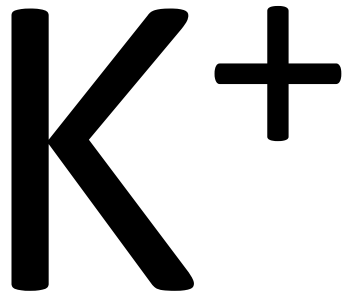
A New Analysis Framework at RIBF

J. Chiba (TUS)

Please keep in mind the word



ANAPaw is NOT a framework

therefore

Need an Analysis Framework!!

Borrow VGs from Prof. R. Itoh (partly written in Japanese)

Event Processing Framework for High Energy Experiment

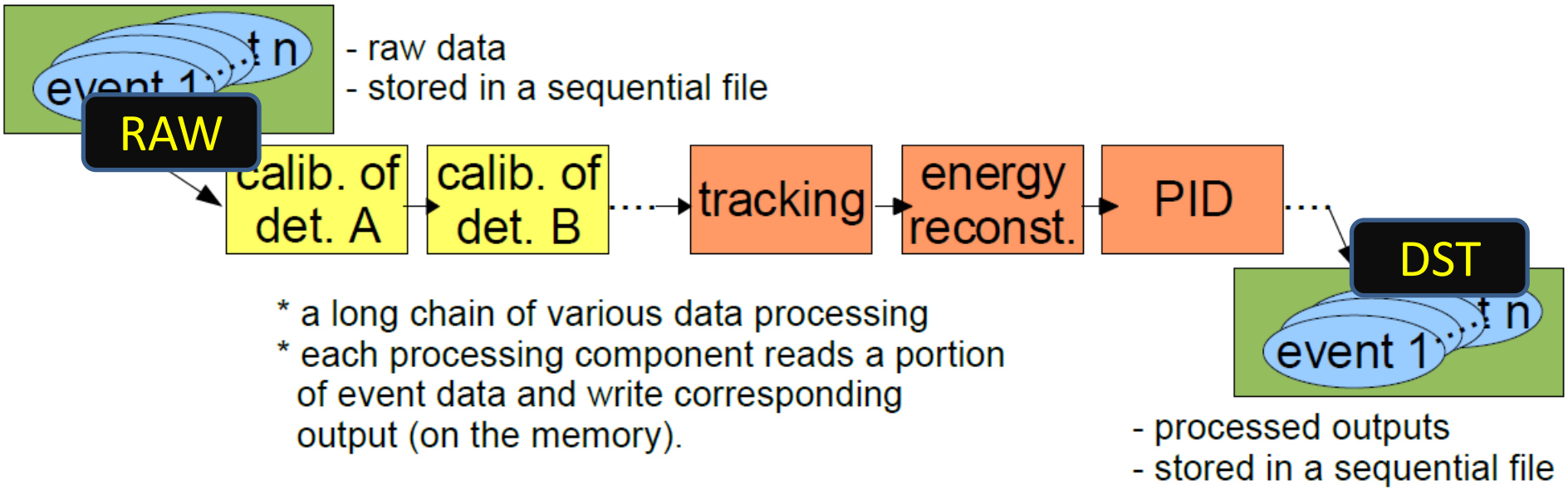
Ryosuke Itoh, KEK
from the Belle Collaboration

Software Workshop, Tohoku U. 10/17/2007

1. Analysis Framework とは？

高エネルギー実験におけるデータ処理

- Data: “event” が集まったもの
  * 1 つの event は header 、検出器の raw data 、software 処理の結果得られた tracking, vertexing の情報、物理解析に用いる 4-momentum や PID のリストが入っている。
  * “event” はそれぞれ独立であり、通常相関がない。
    -> 解析は event by event で行われる。



Analysis Framework の idea

＊ ほとんどの解析 software ではイベントデータを読んだり書いたりする部分は共通である。
＊ Histogram や N-tuple を作るための tool も共通化できる。

＊ 違うのは解析ソフトの本体のみ。
＊ 解析ソフト自身もかなりの部分は共通している。

- 実験のデータ解析で共通の”Analysis Framework”をつくり、ユーザーごとの解析はフレームワークにプラグインする。

- Event Data にアクセスするための共通の方法を用意する。

Analysis Framework と ROOT(or PAW)

- Event data を扱う Framework という意味では同じ

- Analysis Framework は
  * DST production など大量の event をなめる解析
  * 解析に必要な event の subsample(N-tuple を含む)を作る。
  * selection のかかっていない event sample の histogram を作る。

  などに適している。

- ROOT/PAW は
  * N-tuple や selected events を複数回なめて、interactive に histogram などの結果を目で見ながら解析を行う。

  用途に適している。

Analysis Framework を構成する要素

1. ユーザーの解析コードの実行を制御する framework の core
2. 解析コードから event data へ access するための tool
3. 解析コードで使用する histogram/N-tuple を管理する tool
4. event data や histogram/N-tuple と外部記憶との間の I./O を管理するメカニズム
5. ユーザーインターフェイス

もっとも simple な Framework

```
int main (int argc, char** argv )
{
  system_init ();
  user_init();

  file_dscr* fd = open_file ( argv[1] );

  while ( !end_of_file ( fd ) ) {
    Event* evt = read_event ( fd );
    user_event ( evt );
  }

  close_file ( fd );
  user_end();
  system_end();
}
```

＊ ユーザーは太字の部分の function のみあたえる。
＊ 他の部分が Framework の core である。
＊ しかしユーザーのあたえる function はさらに機能別に細分化できるはずである。

↓

細かな機能ブロック（module）を集めて、それを順に実行するような framework に拡張できる。

```
struct Event {
  double px;
  double py;
  double pz;
  double e;
  double mass;
} evt;
```

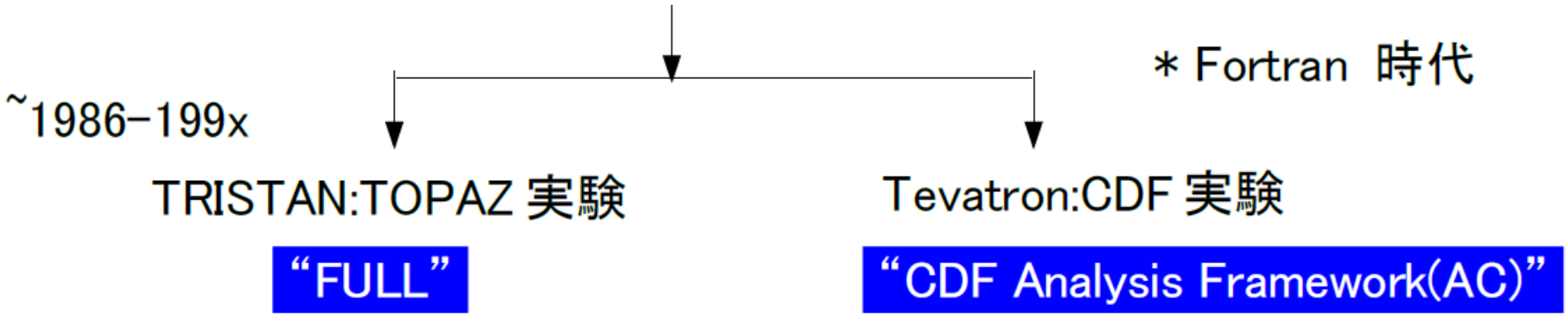
Event 内の data 構造は C の structure や C++ の Class で扱うと簡単
（Fortran のころは array 以外の data 構造が存在しなかったので、複雑な data management package が必要だった。）

Software Bus 構造のフレームワークの変遷

- 最初のフレームワーク

PEP4/TPC 実験における”FULL:”および”DSTANL” （1982-）

＊Bernard Gabioud という人が作った
＊Fortran on VAX/VMS
＊ 一本道の path（途中での実行停止はできた。）
＊Block 単位の原始的なデータマネージメント
＊ 優れた command line interface を持っていた。
（COM/CLI：menu-driven のユーザーインターフェイス）



– 第2世代のフレームワーク

TOPAZ: FULL

- basically similar to PEP4's FULL/DSTANL
- Fortran based. Ported to:
  - * FACOM M380-780 (main frame)
  - * VAX/VMS
  - * UNIX (HP/UX, SunOS4, etc.)
- Data management : TBS (Tristan Bank System)
- Histogram handling : HANDYPAK
- User interface : COM/CLI on main frame
- Modified version was used for the early simulation study of Belle (pre-Belle era.)

CDF: Analysis Framework (AC)

- don't know much about this
- Fortran based
  - * VAX/VMS
  - * RISC UNIX (SGI Challenge)
- Data management : ZEBRA?
- Histogram handling : HBOOK4
- manage multiple "paths" and conditional branch
- User interface : COM/CLI?

- 第3世代の Framework

TOPAZ:FULL

↓

Belle 実験の
B.A.S.F.
(Belle AnalysiS Framework)

CDF: Analysis Framework(AC)

↓

CDF/BaBar 実験の
共通 Analysis Framework
(AC++)

第3世代フレームワークの特徴

- C++ に対応。C++ で coding されている。
- 複雑な path 構造 (条件分岐など) をサポートしている
- 進歩した Data Management と I/O

- 第4世代のフレームワーク

GAUDI/Athena : LHC 実験 (LHCb, ATLAS) のための framework

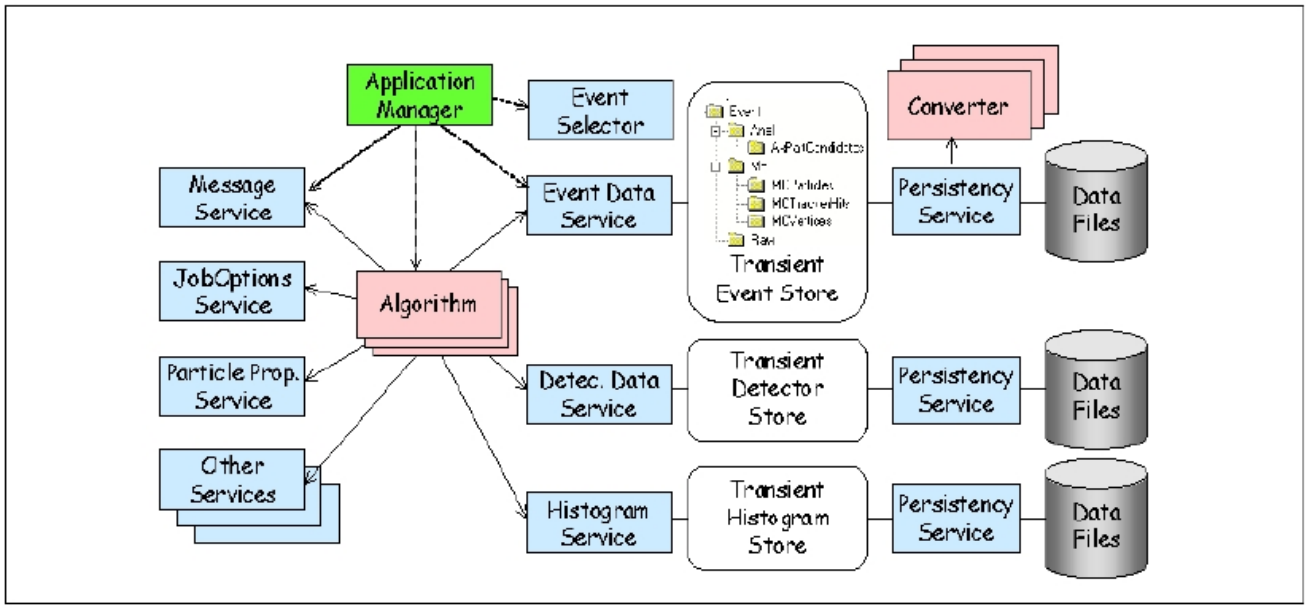


Figure 2.1 Gaudi Architecture Object Diagram

i.e. CMS: ORCA

- “module” は algoriothm と呼ばれる。Sequencer により software pipeline として実行されるが、階層構造などをもたせることもできる。
- Algorithm と Data が明確に分離されている。
- Event I/O は基本的には ROOT I/O (POOL(=ROOT I/O+file catalog) を通している)
- Histogram/N-tuple は AIDA とよばれる I/F を通して ROOT/HBOOK 使用。

RIBF at Present
(when I want to do some analysis)

- Pass Raw (RIDF) data to Users
  - Anapaw (no, I don't want to use)
  - TArt Class Library ~~(anaroot)~~ … I'm using now!
  - Each group makes own program?
- Maybe, lack of something
  - Communication?
  - Documentation?
  - Efforts to integrate software?

Near Future?

- Surely need a Framework … how to make it?
- 1-2 people should become familiar with the Root Class Library!
  - Go4 at GSI (http://www-win.gsi.de/go4/) good example!
- It is easy to use the Classes, (though it is hard to make a new Class with a good design).
- Users should start to use the root ASAP
  - It is very easy to use root if you know how to use Paw.
  - h2root … convert hbk files to root files

ROOT

see http://root.cern.ch

- CERN で開発された interactive な解析システム
  (by R.Brun, who is the author of PAW.)
  * C++ で簡単に event data に interactive にアクセスできる。
  (C++ interpreter 内蔵)
  * Histogram/N-tuple が C++ で非常に簡単につくれる。
  * Histogram/N-tuple を直接操作し、graphical に表示する強力な tool 群

- C++ の Object を serialize して(streaming)、File に sequential にI/O できる。(ROOT I/O)
  <- OODB に対するアンチテーゼ。Objectivety (fully random access) の HEP での利用の失敗をに対する solution 。

- Widely used in HEP community but not only in HEP!
  * Partially at Belle
  * Standard in LHC experiments, BaBar, Tevatron
  * AstroPhysics (GLAST, γ-ray satelite experiments....)