SAMPO

SPD Software&Computing weekly meeting

26/11/24

Updates

New Gaudi image developed:

- OS: CentOS7 -> AlmaLinux9.5
- Gaudi: v36r9 -> v38r3
- GCC: 11.3.0 -> 11.5.0
- Also some dependencies updated

Why it is important:

- Reduced size compared to previous one
- Dockerfile available, but now it looks like total mess…
- Can be used as base image for future releases

VS code as development area

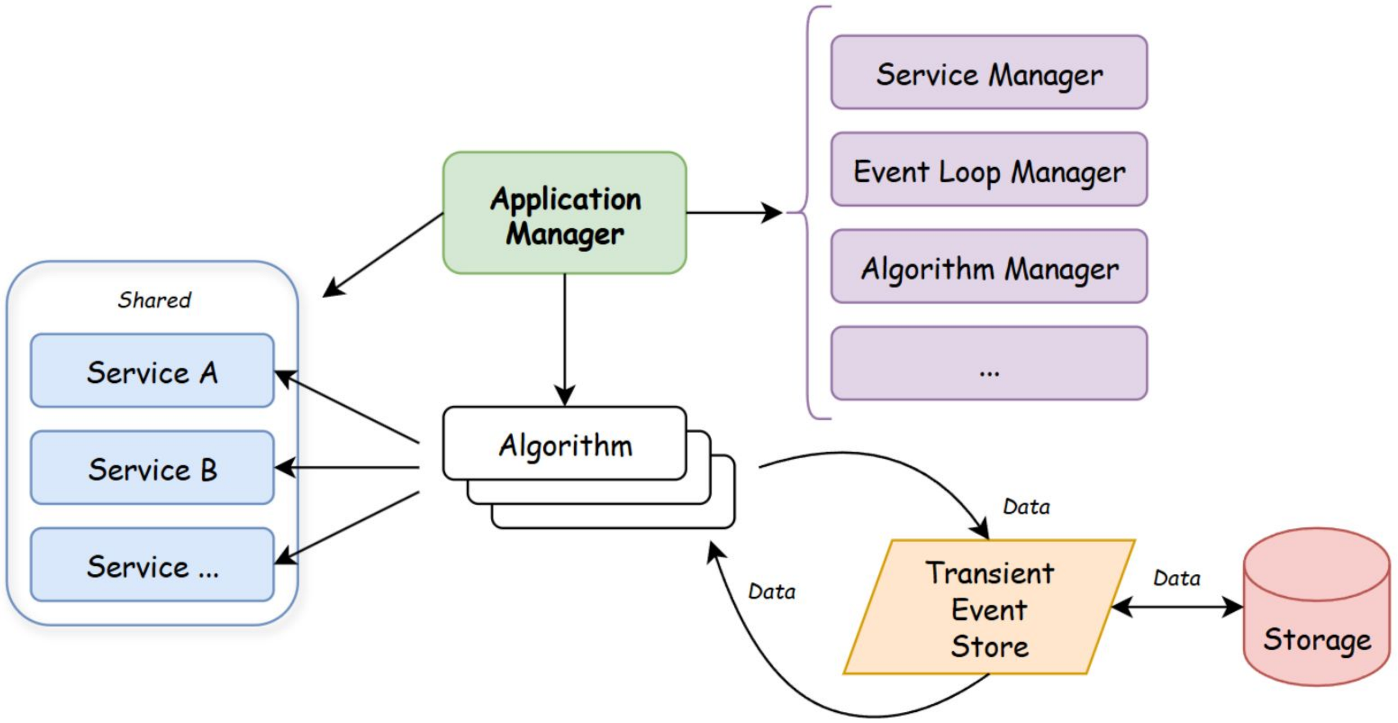
Gaudi is a CMake-configured project with many dependencies, thus it is distributed in container.

If you use VS code, then consider:

-> CMake Tools

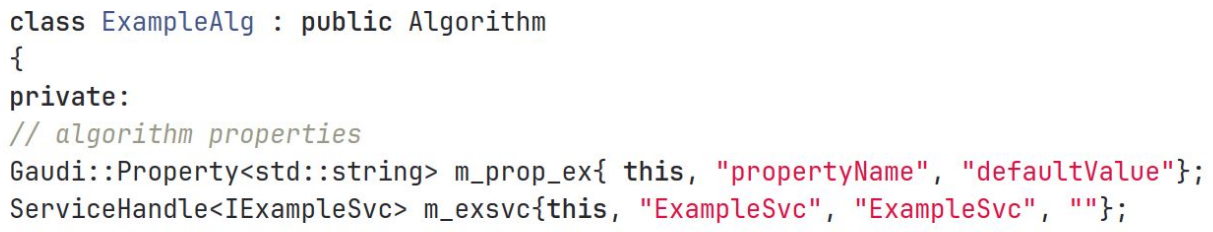
-> Dev containers

Gaudi components reminder



JobOptions

- Job execution order, define components to be loaded
- Input/Output files
- Components’ properties



```
class ExampleAlg : public Algorithm
{
private:
// algorithm properties
Gaudi::Property<std::string> m_prop_ex{ this, "propertyName", "defaultValue"};
ServiceHandle<IExampleSvc> m_exsvc{this, "ExampleSvc", "ExampleSvc", ""};
```

…

JobOptions: how things work

When module.so is compiled, *listcomponents* and *genconf* programms are executed:

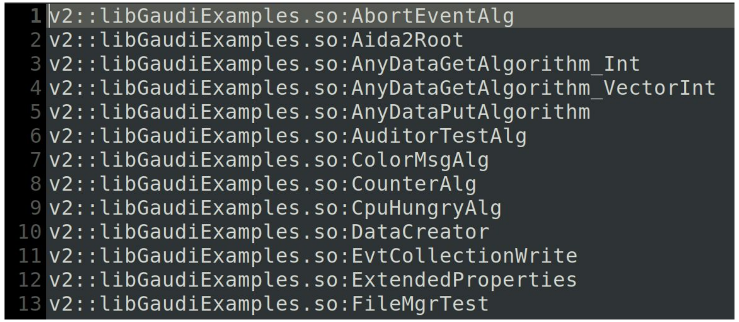
listcomponents -> in which .so file should I look for component?

Module.components file generated as a result.

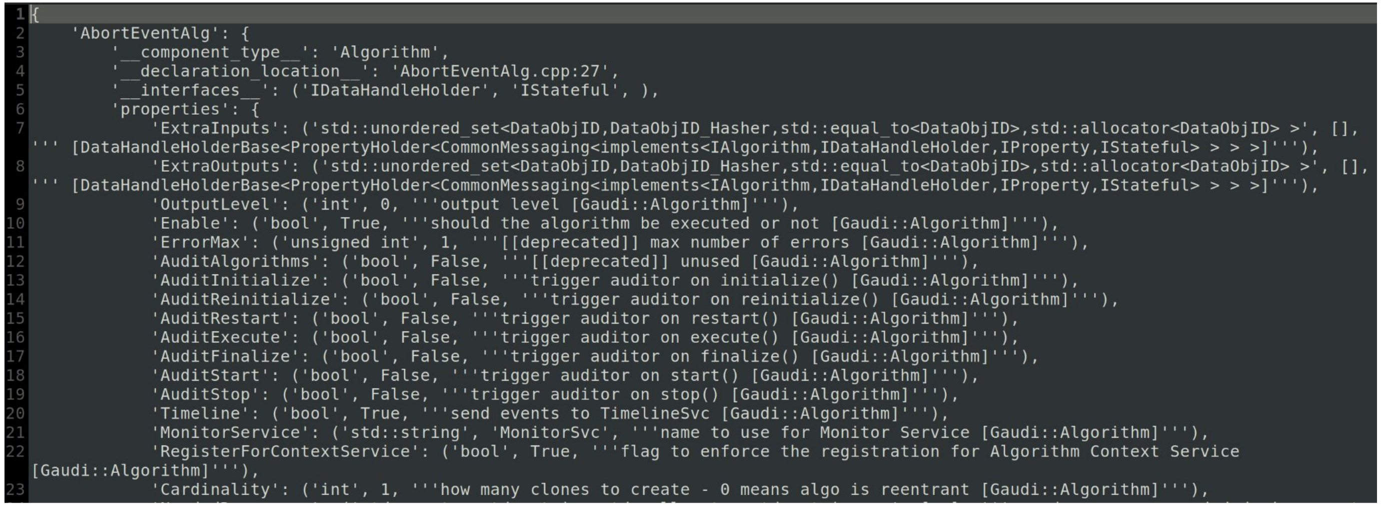
genconf -> what properties does this component have?

Module.confdb file generated as result.

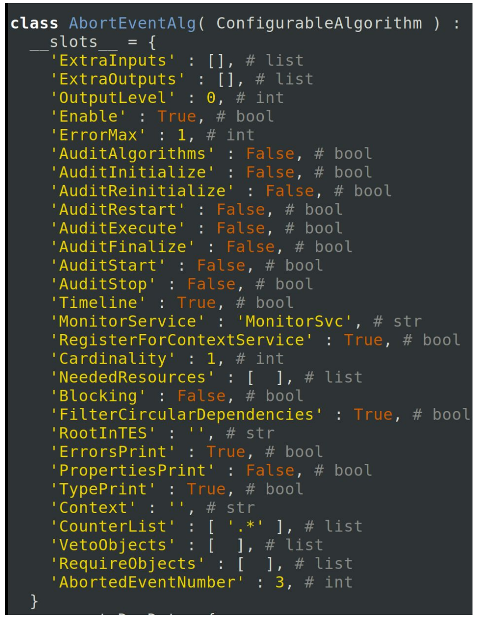
JobOptions: .components and .confdb



```
v2::libGaudiExamples.so:AbortEventAlg
v2::libGaudiExamples.so:Aida2Root
v2::libGaudiExamples.so:AnyDataGetAlgorithm_Int
v2::libGaudiExamples.so:AnyDataGetAlgorithm_VectorInt
v2::libGaudiExamples.so:AnyDataPutAlgorithm
v2::libGaudiExamples.so:AuditorTestAlg
v2::libGaudiExamples.so:ColorMsgAlg
v2::libGaudiExamples.so:CounterAlg
v2::libGaudiExamples.so:CpuHungryAlg
v2::libGaudiExamples.so:DataCreator
v2::libGaudiExamples.so:EvtCollectionWrite
v2::libGaudiExamples.so:ExtendedProperties
v2::libGaudiExamples.so:FileMgrTest
```

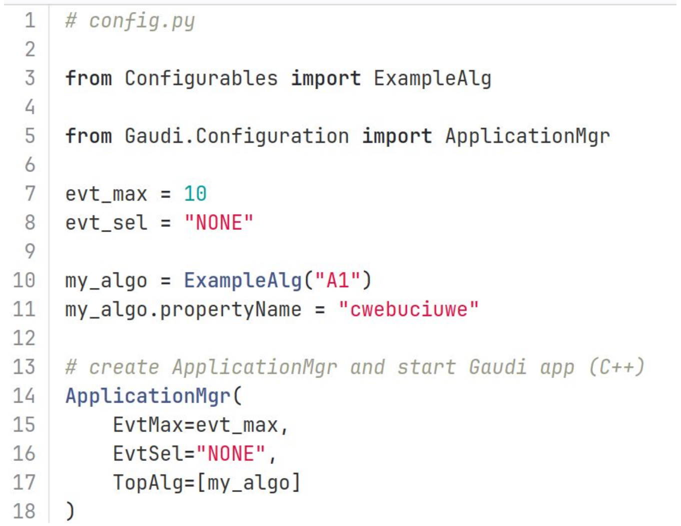


```
{
    'AbortEventAlg': {
        '__component_type__': 'Algorithm',
        '__declaration_location__': 'AbortEventAlg.cpp:27',
        '__interfaces__': ('IDataHandleHolder', 'IStateful', ),
        'properties': {
            'ExtraInputs': ('std::unordered_set<DataObjID,DataObjID_Hasher,std::equal_to<DataObjID>,std::allocator<DataObjID> >', [],
''' [DataHandleHolderBase<PropertyHolder<CommonMessaging<implements<IAlgorithm,IDataHandleHolder,IProperty,IStateful> > > >]'''),
            'ExtraOutputs': ('std::unordered_set<DataObjID,DataObjID_Hasher,std::equal_to<DataObjID>,std::allocator<DataObjID> >', [],
''' [DataHandleHolderBase<PropertyHolder<CommonMessaging<implements<IAlgorithm,IDataHandleHolder,IProperty,IStateful> > > >]'''),
            'OutputLevel': ('int', 0, '''output level [Gaudi::Algorithm]'''),
            'Enable': ('bool', True, '''should the algorithm be executed or not [Gaudi::Algorithm]'''),
            'ErrorMax': ('unsigned int', 1, '''[[deprecated]] max number of errors [Gaudi::Algorithm]'''),
            'AuditAlgorithms': ('bool', False, '''[[deprecated]] unused [Gaudi::Algorithm]'''),
            'AuditInitialize': ('bool', False, '''trigger auditor on initialize() [Gaudi::Algorithm]'''),
            'AuditReinitialize': ('bool', False, '''trigger auditor on reinitialize() [Gaudi::Algorithm]'''),
            'AuditRestart': ('bool', False, '''trigger auditor on restart() [Gaudi::Algorithm]'''),
            'AuditExecute': ('bool', False, '''trigger auditor on execute() [Gaudi::Algorithm]'''),
            'AuditFinalize': ('bool', False, '''trigger auditor on finalize() [Gaudi::Algorithm]'''),
            'AuditStart': ('bool', False, '''trigger auditor on start() [Gaudi::Algorithm]'''),
            'AuditStop': ('bool', False, '''trigger auditor on stop() [Gaudi::Algorithm]'''),
            'Timeline': ('bool', True, '''send events to TimelineSvc [Gaudi::Algorithm]'''),
            'MonitorService': ('std::string', 'MonitorSvc', '''name to use for Monitor Service [Gaudi::Algorithm]'''),
            'RegisterForContextService': ('bool', True, '''flag to enforce the registration for Algorithm Context Service
[Gaudi::Algorithm]'''),
            'Cardinality': ('int', 1, '''how many clones to create - 0 means algo is reentrant [Gaudi::Algorithm]'''),
```

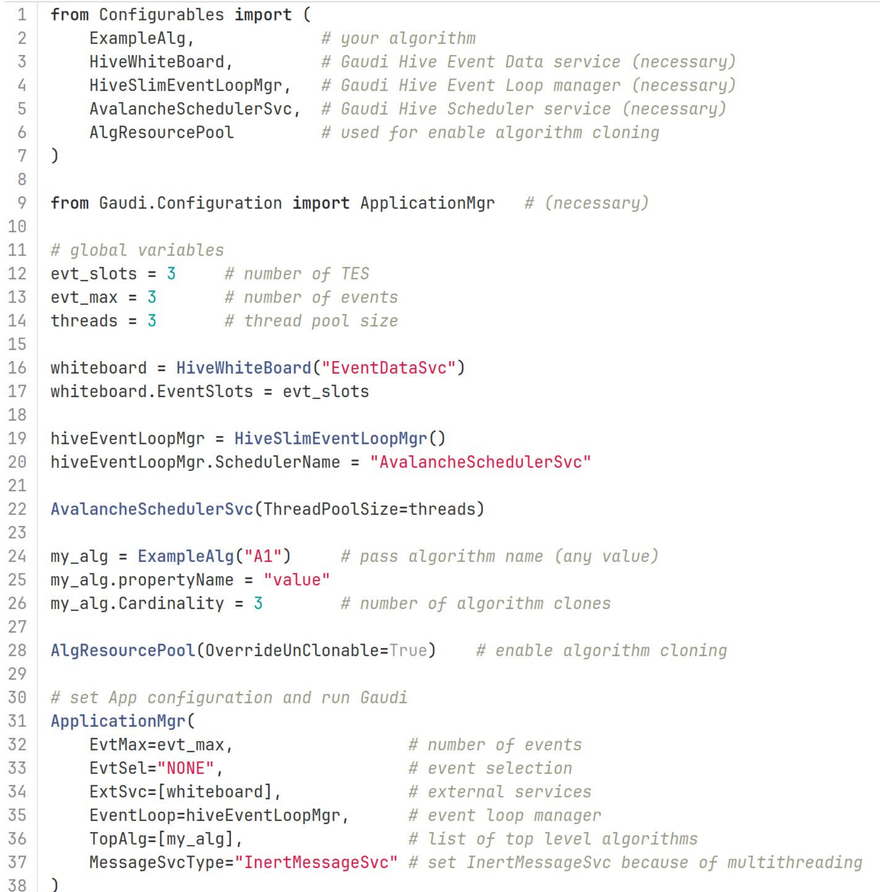


```
class AbortEventAlg( ConfigurableAlgorithm ) :
  __slots__ = {
    'ExtraInputs' : [], # list
    'ExtraOutputs' : [], # list
    'OutputLevel' : 0, # int
    'Enable' : True, # bool
    'ErrorMax' : 1, # int
    'AuditAlgorithms' : False, # bool
    'AuditInitialize' : False, # bool
    'AuditReinitialize' : False, # bool
    'AuditRestart' : False, # bool
    'AuditExecute' : False, # bool
    'AuditFinalize' : False, # bool
    'AuditStart' : False, # bool
    'AuditStop' : False, # bool
    'Timeline' : True, # bool
    'MonitorService' : 'MonitorSvc', # str
    'RegisterForContextService' : True, # bool
    'Cardinality' : 1, # int
    'NeededResources' : [  ], # list
    'Blocking' : False, # bool
    'FilterCircularDependencies' : True, # bool
    'RootInTES' : '', # str
    'ErrorsPrint' : True, # bool
    'PropertiesPrint' : False, # bool
    'TypePrint' : True, # bool
    'Context' : '', # str
    'CounterList' : [ '.*' ], # list
    'VetoObjects' : [  ], # list
    'RequireObjects' : [  ], # list
    'AbortedEventNumber' : 3, # int
  }
```

JobOptions: python scripts



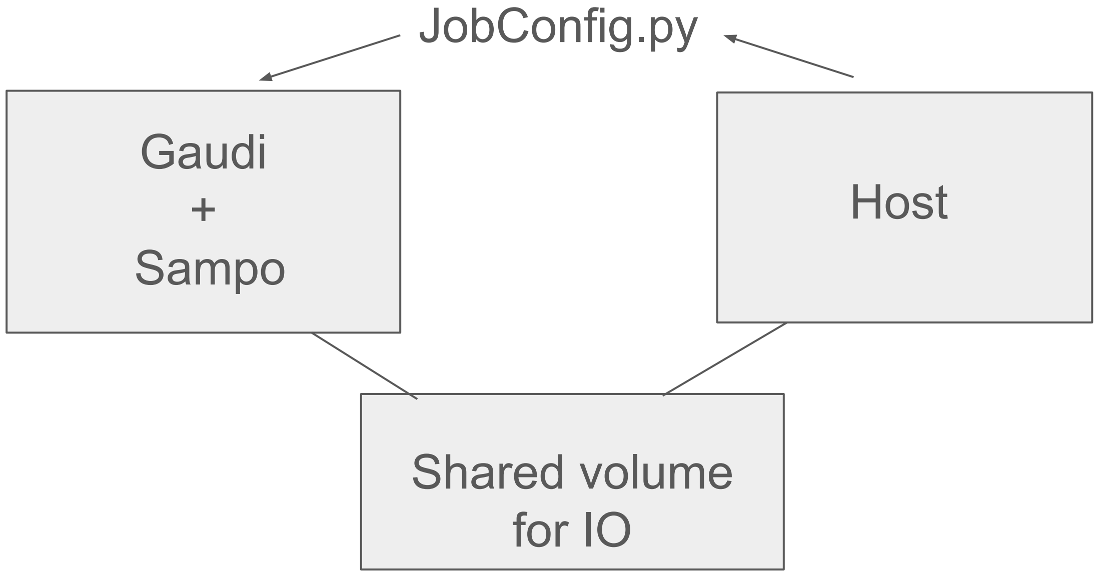
```python
# config.py

from Configurables import ExampleAlg

from Gaudi.Configuration import ApplicationMgr

evt_max = 10
evt_sel = "NONE"

my_algo = ExampleAlg("A1")
my_algo.propertyName = "cwebuciuwe"

# create ApplicationMgr and start Gaudi app (C++)
ApplicationMgr(
    EvtMax=evt_max,
    EvtSel="NONE",
    TopAlg=[my_algo]
)
```



```python
from Configurables import (
    ExampleAlg,              # your algorithm
    HiveWhiteBoard,          # Gaudi Hive Event Data service (necessary)
    HiveSlimEventLoopMgr,    # Gaudi Hive Event Loop manager (necessary)
    AvalancheSchedulerSvc,   # Gaudi Hive Scheduler service (necessary)
    AlgResourcePool          # used for enable algorithm cloning
)

from Gaudi.Configuration import ApplicationMgr   # (necessary)

# global variables
evt_slots = 3     # number of TES
evt_max = 3       # number of events
threads = 3       # thread pool size

whiteboard = HiveWhiteBoard("EventDataSvc")
whiteboard.EventSlots = evt_slots

hiveEventLoopMgr = HiveSlimEventLoopMgr()
hiveEventLoopMgr.SchedulerName = "AvalancheSchedulerSvc"

AvalancheSchedulerSvc(ThreadPoolSize=threads)

my_alg = ExampleAlg("A1")     # pass algorithm name (any value)
my_alg.propertyName = "value"
my_alg.Cardinality = 3        # number of algorithm clones

AlgResourcePool(OverrideUnClonable=True)    # enable algorithm cloning

# set App configuration and run Gaudi
ApplicationMgr(
    EvtMax=evt_max,                     # number of events
    EvtSel="NONE",                      # event selection
    ExtSvc=[whiteboard],                # external services
    EventLoop=hiveEventLoopMgr,         # event loop manager
    TopAlg=[my_alg],                    # list of top level algorithms
    MessageSvcType="InertMessageSvc" # set InertMessageSvc because of multithreading
)
```

What needs to be done

- Create repo for Gaudi dependencies
- Write Dockerfile for Sampo image building upon Gaudi base image
- Automate build process


JobConfig.py
Gaudi
+
Sampo
Host
Shared volume
for IO