江门中微子实验离线软件的发布

林韬

高能所计算中心

on behalf of JUNO Offline Group

高能物理计算和软件会议

2017/06/06 成都

OUTLINE

I. 江门中微子实验简介

II. 江门中微子实验的离线软件简介

III. 离线软件的发布

- 软件发布流程
- 版本控制管理
- 软件的编译
- 软件的部署和外部库的管理
- 软件的测试、性能测量和质量检查

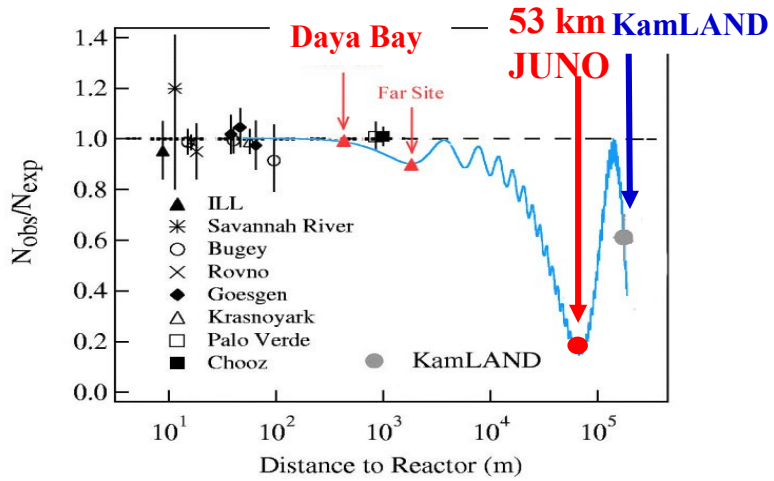
IV. 总结与展望

江门中微子实验 (JUNO)

主要的物理目标：测量中微子质量顺序

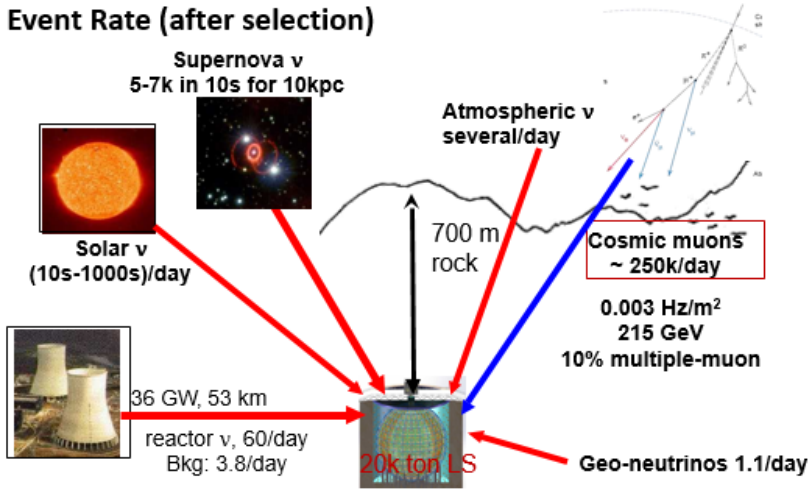
实验的基本参数：

- 地下700m
- 基线53km
- 两万吨液闪
- 3%能量分辨率



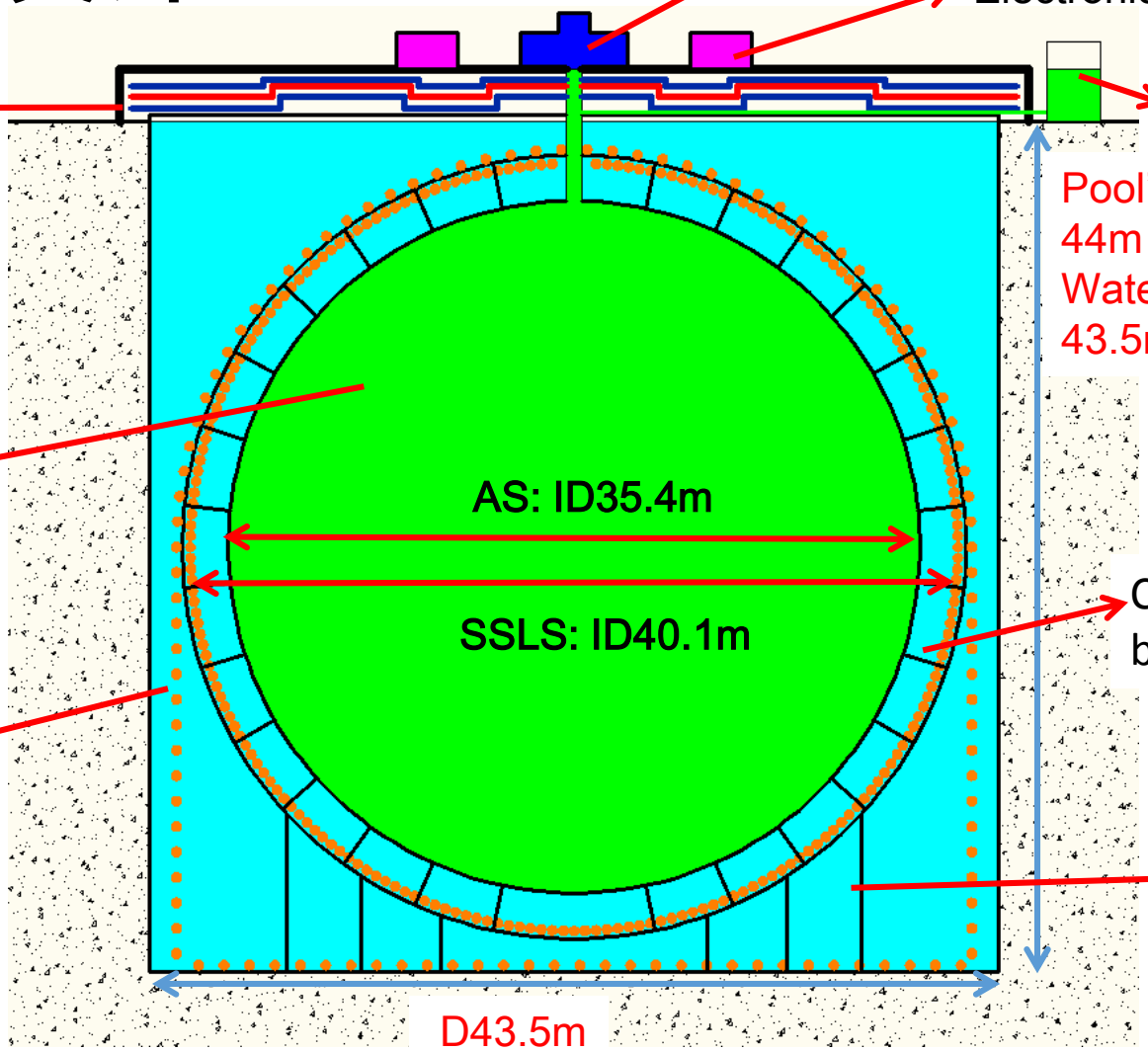
Rich physics possibilities of JUNO

- Precision of three parameters (Δm^2_{21}, Δm^2_{ee} and $\sin^2\theta_{12}$) will reach sub-percent level, several times improvement compared with current precision.
- Probing the unitarity of U_{PMNS} to ~1% level.



Neutrino Physics with JUNO, J. Phys. G 43, 030401 (2016)

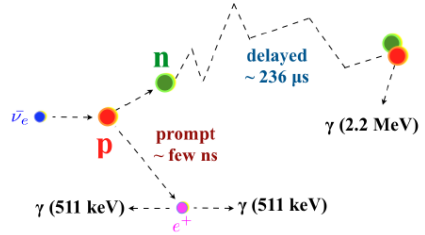
探测器设计



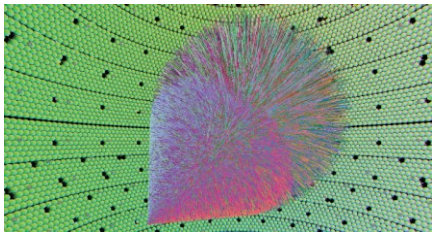
AS: Acrylic sphere; SSLS: stainless steel latticed shell

离线数据处理

- IBD



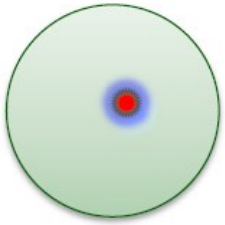
- Muon



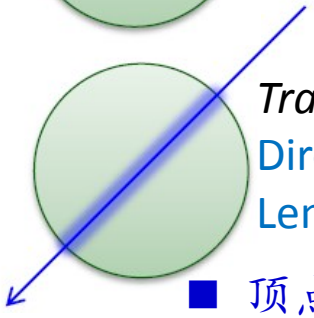
NNVT MCP-PMT | Hamamatsu R12860 | HZC XP72B22

Waveform
ADC/TDC

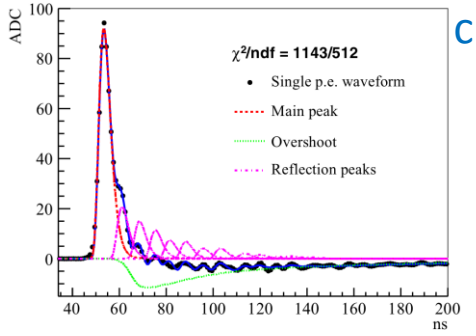
■ 物理产生子 ■ 探测器模拟 ■ 电子学模拟



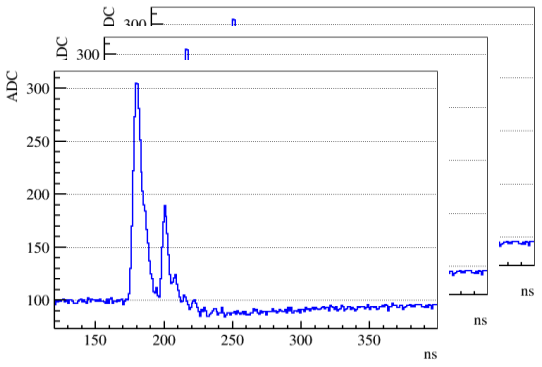
Point-like
Energy/Vertex



Tracking
Direction/Track Length



PMT hits
charge/time



■ 顶点位置/能量/径迹重建 ■ 波形重建/刻度

离线软件的组织结构

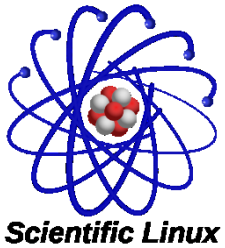
- 核心框架软件SNiPER
- 数据模型管理和输入/输出
- 几何管理
- 事例显示
- 数据库接口和管理
- 物理产生子
- 探测器模拟和电子学模拟
- 事例重建：顶点、能量、径迹
- 刻度
- 物理分析
- 数据产生和质量检查组

软件开发者来自国内外研究机构、大学

- IHEP
- SDU
- SYSU
- NKU
- USTC
- NJU
- SJTU
- INFN-ROMA3
- LLR
- FZ-Jülich
- …

离线软件环境

- 编程语言：C++ and Python
  - C++：算法实现；Python：作业配置
- 操作系统：Linux
  - 官方支持：Scientific Linux 6/GCC 4.4.7
- 软件配置和包管理：CMT
  - 编译软件包，设定环境变量
- 代码管理：SVN/Trac
  - 集中式管理，单一的软件仓库
- 外部库
  - Boost，ROOT，GEANT4，…

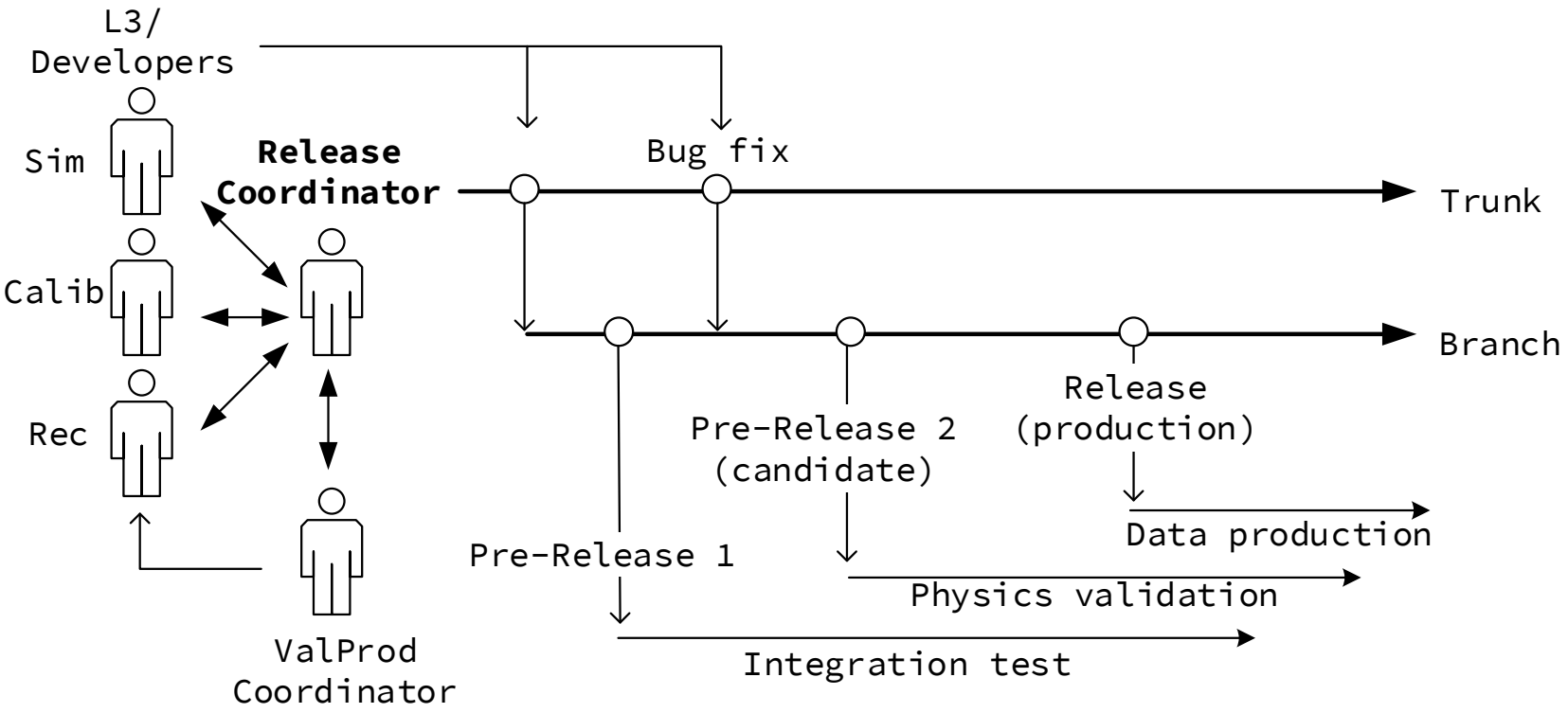



离线软件的发布回顾

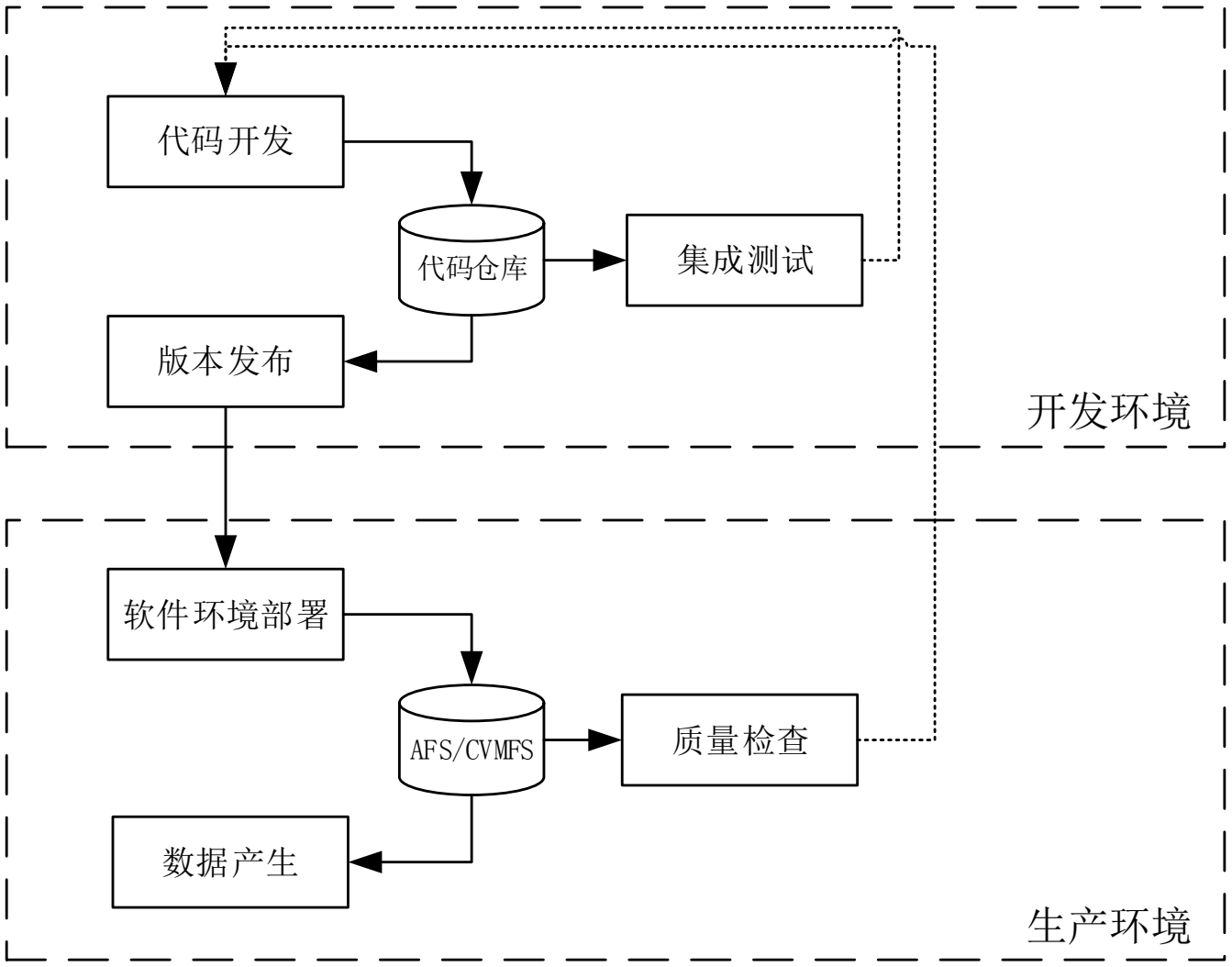
- 2013年开始正式发布，第一个J13v1r1版本包含了数据处理中的重要算法和服务。
- 2014年的J14v1r2，完成大规模的数据产生。
- 2015年的J15版本，全面部署到SL6系统。
- 2016年的J16版本，首次包含完整的数据处理流程，同时加入用于测试和质量检查的JunoTest。数据质量检查组完成大规模数据产生。
- 每年发布约2-3个正式版本。更新主要有软件bug修复，探测器几何和参数的改变，算法改进等。

离线软件的发布模式

- 采用螺旋式的开发模式
  - 正式版本发布前，需要经过多次预发布版本。

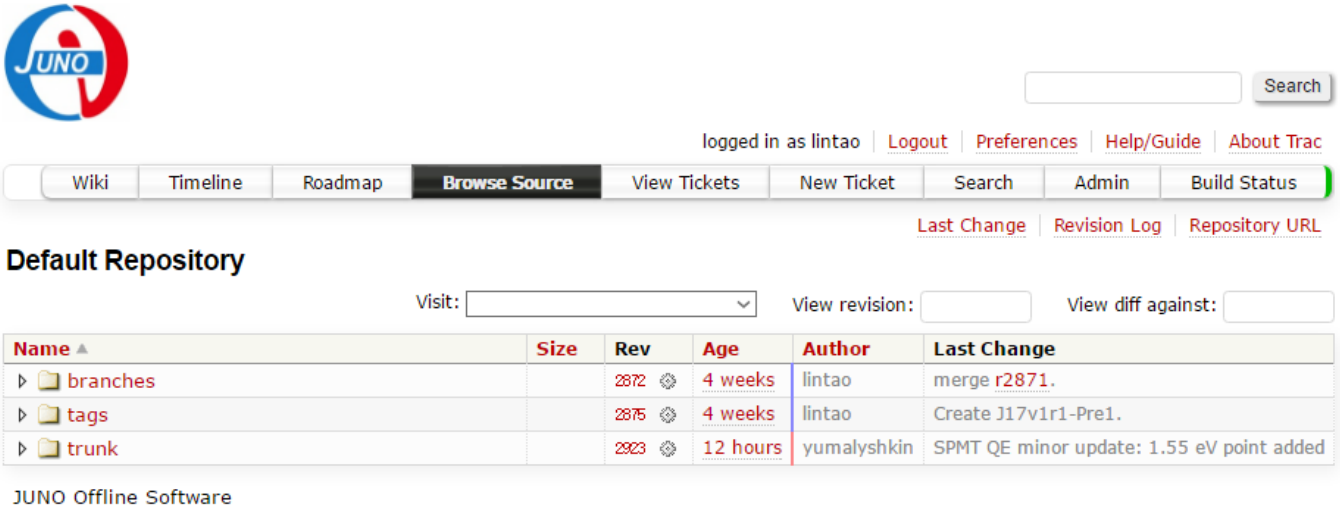


离线软件的发布流程图



软件版本控制

- 采用SVN进行版本控制。
  - 集中式仓库。
  - 允许部分导出。
  - 目录/文件级别的权限控制。
  - 容易定期备份。
- 使用Trac作为管理工具。
  - WebUI，与SVN无缝整合。
  - 支持git，支持多仓库。
  - 浏览源码，追踪bug。
  - 用户和权限管理。
  - 开源，支持插件。



SVN仓库

- Offline软件包全部存储于同一个仓库中。
  - 在AFS上导出需要40s左右的时间。

trunk		2923	12 hours	yumalyshkin	SPMT QE minor update: 1.55 eV point added
Analysis		2775	7 weeks	lintao	update data model path: /Event/GenEvent? -> /Event/Gen?; /Eve
Calibration		2776	7 weeks	lintao	udpate path for elec: /Event/ElecEvent? -> /Event/Elec?.
cmake		2908	5 days	lintao	update CMakeLists.txt: add Geometry dependencies.
cmt		651	3 years	lintao	Add svn:ignore.
CommonSvc		2730	8 weeks	lintao	simplify packages.
Database		2673	2 months	huangwh	fixed some bugs of reading data from any point
DataModel		2825	6 weeks	lintao	compile TimeStamp? using gcc 5.
Detector		2909	5 days	lintao	update CMakeLists.txt: add boost fs.
Doc		2486	6 months	lintao	doc: junoenv.
EventDisplay		2810	7 weeks	lintao	rename RecEvent? to CDRecEvent.
Examples		2911	5 days	yuzy	Update deconvolution parameters
Generator		2896	10 days	xujilei	pdate the rotation angle of experimental hall x axis with real east
installation		2640	2 months	lintao	force install into lib instead of lib64.
JunoRelease		970	3 years	lintao	update the JunoRelease?.
Production		2775	7 weeks	lintao	update data model path: /Event/GenEvent? -> /Event/Gen?; /Eve
Reconstruction		2910	5 days	yuzy	Update the Deconvolution package for J17v1
RootIO		2810	7 weeks	lintao	rename RecEvent? to CDRecEvent.
Simulation		2923	15 hours	yumalyshkin	SPMT QE minor update: 1.55 eV point added
Utilities		2744	8 weeks	lintao	simplify JunoTimer?.
Validation		2885	2 weeks	lintao	add an example ChainAny?.
XmlObjDesc		2811	7 weeks	lintao	update CDRecEvent to support multiple vertexes.
.gitignore	123 bytes	559	3 years	lintao	update .gitignore.
CMakeLists.txt	4.4 KB	2774	8 weeks	lintao	enable Examples again.

关于git的考虑

- 背景
  - GIT是一个分布式的版本管理系统，使用者有完整的仓库镜像。
  - GitHub提供了基于git的软件托管服务，成为开源软件聚集地之一。也有付费的商业版，提供私有仓库。
- GIT和SVN的使用有很多相似之处，但是有几点暂未满足我们的要求。
  - 它的哲理是“一个项目一个仓库”，无法部分导出。
  - 这导致它对大型的仓库支持并不好。需要拆分仓库，同时软件包间的依赖丢失，增加了维护成本。
  - 权限控制问题。

版本控制在多个实验中的思考

- 无论使用GIT或者SVN，版本控制对每个实验的软件都非常重要。
- 如果每个实验都需要自己维护一套代码仓库，不仅重复工作，而且还面临可靠性安全性等问题。
- 一种是将软件托管于商业平台，如GitHub。
  - 以开源形式托管，如CMS实验。
  - 国外已有部分研究机构购买商业版GitHub。
- 另一种是建立私有的统一代码托管平台，提供代码托管的服务。
  - 如使用Trac，GitLab等开源软件进行部署。

软件的编译

- 离线软件基于 CMT[1] 进行编译和配置管理。
  - CMT主要在高能物理领域使用，包括LHCb,BES3,DYB。
- CMT以 包(package)为管理单元。
  - 每个包中编写requirements文件用于指示CMT生成Makefile和setup脚本。可处理软件包的依赖关系。
- CMT支持自定义的编译规则。

```
package RecTimeLikeAlg

use ROOT v* Externals
use Boost v* Externals
use SniperKernel v*
use RecEvent v* DataModel
use CalibEvent v* DataModel
use Identifier v* Detector
```

```
macro_append RecTimeLikeAlg_shlibflags " -lMinuit2 "

macro_append Boost_linkopts " -lboost_filesystem -lboost_system "

library RecTimeLikeAlg *.cc
apply_pattern linker_library library=RecTimeLikeAlg
apply_pattern install_python_modules
```

[1] http://www.cmtsite.net

关于CMT的替代方案考虑

- CMake是一套用于软件编译、测试和打包的软件。
  - 已被ROOT，Geant4等大型软件使用。
  - 支持源码/编译目录分离的方式，允许生成多套配置方式的软件，如优化模式和调试模式。
- 迁移时要考虑的问题
  - 面向的用户是物理学家，要提供简易的使用方式。
  - 能够支持类似CMT的部分软件包编译的方式。
  - 需要为bash/tcsh用户生成环境变量。
- 实现方式：CMake支持自定义的宏/函数
  - 定义相应的宏/函数用于封装编译的细节。

使用CMake进行编译

- 自定义函数，用于隐藏cmake细节，同时有利于将来改进具体的实现。
  - PRJ:用于描述项目
  - PKG:用于具体的软件包
  - EDM和XOD:用于数据模型
    - 数据模型基于xml生成代码
    - 在configure阶段自动生成

```
PKG( RecTimeLikeAlg
    DEPENDS
        RecEvent
        CalibEvent
        Identifier
        Geometry
        EvtNavigator
        DataRegistritionSvc
        BufferMemMgr

        Minuit2
        boost_filesystem boost_system
)
```

- 解决了一部分的问题：
  - 简化了使用方式，比CMT更简洁。
  - 自动生成配置脚本。

对CMake的测试

- 基于trunk版本，分别测试了cmake和cmt

	配置方式	configuire	setup	make
CMT v1r26	默认	0m8.703s	0m0.705s	13m21.914s
CMake 2.8.12.1	默认	0m13.564s	0m0.005s	16m15.296s
	Release	0m27.508s	0m0.006s	16m31.718s

- 目前整体的编译时间相差不大。而且程序运行结果没有差异。
- 关于CMake的下一步工作
  - 优化函数的实现，进一步提升编译时的性能。
  - 研究如何支持部分软件包的导出和编译。

外部库管理：需求分析

- 离线软件依赖于多种外部库，使用固定版本。
  - 外部库的版本会对最终的数据处理结果造成影响。
- 对外部库采用源码编译方式
  - 更加容易控制软件的版本。
  - 便于用户在主流的Linux系统自己安装。
  - 便于对源码打补丁。
- 对编译器的考虑
  - 使用系统自带的编译器。
- 易于修改、添加和移除特定的外部库。

外部库管理：实现

- 采用shell脚本开发，工具集称为junoenv。
  - 更具通用性，Linux系统都可以运行。
  - UI参考了常见的包管理工具，提供常见功能。
  - 源码和offline一起管理，这可确保版本的一致性。
  - 每个外部库都由单独的脚本进行描述，确保扩展性。
  - 可处理依赖关系。
  - 无需管理员权限。

```
$ bash junoenv libs list          # 列出待安装的软件
$ bash junoenv libs all python    # 安装 python
$ bash junoenv libs reuse python # 重复利用已安装的软件
```

子命令 list 的结果

返回信息包括外部库名称，当前版本号，依赖的外部库

```
[x] python@2.7.6
[x] boost@1.55.0 -> python
[x] cmake@2.8.12.1
[x] git@1.8.4.3
[x] gccxml@master -> cmake
[x] xercesc@3.1.1
[x] gsl@1.16
[x] fftw3@3.2.1
[x] cmt@v1r26
[x] clhep@2.1.0.1
[x] ROOT@5.34.11 -> python boost cmake +git gccxml xercesc +qt4 gsl fftw3
[x] hepmc@2.06.09
[x] geant4@9.4.p04 -> python boost cmake xercesc +qt4 clhep ROOT
[x] libmore@0.8.3
[x] libmore-data@20140630 -> libmore
[x] mysql-connector-c@6.1.9 -> cmake
[x] mysql-connector-cpp@1.1.8 -> boost cmake mysql-connector-c
```

软件部署

- 官方的离线软件部署于多个站点
  - IHEP：AFS系统，CVMFS系统
  - INFN CNAF:NFS系统
  - CVMFS允许用户只需安装cvmfs client即可使用。
- 软件目录结构
  - *SITE/OS/Release/Version*
  - 含有外部库、外部库接口、sniper、offline等。
- 用户的使用
  - 对于普通用户，只需source特定脚本，无需其它配置。
  - 对于开发者，需要额外设置自己的工作目录。

JunoTest工具集

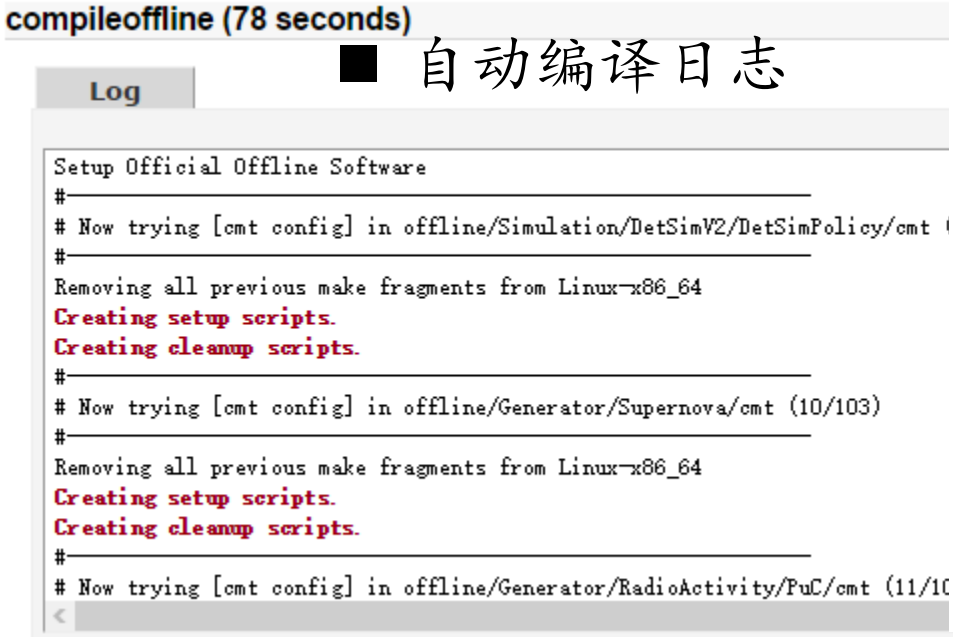
- 部署软件后，需要完成单元测试、集成测试、数据产生和质量检查。
  - 对软件进行全面的测试对于软件质量的控制极其重要。
- 为了复用测试过程中常用的需求和功能，开发了一套基于Python/Bash脚本的通用工具集。
- 基于工具集，建立了高层的应用
  - UnitTest：用于单元测试和集成测试
  - Production：用于数据产生和质量检查
- 对用户提供相同的界面，降低学习难度。

软件的自动编译和集成测试

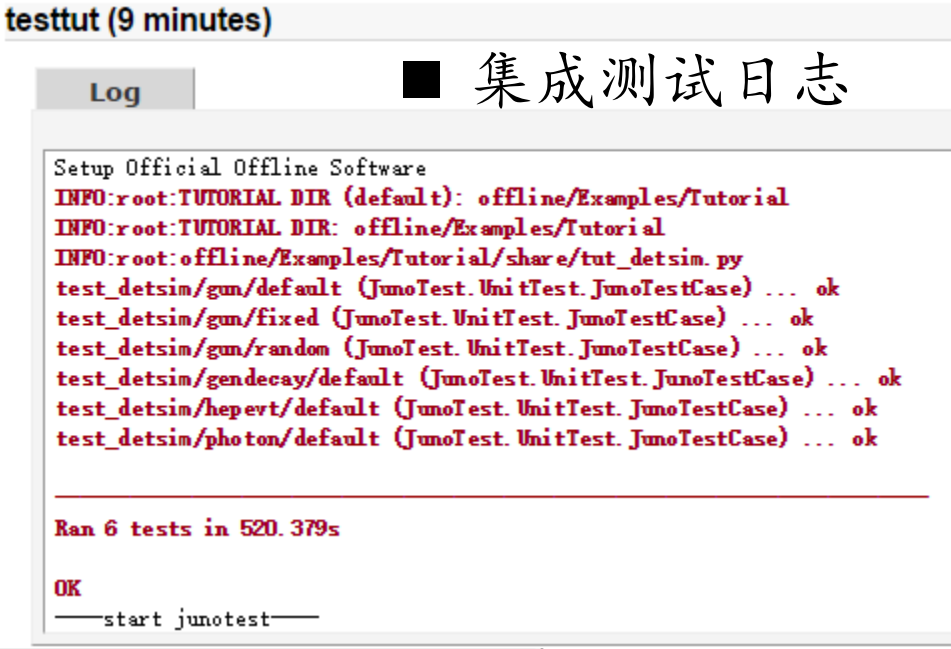
- 为了自动化编译软件并完成集成测试，使用了Trac的插件Bitten。
- 开发者提交完代码后，Bitten server会安排任务。Bitten worker获取任务后开始运行。
- 任务脚本中，需要完成：
  - 软件源码的导出/更新
  - 离线软件的编译
  - 调用JunoTest中的任务
- 支持多种配置方式：worker运行于SL6和SL7
  - 利用IHEP Cloud资源，部署于虚拟机中。

Bitten页面

- 顶层页面显示了软件的版本以及对应的worker信息和运行状态。



- 自动编译日志

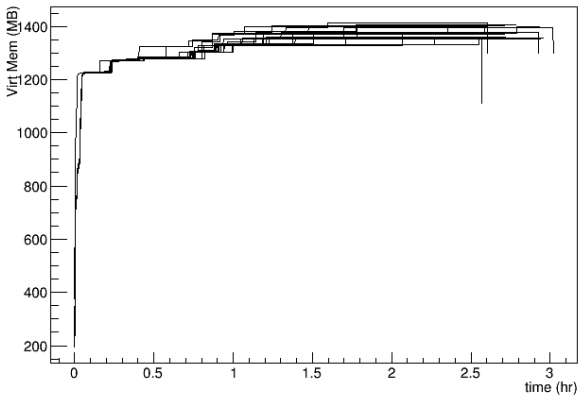


- 集成测试日志

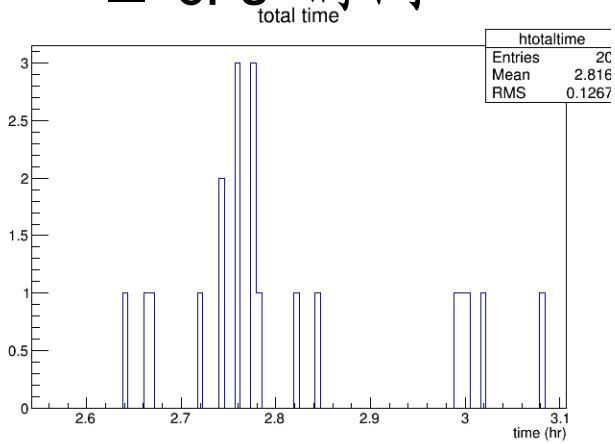
软件的性能测量

- 性能的测量可以提供多种信息，如
  - 软件是否正常，运行的作业节点是否正常
  - 和前一个版本的差异
- 支持两种方式的测量方式
  - 集成于JunoTest UnitTest中，单独测量test case。
  - 集成于作业中，对大规模的作业进行监测。

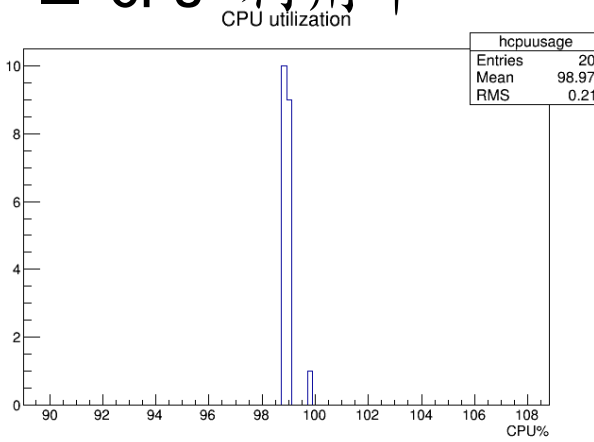
■ 虚拟内存



■ CPU 时间



■ CPU 利用率



数据质量检查和数据产生

- 为了确保软件的物理性能，每个Pre-Release后，都需要完成相应的质量检查。
- 工具“JunoTest Production”可同时用于数据质量检查和产生工作。
  - 自动完成完整的数据处理流程，从模拟、刻度到重建。
  - 自动运行分析脚本，并完成与其它软件版本的比较。
  - 使用.ini配置文件，为用户提供更高层的使用方式。
  - 同时支持IHEP和CNAF两个站点。
  - 通过编写基于shell的driver，可快速扩展。
- 顺利完成2016年的大规模数据产生工作。

Production的配置文件

```
[ChainDecay-detonly]
driver = ChainDecay
seed = 50000
workflow = detsim
evtmax = 5000
njobs = 100
tags = U-238 Th-232 K-40
workDir = BKG-LS-500k-detsim

anaWorkflow = detsim_ana
detsim_ana = @ana-detsim-merge
```

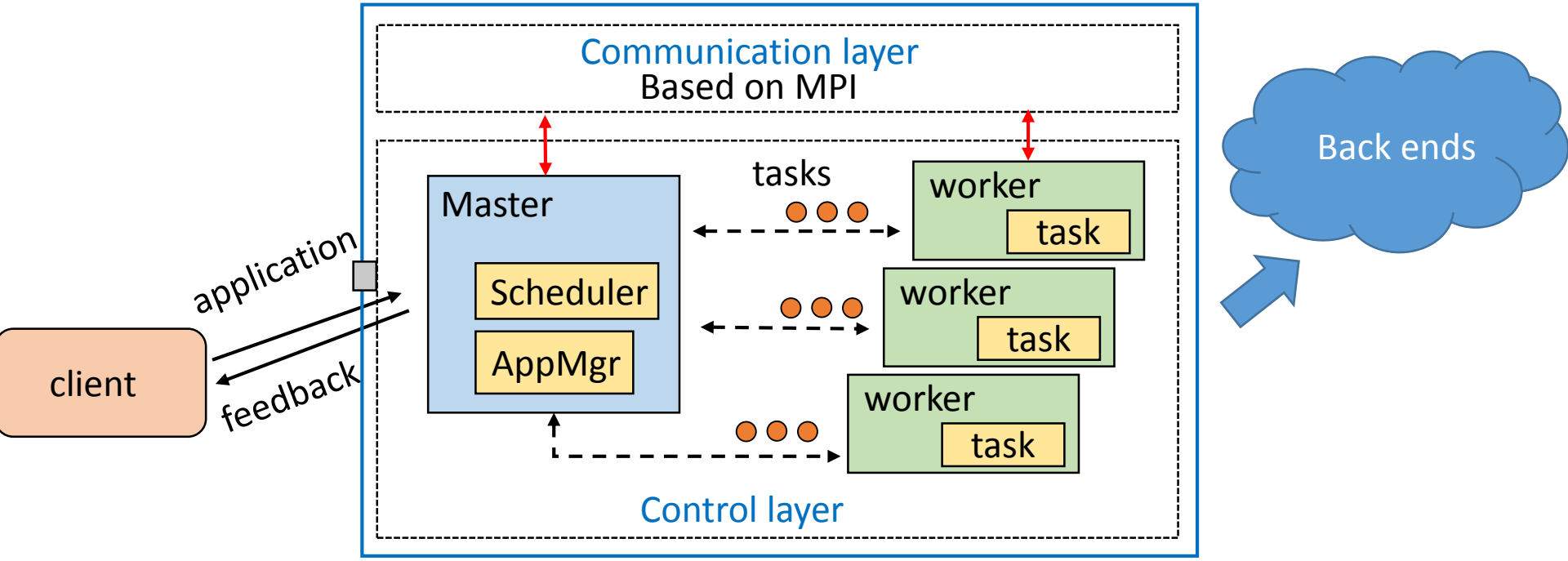
- 用户自定义的section名称
- 底层使用的driver
- 可定制数据处理的流程
- Tag可以作为driver的输入参数。不同tag产生对应的样本。
- 可定制数据分析的流程
- 可定制每个阶段运行的脚本

```
[@ana-detsim-merge]
script = $ROOTIOTOOLSROOT/share/merge.py
cmd = python %(script)s merged.root detsim/detsim-*.root
```

- 可复用，用户只需引用 @ana-detsim-merge 即可。

分布式工具：DistJET

- DistJET: Distributed Juno Execution Toolkit
- 基于MPI实现，并行化单元测试和质量检查。



总结与展望

- 江门离线软件采用螺旋式的开发模式。
  - 软件经过单元测试、集成测试后发布预发布版本。
  - 通过数据质量检查确保预发布版本的正确性，最终发布正式版本用于数据产生。
- 借助开源软件和自主研发的工具，完成了软件发布过程中的规范化和自动化，节约大量人力。
- 展望
  - 江门实验2020年即将取数，软件质量控制极其重要。
  - 逐步完善软件与光学参数、刻度参数的管理。

谢谢！