Optimizing the Rubin DRP
CC–IN2P3, Lyon

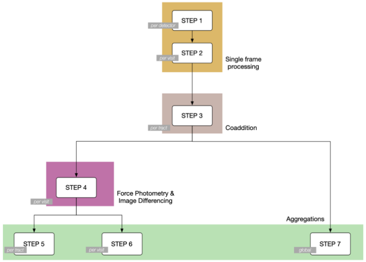
J. Bregeon, D. Boutigny, F. Hernandez, Q. Leboulc'h, D. Parello, C

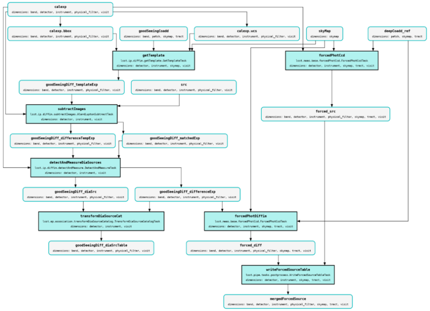
December 14, 2023

Context

- CC-IN2P3 is one of the 3 main Rubin LSST data center (35% US, 25% UK)
  - IN2P3 signed to process 40% of LSST data
  - Rubin **DRP**
- A lot of data
  - 6.4 GB per exposure, i.e. 16 TB per night
  - 15 PB of "catalogues" after 10 years

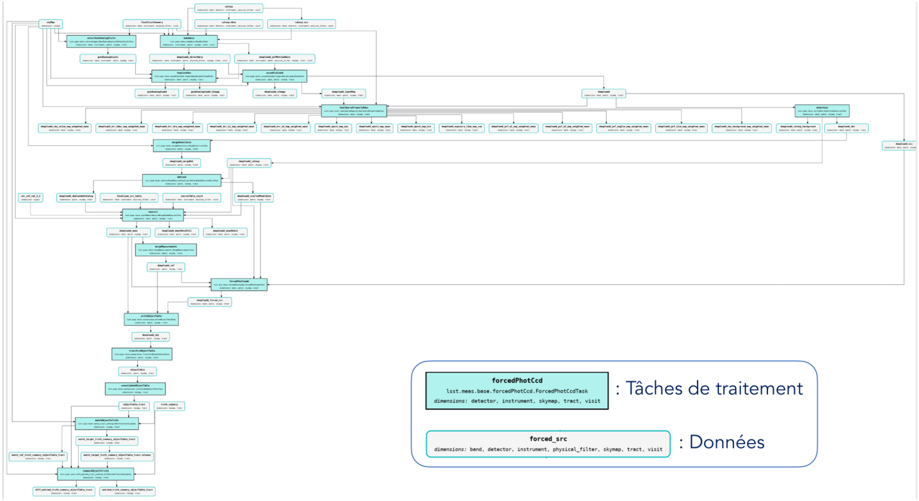
And a complex pipeline




calexp
calexp.bbox
goodSeeingCoadd
calexp.wcs
skyMap
deepCoadd_ref
getTemplate
lsst.ip.diffim.getTemplate.GetTemplateTask
forcedPhotCcd
lsst.meas.base.forcedPhotCcd.ForcedPhotCcdTask
goodSeeingDiff_templateExp
src
subtractImages
lsst.ip.diffim.subtractImages.AlardLuptonSubtractTask
forced_src
goodSeeingDiff_differenceTempExp
goodSeeingDiff_matchedExp
detectAndMeasureDiaSources
lsst.ip.diffim.detectAndMeasure.DetectAndMeasureTask
goodSeeingDiff_diaSrc
goodSeeingDiff_differenceExp
transformDiaSourceCat
lsst.ap.association.transformDiaSourceCatalog.TransformDiaSourceCatalogTask
forcedPhotDiffim
lsst.meas.base.forcedPhotCcd.ForcedPhotCcdTask
goodSeeingDiff_diaSrcTable
forced_diff
writeForcedSourceTable
lsst.pipe.tasks.postprocess.WriteForcedSourceTableTask
mergedForcedSource

- 80 tasks in 7 main steps
- each task is a workflow of its own

really complex



forcedPhotCcd
lsst.meas.base.forcedPhotCcd.ForcedPhotCcdTask
dimensions: detector, instrument, skymap, tract, visit

: Tâches de traitement

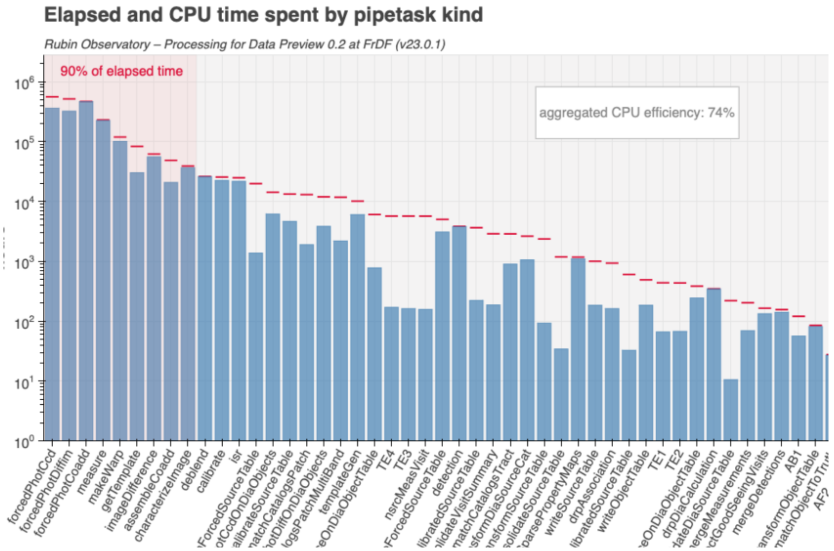
forced_src
dimensions: band, detector, instrument, physical_filter, skymap, tract, visit

: Données

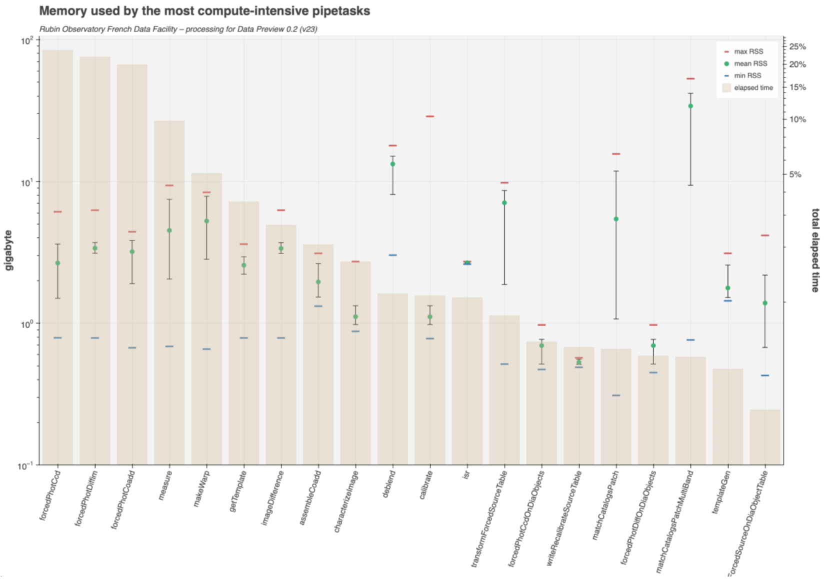
RubinOP

- Interdisciplinary Team
  - Quentin Leboulc’h @ CC-IN2P3
  - Camille Parisel @ APC
  - David Parello @ Univ. Perpignan & LIRMM (Researcher in Computer Science)
  - and Fabio, Dominique and myself. . .
- Goal: Optimize the DRP
  - CPU consumption
  - RAM requirements
  - disk space needs
- Expected results
  - reduce costs and carbon footprint of DRP
  - free some resources for cosmology analysis

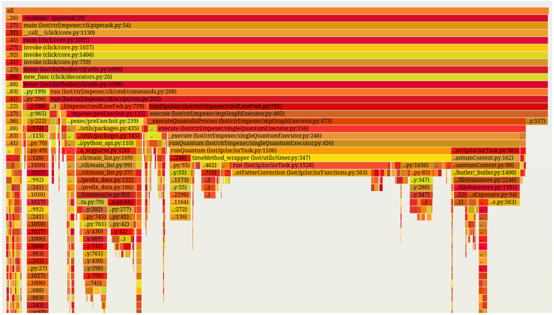
Task level CPU profiling

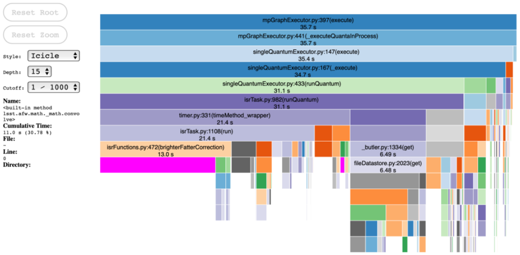


Task level memory profiling



py-spy and cprofile on ISR





Style: Icicle

Depth: 15

Cutoff: 1 ⁄ 1000

Name:
<built-in method lsst.afw.math._math.convolve>

Cumulative Time:
11.0 s (30.78 %)

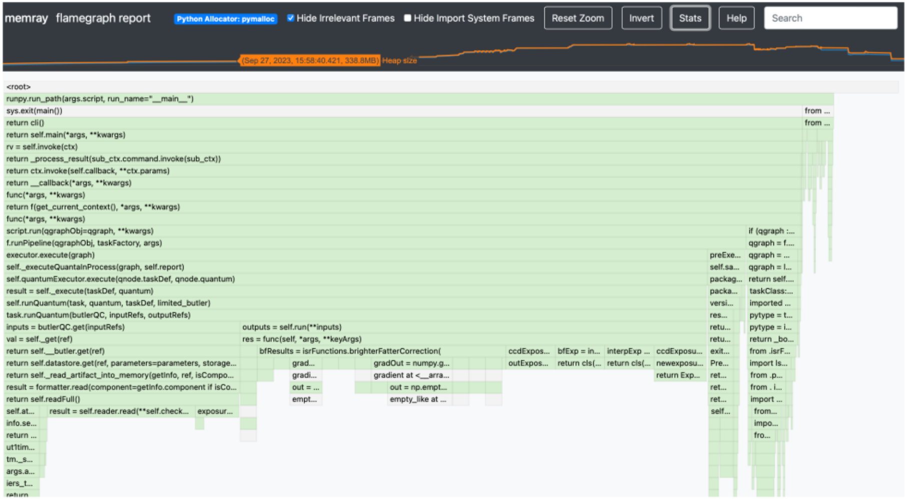
File:
~

Line:
0

Directory:

mpGraphExecutor.py:397(execute) 35.7 s
mpGraphExecutor.py:441(_executeQuantaInProcess) 35.7 s
singleQuantumExecutor.py:147(execute) 35.4 s
singleQuantumExecutor.py:167(_execute) 34.7 s
singleQuantumExecutor.py:433(runQuantum) 31.1 s
isrTask.py:982(runQuantum) 31.1 s
timer.py:331(timeMethod_wrapper) 21.4 s
isrTask.py:1108(run) 21.4 s
isrFunctions.py:472(brighterFatterCorrection) 13.0 s
_butler.py:1334(get) 6.49 s
fileDatastore.py:2023(get) 6.48 s

Memray on Isr


memray flamegraph report
Python Allocator: pymalloc
Hide Irrelevant Frames
Hide Import System Frames
Reset Zoom
Invert
Stats
Help
Search
(Sep 27, 2023, 15:58:40.421, 338.8MB)
Heap size

```
<root>
runpy.run_path(args.script, run_name="__main__")
sys.exit(main())                                          from ...
return cli()                                              from ...
return self.main(*args, **kwargs)
rv = self.invoke(ctx)
return _process_result(sub_ctx.command.invoke(sub_ctx))
return ctx.invoke(self.callback, **ctx.params)
return __callback(*args, **kwargs)
func(*args, **kwargs)
return f(get_current_context(), *args, **kwargs)
func(*args, **kwargs)
script.run(qgraphObj=qgraph, **kwargs)                    if (qgraph :...
f.runPipeline(qgraphObj, taskFactory, args)               qgraph = f....
executor.execute(graph)                        preExe...  qgraph = ...
self._executeQuantaInProcess(graph, self.report)   self.sa...  qgraph = l...
self.quantumExecutor.execute(qnode.taskDef, qnode.quantum)   packag...  return self....
result = self._execute(taskDef, quantum)       packa...  taskClass:...
self.runQuantum(task, quantum, taskDef, limited_butler)   versi...  imported ...
task.runQuantum(butlerQC, inputRefs, outputRefs)   res...  pytype = t...
inputs = butlerQC.get(inputRefs)   outputs = self.run(**inputs)   retu...  pytype = i...
val = self._get(ref)   res = func(self, *args, **keyArgs)   retu...  return _bo...
return self.__butler.get(ref)   bfResults = isrFunctions.brighterFatterCorrection(   ccdExpos...   bfExp = in...   interpExp ...   ccdExposu...   exit...   from .isrF...
return self.datastore.get(ref, parameters=parameters, storage...   grad...   gradOut = numpy.g...   outExpos...   return cls(...   return cls(...   newexposu...   Pre...   import Is...
return self._read_artifact_into_memory(getInfo, ref, isCompo...   gradi...   gradient at <__arra...   return Exp...   ret...   from .p...
result = formatter.read(component=getInfo.component if isCo...   out = ...   out = np.empt...   ret...   from . i...
return self.readFull()   empt...   empty_like at ...   ret...   import ...
self.at...   result = self.reader.read(**self.check...   exposur...   self...   from...
info.se...   impo...
return ...   fro...
ut1tim...
tm._s...
args.a...
iers_t...
return
```

Conclusions

- Optimizing the code is good on all aspects
- Profiling first
- The team has made a lot of progress
  - ⇒ let's keep our momentum