修士論文

ATLAS実験におけるFTKシステムの飛跡パターン生成高速化と処理能力の評価

早稲田大学　先進理工学研究科
物理学及応用物理学専攻
寄田研究室

大矢　章晴

2013年2月7日

概 要

物質を構成している最小の粒子とは何か。宇宙初期状態のエネルギーでは一体どのような現象が起こっていたのか。高エネルギー素粒子物理学ではその答えを見つけるために加速器を用いて、人工的に高エネルギーを作りだし、検出器を用いて粒子のふるまいを検出し、解析を行っている。

LHC(Large Hadron Collider) はスイスジュネーブ郊外にある欧州素粒子原子核研究機構 (CERN) に建設された陽子・陽子衝突型加速器である。現在世界最高の 8TeV の重心系エネルギーまで到達し、順調に稼働している。ATLAS 実験はその LHC に設置された汎用型粒子検出器の一つである。

LHC では 1 秒当たりの瞬間輝度が最大で $7.73 \times 10^{33}[cm^{-2}s^{-1}]$ に到達し 1 回の事象における衝突回数（pile up 効果）が最大で 35 個にまで到達している。この Pileup 効果が高くなるにつれ、主な背景事象である QCD 事象による飛跡が多くなる。一方で記憶できるデータ量には時間的制約があるため、事象選択（トリガー）がより難しくなってきている。そこで、我々は高速に飛跡を演算処理するシステムである FTK システムの開発を行っている。FTK システムは内部飛跡検出器の Hit 情報を受け取り、その情報からあらかじめ用意しておいた飛跡のパターンと照らし合わせ、そのパターン内にある Hit 座標の値を使って、線形一次近似よりトラックのパラメータを求めるシステムである。

この飛跡パターンとパターンごとに用意されている線形一次近似式の定数項はあらかじめ用意しておく必要がある。これらによって飛跡再構成率と分解能が決まるため、ＦＴＫシステムにおいて重要な要素となっている。それにも関わらず現状では、パターンの生成に 2 週間ほど時間がかかってしまっている。このため実データのようにビームスポットが変動し、飛跡パターンが変化してしまうデータを調べることは困難であった。

本研究ではパターンの生成方法の改善と実際のデータを用いた FTK システムの評価を行った。パターン生成の段階で並列処理を行う段階を増やし、グリッドコンピューティングを最大限利用することによって、生成時間が 2 日間ほどに短縮することができた。Minimum Bias の実データをインプットとして FTK シミュレートした結果、オフライントラックに対して約 40% もの飛跡再構成ができないという結果となった。考えられる原因としては内部飛跡検出器の不感領域が存在すること。また飛跡と飛跡の距離が近く、数が多いのでオフライントラックとの比較が正確にできていないという可能性も考えられる。今後はワイルドカードアルゴリズムの導入やミューオンサンプルを使って評価していく必要がある。

目 次

図 目 次

表 目 次

1 序論

現在の物理学では基本的な力として 4 種類の力、強い力、弱い力、電磁気力、重力が考えられている。素粒子物理学ではこの 4 つの力の内、重力を除いた 3 つの力を取り扱っている。この素粒子物理学の中で多くの実験結果と一致しているのが標準理論である。この標準理論の中で唯一未発見の粒子であったヒッグス粒子らしき粒子が CERN にて観測された。

しかしながら観測されたチャンネルはヒッグス粒子とボソンの結合チャンネルのみであり、標準理論のヒッグス粒子であると断定するためには、フェルミオンとの結合の強さ（湯川結合定数）を測定しなければならず、詳細研究が必要となっている。

今後 CERN にある LHC では質量が重く生成断面積が非常に小さい事象を得るためルミノシティの増強を行っていく。瞬間ルミノシティを 2015 年では $1 \times 10^{34}[cm^{-2}s^{-1}]$,2021 年では $5 \times 10^{34}[cm^{-2}s^{-1}]$ に増強する予定である。この増強に伴い衝突回数も増加していき、飛跡の本数も増加していく。

一方でデータ取得量には制限があるため瞬時に記憶すべき事象なのか選択（トリガー選択）を行わなくてはならない。

本研究は LHC のビームラインに置かれた ATLAS 検出器のトリガーアップグレードを行う装置である FTK システムの開発に関する研究である。FTK システムは内部飛跡検出器のうちの Pixel と SCT のヒット情報から高速で飛跡を演算を行う装置であり、Lv1 トリガーと Lv2 トリガーの間に挿入する予定である。まず第 2 章では LHC 加速器と ATLAS 検出器、また ATLAS 実験グループで使われているソフトウエアについても説明していく。

次に第 3 章では FTK システム概要、第 4 章では本研究内容であるパターン生成高速化の手法について説明していく。

第 5 章では FTK グループ内で今まで扱っていない実データのヒット情報を FTK シミュレーションのインプットとして用いて、FTK システムの評価を行った結果を説明する。

2 LHC-ATLAS 実験

2.1 LHC(Large Hadron Colider)

図 2.1: 上空から見た LHC

LHC(Large Hadron Collider) は、CERN(欧州原子核研究機構) にある陽子陽子衝突型加速器である。2012 年 4 月から重心系エネルギー世界最大となる 8TeV で運行している。

LHC は順調に稼働しており、2012 年 8TeV の run では最大瞬間ルミノシティが $L = 7.73 \times 10^{-33}[cm^{-2}s^{-1}]$ に到達し、pile up 数（１回の事象における衝突回数）も 35 まで増えている。また積分ルミノシティも順調に増加しており 8TeV run 約 8ヶ月で 23.3(fb^{-1}) のデータを取得することができた。これに対して ATLAS 検出器がデータ取得した量は 21.7(fb^{-1}) であり、93.1% と安定した動作で運転できているといえる。

2012 年 8TeV の run における瞬間ルミノシティと積分ルミノシティの時間変化をそれぞれ図 2.2 と図 2.3 に示す。また表 2.1 に LHC のデザインを纏めた。

表 2.1: LHC 加速器の主なデザインパラメータ

重心系エネルギー	7TeV+7TeV
バンチ数	2808
バンチ当たりの陽子数	1.15×10^{11}
バンチの長さ	75.5mm
衝突点のビーム半径	16.7μm
衝突頻度	40MHz
瞬間ルミノシティ	$1.0 \times 10^{-34}[cm^{-2}s^{-1}]$

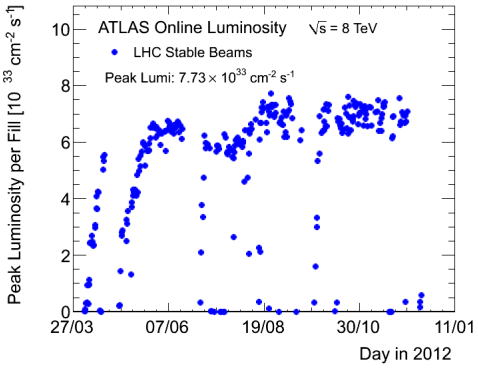

図 2.2: 瞬間最大ルミノシティの時間変化

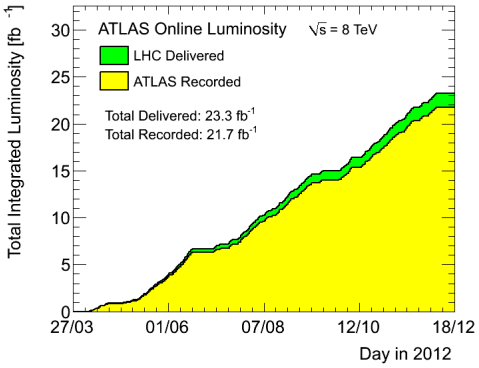

図 2.3: 積分ルミノシティの時間変化

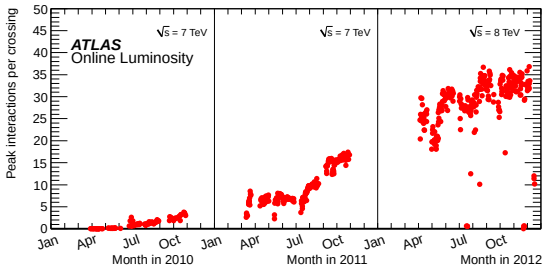

図 2.4: 最大 pile up 数

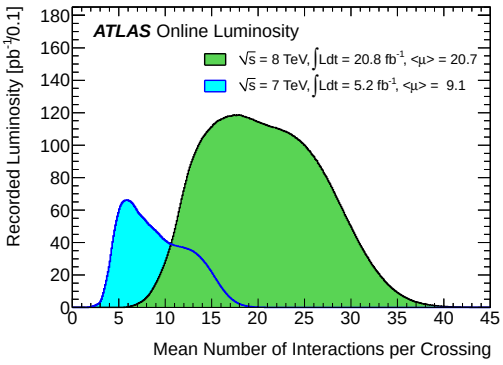

図 2.5: 平均 pile up 数

2.2 ATLAS 実験

ATLAS 実験は LHC のビームライン上に置かれた ATLAS 検出器を使った実験である。この章では ATLAS 実験で使われている検出器とソウトウエアの使い方を述べる。

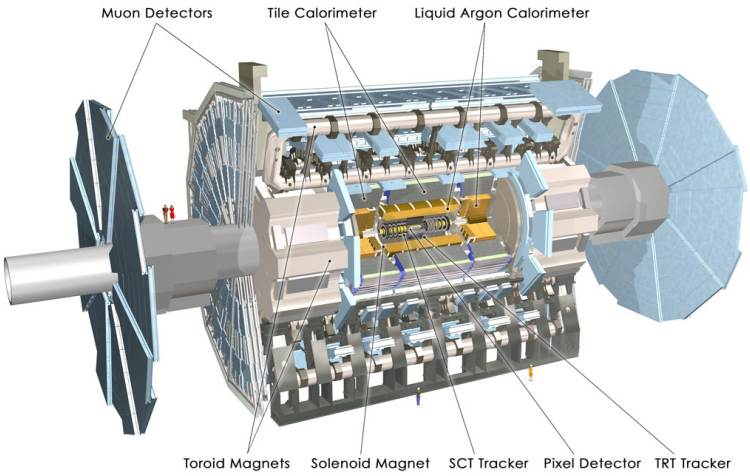

図 2.6: Atlas 検出器全体図

図 4.1 は ATLAS 検出器の全体像である。検出器では粒子の位置情報、運動量、エネルギー、電荷などを測定し、その粒子の種類を同定することが目的である。ATLAS 検出器は内側から順に内部飛跡検出器、電磁カロリーメータ、ハドロンカロリーメータ、ミューオン検出器で構成されている。それぞれについて詳細を記していく。

2.2.1 内部飛跡検出器

図 4.2 に内部飛跡検出器全体図を示す。また、図 4.3 にビーム軸方向から見た時の内部飛跡検出器である。ビーム軸からの角度は、高エネルギー衝突反応では z 軸方向にローレンツブーストされるので、放出角 θ の代わりにローレンツ不変な量である次の値を使う。

$$\eta = -In\tan\frac{\theta}{2} \tag{2.2.1}$$

また内部飛跡検出器には B=2T の磁場をつくる超伝導ソレノイドの内部に置かれている。この磁場を使い荷電粒子を曲げたときの曲率半径 R から横軸運動量を求めることができる。その関係式は (4.1.2) のように表すことが出来る。

$$p_T = 0.3BR \tag{2.2.2}$$

内部飛跡検出器は図にあるように内側から Pixel、SCT(Semi Conductor Tracker)、TRT(Transition Radiation Tracker) の 3 つで構成されている。また検出器が覆っている角度は $|\eta| < 2.5$ である。

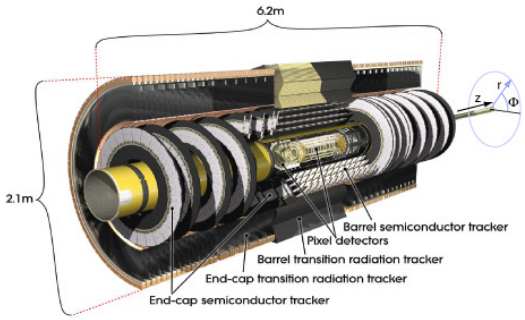

図 2.7: 内部飛跡検出器の全体像

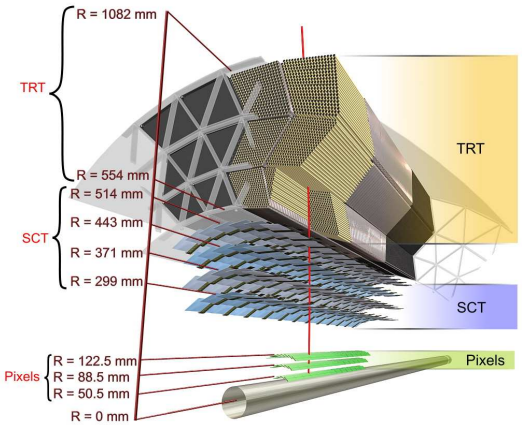

図 2.8: ビーム軸方向から見た内部飛跡検出器

- Pixel
  Pixel 検出器はビームパイプから一番近い距離にある半導体検出器である。一つ一つの Pixel の大きさは $50\times400[\mu\mathrm{m}]$ で、 2 次元読み出しを行うことができる。

- SCT
  SCT は Pixel の外側にある半導体検出器である。SCT のモジュールは 2 枚のシリコンウエハを互いに 40mrad 傾けた構造をしている。またモジュールは全部で 8176 個あり一つのモジュールには 768 個のシリコンストリップがついている。これらのストリップから荷電粒子の通過位置を求めている。

- TRT
  この検出器では荷電粒子が通過する際の遷移放射で放出される X 線から荷電粒子の通過位置を読み取る。軽い粒子ほど遷移放射を出す。また 73 層のストローチューブにより連続的な飛跡を検出を行うことができる。

2.2.2 カロリーメータ

カロリメータは内側から電磁カロリメータとハドロンカロリメータの 2 つから成っている。電磁カロリメータは放射長の短い鉛の吸収体と放射線耐性に優れた液体アルゴンからなる。主に電子と光子の同定とエネルギー測定に用いられる。

またハドロンカロリメータではバレル部が放射長の短い鉄の吸収体とシンチレータ、エンドキャップ部は銅の吸収体と液体アルゴン、放射線強度が高いフォワード部は銅とタングステンの吸収体と液体アルゴンで構成されている。主にハドロンの同定、エネルギー測定、さらにジェットの再構成などを行っている。

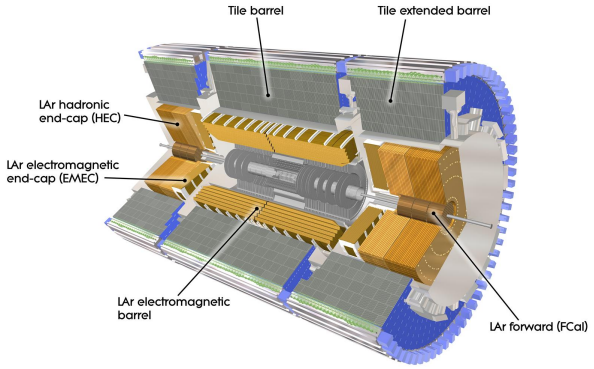

図 2.9: カロリメータの全体図

2.2.3 ミューオン検出器

ミューオンは質量が電子の約 200 倍あるため制動放射を起こしづらいためほとんど電磁カロリメータにエネルギーを落とさない。またレプトンであるため強い相互作用もせず、物質とはほとんど相互作用しない。そのためハドロンカロリメータにもエネルギーをほとんど落とさない。このためミューオンは透過力が高く、一番外側の検出器でほとんど同定することができる。またカロリメータの外側にトロイド磁石が設置されており、この磁石が磁場を作りその曲率半径からミューオンの横運動量を測定することができる。

2.2.4 ATLAS トリガーシステム

トリガーの目的はハードウエア上に検出器情報が記憶できる制限時間内に、事象内の粒子のパラメータをできる限り正しく計算して、それぞれの物理目的に応じた信号事象をより多く取得することにある。

例えば、図 2.10 のように LHC では生成断面積が数十 mb^{-1} 程度である QCD 事象がほとんどであり、信号事象の一つである Higgs 粒子は数十 pb^{-1} で 10^9 倍以上もの違いがある中でトリガーしなくてはならない。そのため、QCD 事象を落とすためにレプトンのある事象を取得するトリガーや飛跡に特徴のある τ 粒子や b クオークなどを同定するトリガー、高いエネルギーを持つ粒子のある事象を取得するトリガーなど様々な工夫されたトリガーがある。

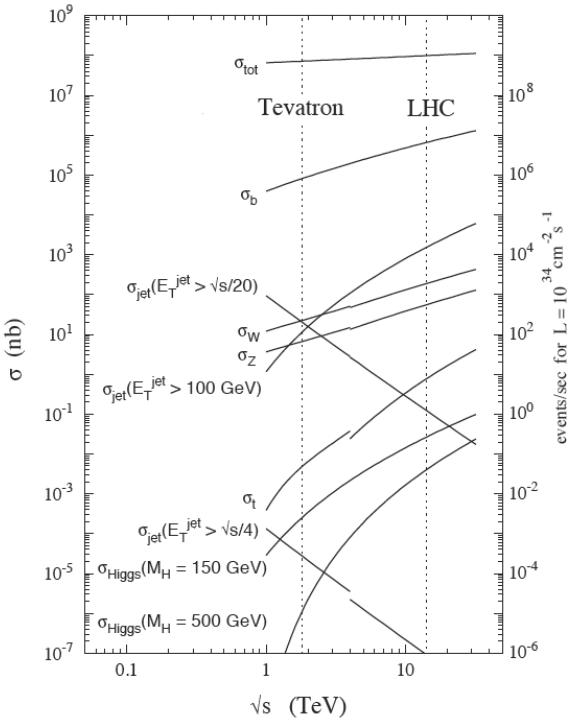

図 2.10: 重心系エネルギーと各事象の生成断面積との関係

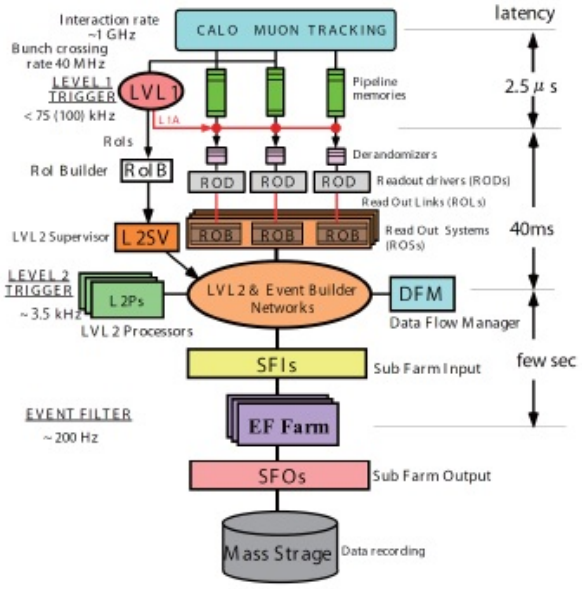

図 2.11: ATLAS トリガーシステム

次に ATLAS トリガーシステムについて説明する。LHC では FTK システム導入時にはビーム衝突を 40MHz の頻度で行い、１衝突当たり平均２３事象起こるため、事象頻度は約 1GHz にもなる。また 1 事象当たりのデータ量は 1.5Mbyte と見積もられているため、1 秒当たりのデータ量は 1.5Pbyte となり非常に大きくなっている。

ATLAS トリガーシステムは 3 段階に分かれており、このデータ量を最終的には 200Hz 程度まで絞り込む。

初めに Lv1 トリガーについて説明する。まず ATLAS 検出器からの全情報は図 p26 のパイプラインメモリーと呼ばれる一時記憶装置に記憶される。パイプラインメモリーは 128 段構造になっていて 25ns×128=3.2μ の間、1 事象の全情報を記憶することができる。この間にカロリーメータとミューオン検出器の全領域の情報を用いて、Lv1 トリガーでエネルギーや運動量を粗く計算する。Lv1 トリガーは約 2.5μs で事象取得するか判断しなくてはならないため、より CPU のクロックと近いアルゴリズムで処理しなくてはならないためハードウエア上の電気回路となっている。

Lv1 トリガーで取得すると判断された事象の L1 Accept 信号が ATLAS の各検出器の読み出し制御部に行き、ROD(Read Out Driver) にその事象の情報がいく。一方その事象の中で粒子が通過した部分を興味ある領域、ROI(Region Of Interest) として ROI Builder で定義を行い、Lv2 トリガーに送られる。この段階で 75kHz 100kHz まで事象レートを絞る。

次に Lv2 トリガーの説明をする。Lv1 Accept された事象のカロリーメータとミューオン検出器の情報、また内部飛跡検出器の情報をコンピュータを使って計算し判断す

る。Lv2 トリガーは約 40ms で判断を下す必要があり、全領域の計算をするには時間が短いため Lv1 トリガーで定義した ROI の範囲を計算し判断を行う。この段階で 3.5kHz まで絞る。

最終的には Event Filter によって 200Hz 程度まで絞り込んでストレージに保存する。

2.3 ATLAS ソフトウエア

ATLAS 実験の共通のフレームワークとして使われている Athena の説明と各パッケージの構造、グリッドコンピューティングの概要と使用方法について述べていく

2.3.1 Athena

Athena は ATLAS 実験において、全てのユーザーがパッケージを共通した方法で使用できるようにするために作られたフレームワークである。また様々なグループがパッケージ開発を行っているため、パッケージ間に依存関係が生じる。そのために徹底したバージョン管理が必要となってくる。Athena はこのバージョン管理も担っている。 この Athena 環境セットアップは例えば以下のようにする（シェル環境の場合)。

```
export AtlasSetup=/afs/cern.ch/atlas/software/dist/AtlasSetup
alias asetup='source $AtlasSetup/scripts/asetup.sh'
asetup TrigMC,17.2.2.6.2 --testarea=/afs/cern.ch/work/t/tohya/FTK_SIM/17.2.2.6.2
```

asetup の後にある 17.2.2.6.2 が使用する Athena のバージョンであり、testarea はこのバージョンの環境で使用するディレクトリ（テストエリア）である。このバージョンで使用するパッケージは全てこのテストエリアの下に置いて使用する。また論文で使用した Athena のバージョンは 17.2.2.6.2 ある。

2.3.2 パッケージ

ATLAS ソフトウエアにおける一般的なパッケージは以下のようなディレクトリで構成されている。

`cmt`:パッケージと `Athena` の環境を結びつけるための要素が入っている
`src`:`Athena` で使用する `C++`のコードが入っている
`Package`:パッケージと同じ名前のディレクトリでヘッダーファイルが入っている
`share`:パッケージの関数を呼びアルゴリズムを走らせるためのジョブオプションファイルが入っている

パッケージを Athena の他のパッケージやオプションと関連づけさせるために必要なことが cmt ディレクトリの中にある requirments に書いてある。この cmt ディレクトリ内で順に cmt config,source setup.sh,gmake とすれば src と Package にあるファイルをコンパイルすることができる。

cmt config,source setup.sh はパッケージのディレクトリを変更したとき、または requirments を変更した時だけ行えばよい。verstion.cmt にはそのパッケージのタグ番号あるいは Athena の revsion 番号が書いてある。

次に Athena で使われている基本的なコードの形について説明する。Athena 上でイベント毎に処理を行わせる C++コードはアルゴリズムと呼ばれ名前に必ず Alg が付くようになっている。このコードは initialize()（一番最初に実行)、execute()（各イベント毎に実行)、finalize()（最後のイベント後に実行）という構成になっている。またこれに excute がないコードをツールと呼び、名前には必ず Tool が付けるようにしている。Athena で解析を行う時はこの二つのコードの関数を呼び、アルゴリズムをイベント走らせ、Tool は関数を呼びデータを読み込む時に使っている。

上記で挙げた以外にも standalone など Athena のフレームワークとは関係なくコンパイルして使うコードの入っているディレクトリがあるパッケージもある。

2.3.3 データ形式

次に ATLAS 実験で使われているデータの形式とその変換方法について説明していく。

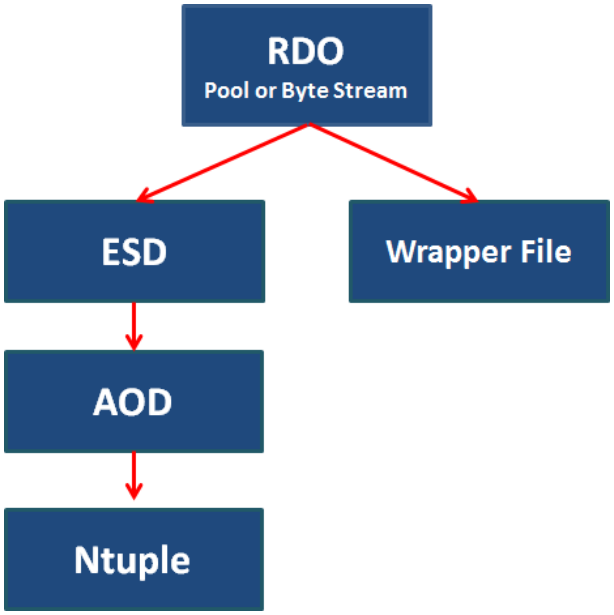

図 2.12: ATLAS データ

図 2.12 の青い四角がデータ形式を表している。それぞれのデータ形式について、特

に内部飛跡検出器や飛跡に関連した項目について説明を行っていく。

- ROD(Raw Data Object)
  実データの場合は Byte Stream 形式、（MC）モンテカルロシミュレーションのデータの場合は Pool 形式で保存されている。RDO には、内部検出器の生のヒット情報が入っており、検出器の詳細やトリガーなど研究などで使用される。
  次章で説明する FTK システムもインプットとして生のヒット情報を用いるので、この RDO を使用する必要がある。

- ESD(Event Summary Data)
  ROD の生のヒット情報から、クラスタリングを行った結果を保存している。また全検出器の情報を使った飛跡の再構成も行っており、それぞれの飛跡がどの物理オブジェクトに入っているのかも判別している。

- AOD(Analysis Object Data)
  ESD から物理解析に必要なコンテナだけを取り出したデータである。

- Ntuple(D3PD)
  AOD から Athena の依存なく、ROOT 形式で扱うことのできるデータとなっている。物理目的に応じて AOD からさらに必要なデータを抜き出したものになっている。通常 Ntuple を用いて解析を行う。

- Wrapper File
  FTK シミュレーションに必要なデータを RDO から ASCII 形式で取り出したものである。具体的には、先頭の文字が'B' のものがバッドモジュール、'R' が run number 'F' が event number と event の始まり、'L' が event number と event の終わり、'S'が生のヒット情報、'P' と'C' がそれぞれクラスタリング後の Pixel と SCT のヒット情報、'E' がオフラインストラック、'T' が MC トラックの真の値、'V' がビームスポットの値を示している。
  将来的に FTK シミュレーションでは、インプットを Wrapper File を使わずに RDO を用いて行うことになる予定である。

次にデータ形式の変換方法について説明する。全ての変換を Athena のコマンドラインから行うことができる。例えば Byte Stream 形式の RDO ファイルから Wrapper File を作成するときの方法について説明を行っていく。
まず Athena のセットアップを行う。次に指定したテストエリアの下で、

```
Reco_trf.py inputBSFile=RAWdata outputTXT_FTKIPFile=ftksim_wrap_raw.dat
```

と入力すると、RAWdata というファイルから ftksim_wrap_raw.dat ができる。inputBSFile というオプションがインプットとファイル形式の指定であり、inputRDOFile にした場合、Pool 形式のファイルをインプットすることになる。

またアウトプットの指定も同じで、outputTXT_FTKIPFile というオプションはアウトプットとファイル形式の指定であり、outputESDFile とすると、ESD 形式のデータがアウトプットされるようになる。

また Reco_trf.py と入力すれば全てのオプションを見ることができる。

2.3.4 グリッドコンピューティング

グリッドコンピューティング（グリッド）は図 2.13 のようにネットワーク上にある複数の計算資源（スーパーコンピュータやコンピュータクラスタなど）を関連付けて一つのシステムとして扱うことのできるサービスである。

この技術の特徴はグリッドのネットワークにつながるコンピュータであれば全ての計算資源を使えること、また並列処理に適していることが挙げられる。並列処理は、同じ処理を複数の CPU に処理させることをいう。例えば 1000 万イベント処理を行わなければならない場合、1000 個に分けてそれぞれ 1 万イベントずつ処理を行わせることによって、理想的に考えると処理時間が 1000 分の 1 になる。さらにグリッドでは各計算資源同士の処理の込み具合がわかるため、空いている計算資源に処理をさせることができるので、並列処理の実現能力も高い。ATLAS 実験ではこのグリッドの技術を使い膨大なデータを処理または生成している。

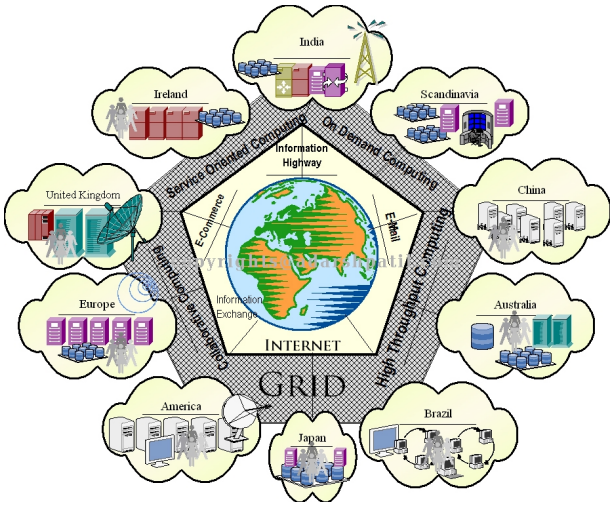

図 2.13: グリッドコンピューティングの概念図

次にグリッドの使い方を説明する。グリッドを使うためには、KEK や CERN などが発行している認証鍵が必要となり、.globus の中に入れてユーザーのホームに置いてお

く必要がある。この.globus はグリッドシステムの機能を管理したりストレージにアクセスするために必要なミドルウエアとして使われるものである。またグリッドを使うためには以下のコマンドを実行する必要がある。

```
1 source /afs/cern.ch/project/gd/LCG-share/new_3.2/etc/profile.d/grid-env.sh
2 voms-proxy-init -voms atlas -valid 100:00
3 source /afs/cern.ch/atlas/offline/external/GRID/ddm/DQ2Clients/setup.sh
4 source /afs/cern.ch/atlas/offline/external/GRID/DA/panda-client/latest
        /etc/panda/panda_setup.sh
```

まず初めのセットアップは gLite(グリッドミドルウェア) の UI(User Interface) と呼ばれるソフトウェアを使うためのセットアップである。グリッドの作業はこの UI を通して行うことになる。

2 番目のコマンドは、代理証明書を取得するためのコマンドである。-valid 100:00 のオプションは権限の期限をセットアップしてから 100 時間まで伸ばすことのできるオプションである。

ATLAS の DDM(Distributed Data Management) では、DQ2 というソフトウェアが使用されている。このソフトウエアを使うためのセットアップが 3 番であり、セットアップを行うことにより dq2-get のコマンドが使えるようになる。このコマンドはデータセットと同じ名前のファイルを各サイトのデータセットから探してダウンロードしてくることのできるコマンドである。

最後のコマンドは PanDA クライアントを使うためのものである。PanDA とはユーザーのジョブを管理するシステムである。以下のサイトから自分のジョブの状態をモニターすることができる。

```
http://panda.cern.ch/server/pandamon/query?mode=jobquery
```

最後にグリッドジョブを投げる際のコマンドについて説明する。グリッドのジョブを投げるコマンドは prun と pathena の二つあり、前者は通常のジョブを投げるときに使い後者は athena のソフトを使ってジョブを投げるときに使う。どちらのコマンドも投げた時のディレクトリにあるスクリプトファイルをサイトに持っていき走らせる。ただし、pathena コマンドはテストエリアに指定した範囲のパッケージも持っていき、そのサイト上でコンパイルすることになる。

3 FTK システム

3.1 序論

この章では FTK システムの概要とシステムの中で使われている装置の詳細について説明していく。

前章で説明した通り、瞬間ルミノシティの増加に伴い pile up 数も増加している。この Pileup の効果が高くなるにつれ、主な背景事象である QCD 事象による飛跡が多くなる。例えば、図 3.1 はバンチが交差する約 10cm の中に衝突点が 25 本ある図である。

現行の Lv1 トリガーでは、カロリーメータやミューオン検出器の情報のみ使っており Pixel/SCT の情報は使われていないため飛跡を再構成することはできない。また Lv2 トリガーにおいても時間の制約があるので ROI の領域のみしか飛跡を計算できない。そのためこの図の衝突点を全ては再構することができない。

しかしながら、衝突点を再構成することは、飛跡が衝突点の周りにできにくい e 粒子や μ 粒子や τ 粒子、また長寿命である B ハドロンになり 1 次崩壊点と 2 次崩壊点に差ができる b クオーク、などに有効なため重要である。そこでトリガーのできるだけ早い段階で飛跡を全領域で再構成できるような FTK システムの開発している。導入する段階は図 3.2 のように Lv1 と Lv2 トリガーの間である。 Lv1 から Pixel/SCT の Hit 情報を受け取って、なるべく早く Lv2 トリガーに正確な飛跡情報を送るようにする。

FTK システムは最大瞬間ルミノシティ $1 \times 10^{-34}[cm^{-2}s^{-1}]$,13TeV の 2015 年の run にバレル部分を実装する予定である。

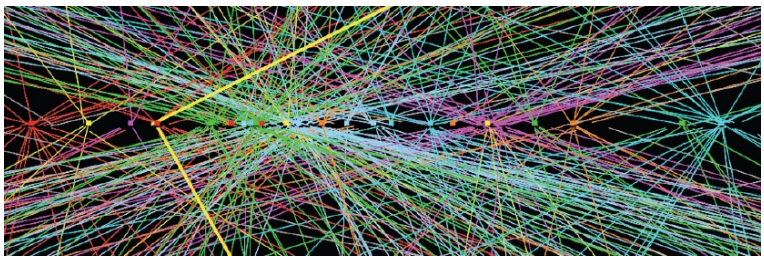

図 3.1: 25 reconstructed vertices

3.2 データの送受信

この節ではFTK システムが内部飛跡検出器の情報を受け取るモジュールである Dual-Hola(High Speed Optical Link for Atlas)、モジュール間の接続方法である Simple-Link(S-Link)、最後に FTK システムが Lv2 へと送信する飛跡のパラメータについて説明していく。図に FTK システム導入後のデータフローを示した。

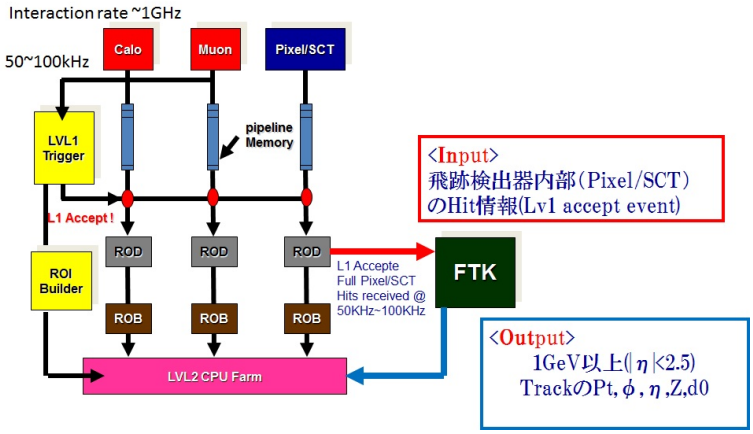

図 3.2: FTK システム導入後の図

3.2.1 Dual-Hola

FTK システムは ROD(Read Out Driver) に乗っている Dual-Hola から、光ファイバを通して Lv1 トリガーで accept された事象の Pixel/SCT の Hit 情報を受信する。もう１つの光ファイバーは本流である Lv2 トリガーの CPU ファームへデータを送信するためにデータを記憶しておくための ROB(Read Out Buffer) に送られる。

図 3.3: Dual Hola

3.2.2 Simple Link

S-Link は検出器のフロントエンドから読み出しを行う際に両方の転送モジュール間のやりとりを簡単に行うための CERN 標準の転送規格のことを言う。図 3.4 に S-Link の Concept を示した。

ケーブルには Duplex LC-LC コネクタが使われており、モジュール間で 2 つの光信号をやりとりすることができる。転送方法には 8bit/10bit 高速シリアル転送方式を用いており、1.28GHz/bit でデータの送受信を行うことができる。8bit/10bit はシリアル転送のため同期させるクロックがないので、8bit のデータを 10bit に変換してクロックをシリアルデータの中に埋め込み、受信側にデータを拾うタイミングを伝えている。またこの変換の際にデータとして送るだけではなく、制御信号として特別な 10bit データとして変換することもできる。このことを利用して、問題が起こった時には上流と下流の間で双方にそのことを伝えることができる。

また受信側は LDC(Link Destination Card)、送信側は LSC(Link Send Card) と呼ばれるファームウエアを使っており、これも S-Link の規格に含まれている。

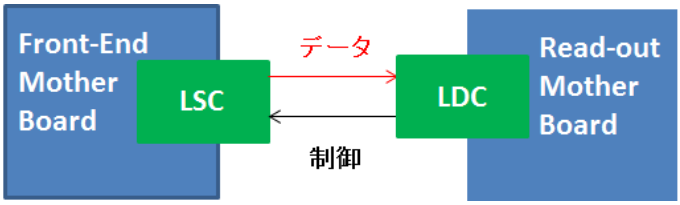

図 3.4: S-Link Concept

図 3.5: LC-LC connector

3.2.3 Output

FTK システムが求める値は $|\eta|$¡2.5,$p_T > 1Gev/c$ の飛跡パラメータ curvature η ϕ d0 z0 である。Lv2 トリガーに送信する飛跡のパラメータとその定義値について説明する。x0,y0,z0 はビーム軸の中心を (0,0,0) とした時の飛跡の始点の x,y,z 座標、また px,py,pz は飛跡の各運動量の成分、q をその粒子の電荷の符号とした。

- curvature
  curvature は飛跡の曲率で、p_T に対応した値になっており以下の定義している。
  $curvature = \frac{q}{2\sqrt{px^2+py^2}}$
- η
  $\eta = -In(tan(\frac{pz}{2\sqrt{px^2+py^2}}))$
- ϕ
  x,y 座標平面における飛跡の始点の原点に対する角度である。
  $\phi = tan^{-1}\frac{y+qpx}{x-qpy}$
- d0
  d0 は通常の再構成された飛跡では primary vertex からの距離と定義されているが、FTK システムでは　beam spot の中心点からの距離を d0 と定義している。d0 は beam spot の (x,y) 座標を x_{beam},y_{beam}　とした時、次のように定義している。
  $d0 = \frac{px(y\_0-y\_beam)-py(x\_0-x\_beam)}{\sqrt{px^2+py^2}}$
- z0
  z0 はビーム軸方向の座標である。

3.3 システム概要

この節では FTK システムの飛跡演算処理をするシステムとその並列処理を行うための準備をする DF(Data Formatter) システムの 2 つの部分に分けて説明していく。

3.3.1 演算処理システム

高速処理を行うため一般的に参照処理と呼ばれる考え方がある。参照処理とは処理を簡略化するために、あらかじめ計算できるデータを算出しておくことをいう。例えば、人間があらかじめ九九を覚えておき数字 2 個から一つの答えを導き出せるようにしておく処理のことである。

FTK システムでは飛跡パターンをあらかじめ作っておくことによって、処理を高速化している。パターンはパターンバンクと呼ばれる大容量のメモリーボード内に記憶されており、検出された Hit 情報と照合する。ただし記憶できるパターンにはハードウエア的に限界があり、またパターンが増加すると処理時間も増大するため、一時的に Hit 情報を束ねてパターンを照合している。この Hit 情報をいくつか束ねたものを SS(Super Strip) と呼んでいる。

一致したパターンはロード（粗いトラック）として算出される。その例を簡略化した図 3.6 に示した。この例では右の図がパターンバンクで 4 つのパターンが記憶されて、左の Super Strip のパターンと一致した上の二つがロードとして算出されている。

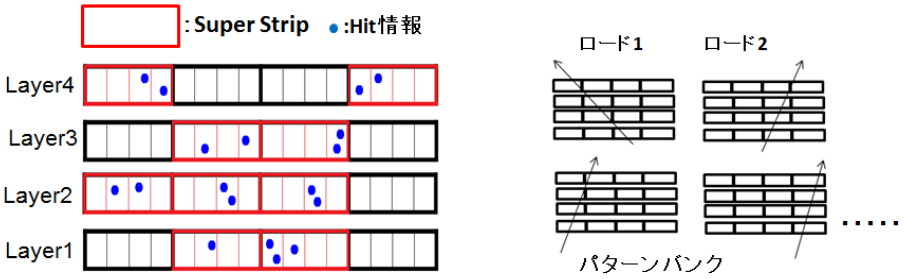

図 3.6: パターン認識の概念図

次にこの二つのロード内に記憶されている Hit 座標より線形一次近似によって飛跡のパラメータを算出する。P_i は各 5 つの飛跡パラメータのことを差し、N は Hit 座標の個数、x_j は Hit 座標、C_{ij} と q_i は定数となっている。定数は各パターンで使われる値が決まっている。

$$P_i = \sum_{j=1}^{N} C_{ij} x_j + q_i \tag{3.3.1}$$

また飛跡のパラメータの質を見るために χ の値を計算してカットをかける。

$$\chi^2 = \sum_{i=1}^{5}(\sum_{j=1}^{N} C_{ij}\delta + q_j)^2 \tag{3.3.2}$$

この関係式より Hit の座標さえ代入することができれば、飛跡のパラメータを求めることができるため非常に素早い処理を可能にしている。ただし、Super Strip のサイズ (SS Size) が大きいとロードの中にある Hit 数が増えてしまい、掛け算で計算する数が増えてしまう。例えば、図 3.7 の緑のロードの場合 32 通りも計算を行わなくてはならない。また SS Size が大きいと線形近似が成り立たなくなり飛跡の質が悪くなってしまう。このパターンと定数の求め方は次章で詳しく説明する。

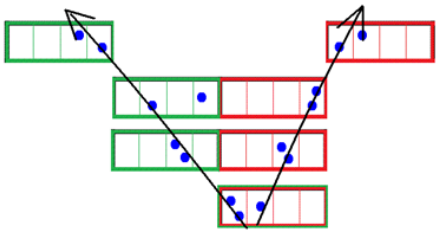

図 3.7: Track Fit

3.3.2 DF システム

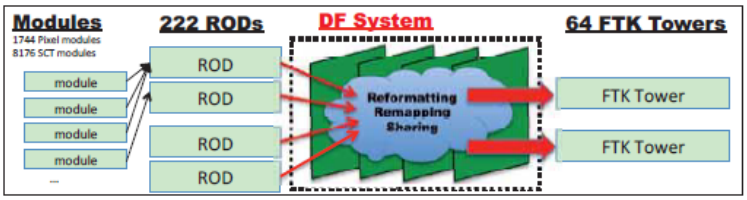

図 3.8: FTK システムの概念図

前述したように FTK システムでは Lv1 Accept された Pixel と SCT の Hit 情報から飛跡のパラメータを求めるので、Hit 数は処理時間と大きな関係がある。最大瞬間ルミノシティが $1 \times 10^{-34}[cm^{-2}s^{-1}]$ の時、1 事象で生じる Hit 数は Pixel で約 50k,SCT で約 100k となり合計で 150k にもなるため、前述したパターン認識システム一つで処理するのは難しい。

そこで FTK システムでは、図 3.8 のように DF(Data Formatter）システムによって、Hit 情報を検出器の 64 領域に分割して処理を行う。つまり DF システムとは、Data(Hit

情報) を FTK システムが扱えるように 64 領域のどの領域に属しているか Format するシステムのことである。

64 領域は図 3.9 と図 3.10 のように ϕ 方向に 16 分割、η 方向に 4 分割している。ϕ 方向は p_T1GeV 以上の飛跡が 10° 曲がるためにそれぞれ 10°over lap する領域がある。また η 方向にはバンチが交差する長さが約 10cm なので、そのラインをカバーするように分割されている。それぞれの方向で over lap する領域があるため DF システムは Hit 情報のやりとりを行わなくてはならない。

FTK グループでは ϕ を 8 分割した 8 つの領域を region と呼び、64 領域に分割した一つを Tower と呼んでいる。

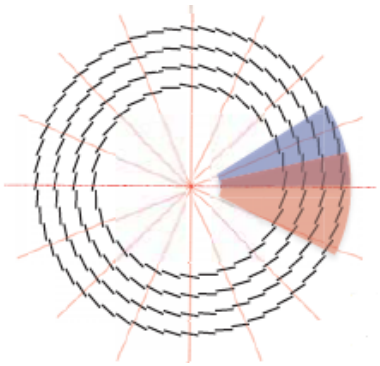

図 3.9: over lap phi region

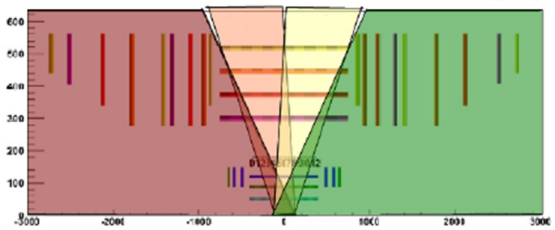

図 3.10: over lap eta region

また送られてく Hit 情報は 3.11 のように粒子が strip を斜めに通過した場合、複数の strip が鳴ることがあり、その strip の束を cluster という。そのためこの cluster から使う Hit 座標を決めなければならない。SCT では 3.11 のように cluster の中心の値を Hit の座標として使っている。

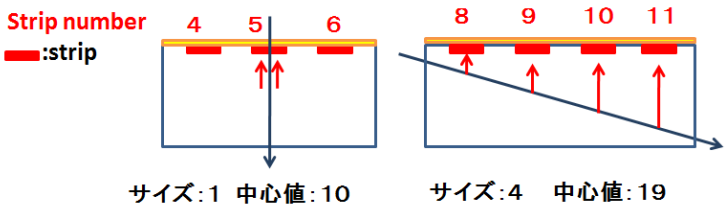

図 3.11: SCT Clustering

3.4 システム詳細

このセクションではFTKシステムを構成しているモジュール毎に詳細を述べていく。

図3.12のように前半は、32枚DFボードとボード1枚につき4つ載っているIM(FTK Input mezzanineカード)によってHit情報からHit座標を決定し、64個のtowerにそれぞれ送信される。

towerに分かれた後は、DO(Data Organizer)にてHit座標をSuper Stripに焼き直して次々にAMボード(Associative memory)に送られる。次にロードとして定義されたものがAMからDOに送られて、そのロード内にあるHit座標と共にTF(Track Fitter)に送られる。TFでは全セクションで説明した線形近似式によりトラックのパラメータが求められる。HW(Hit Warrior)を1つのまとまりとしたProcessor Unitから演算処理を行う。

ここまでを1st Stageと呼び各Towerごとに並列処理が行われる。各Towerで算出された飛跡情報は次々に2nd Stageに送らる。

最終的に全11層のHit情報を使いχ^2をチェックしてLv2トリガーのコンピュータファームに送られるという流れに成っている。

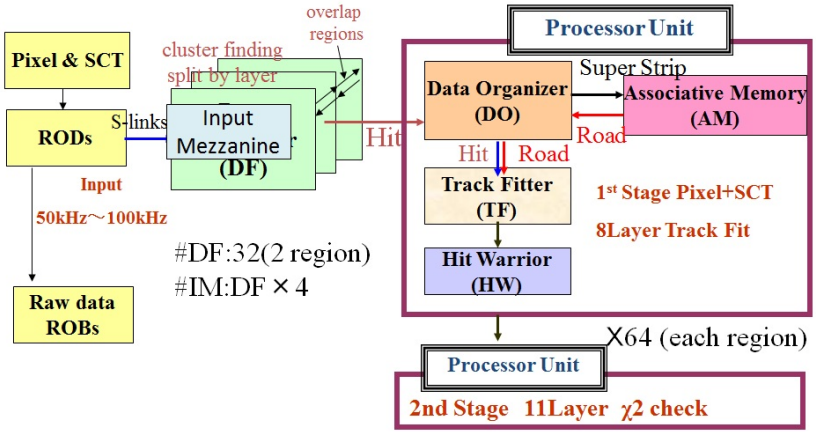

図 3.12: FTKシステムの概念図

3.4.1 Input Mezzanine card

IM card の役割は ROD から S-Link を通じてデータを正確に受信し、Hit 座標とその Hit 座標がどの tower に属しているのかを DF に送信することである。

また ROD と IM Card の接続には気をつけなくてはならない点がある。それは ROD が担当する内部飛跡検出器の位置によって Hit の Occupuncy が異なるため、その配分を均等に分けるように接続しなくてはならないという点である。

この IM Card は早稲田大学が主な担当となって開発しており、2012 年 6 月に v2.1 を作成した。その図が**??**である。

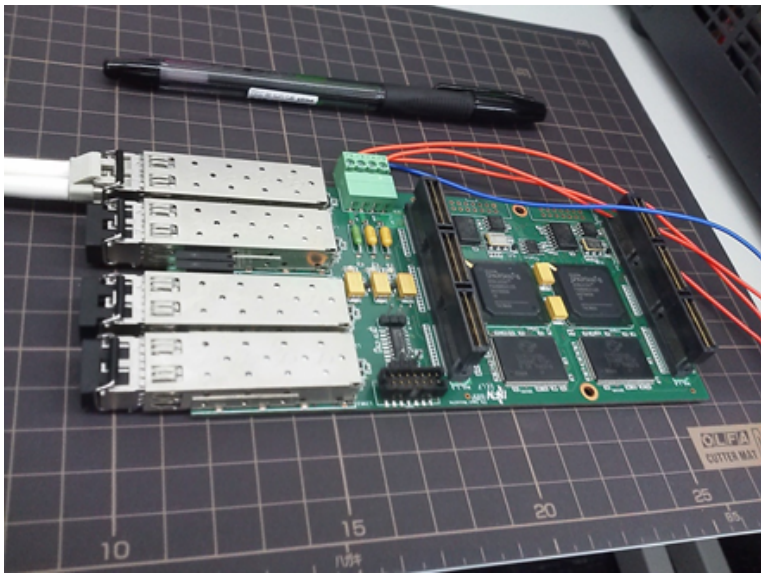

図 3.13: FTK Input Mezzanine

また新しい IM Card の製作も行われている。主な変更点は DF とのコネクタと部分である。このコネクタの変更は FTK システムにおいて Xilinx 社製の FPGA を使っているため、Xilinx 社でよく使われていること、また LVDS(Low voltage differential signaling) 転送方式に適しているので行っている。

LVDS 転送は 2 線間での電圧の違いを利用して送信するためノイズの影響を受けにくく、転送速度を速くすることができる。そのため配線の本数を減らすことができ、カードやボードのデザインがシンプルになり、またコストも減らすことができる。

3.4.2 Data Formatter

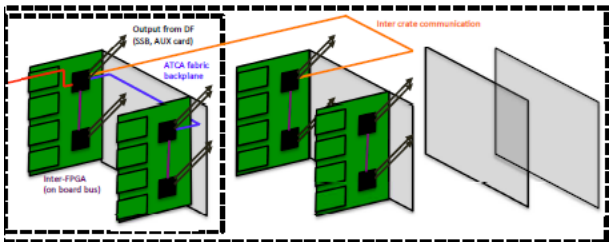

図 3.14: DF の全体図

DF は前節で述べたように DF 同士で自分の tower にある Hit を集めなくてはならないため、図 3.14 のように DF 同士で通信を行わなければならない。そのため FTK システムは相互に通信を行うのに優れた図 3.15 の ATCA 規格を用いている。この ATCA 規格では光ファイバーにより全てのボード同士が通信を同時に行うことができる。

FTK システムでは全部で４つの ATCA を用いて、それぞれに DF を 8 個セットして用いる。つまり１つの DF で 2Tower 分の処理を行うことになる。

DF の PCB の３ D の絵を図 3.16 に、データ転送部である RTM(rear transition module) を図 3.17 に示した。RTM には 10Gbit/s の信号転送ができる口が２つ付いている転送モジュールと４つ付いている転送モジュールがあり、前者が DO と通信を行い、後者が DF との通信を行うようになっている。

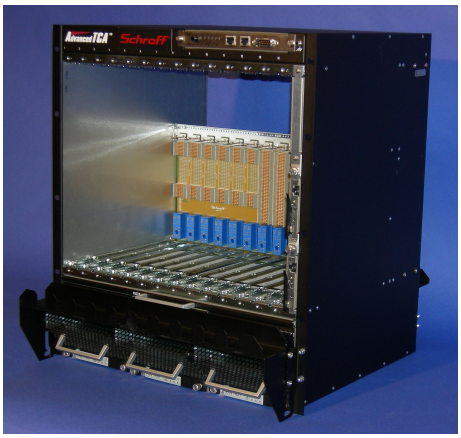

図 3.15: ATCA 規格

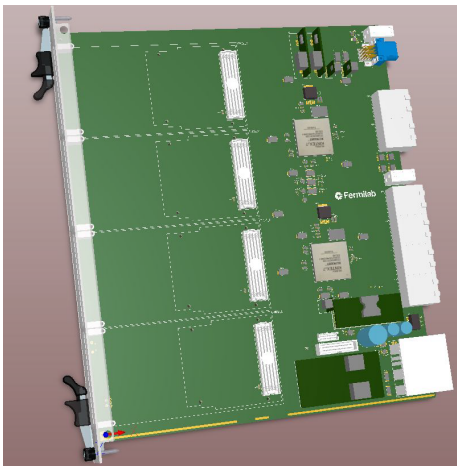

図 3.16: Data Formatter

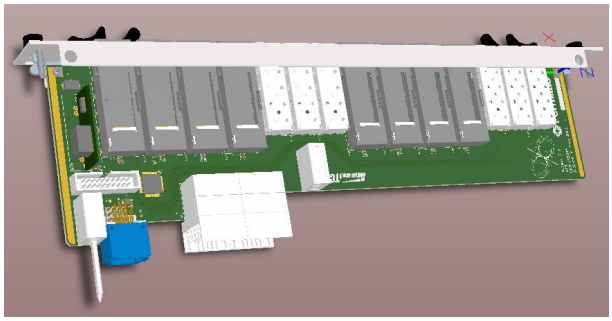

図 3.17: RTM

3.4.3 Data Organizer

DO では DF からヒット位置を各層ごとに受け取り、ヒット位置を Super Strip ID に変換して AM に送信する。また後の TF でヒット位置を使用するので全て DO のバッファに記憶しておく。これらの処理は DO で１００ MHz で行われる。また 1Tower を１つの AM で処理しようとするとロードの読み出しで非常に時間がかかってしまうので、２つの AM で処理を行っている。つまり DO は 1Tower につき 2 つある。また２つの DO は DF から同じヒットを受け取っている。

3.4.4 Associative memory

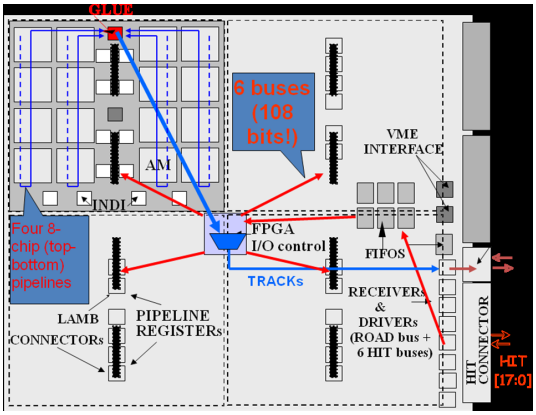

図 3.18: AM borad ブロック図

図 3.19: AM board

図 3.18 は AM のプロトタイプのブロック図であり、図 3.19 は実物である。AM では DO から SSID を中心の FPGA で受信する。そして SSID をその FPGA から４つの LAMB（Local Associative Memory Board）の FPGA へ送くり、全てのメモリーチップに並列に送られる。各メモリーは CAM（Contents Addressable Memory）構造になっ

ており、8 層の SSID（飛跡パターン）を記憶しており、SSID と一致した層は一致したことを記憶しておく。全ての層が一致した場合、図 3.18 の青い線のように読み出してロードの ID DO に送信する。特にこのロードの読み出しは並列処理を行うことができないので、ロードの数はプロセス時間に大きな影響を与える。

各LAMBの読み出しは200MHzで行うことができるので、1つのボード当たり800MHzで処理を行うことができる。もし 1 事象当たり 100kHz で FTK が受信した場合平均 8k ロードまで算出することができる。つまり 1Tower 当たり 16k までロードの算出を行うことができる。

次に AM が記憶できるパターン容量について説明していく。まずチップについてだが、AM Chip03 に変わる新しいチップ AM Chip04 の開発が進められている。Chip04 は Chip03 に比べて処理速度が 2 倍である 100MHz で動き、完成版は 16 倍の 80k パターン記憶することができる。次に LAMB には裏にも同様に Chip が 16 個付いており、合計で 32 個の Chip を搭載している。AM は４つの LAMB からできていて、１ Tower には２つ AM があるので合計で、80k×32×4×2=20.48M パターン記憶できる。

3.4.5 Track Fitter

FTK 概要で説明したとおり、ロードの中にある Hit 情報を使って全コンビネーションでトラックを算出し、それぞれの chi^2 のチェックを行って良いものを次のモジュールに送っている。

処理時間についてだが、1 回の fit を 1ns で行うことができ、1 つの Tower に 8 個あり、80k fit することが可能となっている。

3.4.6 Hit Warrior

HW(Hit Warrior) では TF から受信した Track の Hit 情報を受け取って，２つの Track に使った Hit がある一定数以上同じである場合、，同一のものと判断して削除している。

3.4.7 Second Stage

最後にセカンドステージでは、残り３つの SCT のヒット情報を DF からあらかじめ受信しておき、その情報用いて chi^2 のチェックを行い良いトラックを Lv2 トリガーに送信する。

4 飛跡パターン生成高速化

前章で説明した通り FTK システムは用意するパターンによってその性能が大きく変わる。特に実データの飛跡を再構成する場合様々なことを考慮してパターン生成を行わなくてはならない。それにも関わらず、パターン生成を行うのに従来の方法では約 2 週間ほどかかっていた。この章ではまずパターン生成の概要とその生成方法を改善し高速化を行った目的とその手法について述べていく。

4.1 パターン生成

パターン生成は大きくわけて 2 段階ある。
まず 1 段階目はモジュールを単位とした飛跡パターンであるセクターとそのセクターごとに線形一次近似式の定数を求める。
次にセクターごとに飛跡パラメータを乱数で振って、線形一次近似式より Hit の位置座標を求めて Super Strip を単位としたパターンを作る。それぞれについて詳しく説明していく。

$$P_i = \sum_{j=1}^{N} C_{ij} x_j + q_i \tag{4.1.1}$$

4.1.1 セクターと Fit 定数の求め方

セクターと Fit 定数を求めるには、大きく分けて以下の 3 つのことを行う必要がある。
1. シングルミューオンイベントの生成、トラックパラメータとヒット情報を求める
2. 飛跡からセクターを作る
3. 各セクターごとに Fit 定数の計算を行う

まずトラックパラメータとヒット情報が必要になるので、シングルミューオンイベントを生成してそれらを求めておく必要がある。次にトラックの残したヒット情報からセクターを求め、各セクターごとに Fit 定数の計算を行う。
またトラックの再構成率を考慮し、300M イベントのシングルミューオンイベントを生成して用いる。線形近似の質を考慮するために同じセクターにトラックが 15 本以上集まった場合のみ計算を行う。表 4.2 にシングルミューオンの生成数と検出器の領域カバー率を示した。
また再構成するトラックの kinematics を考慮を入れてシングルミューオンイベントを生成しなくてはならない。表 4.2 の領域で各パラメータがフラットな分布になるように生成している。z0 はビームバンチの交差する範囲で生成している。また d0 は B ハドロンのような長寿命粒子の飛跡のパターンを考慮して中心から 2.2mm の範囲で生成し

表 4.1: シングルミューオンの生成数と検出器の領域カバー率

Coverage	シングルミューオンのイベント数
0.970783	600M
0.966543	500M
0.958326	375M
0.950912	300M
0.933453	200M
0.918158	150M

ている。それぞれ図 4.3 と図 4.4 に生成した飛跡の始点と d0 の分布を示した。これにより飛跡が xy 平面内で均等に生成されていることが分かる。

表 4.2: 飛跡の Kinematics

1.0e-6<1/pT < 1.3e-3
-3.0< η <3.0
0< ϕ <2π
-120mm<z0<120mm
-2.2mm <d0<2.2mm

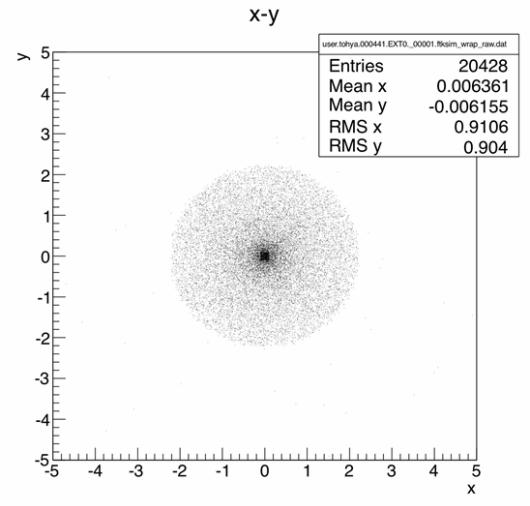

図 4.1: 飛跡の始点

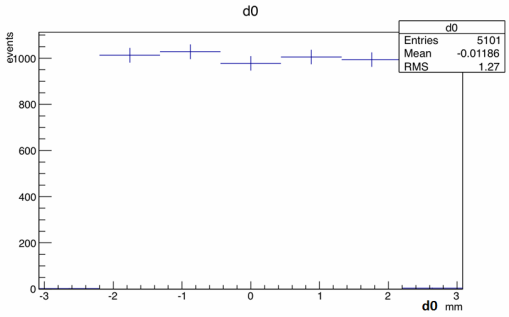

図 4.2: d0 分布

4.1.2 パターン生成

飛跡再構率 90% を要求した場合に必要なパターン数は 1000M 前後と膨大な数となっており、その分だけ MC イベントを生成するのには非常に時間がかかってしまう。

そのため飛跡のパラメータを乱数で振ることにより、擬似的にトラックを作っている。その値と各セクターの線形一次近似の定数の値よりヒット座標を求め、そのヒット座標を SS stripID に焼き直してパターンを作っている。(Pattern From Constant)。また各セクターごとにパラメータの値を取りうる範囲をあらかじめ求めておくことによってより効率的にパターンを求めることができる。このように各セクターごとに乱数を振って計算を行うだけなので、非常に速くパターン生成を行うことができる。

4.2 パターン生成高速化の目的

このセクションでは、パターン生成の際に考慮しなければならないビームスポットの変化や検出器の位置情報、SS ストリップのサイズについて説明していく。

4.2.1 ビームスポット

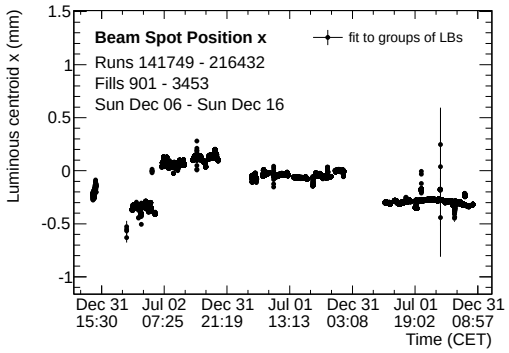

図 4.3: 2012 年 8TeV run beam spot X 座標の変化

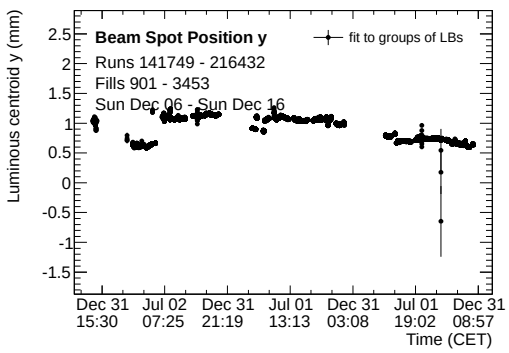

図 4.4: 2012 年 8TeV run beam spot Y 座標の変化

ビームスポットの位置は MC シミュレーションの場合では、原点座標を中心として作られているので、パターン生成を行う際には、原点座標を中心とした 2.2mm の円内を始点とした飛跡からパターン生成を行えばよい。

一方実データのビームスポットは、4.3 と 4.4 のように原点座標を中心としておらず、たえず変化しているため、飛跡パターンも変化してしまう。そのため崩壊点が作ったパターンの範囲外にある場合、飛跡パターンを再構成することができない。

よってビームスポットを中心としたパターン生成を行う必要がある。またこの時のFTK の d0 の定義はこのビームスポットとした座標からの距離にしなくてはならないので注意する必要がる。

4.2.2 検出器の位置情報

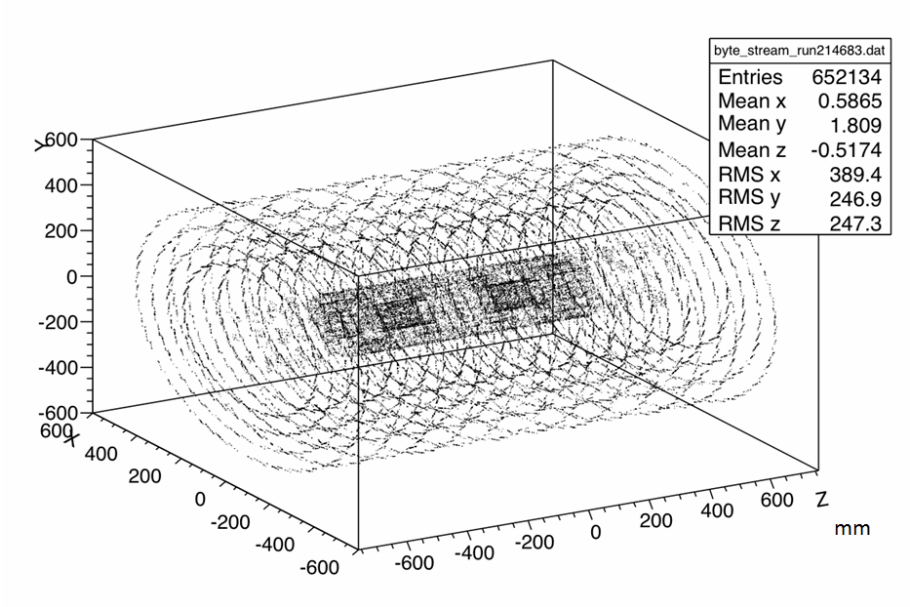

図 4.5: 検出器の位置情報

次に検出器の位置情報について説明する。Pixel と SCT のモジュールとその pixel と strip はそれぞれ座標を持っており、その座標をプロットすると図 4.7 のように内部飛跡検出器の形になる。

検出器の位置情報は Fit 定数を計算する際に使用する。またパターン生成時は、線形一次近似式を用いて、飛跡のパラメータを乱数で振り Hit 座標を得て、その Hit 座標からパターンを生成する。そのためパターン生成を行う時と FTK システムの Input として使うデータの位置情報が異なってしまうと飛跡パターンも異なってしまい、再構成できなくなってしまう。

図 4.6 に MC データと実データの最内層の pixel の x,y,z 座標の差を示した。MC データは ATLAS Geometry tag を ATLAS-GEO-20-00-00、Condtion tag を OFLCOND-SDR-BS7T-05-14 のものを使用し、実データは run212585 を使用した。x 座標は-0.37mm,y 座標は-0.51mm ずれている。このように全てのデータでこの検出器の位置情報が一致しているとは限らないので、常に確認する必要がある。

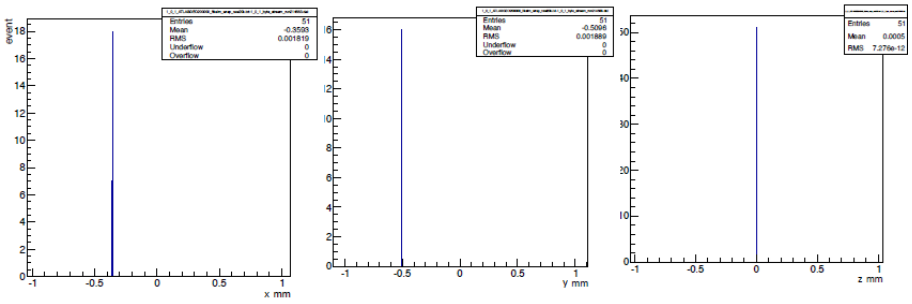

図 4.6: MC データと実データ (run212585) の Hit 座標の差

4.2.3 Super Strip

SS Size はトラックの再構成率、分解能、処理時間に影響を与えるので、FTK システムの中で重要なパラメータとなっている。

まずトラック再構成率と質について着目して説明する。図とのように $22\times(\phi)36\times(z)16$ は再構成率が 90% で要求されている場合、128M パターン必要となっており、$11\times(\phi)36\times(z)8$ は 1280M パターン必要となり、記憶しておくパターン数は 10 倍近くにもなってしまう。しかしサイズが大きいと線形近似が成り立たなくなり、トラックの質は下がってしまう。また擬似トラックも増えることが予想される。

次に処理時間に着目して説明する。SS サイズが大きいと擬似ロードが増えるために AM ボードから DO で読み出す部分で時間がかかってしまう。また TF でもロード内のヒット数が増加し、全ヒットパターンで計算を行うため時間がかかってしまう。

このように SS Size は FTK システムの処理能力に大きく関わっている。また再構成するべき飛跡の数が変化すると、処理時間も変わってしまうため瞬間ルミノシティの増加に伴いこの SS Size の最適化を行っていく必要がある。

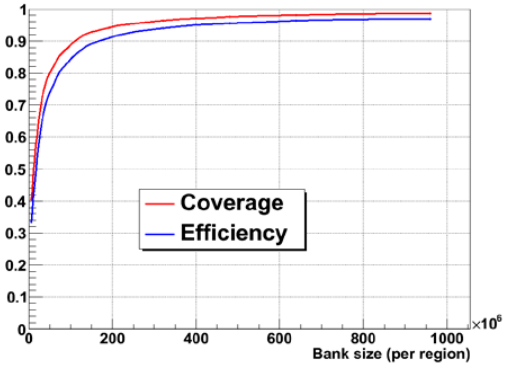

図 4.7: SS サイズが大きい時の記憶パターンと飛跡再構成効率の関係

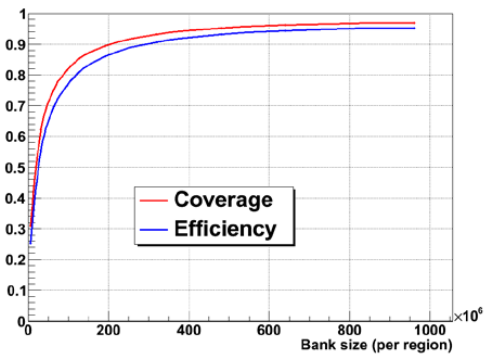

図 4.8: SS サイズが大きい時の記憶パターンと飛跡再構成効率の関係

4.3 MC サンプルの生成方法

このセクションではパターン生成に使用するシングルミューオン事象の MC サンプルの生成方法について説明する。具体的には Pythia によるイベント生成から Wrapper File の作成するところまで説明を行っていく。

4.3.1 MC 生成の流れ

まず MC 生成の流れについて説明していく。4.9 のように Wrapper File 生成までにはイベント生成、検出器シミュレーション、デジタイゼーション、再構成の 4 つの段階がある。それぞれについて説明していく。

イベント生成では、シングルミューオン事象の生成を行う。この時に前節で説明した Kinematics の条件で生成する。

次に検出器シミュレーションではイベント生成で作ったミューオンイベント事象が ATLAS 検出器内で発生した時のシミュレーションを行う。この時に ATLAS 検出器の位置の定義、また状態を決定しなければならない。この位置や状態はバージョン管理されており指定する必要がある。

デジタイゼーションではヒットデータを元に検出器のチャンネルからの信号をシミュレーションする。また通過時刻、場所とエネルギー損失から信号の発生時間や大きさなどの計算も行う。アウトプットとして RDO ファイルを生成する。

再構成ではヒット情報を元に飛跡の再構成やヒットのクラスタリングなどを行っている。この情報を ASCII 形式に焼き直して、Wrapper File を生成する。

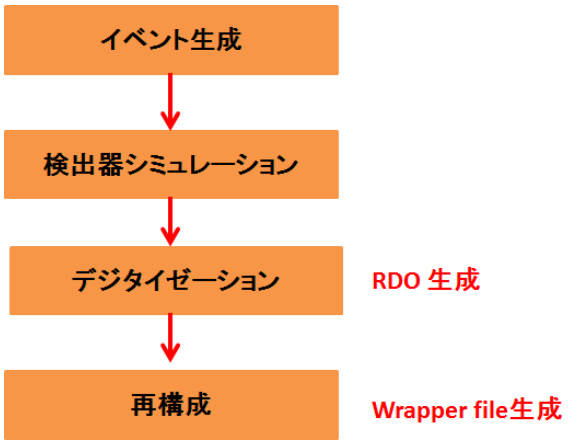

図 4.9: MC 生成の流れ

4.3.2 使用するパッケージのチェックアウト

次に MC 生成を行うのに必要なパッケージと具体的な生成方法について説明していく。まず以下のパッケージを取ってくる必要がある。これは FTK のサンプルトラック用に必要な変更を加えるためである。これらのパッケージはバージョンによってその内容が変わってくるため、バージョン変更する際は前のものとどこが違うのか確認する必要がある。またパッケージの取ってき方は、Athena のセットアップをした後に自分のテストエリアのディレクトリで、

cmt co -r パッケージ名とバージョン　バージョン管理されているディレクトリ

例　cmt co -r ParticleGenerator-00-00-54 Generators/ParticleGenerator

とチェックアウトすることができる。
Generators/ParticleGenerator-00-00-54
Generators/EvgenJobOptions00-01-77
InnerDetector/InDetConditions/InDetBeamSpotService-00-01-13
InnerDetector/InDetDigitization/SCT_Digitization-00-12-24
InnerDetector/InDetDigitization/PixelDigitization-00-11-04
Reconstruction/RecCBNT_Algs/CBNT_Particle-00-07-05

4.3.3 修正パッチとコンパイル

またこれらのパッケージは前述した通り修正を加える必要がある。その修正を行うための実行ファイルは
curl http://hep.uchicago.edu/ jswebster/save/patches/16.0.3/20111221/patch_athena_160003.sh

patch_athena_160003.sh
と取ってくることができる。このスクリプトを実行するとパッケージにパッチが当てられ、修正を行うことができる。またこの修正はそれぞれのパッケージのバージョンがあがると、すでに修正されている場合もあるので、バージョン間での変更点の確認を行う必要がある。
次にパッケージのコンパイルを行う。全てのパッケージを一括でコンパイルする場合は、setupWorkArea.py コマンドを使って WorkArea パッケージを作って行うと楽である。次にこのパッケージの cmt ディレクトリにて

```
cmt broadcast make
```

とすると全てのパッケージをコンパイルすることができる。

4.3.4 MC 生成

最後に取ってきたパッケージを使用してイベント生成から Wrapper File の生成までを行う。以下のコマンドでパッケージ内のアルゴリズムを動かすための実行ファイルを取ってくる必要がある。

```
curl   http://hep.uchicago.edu/~jswebster/save/patches/16.0.3/20111221
/ftktdr_run_scripts.tgz
 > ftktdr_run_scripts.tgz
```

次にこのファイルを解凍し、そのファイルの中にある run_generation.sh,run_simulation.sh, run_digitization.sh,run_reconstruction.sh を順番に実行する。それぞれイベント生成、検出器シミュレーション、デジタイゼーション、再構成に対応したスクリプトになっている。実行した結果、RDO ファイルと Wrapper file ができる。従来のパターン生成では Wrapper file を使用していたが、新しい手法では RDO ファイルを使用する。

またパターン生成では 300M イベントと膨大な量のサンプルを生成しなくてはならないので、数万イベントごとにグリッド上で並列処理を行い生成している。

4.4 従来の手法

この節ではWrapper Fileからセクターと Fit 定数をつくる従来の手法の概要と所用時間の説明をしていく。手順は以下のようになっている。

1. Grid 上でシングルミューオンイベントの Wrapper File を 300M 事象生成
2. Grid 上のデータベースから 1Tera Byte の容量の Wrapper File をローカル PC にダウンロード
3. 300M 事象のトラックから検出器を 8region に分けて、それぞれセクターリストを作る
4. 各 region ごとに各セクターの Fit 定数の計算を行う。(8 並列処理)
5. 計算に失敗してしまったセクターを取り除く

次にそれぞれの段階でかかる時間を説明していく。1 ではグリッド上のどのサイト上で行うか、またグリッドを使用できる優先度によっても変わってくるが、コンピュータークラスターが多く、5 人分のグリッド権限があれば平均して 2 日ほどでできる。2 ではダウンロードするだけで約半日かかり、また 1TB の容量があるストレージを用意する手間がかかる。３では４で並列処理ができるよにするために行っている。これは 300M 事象もの処理を行うため約 1 日かかる。４は 8 台のローカル PC を用いて並列処理を行う。３と同様に 300M 事象処理しなければならないので約 1 日かかる。最後に計算に失敗してしまったセクターをそれぞれ取り除かなくてはならない。この作業の処理時間は 30 分ほどでできる。

しかしこの方法にはいくつか問題点がある。まず、セクターと Fit 定数を求めるために必要なパッケージが Athena とは独立しているため、コンパイルや関数の呼び方が独自のものとなっているため使える人が限られてしまうという点がある。

次に全ての段階でジョブがこけることなく成功したとしても、5 日以上かかってしまうという処理時間に問題点がある。また手間も多いためそれ以上の時間がかかってしまう。さらに３や４の段階でジョブがこけた場合、ジョブを投げ直すとまた 1 日かかってしまう。そのためパターン生成にかかる時間は最大で約 2 週間となっている。

4.5 新しい手法

4.5.1 概要

従来の方法との変更点は大きくわけて 3 つある。まず一つ目は Athena のフレームワークで動かすことのできるパッケージになっているという点が挙げられる。よって第 2 章で説明したコンパイル方法で使用することができる。

次にグリッドを全行程で使用することができる点が挙げられる。

最後に線形近似式の考え方を利用した算出方法の改善をした点である。図 4.10 に曲率の場合の例を示す。微小な範囲であるパターン内では、曲率はその縁から距離 x に比例していると近似できる。よって定数 C を求めるためには、同じパターン内にある各トラックの位置と曲率からなる変数を足し上げ、最後にトラックの数で割ることにより求めることができる。つまり、セクターとそのセクターのトラックのヒット位置とトラックパラメータからなるトラックの定数を求めておき、あとで同じパターンを探してトラックの定数を足しあげてトラックの数で割ることにより求めることができる。

新しいパッケージでは、このことを利用して三段階に分け Fit 定数の計算を行っている。

1. Grid 上でシングルミューオンイベントを 300M 事象生成してセクターとトラックの定数を求めておく。
2. 同じセクターを探して、飛跡定数を足しあげる。
3. Fit 定数を計算する。

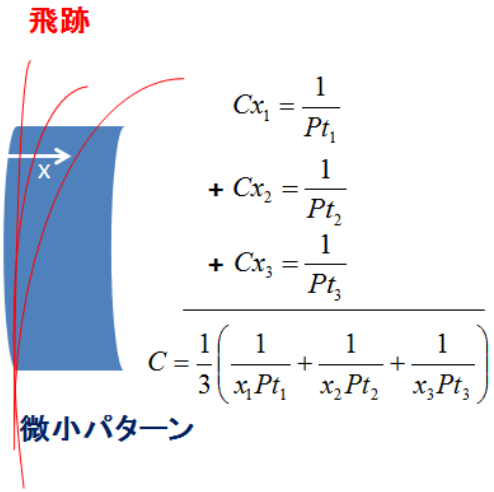

図 4.10: 線形近似の例

4.5.2 TrigFTKBankGen パッケージ

TrigFTKSim と TrigFTKBankGen パッケージを使用する。TrigFTKSim は RDO からヒットとトラックの情報を受け取るために使用する。TrigFTKBankGen は新しい手法のために開発したパッケージである。このパッケージでできることは以下のようになっている。

1. TrigFTKSim パッケージの FTK_SGHitInput.cxx から FTKDataInput.cxx を経由してクラスタリングしたヒットとトラック情報を受け取りセクターとトラックの定数を求める。
2. 同じセクターを探して、飛跡定数を足しあげる。
3. Fit 定数を計算する。
4. subregion に分ける
5. slice file を作成する。

4 は FTK シミュレーションを走らせる際、より並列処理を行うためにセクターファイルを分割することを意味してる。またセクターファイルにあるセクターはそのセクター内のトラック数順に並んでいる。subregion は例えば 4 つのセクターファイルに分けたい時、4(n-1) 番目のセクター、4n-3 番目のセクター、4n-2 番目のセクター、4n-3 番目のセクターといったように分割している。
5 の slice file は各セクター ID とそのセクターを作ったトラックのパラメータを簡易に記憶したものである。このファイルは Pattern From Constant でパターン生成をする際に使用するため作っておく必要がある。
次にパッケージの各ソースファイルの説明をする。

- FTKBankGenAlgo.cxx
  FTKDataInput.cxx からクラスリングした後のヒットとトラック情報を受け取り、セクターと飛跡の定数の計算をおこなっているファイル。第一段階で使用する。
- FTKPattKDTree.cxx
  同じパターンを探して飛跡の定数の足しあげを行っている。パターンの探索方法は 2 分探索木というアルゴリズムを用いて行っている。
- FTKConstGenAlgo.cxx
  initialize() でいくつかの関数を呼んでいる。merge() は第 2 段階で同じセクターを見つけて定数の足し上げを行っている。constantgen() は Fit 定数の計算を行う関数である。make_subregion() は subregion を作成するための関数である。

- atlparslice_root.cxx
  Slice File の作成を行うために使用している。第三段階の Fit 定数を計算した後に計算を行うことができたセクターとそのセクターの簡易なトラックパラメータを用いている。

4.6 従来の手法との比較

この章では従来の手法と新しい手法との比較を行う。具体的にはそれぞれのパターンバンクを使用してシングルミューオンイベントの飛跡再構成率と分解能を見ていく。

従来の手法は 2010 年の FTK Technical Proposal より抜粋したものを使用しており、11Layer 全ての層で Fit をしたものである。

また新しい手法のパターンバンクのコンフィギュレーションを示した。FTK シミュレーションで用いたシングルミューオンイベントのデータセットを表に記した。

表 4.3: バンクのコンフィギュレーション

ATLAS 検出器タグ	ATLAS-GEO-16-00-00
ATLAS コンディションタグ	OFLCOND-SDR-BS14T-IBL-03
Layer 数	8layer option C
SS Size 小	24×20×36
SS Size 小 パターン数	480M pattern
SS Size 大	50×64×144
SS Size 大 パターン数	48M pattern

表 4.4: シングルミューオンデータセット

事象数	100k events
ATLAS 検出器タグ	ATLAS-GEO-16-00-00
ATLAS コンディションタグ	OFLCOND-SDR-BS14T-IBL-03

4.6.1 飛跡再構成率

飛跡再構成を SS Size 大の場合と小の場合について求めて、それぞれ TDR の結果と比べた。その結果を図 4.11 から 4.16 まで示した。飛跡再構成率は、SS Size 大では TDR に比べて大きく、また SS Size 小では TDR に比べて小さい結果となった。この結果は TDR が 11Layer 使っているため SS Size 大より小さく、また SS Size 小ではパターン数が足りないために飛跡再構成率が悪くなっていると考えられる。

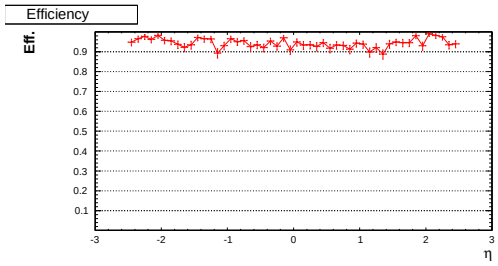

図 4.11: SS Size 大飛跡再構成率 η 分布

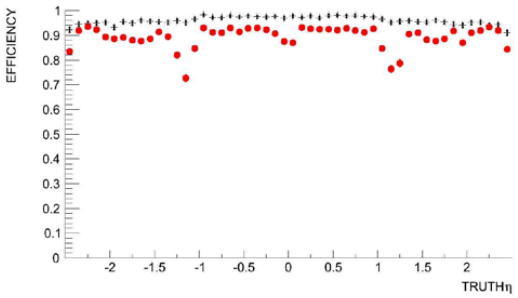

図 4.12: TDR 飛跡再構成率 η 分布

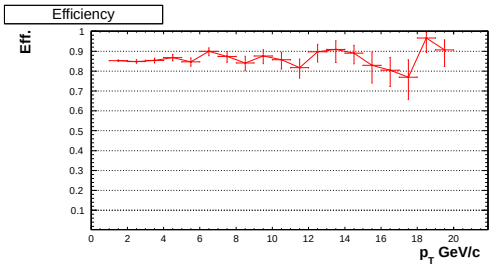

図 4.13: SS Size 大飛跡再構成率 pT(GeV) 分布

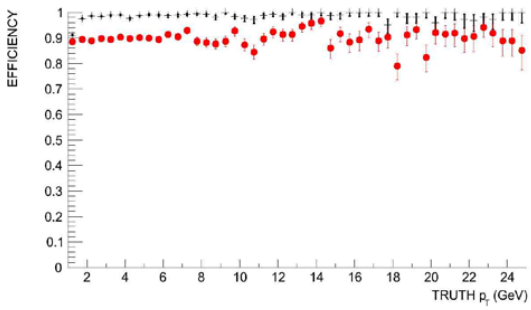

図 4.14: TDR 飛跡再構成率 pT(GeV) 分布

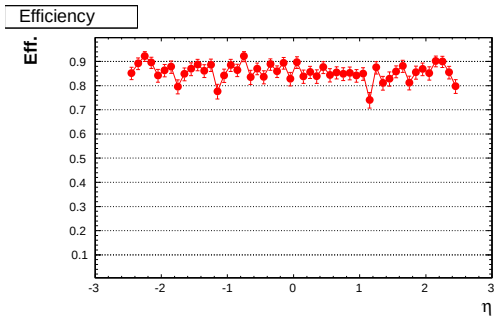

図 4.15: SS Size 小 飛跡再構成率 η 分布

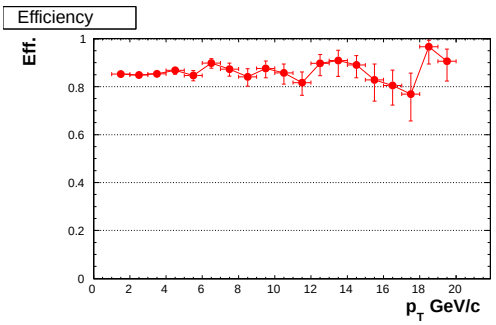

図 4.16: SS Size 小 飛跡再構成率 pT(GeV) 分布

4.6.2 飛跡の分解能

次に飛跡の分解能について議論していく。比較はSS Size大では十分な線形近似が出来ていない可能性があるため、SS Size小を用いて行った。それぞれの結果を4.17から4.26に示す。

その結果ϕとcaruvatureはSS Size小の方が良い結果になりその他は悪い結果となった。これは11Layer全てを使ったTDRの結果の方が全てのパラメータで良くなるという予想に反した結果となった。

しかしながら、TDRと今回の手法ではMCシングルミューオンイベント生成でも異なった点が存在する可能性がある。今後は同じ手法で生成したMCシングルミューオンイベントで同じに結果となるのかを確かめる必要があると考えられる。

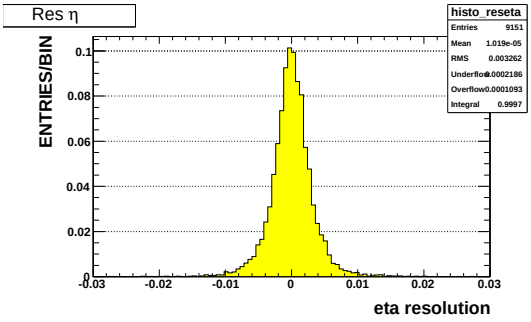

図 4.17: SS Size小 飛跡分解能 η

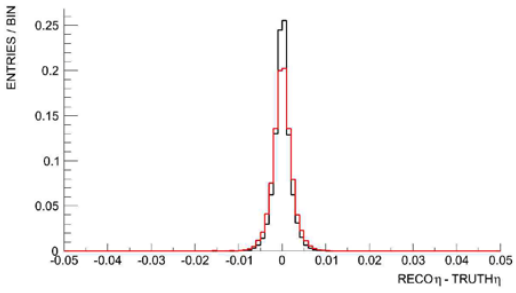

図 4.18: TDR 飛跡分解能 η

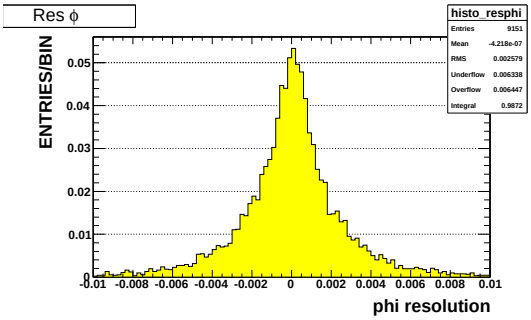

図 4.19: SS Size小 飛跡分解能 ϕ

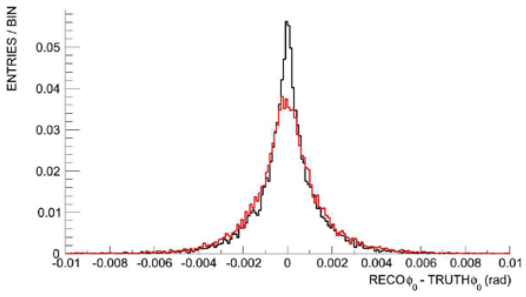

図 4.20: TDR 飛跡分解能 ϕ

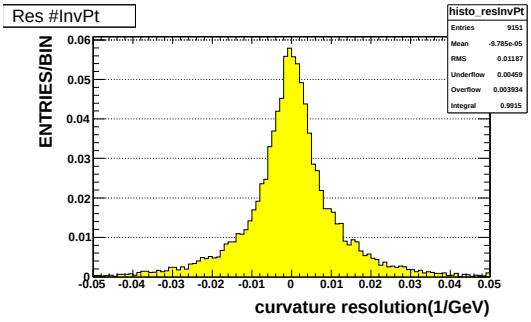

図 4.21: SS Size 小 飛跡分解能 caruvature

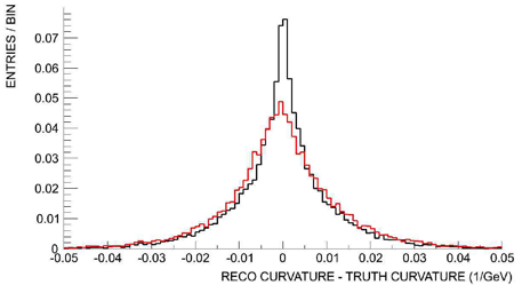

図 4.22: TDR 飛跡分解能 curvature

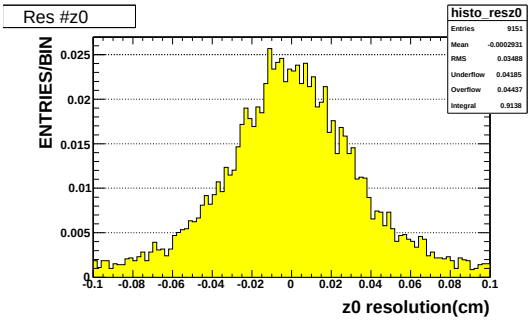

図 4.23: SS Size 小 飛跡分解能 z0

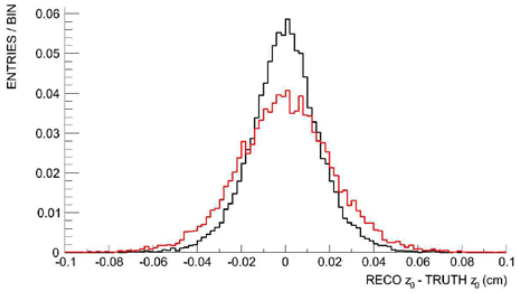

図 4.24: TDR 飛跡分解能 z0

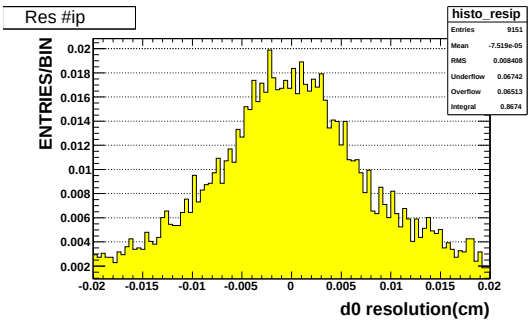

図 4.25: SS Size 小 飛跡分解能 d0

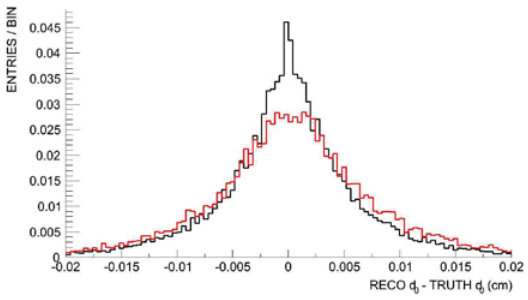

図 4.26: TDR 飛跡分解能 d0

4.7 纏めと考察

FTK システムを実機に入れる際には、様々なことを考慮に入れる必要がある。特に用意しておくパターンはビームスポットや検出器のアライメントの変化などによって変更をする必要があった。しかしながら、従来の手法では最大で約 2 週間もの時間がかかってしまっていた。

本研究では、線形近似の考え方を利用して、並列処理できる段階を増やすことにより生成時間を大幅に短縮することができた。また Athena のフレームワークで使用できるようにしているので、より使い勝手が良くなったといえる。

しかしながら、従来の手法とトラックの分解能に差ができてしまっているため今後は同じ Layer 数、同じ MC 生成方法を用いての比較を行っていかなければならないと考えている。

5 FTK システムの性能評価

この章では実際のデータから得られた Pixel/SCT の Hit 情報を用いシステムの性能評価について議論していく。

5.1 データセット

表 5.1 に今回使用したデータセット名、表 5.2 にデータセットの状態をまとめた。また表 5.3 に使用したバンクの情報をまとめた。

ESD ファイルよりヒット情報を得たため、オフラインでクラスタリングされているヒット情報を使用して FTK シミュレーションを行った。

表 5.1: データセット名

user.tompkins.data11_7TeV.190728.ESD_MinBias.f411.FtkWrapTrigSkim.v2.0/

表 5.2: 使用した run の状態

run	190728
ルミノシティブロック	233
事象数	315
ビームスポット	x=-0.057 y=1.064
瞬間最大ルミノシティ	4.86×10^{30}
パイルアップ数	平均 25 個
トリガー	Minimum bias Trigger
ファイル形式	ESD ファイル

5.2 解析結果

5.2.1 オフライントラックと FTK トラックの数

FTK トラックと比較するオフライントラックは、パターン生成している Kinematics 内（表 4.2 を参照）のトラックを使用した。その範囲内のオフライントラックと全 FTK トラックの数とその比を 5.1 と 5.2 に示した。現時点ではトラックの数は良い一致をしているようにみえる。

表 5.3: 使用したバンクのデータセット

ATLAS 検出器タグ	ATLAS-GEO-16-00-00
ATLAS コンディションタグ	OFLCOND-SDR-BS14T-IBL-03
Layer 数	8layer option C
SS Size	50×64×144
SS Size パターン数	48M pattern

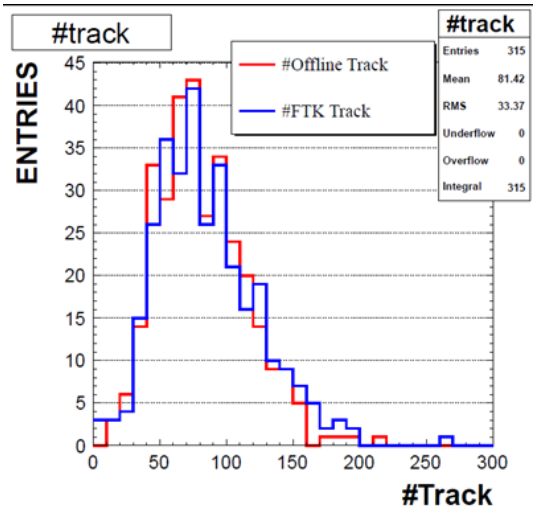

図 5.1: Offline Track と FTK Track の数

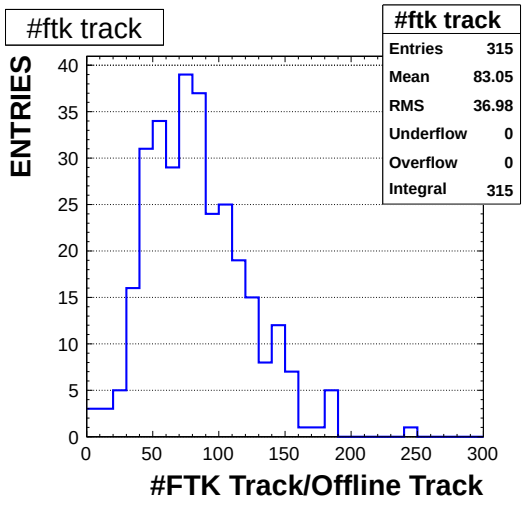

図 5.2: 事象毎の FTK と Offline Track の割合

5.3 トラックのマッチング

次にオフラインと FTK トラックの比較を行っていく。まずオフラインと FTK トラックの距離を見て、同一のトラックかどうかの判断をしていく。オフライントラックから最も近い FTK トラックとの距離を 5.3 に示した。dR=0.1 付近のトラック数が一番低くなっており、同一のトラックであるものとそうでないトラックの境になっていることがわかる。

5.3 にオフライントラックから dR<0.1 の FTK トラックの数を示した。このように FTK トラックと全くマッチしなかったオフライントラックが約 40% もいる結果となった。つまり FTK システムで約 40% ものトラックを再構成できていないという結果を意味する。

この結果は MC シングルミューオンのトラックを 95% 再構成できるという結果と大きく異なっている。つまり MC トラックとは違い実データのトラックには何かしらの問題があり、再構成することができないということを意味している。

そこで全てのオフラインイントラックとマッチしたオフラインイントラックの Kinematics の分布の比較を行うことで、どのようなトラックを再構成できていないか調べた。オフラインイントラックの dR<0.1 に FTK トラックが 1 個あるいは 2 個いた場合で、一番距離の近い FTK トラックをマッチしたものとして評価を行った。

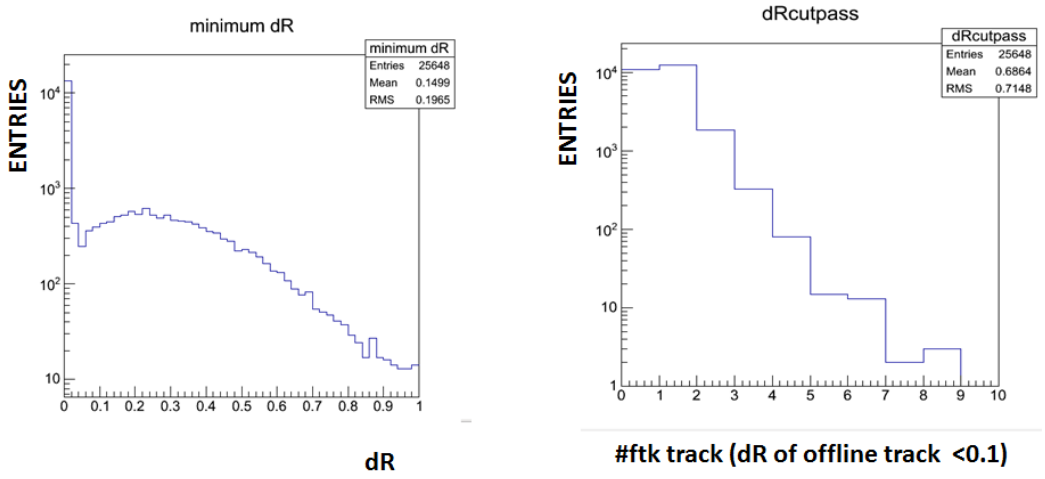

図 5.3: Offline Track から最も近い FTK Track との距離とオフラインイントラックから dR¡0.1 の FTK トラック数

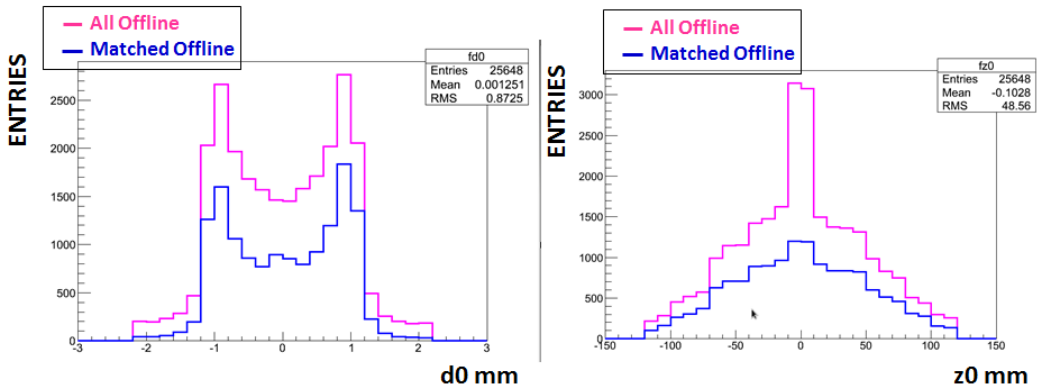

図 5.4: マッチしたオフラインイントラックとの比較 d0,z0

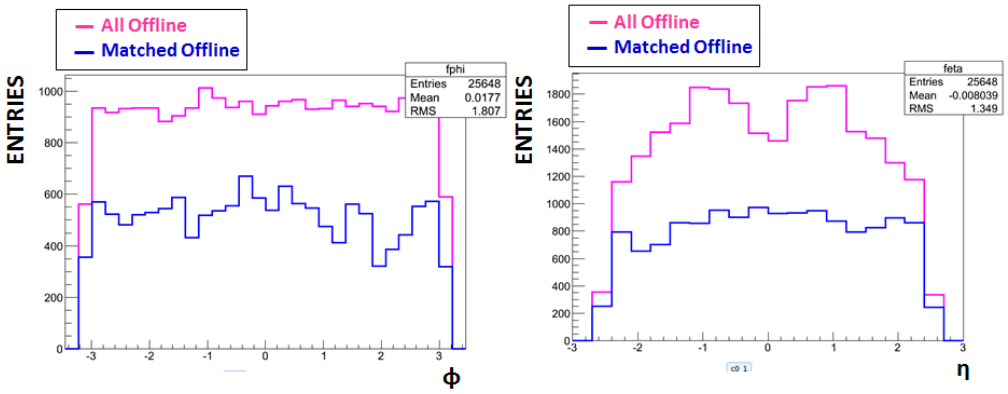

図 5.5: マッチしたオフラインイントラックとの比較 ϕ,η

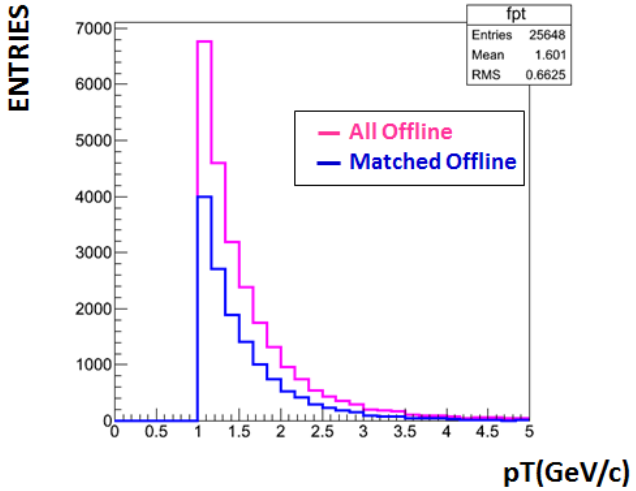

図 5.6: マッチしたオフラインイントラックとの比較 pT

5.3.1 考察

FTK トラックとマッチしたオフラインとマッチしなかったオフライントラックの分布にはいくつか異なる点がある。

まず z0 の中心部分のオフライントラックが再構成できていないことがわかる。これはオフライントラック全体の約 10% もある。

またマッチした ϕ の分布はがたがたしている。これは内部飛跡検出器は放射線ダメージなどにより損傷している場合があり、ヒット情報を送ることのできないモジュールが存在するためこのようながたがたな分布になってしまっていると考えられる。

6 纏めと今後の展望

本研究ではパターン生成高速化と実データをインプットとして用いたFTKシステムの評価を行った。

パターン高速化では並列処理を行う段階を増やし、またグリッドコンピューティング技術を駆使することにより、最大で２週間かかっていた時間を約２日間でできるようにした。またパッケージをAthenaのフレームワーク内で使えるようにしたため、パターン生成が従来の手法と比べ簡単なものになった。

実データをインプットとして用いたFTKシステムの評価だが、オフラインイントラックを約40%再構成できていないことがわかった。再構成できなかったトラックの分布を調べた結果、オフラインイントラックのz0がビームの中心に近いところで約10%ものトラックを再構成できていないことがわかった。またϕの分布もフラットになっておらず、これは内部検出器の中で壊れているモジュールがあることが原因だと考えられる。

最後に今後の展望について述べていく。まずパターン生成については、実データのトラックを再構成することを意識して実データのオフラインイントラックを使ってパターン生成を行う手法についても考えていきたい。また放射線損傷のため内部飛跡検出器の中で信号を送信することができなくなっている不感モジュールが今後増加していくことが予想される。この不感モジュールをパターン生成を行う際に考慮に入れることを考えたい。

次に実データの解析について述べていく。今回使用したMinimum Biasデータの場合、同じVertexからの飛跡同士の距離が近いためFTKのトラックの評価を行うことは難しい。よってオフラインイントラックでミューオンとして同定されたトラックのみでFTKの評価を行っていきたいと考えている。不感モジュールの部分に予めワイルドカード(WC)と仮定のヒット情報を使用しトラックの再構成を行うアルゴリズムの導入も行っていきたいと考えている。

7 付録

7.1 便利なリンク集

便利なページのリンクを紹介する。

1.Good run list page http://atlasdqm.web.cern.ch/atlasdqm/grlgen/All_Good/
Good run を調べたい時に使うページ。

2.Run query page http://atlas-runquery.cern.ch
run の状態が載っているページ。
run の beam spot や瞬間ルミノシティなど調べたいときに使うページ。
find run 189280 / show all とすればその run の状態を見ることができる。

3.AMI page http://ami.in2p3.fr
ATLAS の data set 名を検索することができる。
自分の調べたいデータがあるときに使うページ。
ただし AMI にアカウント登録しておく必要がある。

4.Atlas Package https://svnweb.cern.ch/trac/atlasoff/browser/Trigger/TrigFTK
ATLAS で使うソフトウエアのパッケージが全て見ることができる。
またパッケージは svn で管理されており、パッケージ名の下のディレクトリ
にある trunk に最新版があり、tag の下にパッケージのバージョンごとに
置かれている。

5.ATLAS Geometory tag page
https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGeomDBTags
Geometory tag の Release 情報や詳細な変更事項などが載っている。

7.2 パターン生成の pathena コマンド

第 4 章で紹介したパターン生成を行う際に、グリッド上でイベント生成からマトリックスファイルを生成するまでのコマンドを以下に記した。

```
athena --split 520 --extOutFile=matrix_11L_14Dim.root,
--dbRelease=ddo.000001.Atlas.Ideal.DBRelease.v200501:DBRelease-20.5.1.tar.gz
--outDS user.tohya.beamoff190728.Jan30_100k500j_v3 --trf "time athena.py -c
'RandomSeed1 = %RNDM=780100 ; RandomSeed2 = %RNDM=78001600
'jo.TrainingMuons.py ; time ./AtlasG4_trf.py --ignoreerrors=ALL
--omitvalidation=ALL inputEvgenFile=ParticleGenerator.pool.root
outputHitsFile=hits.pool.root maxEvents=-1 skipEvents=0 randomSeed=%RNDM=7113600
geometryVersion=ATLAS-GEO-16-00-00 physicsList=QGSP_BERT
preInclude=VertexFromCondDB.py
DBRelease=%DB:ddo.000001.Atlas.Ideal.DBRelease.v200501:DBRelease-20.5.1.tar.gz
conditionsTag=OFLCOND-SDR-BS14T-IBL-03 IgnoreConfigError=False AMITag=NONE ;
time ./Digi_trf.py --ignoreerrors=ALL --omitvalidation=ALL AddCaloDigi=False
inputHitsFile=hits.pool.root outputRDOFile=rdo.pool.root maxEvents=-1
skipEvents=0 geometryVersion=ATLAS-GEO-16-00-00 preInclude=VertexFromCondDB.py
digiSeedOffset1=%RNDM=790100 digiSeedOffset2=%RNDM=785600 doAllNoise=False
samplingFractionDbTag=QGSP_BERT
DBRelease=%DB:ddo.000001.Atlas.Ideal.DBRelease.v200501:DBRelease-20.5.1.tar.gz
digiRndmSvc=AtRanluxGenSvc triggerConfig=DEFAULT
conditionsTag=OFLCOND-SDR-BS14T-IBL-03 IgnoreConfigError=False
AMITag=NONE DataRunNumber=1 ; time athena.py  FTKBankGen_joboptions.py;"
```

また以下に重要なコマンドを記しておく。
–split:投げるジョブ数の指定
–extOutFile:アウトプットファイルの指定
–outDS:アウトプットファイル名の指定。自分の user 名をいれておく必要がある。
–randomSeed:MC によって計算を行うための乱数の指定を行う。同じ数字を用いた場合、同じ 4 元運動量を持つ粒子ができるので、並列処理を行う時に注意する必要がある。
–DBRelease:Geometory tag と Condtion tag に使うファイルのバージョンの指定
–geometryVersion:Geometory tag の指定
–conditionsTag:Condition tag の指定を行っている。

7.3 Pattern From Constant

第 4 章で紹介したパターン生成をグリッド上で行う方法を説明する。まず使用するパッケージは Athena のフレームワーク外のパッケージなので、コンパイル方法の説明を行った後、グリッドにジョブを投げるコマンドを紹介する。

7.3.1 使用するパッケージ

使用するパッケージは TrigFTKSim と TrigFTKLib である。以下のようなコマンドを打てば、取ってくることができる。

```
cmt co -r TrigFTKLib-00-03-01 Trigger/TrigFTK/TriFTKLib
cmt co -r TrigFTKSim-00-06-15 Trigger/TrigFTK/TriFTKSim
```

7.3.2 コンパイル方法

TrigFTKLib は FTK 開発初期に使われていたパッケージであり Athena のフレームワークでは動かない。また TrigFTKSim も standalone 版で動かす必要がある。それぞれのコンパイルの方法を説明する。まず Grid のセットアップを行う。次に asetup コマンドを実行して ROOT を使用できるようにしておく。

TrigFTKLib のコンパイルは以下のコマンドを standalone ディレクトリで実行する。

```
source bootstrap.sh
source build_standalone.csh
```

TrigFTKSim のコンパイルは以下のコマンドを standalone ディレクトリで実行する。

```
source bootstrap.sh
source ../grid/dependencies.sh
gmake -f Makefile.grid.mk
```

7.3.3 パターン生成コマンド

最後にパターン生成のコマンドについて紹介していく。コンパイルが上手く通ったことを確認した後に TrigFTKSim の script ディレクトリの下でまず以下のコマンドを実行する。

```
source grid.sh
site=ANALY_MWT2
unset extra loops regions subregions inputs
runstring=raw_8Lc_24x20x36_run190728
m=16
baseDS=user.tohya.FTKV17_base8Lc_190728_siteANALY_MWT2/
n=30000000
loops="0..150"
regions="0..7"
makeBank
```

コマンドのオプションの説明を行う。site:グリッドで使うサイトの指定を行う。

runstring: 8Lc の部分は使用する Layer 数の指定を行っている。7L の場合”7L” とする。これは base に使った sector ファイルと同じにしなくてはならない。また 24×20×36 が SS Size の指定部分になる。run190728 の部分は作成するパターンファイルの名前で、通常 base に使った sector ファイルと constant ファイルの名前と一致させておく。

m:subregion の数の指定を行う。

baseDS:使用する sector ファイルと constant ファイルの指定を行う。これらのファイルは subregion に分けるまえのファイル (unsplit file) を使用しなくてはならない。

n: 乱数を振る回数の指定を行う。

loops:何個ジョブを投げるか指定する。このコマンドの場合は 30M 回乱数を振ってパターン生成を行うジョブを 151 回投げるということになっている。それぞれのジョブは 30M であれば約 1 時間ほどで出来る。

region:パターン生成する region の指定を行っている。

7.3.4 パターンのマージコマンド

次にそれぞれのジョブのパターンをマージする必要があるので以下のコマンドを実行する。

```
source grid.sh
site=ANALY_MWT2
unset extra loops regions subregions inputs
runstring=raw_8Lc_24x20x36_v17
m=16
baseDS=user.tohya.FTKV17_base8Lc_190728_siteANALY_MWT2/
n=30000000
regions="0..7"
ftkDS=user.tohya.raw_8Lc_24x20x36_run190728_151NLoops_unmerged/
mergeBank
```

オプションは全てパターン生成を行ったものと一緒にしなければならない。また ftkDS はパターン生成のアウトプットのデータセット名を指定する。マージのジョブ数は region 数 ×subregion 数となっていて、約 1 時間ほどでできる。

8 Reference

(1) 長島順清著, 朝倉物理学大系　素粒子物理学の基礎, 朝倉書店,1998
(2)F. ハルツェン・A.D. マーチン共著, 小林　郎・広瀬立成訳,
　クオークとレプトン　現代素粒子物理学入門, 培風館,1986
(3)FTK Group Page
　https://twiki.cern.ch/twiki/bin/viewauth/Atlas/FastTracker
(4)FTK TDR 2010
FTK: a hardware track finder for the ATLAS trigger Technical Proposal April,13,2010
(5)ATLAS ソフトウエア講習会ホームページ
　http://www.icepp.jp/atlas-japan/tutorials/tutorial2011/
(6)Run query page
　http://atlas-runquery.cern.ch

9 謝辞

最後に研究を通してお世話になった方々への感謝を述べたいと思います。

まず研究に対する姿勢、研究方針、研究発表の工夫など丁寧に教えてくださった寄田准教授に心から感謝を致します。物理を通して人生や物事に対する態度や考え方などを深く教えてくださったことを重ねて感謝を致します。

研究を行う際のコンピュータ環境や生活を支えてくださり、また時には飲みに連れていってもらい人生に関するアドバイスをくださいました蝦名幸二氏、やさしく丁寧な指導をしてくださった永野間淳二氏、ミーティング中などにおそろしく的確なアドバイスをしてくださった田中雅士次席研究員に感謝いたします。

また 2 度のイタリア出張で熱心に物理やものの仕組みの指導してくださり、また生活面においても通学時の車の運転や海外での慣れない生活を支えてくださった木村直樹助教に感謝いたします。

更に、FTK グループやイタリア出張の際にお世話になった方々にも感謝しています。特に家を貸してくださった Paola 氏、FTK シミュレーションのサポートやミーティングで議論をしていただいた Guido 氏、AM のことを教えていただいた Marco 氏に感謝いたします。

同期である飯沢君、岡本君、杉田君、藤崎君の 4 名とは 3 年間という短い時間でしたが非常に濃く付き合えたと思っています。私が酔いつぶれている時に介抱してくれたり、私のしょうもない議論に付き合ってくれたりなど様々な場面で助けてもらいました。ありがとうございました。そしてまた会いましょう。そのときは別々のところにいますが楽しいお話ができるようにこれからも頑張っていきたいと思います。

最後に大学生活はもちろん、私が社会に出るまでの生活を支えてくださった父、母に心から感謝を述べたいと思います。