从Gaudi和SNiPER谈高能物理软件框架

邹佳恒

中国科学院高能物理研究所

2023.06.10

软件框架

- 软件框架
  - 面向特定领域
  - 介于传统软件库与完整软件之间的“半成品”
  - 高度的软件复用
- 基于框架的高能物理数据处理软件系统
  - 专业的软件人员：基础、通用的底层软件模块
  - 普通用户：一般只需要关注对数据的处理算法
    - 不需接触底层复杂繁琐的技术细节
    - 大幅降低对软件开发知识和技巧的要求
  - 像搭积木一样组合不同的功能模块，处理各种不同的任务

毛坯房 → 精装修

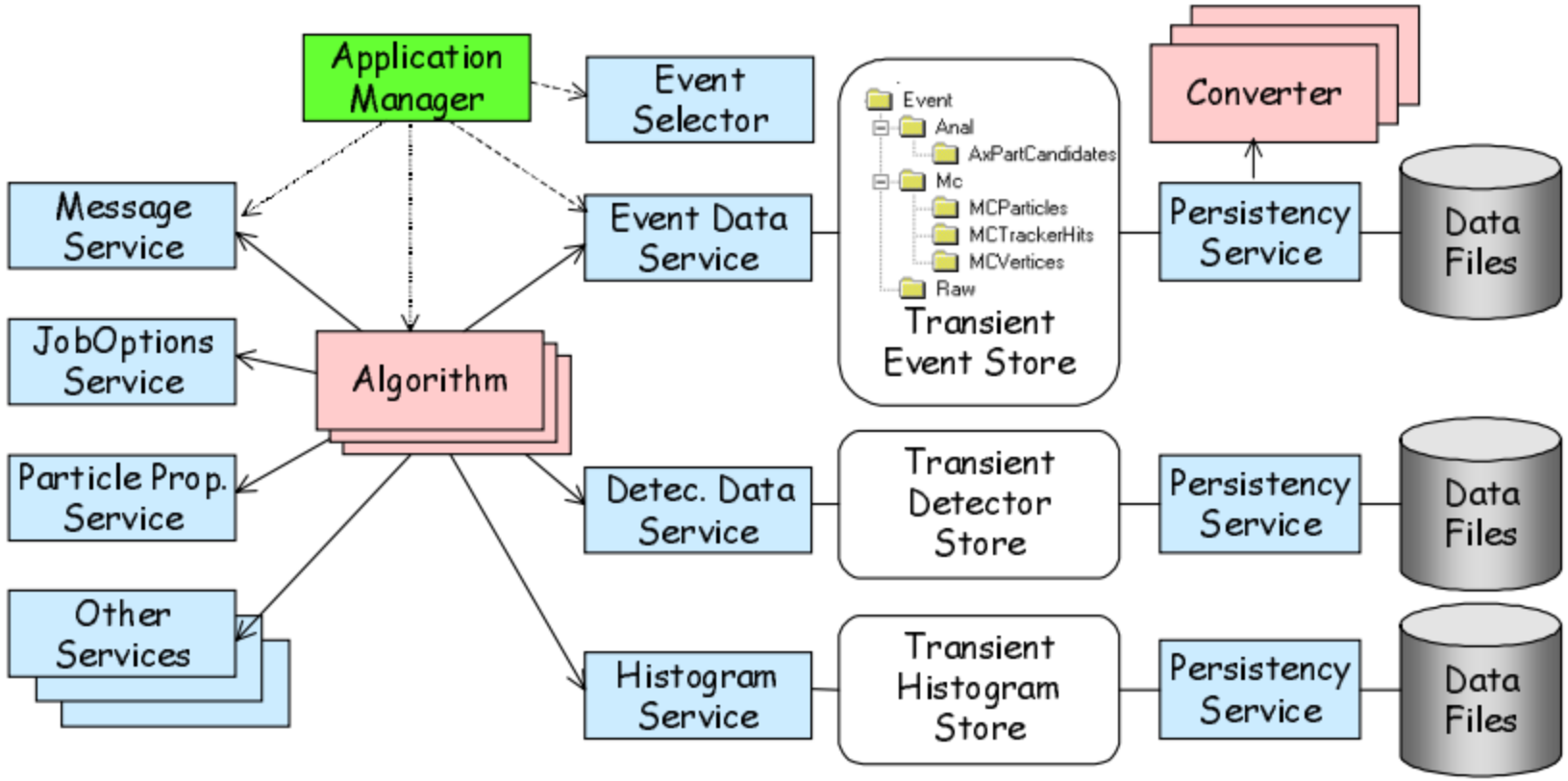
国内部分实验离线软件系统

	项目开始时间	实验类型	软件框架	主要开发语言	任务配置
DRUNK @ BESII	1990s	对撞机	?	Fortran	Text
BOSS @ BESIII	2000s	对撞机	Gaudi	C++	Text
NuWa @ DYB	2006	中微子	Gaudi、LAF	C++	Python
junosw @ JUNO	2012	中微子	SNiPER	C++	Python
LoadStar @ LHAASO	2016	宇宙线	SNiPER	C++	Python
OSCAR @ STCF	2019	对撞机	SNiPER	C++	Python
cepcsw @ CEPC	2019	对撞机	Gaudi	C++	Python

- 开发语言：C++（基本）淘汰了Fortran
- 任务配置：Python等脚本语言替代了静态文本
- 软件框架：Gaudi、SNiPER各有优势，可以预见将长期并存

Gaudi

- 上世纪90年代CERN为LHCb开发，不断发展持续至今（最新 v36r13）



- 在同为对撞机实验的BESIII上应用非常成功

DYB实验中采用Gaudi的经验 I

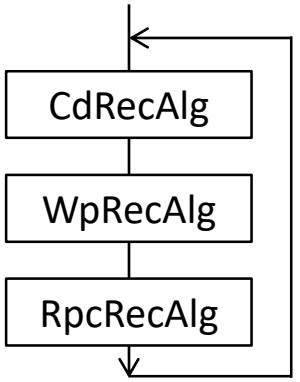
- 在DYB（和JUNO）实验中，中微子的测量是通过其反beta衰变过程

$$\bar{\nu}_e + p \rightarrow e^+ + n$$
$$e^+ + e^- \rightarrow 2\gamma$$
$$n + Gd \rightarrow Gd^* \rightarrow Gd + \gamma_s \ (8\mathrm{MeV})$$

  - 快信号和慢信号复合构成一个逻辑上的完整物理事例
  - 两个信号分别触发探测器，存在时间差，探测器观测到两个“事例”
  - 寻找中微子，需要分析“事例”间的时间关联信息
  - 模拟任务中，一个事例分裂为多个事例；分析任务中，多个事例合并为一个事例
- 数据特点
  - 各子探测器分别触发，来自不同子探测器的不同类型的“事例”混合在一起

DYB实验中采用Gaudi的经验 II

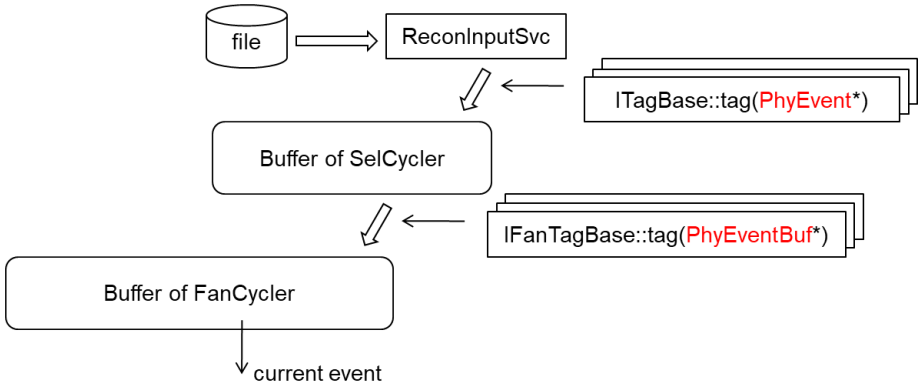
- Gaudi原本是为对撞机实验设计
  - Gaudi TES只能存放单个事例数据，事例循环过程中每次都会清空之前的数据
  - Gaudi事例循环过程中，每次都会执行所有top算法
    - 对撞机实验一般不用考虑事例分裂或合并的情况
- NuWa中的解决方案
  - 专门开发了AES，用于缓存之前一段时间窗口内的事例数据
    - 单向窗口，使用场景受限
    - 运行性能有严重瓶颈
  - NuWa中与探测器类型相关的重建等算法
    - 算法开始先判断事例类型，对不符合类型的事例直接返回
  - 涉及事例分裂的电子学模拟等软件存在一些缺陷

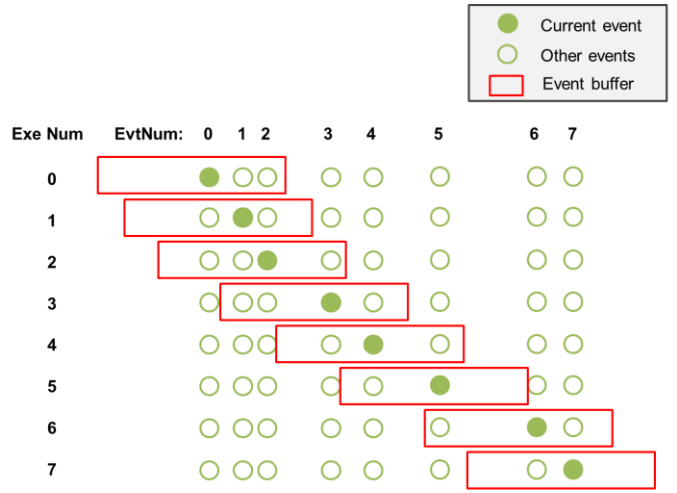


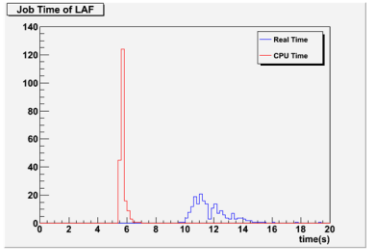
事例循环示意图

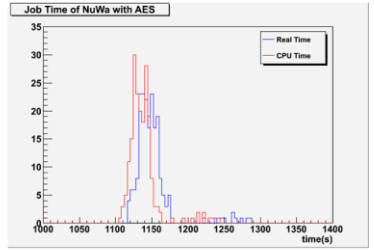
DYB实验中采用Gaudi的经验 III

- 为物理分析开发了新的不同的软件框架 LAF
  - Lightweight Analysis Framework (轻量，分析专用)
  - 双向时间窗口
  - 双层嵌套事例循环
    - 解决分析过程中本底过滤与事例合并的问题
  - 运行性能相较NuWa有~100倍提升



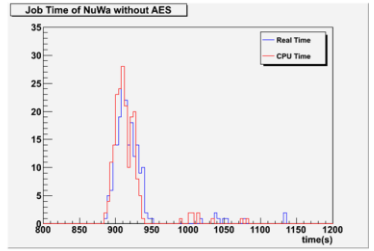






	CPU Time	Real Time
LAF	5.7	11.9
NuWa (0.1s AES)	1138	1151
NuWa (no AES)	917	922

average time (s) to process a recon. file



SNiPER的起因

- ~2012年我们开始考虑JUNO实验的离线软件系统
  - 当时世界上并没有适合中微子实验的通用软件框架
  - 基于NuWa的经验，Gaudi被排除
  - 基于LAF的经验，有了自主开发适用于中微子实验软件框架的信心
- SNiPER: Software for Non-collider Physics ExpeRiments
  - 以通用软件框架为目标
    - 适用于包括模拟、重建和分析的所有数据处理任务
    - 适用于JUNO之外的其它实验
  - 以Gaudi为师，沿用Gaudi中的算法、服务等概念
  - 以Gaudi在NuWa中暴露的不足为鉴，全新设计从0开发

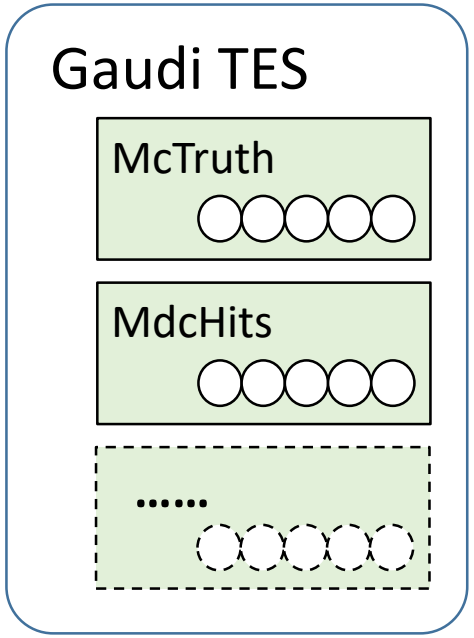
Gaudi & SNiPER

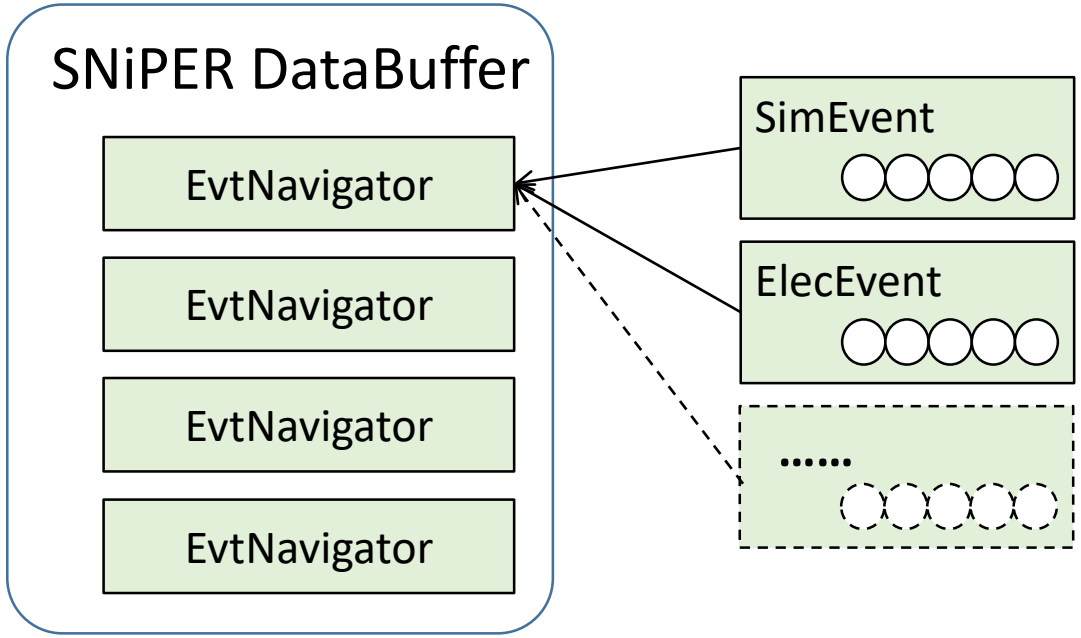
- Gaudi开始时间不晚于1998年
- 代码量：GaudiKernel ~5w行，整体10w+行
- GaudiKernel依赖ROOT、Boost、tbb和Python等，可选组件有XERCESC、CLHEP等更多依赖
- TES使用特定的数据模型基类（DataObject）
  - 优点：文件格式与内存数据解耦
  - 复杂的数据I/O和converter机制
  - 对分析等简单任务有较大性能影响
- 集成了数值单位、ParticleProperty、CLHEP等常用的物理软件功能
- 历史包袱多，有大量Histogram、Ntuple相关代码

- SNiPER开始于2012年
- 代码量：整体 ~1w行 （但将长期持续发展）
- SniperKernel只使用标准库和Linux系统调用，可选组件依赖ROOT、Boost.Python~~和tbb~~
- DataBuffer模板化，甚至可自定义DataBuffer
  - 内存数据模型可与数据文件格式关联
  - 没有converter的开销，性能更好
  - 文件格式不一致时，可在I/O服务中实现converter
- 集成ROOT处理用户的Histogram和TTree数据
- 保持轻量和简单依赖，减轻软件长期维护压力
- 目前还没有历史包袱

后面主要从SNiPER与Gaudi的不同之处，介绍两者的特点

事例数据内存管理

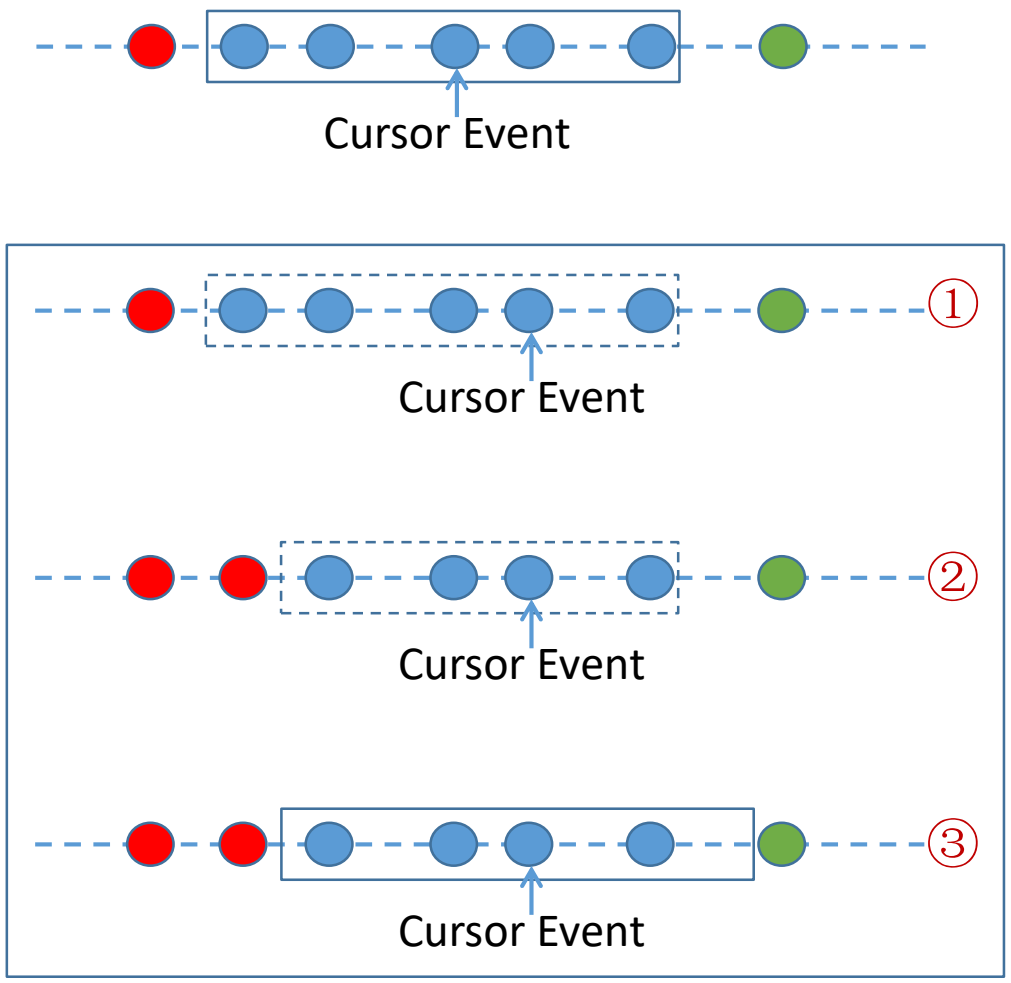




- TES中的所有数据属于同一个事例
- 下一个事例开始时，清空前一个事例的数据

- DataBuffer中存放一段时间内的多个事例
- EvtNavigator作为每个事例的入口
  - 多了一个维度，读取具体数据对象一般要多一个步骤
  - 但可借助辅助类型简化对cursor事例的简化步骤
- 事例循环过程中，DataBuffer中的事例滚动更新

JUNOSW中DataBuffer的滚动更新



The DataBuffer is updated automatically during the event loop

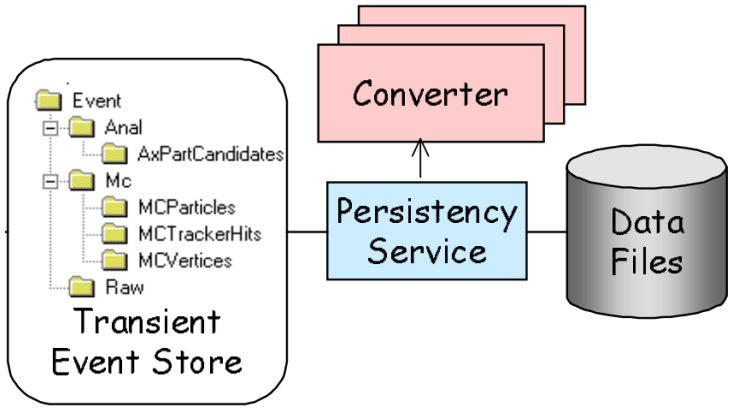
Suppose the time window is [t1, t2)

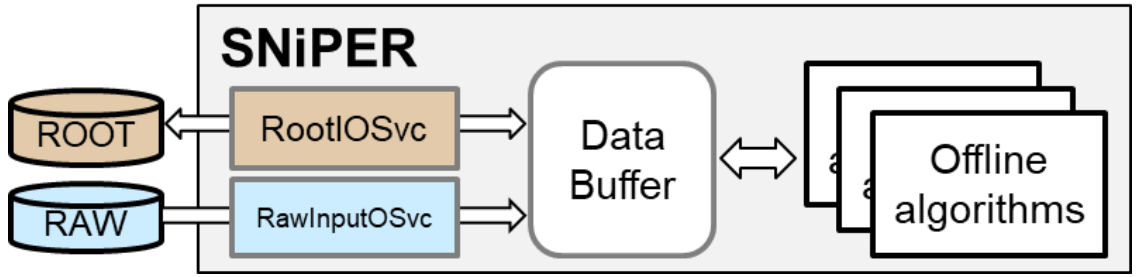
At the beginning of each event:

1. Move the cursor event to the next event
   - T0 is updated to the timestamp of the new cursor event
2. Recalculate the lower boundary of the time window
   - Remove any event that T-T0 < t1 from the DataBuffer
3. Recalculate the higher boundary of the time window
   - Fill any new event that T-T0 < t2 to the DataBuffer

The number of events to be removed in step 2 and to be filled in step 3 are not fixed

数据I/O

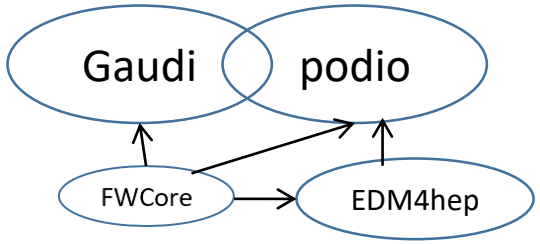




- Gaudi内存中的数据必须为DataObject的子类型
- 使用converter兼容各种不同格式的文件
- I/O service工作模式有一套完整规则

- SNiPER的DataBuffer实际上需要每个实验自己定制
- I/O service直接向DataBuffer提供数据对象
  - 特定情况下可省略converter的开销
- SNiPER不介入I/O service的内部行为

- Gaudi前述特点与podio存在功能重合和冲突
- 在Key4HEP中，提供了k4FWCore
  - 为了性能没有使用converter（用DataObject子类型简单包装）
  - 为了适配Gaudi，重复了podio中的部分代码
  - 我个人认为不论从podio还是Gaudi的角度看，都有点奇怪



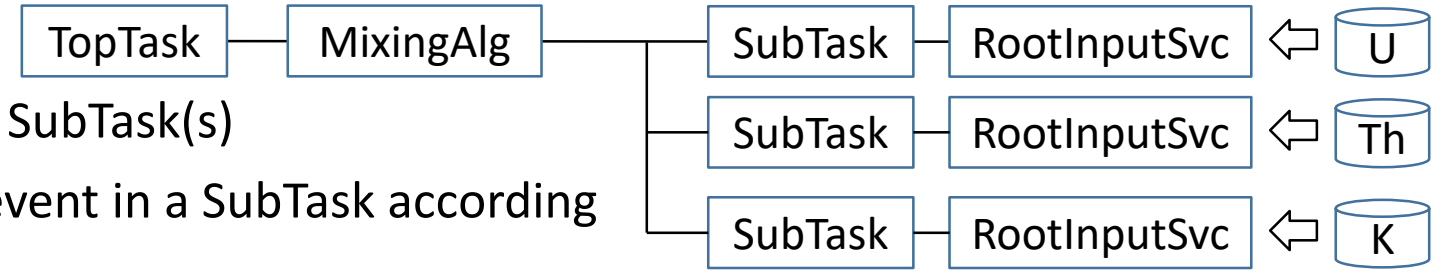
任务分支

- Gaudi中的子算法功能支持简单的任务分支
  - 软件复用粒度：算法复用
  - 父、子算法单层嵌套，它们仍在唯一的ApplicationMgr中，共享相同的TES和各种服务
- SNiPER中Task负责事例循环控制（类似Gaudi中的ApplicationMgr）
- SNiPER使用SubTask支持任务分支
  - 软件复用粒度：任务复用，复用度更高
  - Task可以多层嵌套
  - 每个子Task都可以通过配置选择自己管理DataBuffer或与其它Task共享DataBuffer
  - 不共享DataBuffer的Task之间，也能够互相访问数据

Sub-Task Use Cases

1. Event mixing (a demo)

- A simple example to demonstrate SubTask(s)
- Each time the MixingAlg get one event in a SubTask according to the event rate
- The mixed data stream is wrote out by TopTask

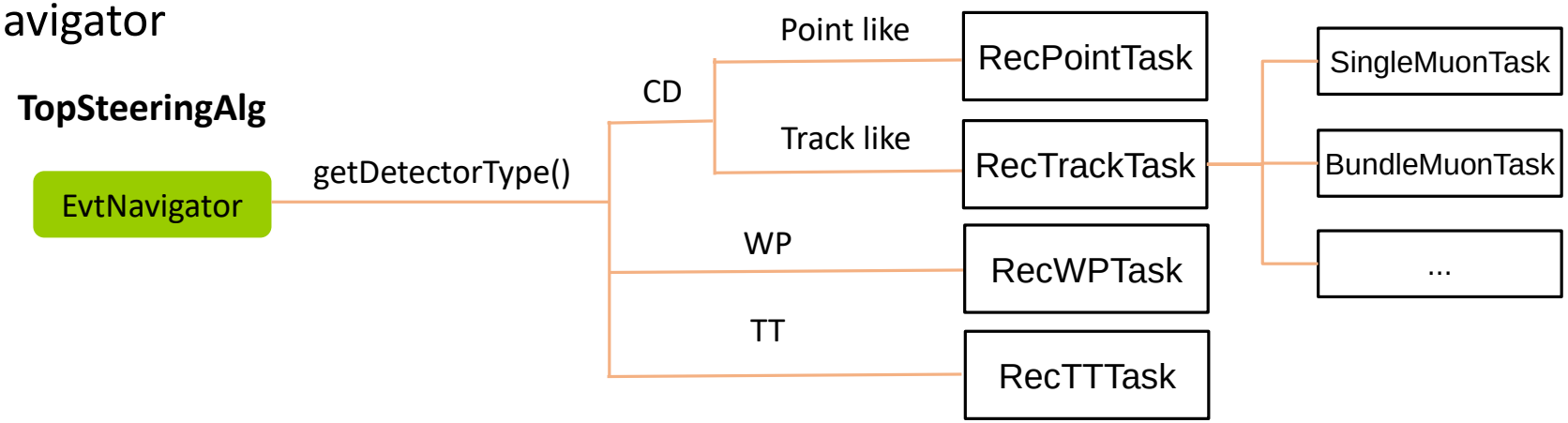


The SubTask only read event data without any algorithm in this demo

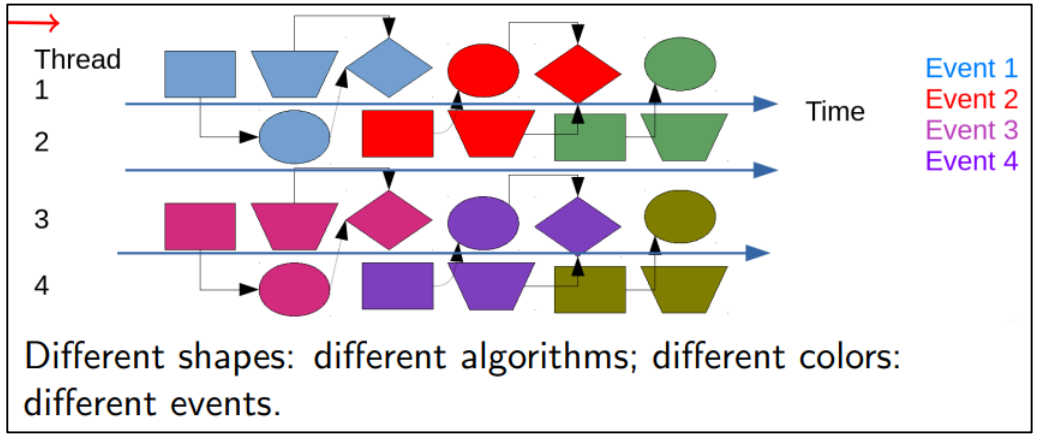
Gaudi中无法同时加载两个InputSvc，所以BOSS中MC数据混random trigger噪声时，重复了RawDataInputSvc中的一部分工作

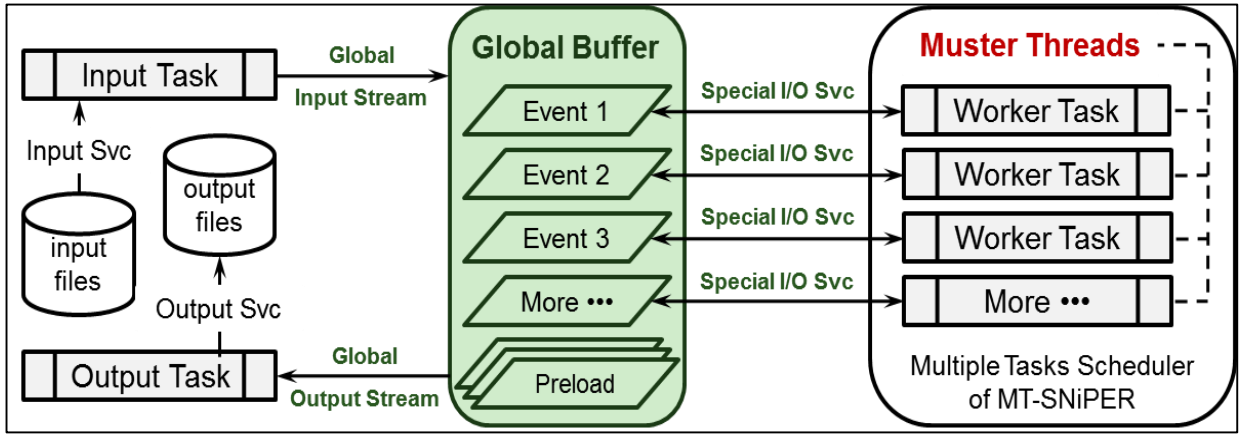
2. Reconstruction Steering (@ junosw/Reconstruction/TopSteering)

- Invoke different algorithm sequences in SubTasks according to the event classification results in EvtNavigator



多线程计算





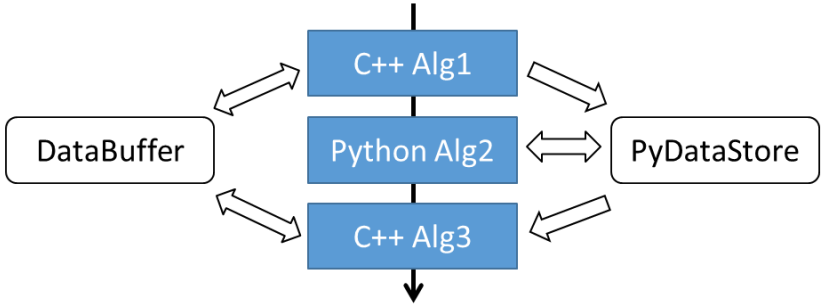
- 基于oneTBB
- 算法级并行
- AthenaMT运行性能有不错的表现

- 基于oneTBB但未使用其高级功能，可用pthread替换
- 事例级并行
- MT.SNiPER对不同数据样本性能表现差异大
  - Global Buffer需要保序，其同步要等待特别慢的事例

- 当有特别慢的算法时（如JUNO的波形重建算法），算法级并行仍不足以解决问题
- oneTBB中的任务不能暂停后再恢复，这一限制会使算法内并行变得复杂且难以通用
- 对JUNO实验来说，算法内并行势在必行
- 算法修改小的通用解决方案，是MT.SNiPER当前面临的一个艰难而紧迫的任务

Python Algorithms and PyDataStore

- A hybrid of C++ algorithms and Python algorithms
  - C++ algorithm: the recommended way to implement ordinary algorithms
  - Python algorithm: for some specific requirements, such as machine learning
- PyDataStore: a service to exchange data between C++ and Python algs

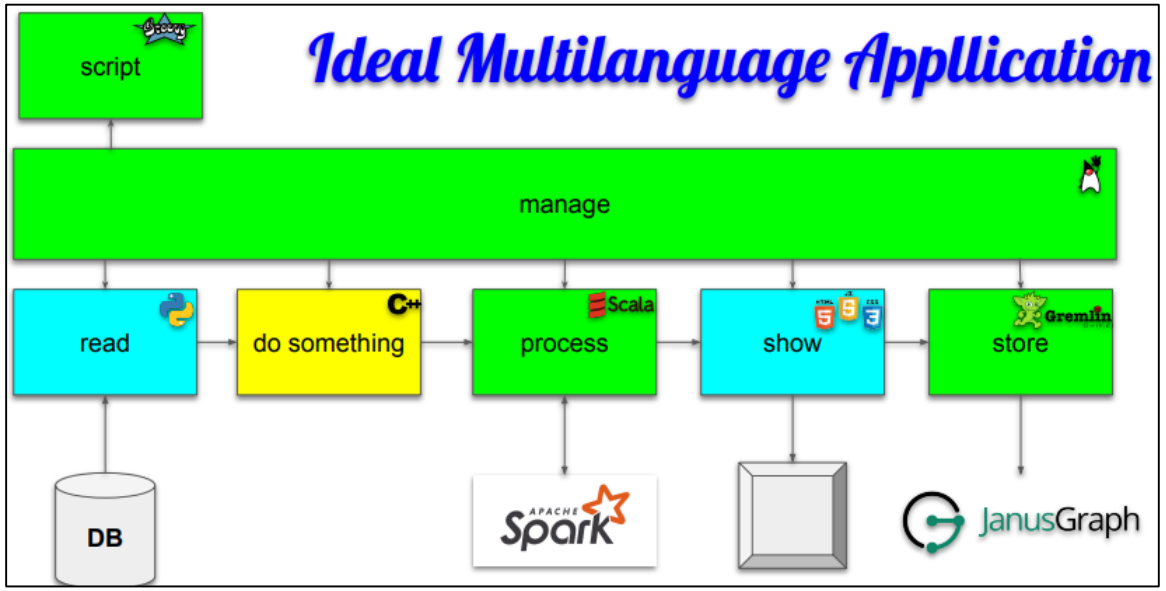


- Examples
  - A simple example to hybrid C++ and python algorithms:
    - *https://github.com/SNiPER-Framework/sniper/tree/master/Examples/PyUsages*
  - Another example with numpy, PyTouch and Tensorflow:
    - *https://github.com/JUNO-collaboration/offline-example-pyalg*

CHEP 2023：未来有无数可能

- Dune 的探索性项目 Meld
  - 函数式编程思想，将数据处理过程抽象为高阶(high-order)函数
  - 从最初设计开始高度重视并行化

Supported construct	User function	
Transform (Map)	$f(a) \to b$	*Standard data-processing idioms*
Filter	$f(a) \to$ Boolean	
Monitor	$f(a) \to$ Void	
Reduction (Fold)	$f_c(a) \to c$	*For splitting and then combining events*
Splitter (Unfold)	$f_n(a) \to (d)_n$	
Zip	—	*For combining arguments to user functions*
Sliding window	—	*To do: For sliding over adjacent events*



- 使用合适的语言或工具做合适的事情
  - Scala：并行计算
  - JavaScript：图形化
  - …
- 一种可能的方案：GraalVM
  - 多种语言同时运行于一个环境空间中

总结与展望

- SNiPER
  - 能够适应多种类型（中微子、对撞机、宇宙线、卫星等）高能物理实验
  - 框架核心功能完备，灵活性高
  - 可选组件目前少，但高度开放容易扩展
- Gaudi
  - 适合对撞机实验
  - Key4HEP默认框架，有利于算法级的软件复用
  - 国际化合作程度高
- 展望
  - 对新技术开放心态、勇于探索，取长补短，不断发展

Thanks !