS (адрес начинается со схемы https://).
- **n_threads** (int): Количество потоков для вычислений (по умолчанию 5).
- **calibrate** (str): "prefer_server" или "from_scratch". По умолчанию "prefer_server". "from_scratch" удаляет все уже имеющиеся калибровки в списке задач и заставляет калиброваться всё заново.
- **skip_postprocessing** (bool, опционально): Пропустить выравнивание и сглаживание амплитуды/фазы усилений для антенн СРГ. По умолчанию False, True не рекомендуется.
- **skip_images** (bool, опционально): Только откалибровать (и выполнить постобработку) задачи и не выполнять этап CLEAN и синтез изображений. По умолчанию: False.
- **input_dir** (str, опционально): Директория, содержащая входные файлы, в случае, если SRH NAS смонтирован там. Для локальных расчётов всегда должно быть None.
- **cluster_object** (str или object): Тип кластерного объекта для использования для вычислений, 'local' для локальных вычислений (по умолчанию 'local') и 'badary' для кластера в Бадарах. Другие варианты включают передачу пользовательских объектов Dask.distributed (например, SSHCluster).
- **run_id** (str, опционально): Префикс строки для Redis для хранения прогресса задачи (и списка задач) во время вычислений. По умолчанию None, что приведёт к случайной строке.

Возвращает:

- **results**: Список задач с результатами вычислений, либо с ошибками.

Примечание:

- Если `task_list` является строкой или объектом pathlib.Path, он будет загружен как JSON файл.
- Загружает необработанные файлы с FTP сервера в директорию кэша.
- Калибрует задачи, используя серверные калибровки или локально.

- Постобработка калибровок выполняется методом, разработанным Сергеем Анфиногентовым.
- Все ошибки в вычислениях будут сохранены в возвращаемом объекте или сохранены в результирующем файле в формате JSON.

Пример использования

Простой случай нескольких изображений в течение дня

```
import srhimages

time1, time2 = "2024-06-01 02:00:00", "2024-06-01 02:00:05"
frequencies = [2800, 3000]

task_list = srhimages.create_synth_tasks(time1, time2, cadence="3s", frequencies=frequencies,\
    save_to="single-day.json",
    average_width=5, average_unit="seconds", average_mode = "visibilities",
average_position="after",\
    output_polarizations = "IV", naxis=512, cdelt=4.9, clean_disk=True, compressed=True)

if __name__ == "__main__":
    for i, results1 in enumerate(srhimages.run_computation("./single-day.json", "globa")):
        results1.to_json_file(f"./single-day-results-{i}.json")

# изображения появятся в каталоге ./images/out
```

Обработка солнечной вспышки

Здесь требуется уже откалибровать данные на достаточно большом интервале времени, поскольку во время вспышки резко снижается вклад коротких баз радиотелескопа. Поэтому расчёт будет проходить в 2 этапа:

Первый этап - калибровка на длинном интервале времени раз в 5 минут, чтобы отследить тренд "уплывания" коэффициентов усиления антенн в течение дня. Обязательно используется постобработка калибровок и их сглаживание. Рекомендуются интервалы от нескольких часов, самое идеальное - весь день.

```
import srhimages

freq_list = [3000, "5500-6200"]
time1, time2 = "2024-02-06 02:10:00", "2024-02-06 04:00:00"
```

```
calib_tasks = srhimages.create_synth_tasks(time1, time2, cadence="5min",\
    frequencies=freq_list, save_to="calibs.json")

# в файле calibs.json появятся задания для вычисления первоначальных калибровок

if __name__ == "__main__":
    for i, calibrations in enumerate(srhimages.run_computation("calibs.json", "globa",\
        skip_images=True)):
        if len(calibrations) == 0:
            continue
        freq = calibrations.frequencies[0]
        calibrations.to_json_file(f"calibs_ready_{i}_{freq}.json")

    # параметр skip_images нужен, потому что
    # нам нужны только калибровки, а не сами изображения

# если всё хорошо, то в файлах calibs_ready**.json будут готовые калибровки.
```

Второй этап. Когда у нас уже имеются постобработанные калибровки, то можно на их основе посчитать картинки с максимальным временным разрешением. `cadence="0s"` означает, что используется максимально возможное временное разрешение. Список частот должен быть тем же самым, на котором шла калибровка. Параметр `resample_from` означает, что для новых заданий берутся уже посчитанные калибровки и переносятся на новую сетку по времени. `smooth_gains` означает, что интерполяция коэффициентов усиления антенн на нужный кадр будет производиться с помощью сплайна, а не ближайшего соседа. Эта настройка рекомендуется, когда временное разрешение финальных изображений чаще, чем разрешение калибровок.

```
synth_tasks = srhimages.create_synth_tasks(time1, time2, cadence="0s", frequencies=freq_list,\
    resample_from="calibs_ready.json", smooth_gains=True, save_to="synth.json")

if __name__ == "__main__":
    for i, results in enumerate(srhimages.run_computation("synth.json", "anfinogentov")):
        if len(results) > 0:
            dt = results.dates[0]
            freq = results.frequencies[0]
            results.to_json_file(f"result_{freq}_{dt}_{i}.json")
```

В списке выполненных (или невыполненных) задач, который появится в переменной `results` или в файле `synth**.json`, будут уже прописаны полные пути к полученным изображениям

Загрузка калибровок на сервер

```
from srhimages import TaskCollection
calibrated_tasks = TaskCollection("synth_progress.json")
calibrated_tasks.extract_gains().upload()
```

Диагностика и ремонт АП

Пошаговое руководство диагностики и ремонта антенных постов (АП)

Неполадки мехатронного модуля (ММ)

Список неполадок:

- заклинивание;

- нет связи ММ;

- отказ энкодера;

- неравномерный ход/трясучка;

Заклинивания

Нет связи ММ

Отказ энкодера

Неравномерный ход

Неполадки приёмного модуля (МП)

Список неисправностей МП:

- отсутствует сигнал;

- отсутствует одна поляризация;

- сигнал от антенны шумит/прыгает;

Отсутствует сигнал

Отсутствует поляризация

Сигнал шумит или прыгает

Отсутствует сигнал от антенны

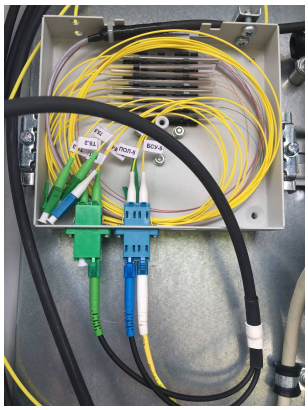
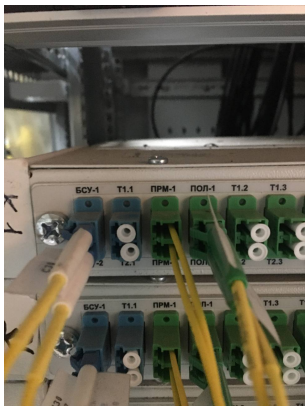
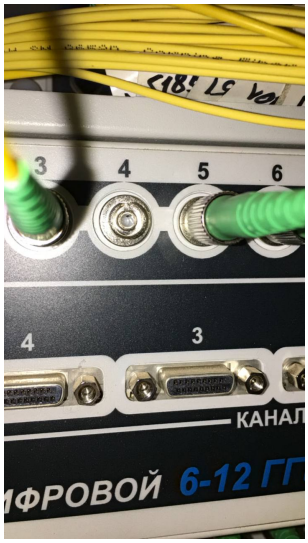
От антенны нет сигнала, порядок действий:

- Смотри UV plane;
- Смотрим в программе регистрации;
- Запускаем ПО Heliograph_antenna;

Блок схема приемного тракта и антенного поста

1. Приемный тракт

Вид	Описание	Неисправности	Диагностика
	Облучатель	Не переключается поляризация. Ток облучателя меньше 80мА. Ток облучателя больше 300мА.	Целостность линии. Параметры телеметрии. Показания тестера.
	Модуль приемный (МП)	Смотреть Таблица 6 - Светодиодная индикация ошибок	Световая индикация. Параметры телеметрии. Показания напряжений в heliograph_antenna

Вид	Описание	Неисправности	Диагностика
	Шкаф БСУ		Визуальный осмотр на предмет заломов оптического кабеля. Качество соединения разъемов.
	Кросс оптический в БСУ		Проверка соединения в оптических муфтах. Отсутсвие заломов оптического кабеля.
	Кросс оптический в тоннеле		Проверка соединения в оптических муфтах.
	Цифровой приемник		Проверка соединения в оптических муфтах.

Меры безопасности

ЗАПРЕЩАЕТСЯ ПРИ ВКЛЮЧЕННОМ ИЗДЕЛИИ:

- ОТКРЫВАТЬ ЕГО СОСТАВНЫЕ ЧАСТИ ДЛЯ РЕМОНТА И ОБСЛУЖИВАНИЯ;
- ОТСОЕДИНЯТЬ И ПОДСОЕДИНЯТЬ КАБЕЛИ.

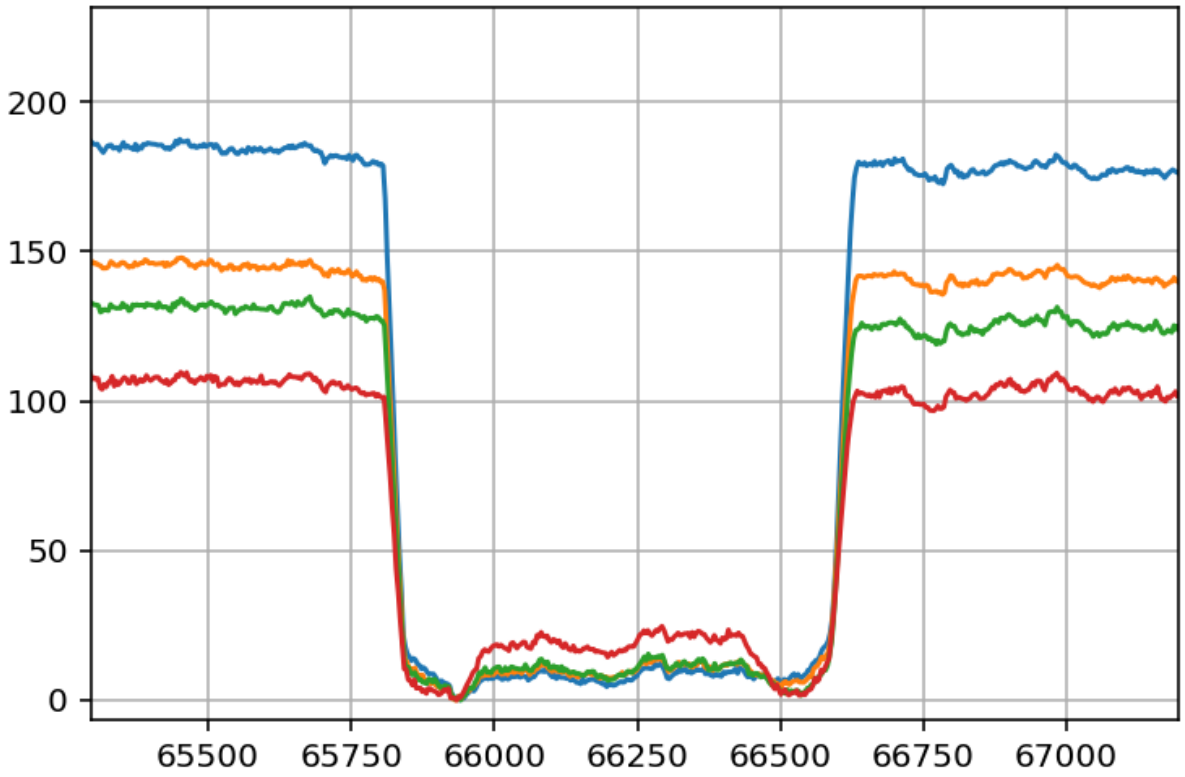
Текст текст Текст с новой строки

2. Антенный пост

Кассиопея А и Луна

27-28 июля 2024

27 и 28 июля 2024 пытался наблюдать Кассиопею А и Луну решеткой 36.



На рисунке по X UTC секунды, сначала потоки (сумма откликов одиносных антенн) от Луны на частотах 36[4,7,9,13], затем от Кассиопеи А, затем снова от Луны.

Основные результаты:

1. Луна наблюдается уверенно.
2. Кассиопея А слабо, но видна не частотах 36[4,7,9,13].
3. Рядом с Кассиопей А наблюдался поляризованный узкополосный источник либо медленно движущийся, либо медленно затухающий. Пока не очень понятно что это.
4. Выяснялдось, что чувствительность сильно меняется на слабом сигнале в диапазон частот. Возможно, что это из-за гармоник гетеродина, проходящих через смеситель.

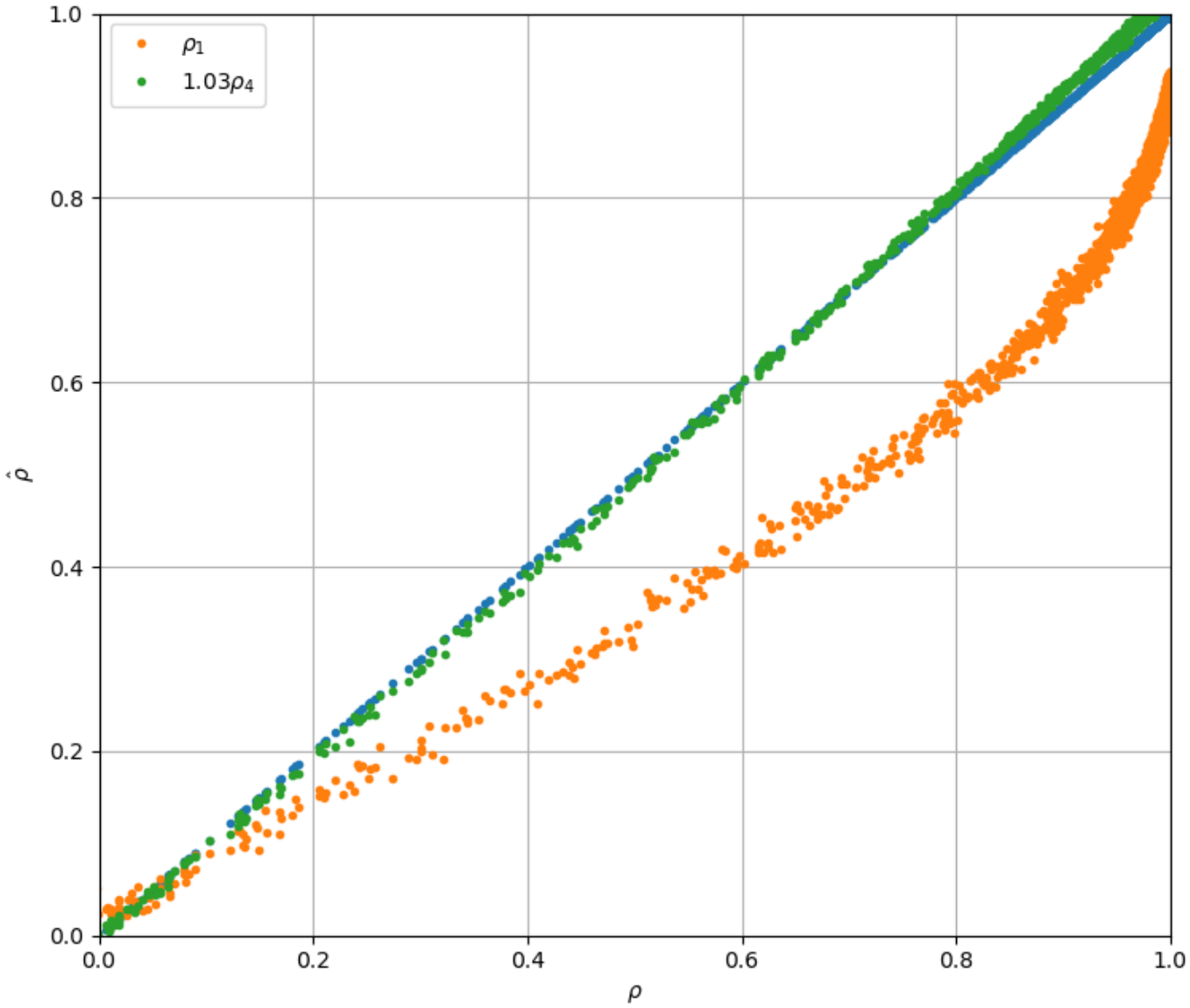
Изменения в ПО сбора данных 20241128-20241202

28 ноября - 2 декабря 2024

1. Скрипт SunEphems.py создает файл солнечных эфемерид на пять дней: +-2 от текущей даты. Формат файла такой-же, как и у soldat.txt, но заполняются только те поля, которые использует QSoldatFile. SunEphems исполняется каждый день в 23 часа. Находится в той же папке, что и soldat.txt.
2. srh*DataAcquire указывает файл эфемерид soldat_YYYYMMDD.txt.
3. В srh*DataAcquire добавлена опция установки времени коррелятора: можно установить в корреляторе локальное время компьютера, можно отправить запрос на синхронизацию коррелятора с NTP. Осуществляется это переменной конфигурационного файла set_correlator_time_ntp <true | false>.
4. В метод initDigitalReceiver добавлена запись 0 в регистры коррелятора до инициализации.
5. Анализ смещений Солнца в моменты -20 минут от кульминации, в кульминацию, +20 минут после кульминации показывает, что возможной причиной смещений может быть ошибка в частоте гетеродина порядка 5-10 МГц. Характер смещений одинаков на всех решетках. Отмечено, что реальная инициализация цифровых приемников происходит не всегда, поэтому возможно работа приемников с параметрами предыдущего сеанса наблюдений.

Об использовании амплитуд, измеренных коррелятором

Коррелятор Радиогелиографа измеряет ковариацию (кросс-корреляцию) двух 4-битовых или 1-битовых сигналов от пары антенн. Назовем эти сигналы квантованными сигналами, а устройство, получающее эти сигналы из исходных - квантователь. Разрядность квантователя, вид его характеристики и вес одного разряда задаются при инициализации цифровых приемников Радиогелиографа перед началом наблюдений. Для обозначения квантованных сигналов будем использовать символ $\hat{ }$. Так как коррелятор измеряет ковариации всех пар антенн, то часть из них являются автокорреляциями или дисперсиями квантованных сигналов ${\hat{\sigma}_x}^2$, где x - индекс антенны, $\hat{\sigma}$ - среднеквадратичное отклонение. Тогда квантованный коэффициент корреляции определяется как $\hat{\rho}=\frac{\left<\hat{x}\hat{y}\right>}{\hat{\sigma}_x\hat{\sigma}_y}$. В случае 1-битового квантователя $\hat{\sigma}_x=\hat{\sigma}_y=1, \hat{\rho}=\left<\hat{x}\hat{y}\right>$ и истинный коэффициент корреляции определяется коррекцией ван Флека $\rho=sin\left(\frac{\pi}{2}\hat{\rho}\right)$. В случае многобитового квантователя истинный коэффициент корреляции равен $\rho = gV\left(\hat{\rho},\sigma_x,\sigma_y,\right)$, где gV обобщенная коррекция ван Флека. Заметим, что для вычисления истинного коэффициента корреляции из измеренных ковариаций необходимо сначала вычислить истинные дисперсии σ^2.



На рисунке показаны модельные зависимости квантованных коэффициентов корреляции для разрядность 1 и 4. Видно, что 4-битовый коэффициент корреляции отличается от истинного более чем на 3% только в области значений > 0.8. Поэтому в большинстве случаев можно использовать 4-битовый коэффициент корреляции и без обобщенной коррекции ван Флека. Хотя, для исследования всех событий и для достижения максимальной достоверности обобщенная коррекция ван Флека необходима. Получить 4-битовый коэффициент корреляции из набора ковариаций, записанных коррелятором можно следующим образом. Данные ковариаций находятся в колонках 'vis_lcp' и 'vis_rcp' исходных фитсов. Данные квантованных дисперсий (амплитуд) находятся в колонках 'amp_c_lcp' и 'amp_c_rcp'. Допустим мы обрабатываем ковариацию (видность) антенн A и B для левой круговой поляризации. Тогда $\hat{\rho} = vis\_lcp[\nu,t,pair(A,B)] / \sqrt{amp\_c\_lcp[\nu,t,A]amp\_c\_lcp[\nu,t,B]}$. Разрядность квантователя (коррелятора), тип его характеристики и вес разряда задаются в конфигурационном файле программы сбора данных. В исходных фитсах эта информация содержится в полях Q_LEVELS, Q_TYPE и Q_STEP. Q_LEVELS - число уровней квантователя. Для 4-битового это 15 или 16 в зависимости от типа характеристики квантователя Q_TYPE 0 или 1. Q_STEP определяется уровнем входного сигнала. Например для наблюдения Луны или дискретных источников вес разряда нужно

уменьшить. Но при этом нужно иметь в виду, что с его уменьшением возрастает влияние постоянного смещения на выходах АЦП, что может привести к сильной нелинейности при измерении ковариации.

Изменения в ПО сбора данных 20241219-20241224

О частотах

Подозрения о том, что стандарт частоты или гетеродины работают с отклонениями, к счастью, не подтвердились. С помощью Agilent 53150A измерил выход 10 МГц стандарта частоты и гетеродина 3-6. До точности 10^{-6} все на месте.

Об эфемеридах

Было подозрение, что в ежедневных файлах эфемерид не хватает пары строк (дней) для корректного вычисления вторых производных. Добавил пару дней.

Об инициализации цифровых приемников

Так и не удалось выяснить всегда ли проходит инициализация. Смущает то, что у сервера коррелятора нет очереди для входных пакетов, содержащих конфигурацию приемников. Добавил usleep(120000) на всякий случай, чтобы задержка между пакетами была гарантированно больше времени накопления (на случай если коррелятор зависает в режиме сбора). Но результата это не дало. Коррелятор имеет право игнорить пакеты и пользуется он этим правом или нет пока не понял. Сброс питания стоек обременен тем, что коррелятор долго (не менее 5 минут) ищет NTP. Пока предлагается следующая процедура:

- стойки сбрасываются за 10 минут до наблюдений ($\textcolor{red}{питание}$ облучателей 12-24 должно быть $\textcolor{red}{выключено}$)
- на машинах сбора данных открываются веб-страницы корреляторов и проверяется синхронизация. После сброса стойки и до синхронизации коррелятора дата будет 19700101 и время около 0. У меня получалось дождаться синхронизации за 5 минут. Это гарантирует, что инициализация цифровых приемников будет правильной.

Поездка 20250118-20250123

srh1224DataAcquire

Увеличена задержка между отправками пакетов инициализации цифровых приемников - usleep(120000).

amp_ant2web

Написаны скрипты для отображения через web состояний антенн всех решеток в виде X - номер антенны, Y - мощность, слайдер для выбора частоты. Порт 8056 на машинах 10.1.1.3, 10.1.1.6, 10.1.1.12. ant2web продолжают работать через порт 8055 и отображают X - время, Y - мощность для каждой антенны усредненная по частоте, слайдер для выбора приемника.

calib_forest_0306_single_fits

Написан скрипт для калибровки антенн 0306 по температуре окружающей среды и сигналам от леса-неба. Посчитаны калибровки за 20240101-20250121. Функция яркостной температуры леса пока получается такая: $T_b=200 + 4T_{amb}$, где T_b в K, T_{amb} в C. Зависимость T_b от частоты пока не оченнь понятна. Все данные для калибровок теперь хранятся в одном файле, имена переменных сохраненны для совместимости.

Корреляторы - работа с NTP

22.01.2025

перевели коррелятор 3-6 на работу в режиме внутреннего времени,

параметр в конфиге set_correlator_time_ntp=false

24.01.2025

6-12 и 12-24 так же переведены в режим set_correlator_time_ntp=false

20250207

Мощность гетеродина 36 была уменьшена с 5 до 2 дБм. После наблюдений мощность была возвращена на 5 дБм.

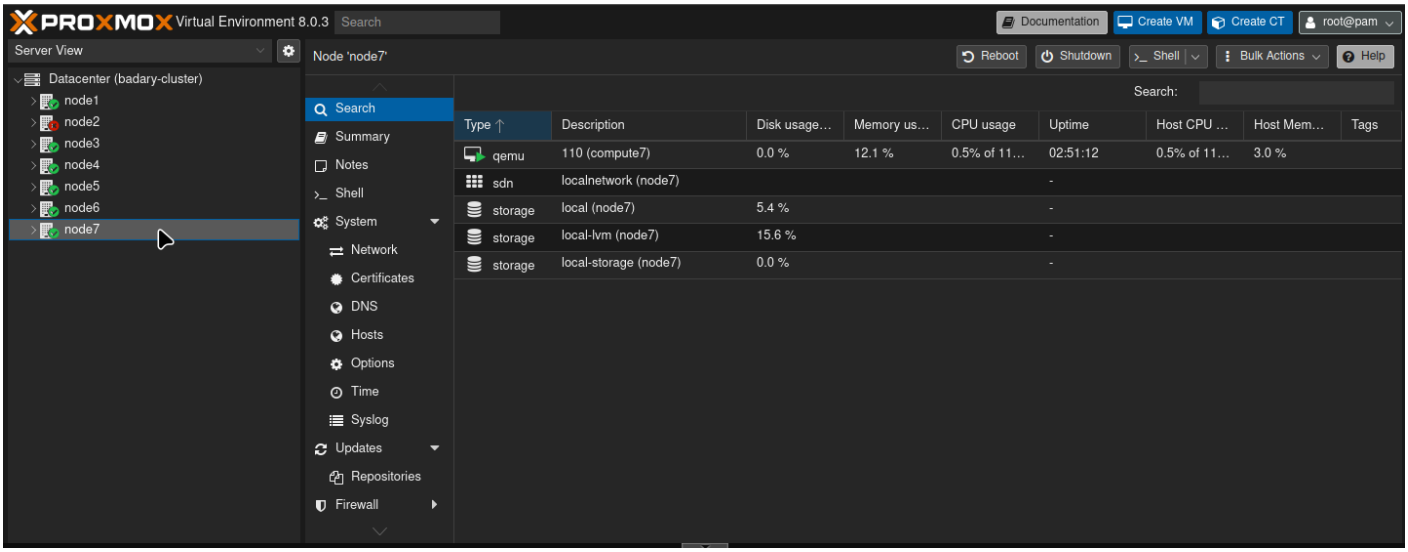
Ручное отключение питания ЦОД и СХД

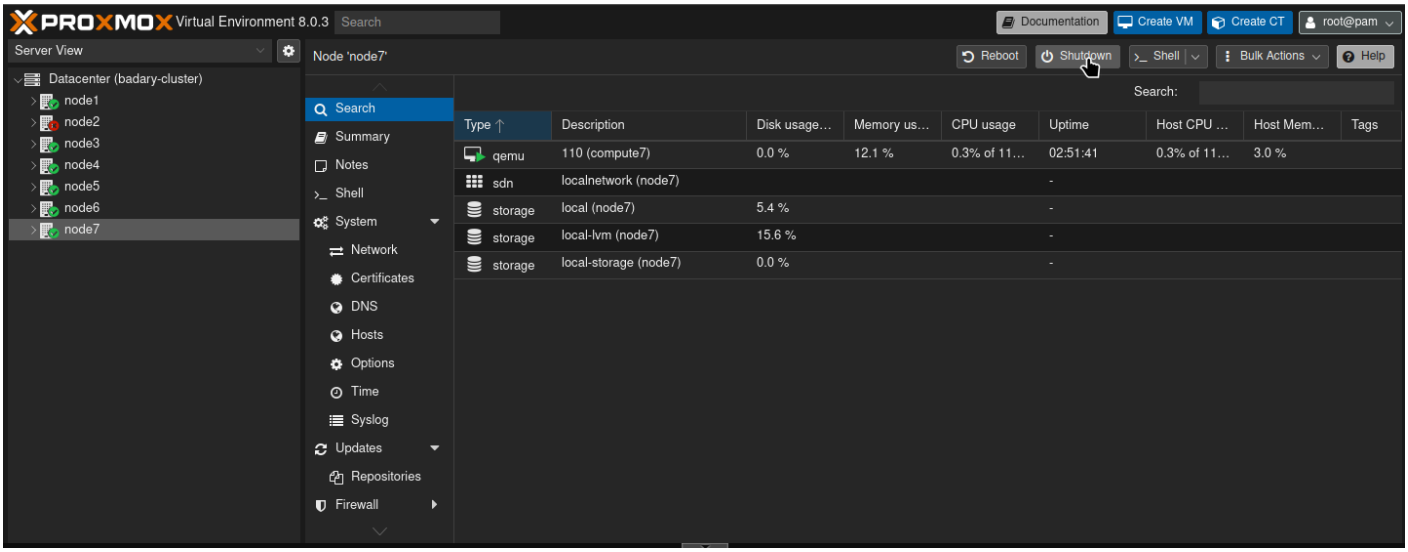
Порядок выключения

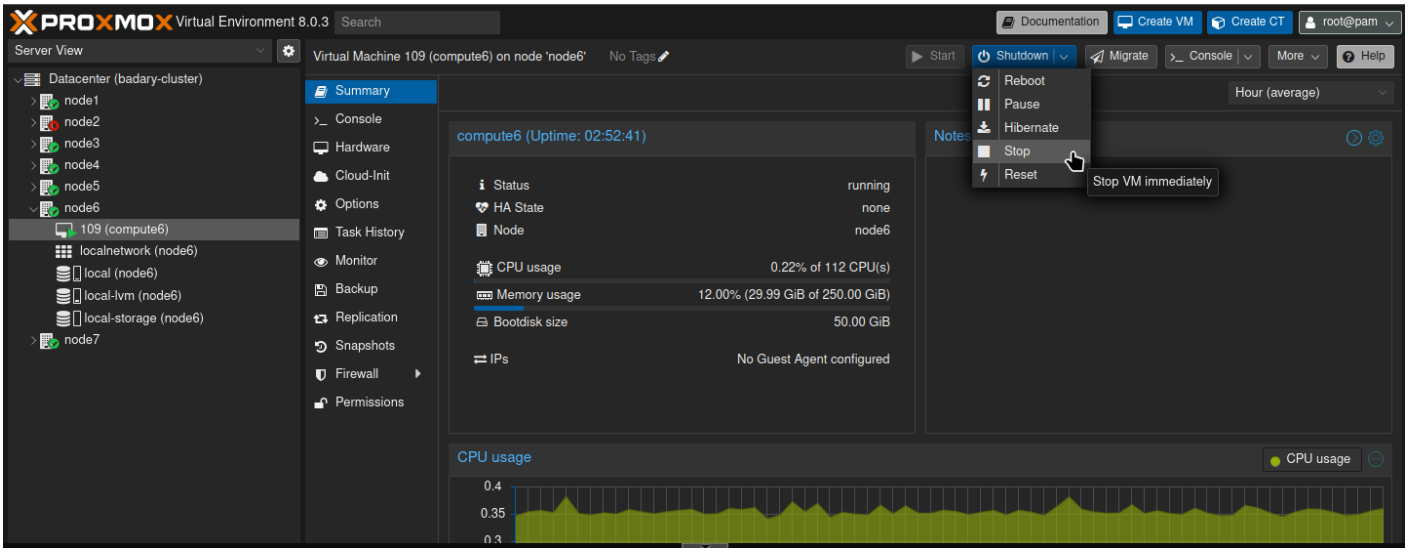
1. Сначала выключаем ЦОД, так как он зависит от хранилища + потребляет больше ресурсов
2. После ЦОД выключаем СХД

Алгоритм выключения ЦОД

1. Заходим на https://10.1.2.30:8006, вводим пароль
2. Слева в панели управления кластером видим список из 7 узлов. Выключаем каждый кнопкой Shutdown, которая находится справа сверху.
3. Выключение начинать сначала у всех узлов, кроме `node1` (т.е. 2, 3 и до 7). Убедиться, что каждый из них загорелся красным в панели слева.
4. После того как все узлы кроме `node1` горят красным, выключить `node1`
5. Попытаться обновить страницу в браузере. Если она не обновляется, недоступна, значит питание отключилось успешно
6. Если какой-то из узлов выключаться не хочет, завис, то, скорее всего, дело в какой-либо из виртуальных машин. Надо найти виртуалку, которая блокирует выключение, и нажать в меню справа сверху кнопку `Stop`, предварительно дождавшись.

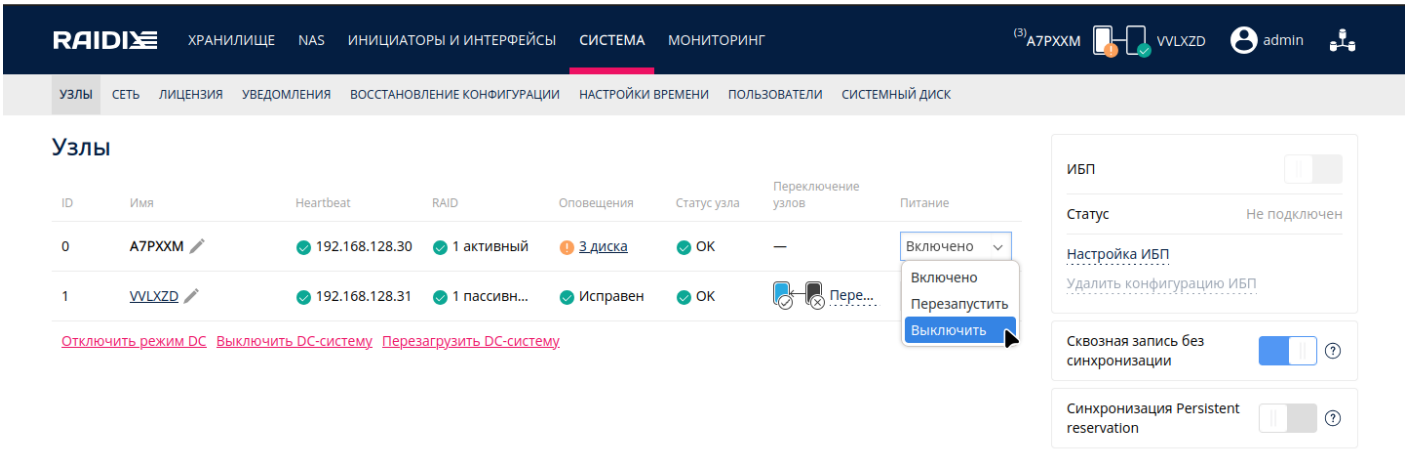






Алгоритм выключения СХД

1. Точки входа на СХД: http://10.1.2.45, http://10.1.2.46, http://10.1.2.47, http://10.1.2.48.
2. Заходим по каждому из этих адресов, жмём на вкладку "Система" и видим список узлов СХД. Выбираем для каждого в выпадающем меню "Выключить", жмём ОК.
3. Пробуем обновить страницу, проверяем, что страница не открывается, выдаётся, ошибка, и.т.д.


RAIDIX
ХРАНИЛИЩЕ
NAS
ИНИЦИАТОРЫ И ИНТЕРФЕЙСЫ
СИСТЕМА
МОНИТОРИНГ
(3) A7PXXM
VVLXZD
admin
УЗЛЫ
СЕТЬ
ЛИЦЕНЗИЯ
УВЕДОМЛЕНИЯ
ВОССТАНОВЛЕНИЕ КОНФИГУРАЦИИ
НАСТРОЙКИ ВРЕМЕНИ
ПОЛЬЗОВАТЕЛИ
СИСТЕМНЫЙ ДИСК
Узлы
ID
Имя
Heartbeat
RAID
Оповещения
Статус узла
Переключение узлов
Питание
0
A7PXXM
192.168.128.30
1 активный
3 диска
OK
—
Включено
1
VVLXZD
192.168.128.31
1 пассивн...
Исправен
OK
Пере...
Включено
Перезапустить
Выключить
Отключить режим DC
Выключить DC-систему
Перезагрузить DC-систему
ИБП
Статус
Не подключен
Настройка ИБП
Удалить конфигурацию ИБП
Сквозная запись без синхронизации
Синхронизация Persistent reservation