CTA 大口径望遠鏡用高速データ収集システムの開発

Development of a High-Speed Data Acquisition System for the Large-Sized Telescopes of CTA

石尾　一馬

2015/2/4

目次

概要

高エネルギー宇宙の観測は、超高密度物質、ブラックホールなどの極限時空、超高エネルギー宇宙線や暗黒物質粒子の対消滅ガンマ線の探索など、宇宙論や基礎物理学の発展に重大な貢献をもたらす可能性がある。解像型大気チェレンコフ望遠鏡は、地上からのガンマ線観測によってこれら物理学の重要な課題に迫ることを目的としており、100GeV から 100TeV 以上にわたる広範なエネルギー領域において最も成功を収めている望遠鏡である。そして現在、更に高感度・高性能のチェレンコフ望遠鏡を建設するため、Cherenkov Telescope Array (CTA) 計画が進行中である 。CTA 計画は、ガンマ線天文台として北半球と南半球に次世代大気チェレンコフ望遠鏡群を建設する。20GeV から 100TeV 以上にわたる広範なエネルギー領域に対し現行望遠鏡よりも 1 桁深い観測感度を達成するため、高・中・低それぞれのエネルギー領域に特化した小・中・大口径の望遠鏡を開発する。低エネルギー側は大口径望遠鏡 (Large Sized Telescope ,LST) と呼ばれ、日本が中心となって開発を行っている。

LST のデータ収集システムは、望遠鏡の焦点面に並ぶ 1855 本の光電子増倍管から読出し回路ボード (Dragon FEB)265 枚に読出されデジタル化された空気シャワー由来大気チェレンコフ光の波形情報を、それぞれの FPGA に搭載された SiTCP 技術により TCP/IP 通信でカメラサーバーに転送する。LST は低エネルギー領域での感度向上を目指すため、平均 20kHz の非常に高いトリガレートに対応する必要があり、そのため観測データの転送レートは望遠鏡一台あたり平均 40Gbps と膨大な観測データの超高速転送が必要になる。

これを実現するため、本論文では LST のデータ収集システムの開発を行った。望遠鏡のカメラを構成する 265 枚の Dragon FEB から TCP/IP 非同期通信によりデータを取得し、得られたデータをイベント毎に結合し、後段の処理に回す。このデータ取得とイベント結合をマルチスレッドプログラミングにより開発した。性能評価では、データ取得性能のマルチスレッド化効果、イベント結合機能の試験と性能評価を最大 26 接続で行い、それを元にサーバー分割最小単位の 45 接続、望遠鏡単位の 265 接続における性能を予測した。このスケールテストは Dragon FEB のデータ送信機能のエミュレーターを開発して行った。結果、データ取得性能は 2 スレッド 26 接続において 45kHz に追随し、45 接続における予測値 30kHz は要求値のほぼ 2 倍を満たした。また、イベント結合機能は、20 接続 40kHz で 20 分、16 接続 45kHz で 10 時間の機能試験に対して安全に稼働でき、一部の FEB がトリガ受信を失敗することによるトリガ番号の部分的な抜けにも柔軟に対応してイベント結合ができた。

第 1 章

はじめに

1.1 ガンマ線宇宙物理学の概況

ガンマ線宇宙物理学は 1967 年の OSO-3 衛星による宇宙ガンマ線初検出以来、観測が難しいため進歩が遅れていたが、1991 年打ち上げの Compton ガンマ線衛星による数十 keV〜10GeV での観測と全天のガンマ線天体カタログ作成を皮切りに、ここ 20 年ほどで劇的な進歩を遂げた。100MeV から 100GeV の領域では 2008 年に打ち上げられた Fermi 衛星による観測が大きな成功を収め、1800 を超える天体が発見されている。TeV 領域では 1989 年の Whipple 望遠鏡により初めて宇宙からの TeV ガンマ線が観測されたのを皮切りに、チェレンコフ望遠鏡が大きな発展を遂げ、現在では 150 天体以上が発見され、世界で 3 箇所の望遠鏡が観測を続け、新しい発見をもたらしている。そして次世代のチェレンコフ望遠鏡として CTA が計画され、感度が現行望遠鏡のさらに 10 倍になることで、1000 を超える天体の発見が期待されている。

1.2 物理学実験におけるデータ収集システムの概況

近年、物理学実験は大きな変貌を遂げようとしている。そのひとつに、測定データ量の大幅な増加が挙げられる。高速サンプリング ADC(Analog to Digital Converter) が登場し、多チャンネル化 (さらに画像のサンプリング においては高解像度化) が進んでいることがその原因であるが、これにともなって、データ集積点となる 中央ストレージ装置においても、容量のみならず、I/O 性能向上とその評価は大きな課題になっている。この課題はもちろん中間集積点となるネットワークスイッチやネットワークインターフェースにおいても同様である。

また、本研究が対象とする実験では SiTCP 技術を用いている。これは FPGA という集積回路上に IP 通信を実装する技術である。この技術を埋め込むことで、検出器からのデータ取得が非常に簡便になる。さらに SiTCP 技術は汎用性が非常に高い。そのため物理学実験において規模を問わず急速に広まりつつあり、開発元の高エネルギー加速器研究機構 (KEK) 主催のワークショップが活発に開催されている。一方で大規模なデータ収集を念頭においた場合、その収集システムの構築と評価専用に FPGA を用意することは費用面と手間から考えると簡単ではない。今回の実験手法は、既存の PC 資源を有効活用することにより低コストで実施できるため、SiTCP を用いたデータ収集システム構築 f において、広い範囲で応用が可能である。

以上のように、多数のノードからの超高速データ収集を行う実験がますます重要になっている。

1.3　本論文の目的と概要

Cherenkov Telescope Array (CTA) は、ガンマ線天文台として北半球と南半球に建設が計画されている次世代大気チェレンコフ望遠鏡群である。20GeV から 100TeV 以上にわたる広範なエネルギー領域に対し現行望遠鏡よりも 1 桁深い観測感度を達成するため、高・中・低それぞれのエネルギー領域に特化した小・中・大口径の望遠鏡を開発する。低エネルギー側は大口径望遠鏡 (Large Sized Telescope ,LST) と呼ばれ、日本が中心となって開発を行っている。LST は低エネルギー領域での感度向上を目指すため、平均 20kHz の非常に高いトリガレートに対応する必要があり、そのため観測データの転送レートは望遠鏡一台あたり平均 40Gbps と膨大な観測データの超高速転送が必要になる。

本研究ではこの LST のためのデータ収集システムの開発を行った。SiTCP 技術により TCP/IP 通信が可能である複数の FEB からデータを非同期で読出し、データを結合するイベント結合処理を行うまでの機能をマルチスレッドのプログラムとして開発し、要求性能を満たして稼働することを確認した。このために、エミュレータを開発した。

第 2 章

ガンマ線天文学

この章ではガンマ線天文学を概観する。ガンマ線領域は最も短波長または高エネルギーの領域であり、宇宙の高エネルギー現象を最も直接的に調べることができる。高エネルギー宇宙の観測は、現在、銀河系内、銀河系外に、PWN、SNR、AGN など多種多様な最高エネルギーガンマ線源が約 150 天体発見されている。また、その数のみならず、近傍の明るい天体に関しては、観測の高精度化がすすみ、天体での物理現象をより詳細に研究する事が可能になってきた。今後は、超高密度物質、ブラックホールなどの極限時空、超高エネルギー宇宙線や暗黒物質粒子の対消滅ガンマ線の探索など、宇宙論や基礎物理学の発展に重大な貢献をもたらす可能性がある。[11]

2.1 ガンマ線観測の歴史

2000 年ごろまでガンマ線天文学を牽引してきたのは、人工衛星による観測であった。宇宙から到来したガンマ線は大気によって吸収されてしまうため、地上で直接検出できないからである。人工衛星の技術は飛躍的な進歩をし、現在では 2008 年打ち上げの Fermi 衛星が観測を行っている。Fermi 衛星は 20 MeV - 300 GeV の有効エネルギー領域を持った LAT という検出器を搭載していて、2013 年までの 5 年間の観測により、到来するガンマ線を元に鮮明な全天サーベイマップが作成された。これまで発見されたガンマ線天体は 1800 を超える。

しかし、更に高エネルギーの領域では観測が難しい。なぜなら、ガンマ線光子の到来頻度はエネルギーが高くなるほど低くなるからである。

この人工衛星で観測が難しいエネルギー領域については、逆に地上からの観測が優位になる。直接の検出はできないが、高エネルギーガンマ線が大気に突入した際に起こす相互作用によりチェレンコフ光のフラッシュを生成するためである。これを観測する技術が解像型大気チェレンコフ望遠鏡（Imaging Atmospheric Cherenkov Telescope、IACT）である。チェレンコフ望遠鏡は、宇宙から到来するガンマ線が地球大気で発するチェレンコフ光を捉える。この方法では地球の大気圏を検出器として使うことができるため、人工衛星での観測に比べて 10000 倍以上の有効面積となる。

この圧倒的な統計を稼ぐ効果は、とくに 10GeV を超える高エネルギーガンマ線観測に威力を発揮する。宇宙から到来する粒子・光子は、高エネルギーではエネルギーの冪乗で到来頻度が減ってしまうからである。1989 年に初めて観測に成功して [16]、現在は世界 3 カ所に観測所があり、150 程度の高エネルギーガンマ線源が発見されている。

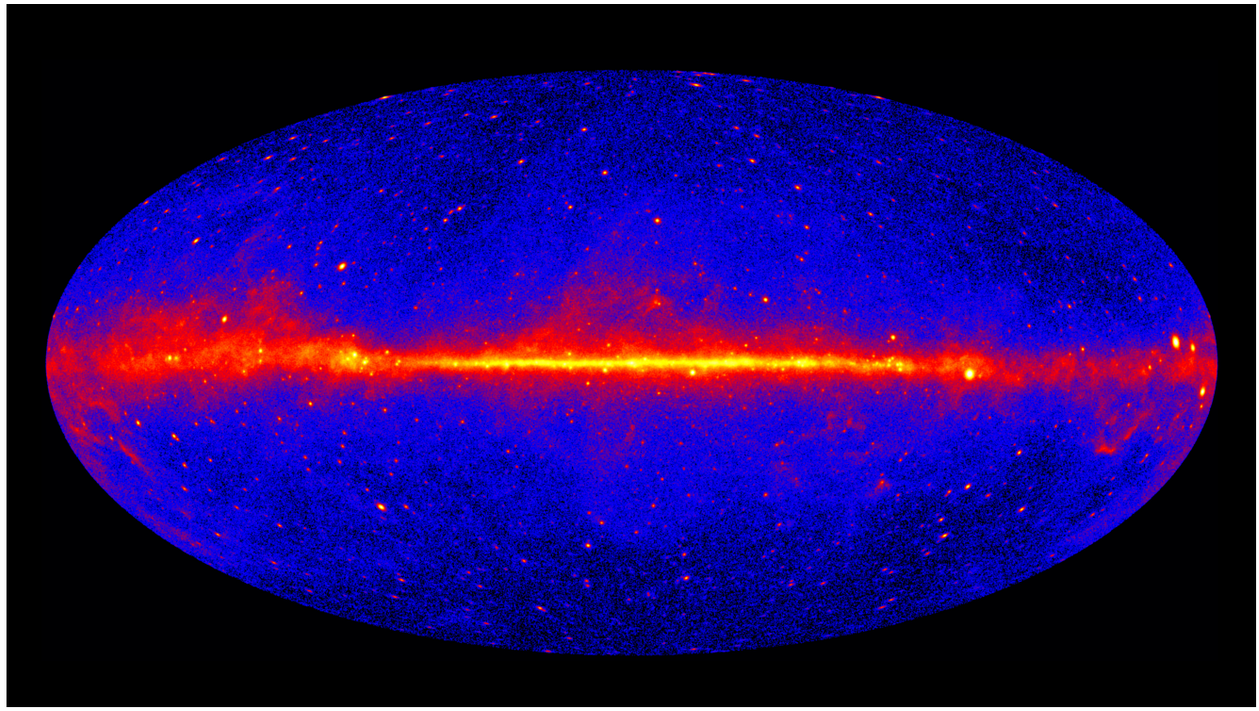


図 2.1　ガンマ線の全天マップ。Fermi 人工衛星による観測で作成された。

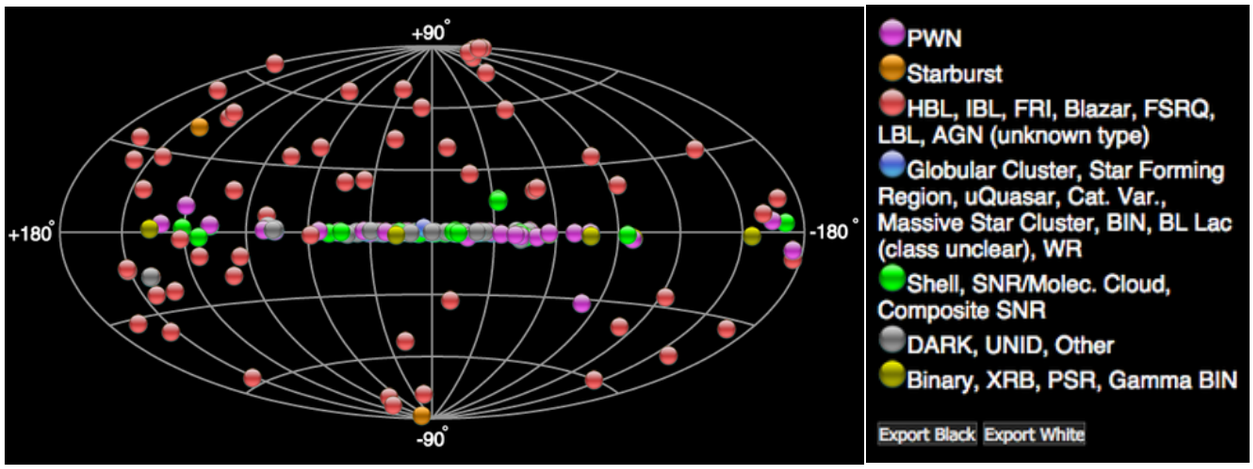


図 2.2　チェレンコフ望遠鏡により高エネルギーガンマ線が検出された天体の一覧。[9]

2.2　ガンマ線を生成する反応

熱的放射によりガンマ線が放射されたと仮定した場合、プランク分布の光子エネルギースペクトルによって典型的な温度を見積もるとすると、$kT \sim h\nu$ より、1 GeV に対応する温度はすでに 10^{13}K にもなってしまう。また、観測される高エネルギーガンマ線のエネルギースペクトルはプランク分布を持たない。したがって非熱的放射が要請される。ここでは、このような GeV、TeV 領域で考えられるガンマ線発生機構を概観する。

2.2.1 π^0 崩壊

π^0 粒子の寿命は 8.52×10^{-17}s と非常に短い。崩壊は 98.8% が

$$\pi^0 \to 2\gamma \tag{2.1}$$

と辿る。[6] また、π^0 の質量は約 135MeV である。そのため π^0 の静止系で約 70MeV のピークとなるガンマ線が放出される。π^0 粒子は高エネルギーの原子核（多くは陽子）が物質中を通過すると、物質を構成する原子の原子核との核破砕反応によって生成される。このような反応は、宇宙線が大気に突入した際の相互作用でも見られる。詳しくは後述する。

2.2.2 制動放射

制動放射は運動する電子が原子核のクーロン力によって加速度を受けて曲げられることに伴って、電気双極子の加速度運動が電磁波を放射してエネルギーを失う現象である。電子同士や陽子同士では電気双極子にならず、陽子と原子核の場合は非常に弱いので、これは電子と原子核との間で特有の現象である。

電子の運動が非相対論的な場合はイオン化損失が支配的だが、相対論的な状況では制動放射が支配的になる。

数密度 N、原子番号 Z の媒質中を電子が速度 v で通過することにより起こる制動放射のスペクトルは

$$I(\omega) = \frac{Z^2 e^6 N}{12\pi^3 \epsilon_0^3 c^3 m_e^2 v} \ln(\frac{192v}{Z^{1/3}c}) \tag{2.2}$$

となり、スペクトルは波長依存性がなくフラットになり、カットオフは電子が一回の衝突によりエネルギーの全てを失ってしまう $\hbar\omega = (\gamma - 1)m_e c^2$ に対応する辺りになる。

また、これによる電子のエネルギー損失は

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{Z^2 e^6 N E}{12\pi^3 \epsilon_0^3 c^4 \hbar} \propto E \tag{2.3}$$

これらはある速度の相対論的電子に対しての式だが、熱平衡にあるような高温プラズマでは、電子とイオンの速度が Maxwell 分布をしている状況で、イオンに対し相対的に速い速度で運動する電子がイオンのクーロン力によって制動放射を起こしている。このような状況での制動放射は熱制動放射または自由-自由遷移と呼ばれている。

2.2.3 シンクロトロン放射

シンクロトロン放射は、磁場中で相対論的な速度で運動している電子から電磁波が放射される現象である。磁場中の電子は、運動方向と磁力線の方向に垂直な方向のローレンツ力を受ける。それによって電子は磁場中を螺旋運動するが、進行方向を曲げられることに伴って、電気双極子のこの加速度運動により電磁波を放射する。この放射の特徴は、電子の螺旋運動の接線方 向に強いビーム状の放射をすること、放射はローレンツ力の方向に 100% の直線偏光をしていることである。

シンクロトロン放射による電子 のエネルギー損失は

$$-\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{4}{3}\sigma_T c \beta^2 \gamma^2 U_B \propto E^2 \tag{2.4}$$

σ_T はトムソン散乱断面積、U_B は磁場のエネルギー。

電子のスペクトル指数が $-p$ の時には、シンクロトロン放射もべき関数となり、その指数は $-(p-1)/2$ となる。

2.2.4　逆コンプトン散乱

高エネルギーの電子がマイクロ波や赤外線のようなエネルギーの低い光子と弾性散乱を起こし、エネルギーの高いガンマ線を生成する現象を逆コンプトン散乱という。コンプトン効果は高エネルギーの光子と低エネルギーの電子の弾性散乱により低エネルギー光子が生じる現象であり、基本的には同一の過程である。

$$-\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{4}{3}\sigma_T c\beta^2\gamma^2 U_{ph} \propto E^2 \tag{2.5}$$

σ_T はトムソン散乱断面積、U_{ph} は光子のエネルギー密度

2.2.5　対消滅

物質と反物質が衝突することにより、対消滅と光子の生成が起こる。電子ー陽電子の対消滅の場合、

$$e^+ + e^- \rightarrow \gamma + \gamma \tag{2.6}$$

という反応となる。これと同様の反応過程として、暗黒物質の対消滅によるガンマ線の生成の可能性が示唆されている。

2.3　ガンマ線宇宙物理学の目指す物理

超高エネルギーガンマ線天文学での狙いは宇宙における高エネルギー粒子の生成と伝搬、つまり非熱的宇宙の探求である。宇宙を伝搬し地球に届く放射光の多くは、星光のような高温物体からの熱的放射である。極限の条件では熱的放射は keV を超え、さらに高いエネルギー領域に達する。しかし、宇宙には熱的過程からは生成され得ない粒子があり、そしてこれらは熱的過程の代わりに、放射源から比較的少ない量の粒子がエネルギーを集めて持ち出す特別なメカニズムによって生成されているはずである。そのような非熱的な粒子の分布の一番良く知られている例は宇宙線である。エネルギースペクトルはべき乗分布のため特定の温度スケールを持たず、そしてそのエネルギーは 10^{20} eV、それより上にも達する。これは熱的放射メカニズムを考えることは到底無理である。

非熱的な宇宙の探求は多くがまだ未解決であり、TeV 領域を超える粒子の加速源の特定と加速機構の解明は理論的な推察と観測による確認は大きな課題である。そして、非熱的宇宙の理解は宇宙の進化、天体の進化、そして宇宙を超えて物理全体を理解する上でも非常に重要である。

ガンマ線宇宙物理学の目的は、このような非熱的粒子の加速、伝播、そして相互作用の理解を深めることである。特に以下の課題は、次章以降で述べるチェレンコフ望遠鏡により明らかにされると期待される物理である。

- 宇宙線加速源の特定
- ガンマ線天体の性質解明
- 銀河間背景光の測定による星形成史の解明
- 暗黒物質対消滅ガンマ線の探索
- ローレンツ不変性の検証

第 3 章

解像型大気チェレンコフ望遠鏡

解像型大気チェレンコフ望遠鏡（Imaging Atmospheric Cherenkov Telescope、IACT）は、ガンマ線を地上から間接的に検出する望遠鏡である。

宇宙から到来したガンマ線が大気に突入すると電磁シャワーを起こす。また、宇宙線はハドロンシャワーを起こす。これらのシャワーは大気チェレンコフ光の塊を発生させる。チェレンコフ望遠鏡はこれを地上で捉え、バックグラウンドとなる宇宙線由来ハドロンシャワーによるイベントを除去し、残った事象を統計的に解析してガンマ線の到来を判断する。この統計処理の中でガンマ線天体の方向やフラックスを知ることができる。また、チェレンコフ光の光量はガンマ線光子の入射時のエネルギーに依存する。このことを利用してエネルギースペクトルを求めることもできる。この章では、

- 大気チェレンコフ光発生の原理
- チェレンコフ望遠鏡による観測原理

について、順を追って説明する。

3.1 大気チェレンコフ光発生の原理

高エネルギーの宇宙線やガンマ線が大気に入射し、相互作用を繰り返し膨大な 2 次粒子のシャワーを大気中に発生させる現象を空気シャワー呼ぶが、ここでは特にガンマ線由来を電磁シャワー、宇宙線由来をハドロンシャワーと分類する。空気シャワーで作られたそれぞれの 2 次粒子は大気中での光速を超えて移動するため、チェレンコフ光を発生させる。2 次粒子は膨大な数が生成されるので、チェレンコフ光は塊となって地上まで到達する。電磁シャワーとハドロンシャワーではシャワーの発達の仕方に違いがあるため、チェレンコフ光の降り注ぎ方に違いがある。大気チェレンコフ望遠鏡はこの違いを利用してバックグラウンド除去を行っている。この節では、シャワーの発達とチェレンコフ光の発生原理について、電磁シャワーとハドロンシャワーの違いに注意しながら説明する。

3.1.1 電磁シャワー

数十 GeV を超える宇宙ガンマ線が大気に突入すると、大気中の原子核と相互作用により電子対生成を起こす。この $e^{\pm}$ はそれぞれ制動放射を起こし、再び高エネルギーの光子が発生する。この光子→電子対→光子・・・の連鎖反応により多数の電子対となって大気中に降り注ぐことを電磁シャワーと呼ぶ。電磁シャワー中で発生した電子、陽電子は、非常に高エネルギーのため、媒質中の光速を超えて飛ぶ。そのためチェレンコ

フ光が発生する。

ガンマ線の物質との相互作用は、エネルギーの大きさにより変わってくる。ガンマ線は物質を通過する際に相互作用を起こし、電子を弾き飛ばす。この作用はエネルギーが高くなるにつれて光電吸収、コンプトン散乱、電子－陽電子対生成と過程が変わっていく。入射ガンマ線が 0.511MeV×2=1.022MeV を超えると電子－陽電子対生成が起きはじめ、さらに高いエネルギーを持つと、生成した電子、陽電子が空気中の原子核のクーロン場により加速し制動放射を起こして再びガンマ線を放出する。この過程の連鎖が電磁シャワーという。この時生成された電子、陽電子は多くが真空中の光速となるため、媒質（空気）中の光速を上回っているので、チェレンコフ光が発生する。

チェレンコフ効果は、粒子が運動する場の反作用によって媒質が放射するもので、制動放射は原子との衝突が起きた時、運動する電子自身から放射される制動放射とは独立に起こる。

3.1.2　ハドロンシャワー

宇宙線の主成分は陽子などの原子核で、これが大気に突入すると大気中の原子核と相互作用をして多くの二次粒子の生成が繰り返される（核カスケード）。二次粒子は K 中間子も含まれるが、90% は π 中間子で、π^+、π^-、π^0 がほぼ同数生成される。

π^+ はほぼ 100% が

$$\pi^+ \to \mu^+ + \nu_\mu \tag{3.1}$$

と崩壊し、寿命は 2.60×10^{-8}s る。π^- はこれの荷電共役として考えれば良い。

先述のように、π^0 は 98.8% が

$$\pi^0 \to 2\gamma \tag{3.2}$$

と崩壊し、寿命は 8.52×10^{-17}s と非常に短い。

このように、高エネルギー宇宙線は大気との相互作用の結果、典型的には約 1/3 のエネルギーが π^0 に渡され、その π^0 がガンマ線を生成することにより、電磁シャワーがサブシャワーとして多数発生する。これによりガンマ線由来の電磁シャワーと同様のチェレンコフ光が発生することになるが、以下の様な特徴を持つ。

- 横方向のシャワー発達
  原子核の相互作用では横方向の運動量が与えられて二次粒子が拡散して飛ぶため、電磁シャワーと比べてハドロンシャワーは広がったイメージになる。
- ミューオン
  荷電 π 粒子の崩壊により μ 粒子がシャワーの発達段階で発生するが、μ 粒子の寿命は 2.20×10^{-6}s で、相対論的な寿命の伸びによって地上付近でもチェレンコフ光を発生させる。

3.1.3　チェレンコフ光

チェレンコフ光は、チェレンコフ光の放出される方向（粒子の進行方向とのなす角 θ）は屈折率 n に依存し、

$$\cos\theta \sim \frac{1}{n} \tag{3.3}$$

地表付近では $n \sim 1.00028$ で、これに対する $\theta \sim 1.4°$ である。大気上層では n が減少するので θ も小さい。また、速度がほぼ光速のまま電磁シャワーを起こし続けるので、チェレンコフ光が重なりあって発生し、光のフ

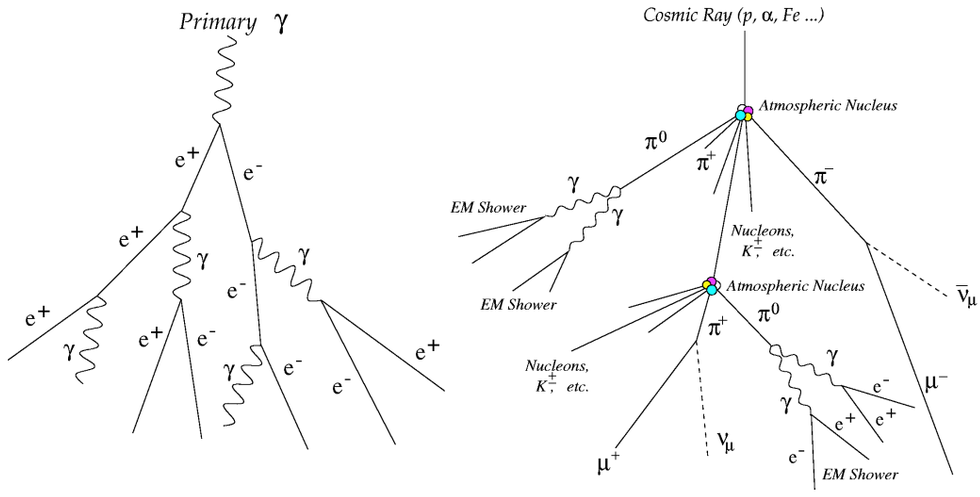


図 3.1　電磁シャワーとハドロンシャワーの発達の違い。[14]

ラッシュとなる。地上にはほぼ同時にチェレンコフ光子が到着する。地面に垂直な入射の場合、地上には直径約 300m、厚さ 1m の「円盤」として降り注ぐ。

3.2　チェレンコフ望遠鏡による観測原理

これまで述べたとおり、空気シャワー現象はチェレンコフ光の塊を作り出し、地上には直径約 300m、厚さ 1m の「円盤」として降り注ぐ。これを地上で捉えて「写真」を作り、シャワーの到来方向を判断する。

しかしこのチェレンコフ光は非常にかすかな光のフラッシュであり、わずか 10 ナノ秒程度の間に $1m^2$ あたり 50 個程度の光子が降り注ぐだけである。そのため、かすかな光を検出できる検出器が必要になる。そこでチェレンコフ望遠鏡では、大型の放物面鏡により集光を行い、焦点面に多数の光電子増倍管（Photo Multiplier Tube、PMT）を設置して、各 PMT にカメラの 1 ピクセルの役割を担わせる。

PMT は、光電面に光子が当たると電流となって取り出すことができる検出器である。電流の変化を信号波形として記録すると、多数の光子が到来するほど高い波形が得られ、そこから到来光子数を判断できる。空気シャワーによるチェレンコフ光のフラッシュが到来した瞬間に全 PMT から波形情報を取得すれば、ピクセルごとの明暗によるイメージとして取り出すことができる。

(図 3.4) はチェレンコフ望遠鏡がシャワーを捉えた様子をシミュレーションした結果である。望遠鏡は CTA 計画で建設予定の大口径望遠鏡（Large Sized Telescope、LST）で、この望遠鏡について詳しくは次章で述べる。図の左側はガンマ線によるシャワーイメージ、右側は宇宙線陽子によるシャワーイメージで、共に入射時のエネルギーは 500GeV である。それぞれの粒子を天頂から降らせ、望遠鏡も天頂を向けた状態でシミュレーションしているため、カメラ視野のほぼ中心からシャワーが到来しているイメージになっている。

また同じエネルギーを持ち、望遠鏡からほぼ同じ距離の地上位置に到達しているが、捉えられた光子数の総数はガンマ線のほうが約 3 倍程度高い。前章で述べたように、陽子などのハドロンシャワーにより大部分がパイオンになるが、π^0、π^+、π^- が同数ずつ生成され、そのうちの π^0 のみが 2γ 崩壊をすることによってチェレンコフ光を伴う電磁シャワーとなるためである。

この図で最も注目すべきなのは、ガンマ線由来のイメージではほぼ楕円の形をしているのに対し、陽子由来のイメージでは形の定まった塊を持たないことである。このような違いに注目し、陽子由来のイベントを除去

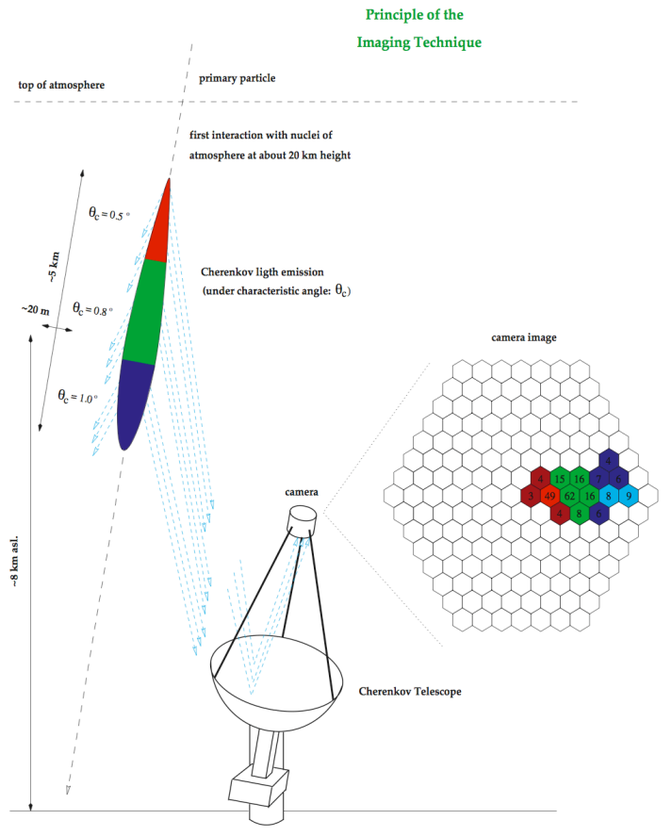


図 3.2　空気シャワーにより生成されたチェレンコフ光が望遠鏡に届くまでの様子。[14]

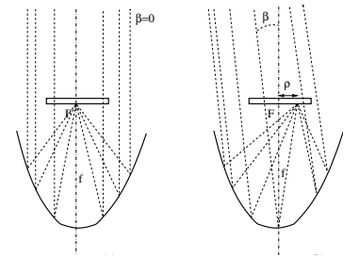


図 3.3　空気シャワーにより生成されたチェレンコフ光が望遠鏡に届くまでの様子。[14]

することでガンマ線事象を浮かび上がらせる方法を「イメージング法」という。

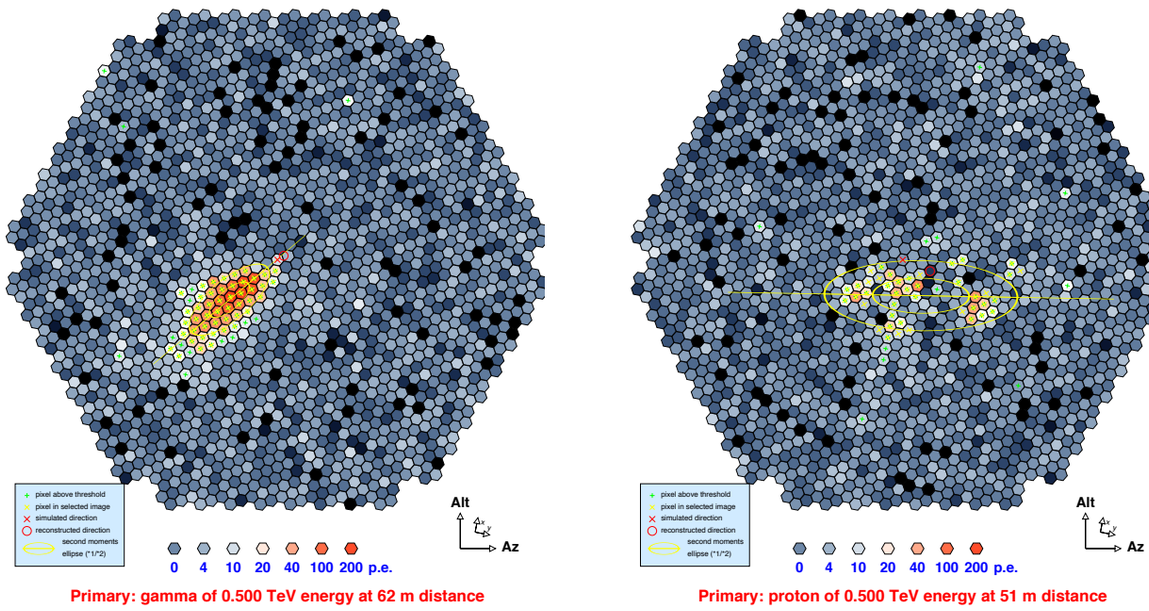


図 3.4　チェレンコフ光を LST で捉えたときのカメライメージシミュレーション例。左：500GeV のガンマ線、右：500GeV の陽子。CorsikaSimtelarray によるシミュレーションによって生成した。小　さな六角形で表された 1855 本の PMT が全体として一つの 1855 ピクセルのカメラを構成している。上の画像はで 47 ピクセルがシャワー由来として判定され。それらで検出した光子の数は 2972.6p.e.。一方で下の画像は 42 ピクセルによって 806.9p.e. を検出している。コアポジションがほぼ同じながら丁度 1/3 程度の光子数になっている。

3.2.1　イメージング法

イメージング法では、ガンマ線由来のシャワーイメージが楕円に近い形を持っていることに注目して、シャワーイメージからイメージパラメターを計算することでシャワーの特徴を抽出する方法である。カメラの視野にデカルト座標を定義し、各ピクセルを各シャワーイメージを楕円と考えて、計算により視野内での楕円の重心、楕円の長軸、楕円の長径、短径、視野中心から重心までの距離を定義する (図 3.5) (詳しい導出は Appendix 参照)。

- 楕円の重心 (c.o.g , center of gravity) $(< x >, < y >)$
- 楕円の長軸 $y = ax + b$
- 楕円の長さ（$Length$）
- 楕円の幅（$Width$）
- ガンマ線天体を期待する方向から重心までの距離 ($Distance$)
- ガンマ線天体を期待する方向から長軸までの距離 ($Miss$)

先述のようにガンマ線が持つ楕円形イメージに近い事象のみを抽出することで、宇宙線由来のシャワーイメージを除去することができる。これらのパラメターの中で特に効果を発揮するのが $Width$ パラメターである。空気シャワーのうち 99% 以上が宇宙線由来のバックグラウンド事象でありガンマ線由来のシャワーは非常

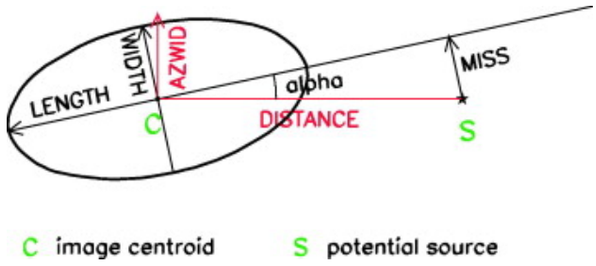


図 3.5　Hillas Parameter の定義 [1]。この図では視野中心にガンマ線天体の存在可能性を仮定した観測のため、potential source から長軸までの距離を MISS としている。また、AZWID はイメージ重心と天体方向を結んだ線からの 2 次モーメント。

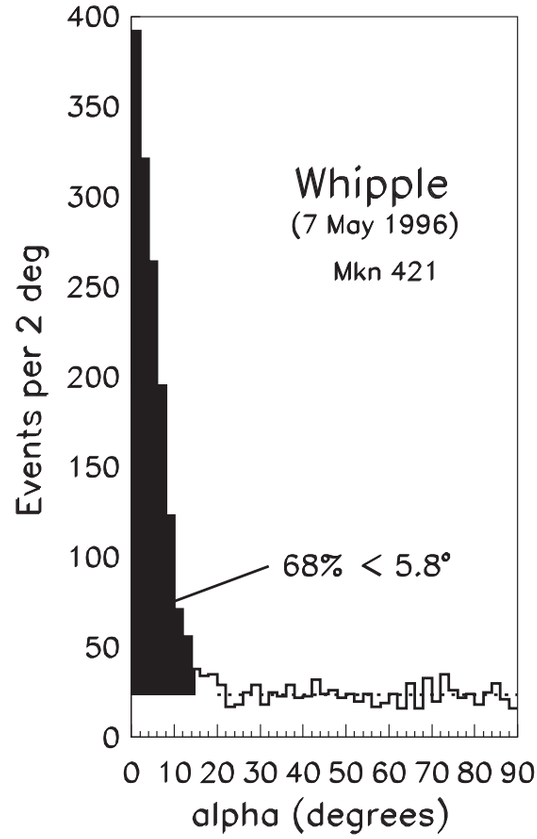


図 3.6　Mkn421 の観測結果。[7] ガンマ線らしい事象を残した上で、α の値でヒストグラムにしたもの。

に少ない割合だが、これらのパラメターを使いガンマ線の特徴に値しない事象を除去することで、98% のバックグラウンドが除去できる一方で 2/3 のガンマ線事象を保持することができる [8]。これにより 1989 年の初観測が成功した。イメージング法は、チェレンコフ望遠鏡の技術の中核をなすものである。現在の望遠鏡はのステレオ観測などにより 99.9% の確率で排除することが可能になっている。これについては後述する。

シャワーの到来方向は長軸から判断できる。シャワーイメージが楕円形を示すのははガンマ線のシャワーが一直線上に固まって発達するためで、(図 3.3) のように長軸上の片方の端が相互作用の開始点から到達する光子、もう一方の端はシャワー発達の最終段階から到達した光子ということになる。天体方向は視野の中では無限遠に近似できる一方、相互作用の開始点は約 20km 上空のため、天体方向は楕円の端からずれ、長軸の延長上になる。この事を利用して α の値でヒストグラムを得れば、α の値が小さい事象が集中的に観測されることになる (図 3.6)。

3.2.2　ステレオ法

さらに観測精度を向上し、バックグラウンドを低減する技術として、ステレオ法が導入されている。

以下のように到来方向の決定精度向上の寄与が大きい。(図 3.7) は CTA 計画で建設される予定の 4 台の

LST で観測のモンテカルロシミュレーションをしたものである。図の下左のように一辺 100m の正方形の角に当たる部分に望遠鏡を配置し、天頂を観測させた上で天頂から 500GeV のガンマ線を降らせている。コアポジションが 4 台のほぼ中心の場所の時、各カメラで捉えたイメージは図の上、中の 4 枚のようになる。下左の図中の 1 から 4 の番号に対応するイメージは順番に、上右、中右、上左、中左である。これらの図を重ねあわせると下右の図になり、全ての長軸はほぼ 1 箇所で交わっている。これがガンマ線の到来方向になる。再構成したガンマ線到来方向と実際のガンマ線天体の方向との角距離は θ と名前がついていて、(図 3.8) に示してある長軸の交点が再構成したガンマ線到来方向、黒い星印が天体方向、その間の距離が θ である。ヒストグラムを θ^2 に対してとって α プロットと同様に天体方向が定まる。このヒストグラムを $\theta^2 plot$（シータスクエアプロット）といい、で表す方法が一般的になっている。現在は $\theta^2 plot$ が一般的である。

複数台の望遠鏡で 1 つのシャワーを見ることによって 2 地点から方向がわかれば、三角測量の方法でシャワーを立体的に再構成できるという利点があり、この方法の研究が現在進められている。

さらにこの方法でコアポジションも簡単に求めることができる。コアポジションはインパクトポイントとも言い、シャワー軸が地上と交わる点のことである。先述のようにシャワーイメージの長軸の片方の延長上は粒子の到来方向だが、これと反対の方向はシャワーの発達方向である。そのため、4 つのイメージを元に反対方向に地上をたどると交点が結べて、コアポジションに値する。

バックグラウンドの低減効果は、トリガの発行を望遠鏡間でコインシデンスをとることで発揮される。チェレンコフ光のフラッシュは非常にかすかなため、星の光や銀河面の塵の反射、流れ星、大気蛍光、飛行機や人工衛星を始め、様々な夜光がノイズとしてトリガされてしまう。単体の望遠鏡では、予め星の分布を確認したり、隣接するピクセルが信号を出しているか判断することでノイズによるトリガを防いでいるが、望遠鏡間でのコインシデンスを取ることでさらなるノイズ由来トリガ防止が期待できる。

また、宇宙線由来のシャワーに対しても効果がある。宇宙線由来のシャワーは先述のように π^0 を発生させて、その 2γ への崩壊から電磁シャワーを引き起こしているため、ガンマ線由来の電磁シャワーと区別の付かないイメージが紛れ込んでしまう。一方で宇宙線由来のシャワーは大量のミューオンを発生させる。ミューオンは寿命の伸びにより地上付近でもチェレンコフ光を出すことでリング状のイメージが検出されることがある。このイメージは複数の望遠鏡の中に入らないため、宇宙線由来のシャワーとして判定できる。

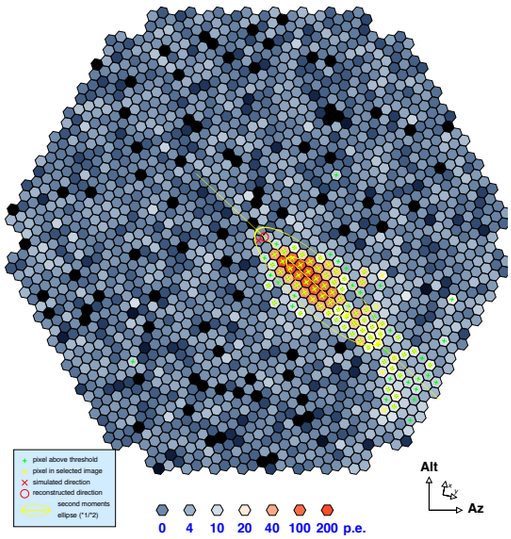
3.2.3 Displacement 法

現在ではステレオ法を発展させた Displacement 法 (図 3.8) が用いられる。Displacement 法では、各シャワーイメージの長軸を到来方向を判断する代わりに、estimated distance を使う。estimated distance はコアポジションを元に得る値で、コアポジションが望遠鏡から遠いほどカメラの中では長軸が伸びたシャワーになることを利用して求めている。長軸上を estimated distance の長さだけ辿って到達した点を平均することで、ガンマ線の到来方向を判断する。望遠鏡の性能向上により、この方法の方が精度の良い方向決定ができるようになっていて、特に長軸同士の角度が小さい場合に大きな効果を発揮する。

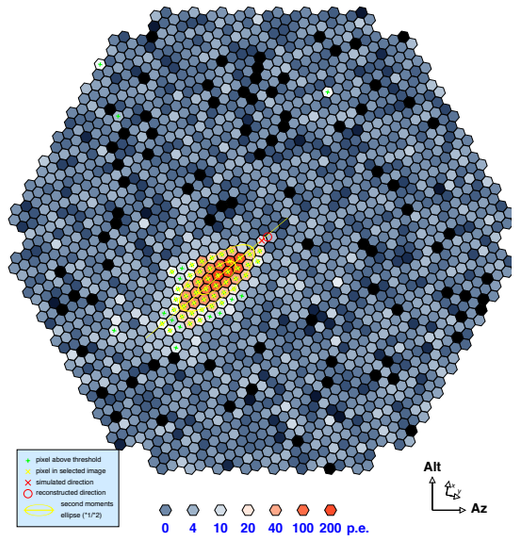
この Displacement 法で再構成したシャワー到来方向と実際のガンマ線天体の方向との角距離を θ として、この値で $\theta^2 plot$ を表す方法が一般的になっている (図 3.9)。

3.2.4 バックグラウンド観測

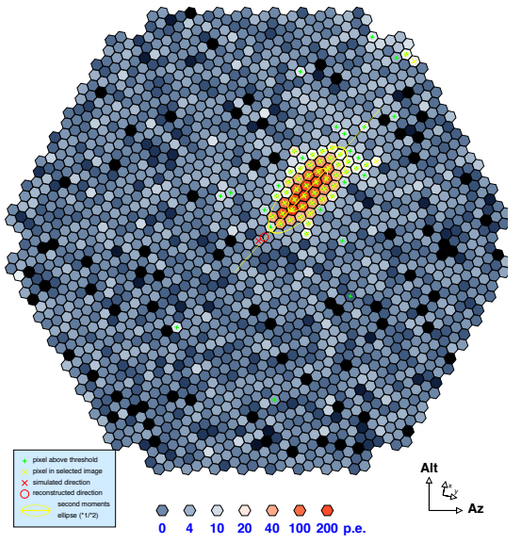
バックグラウンドは完全には無くならない。そこで目標とする天体方向の観測を行うと同時に、ガンマ線の到来が無いと判断する方向も観測を行う。これらの観測方向をそれぞれ on source、off source と呼んでいる。



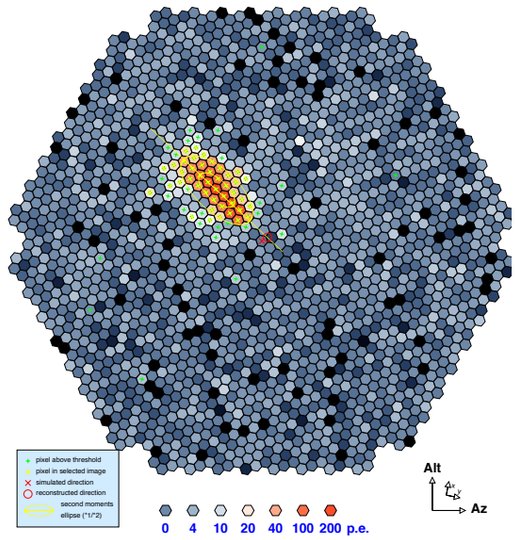
Primary: gamma of 0.500 TeV energy at 61 m distance



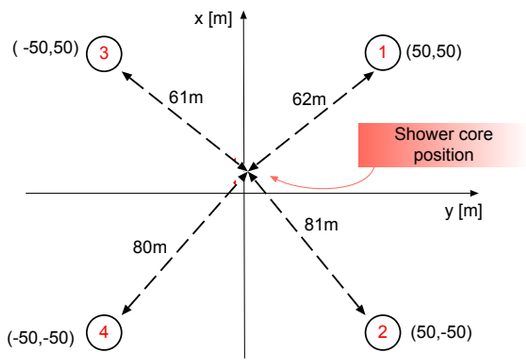
Primary: gamma of 0.500 TeV energy at 62 m distance



Primary: gamma of 0.500 TeV energy at 80 m distance



Primary: gamma of 0.500 TeV energy at 81 m distance



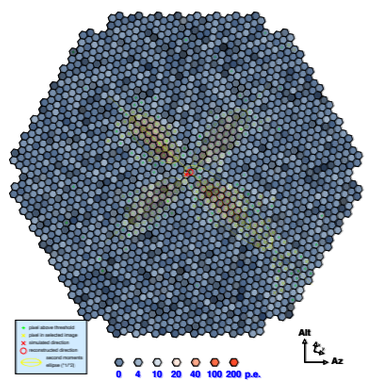


図 3.7　4 台の LST を用いて一つのシャワーを観測した場合の、各望遠鏡焦点面のピクセルイメージ。シミュレーションによる。

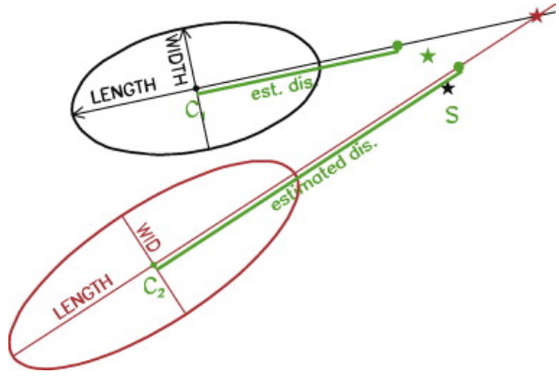


図 3.8　2 台の望遠鏡で観測した場合のステレオ法。ガンマ線天体の存在すると思われる方向が黒い星印、これと 2 つの楕円の交点との距離を θ とするのが初期のステレオ法、est. dis. の平均から求めた点との距離を θ とするのが displacement 法 [1]。

off source で得られたデータはバックグラウンドのみと仮定すれば、on source で得られた結果との差がガンマ線のシグナルとして浮かび上がる。この方法でより信頼性の高い方法は on source と off source のデータから有意度を求める方法である。それぞれの観測時間 $t_{\rm on}$、$t_{\rm off}$ で、$N_{\rm on}$ と $N_{\rm off}$ のイベントを得た時、有意度は Li&Ma の式して知られ、

$$S = \sqrt{-2\ln\lambda} = \sqrt{2}\left\{ N_{\rm on}\ln\left[\frac{1+\alpha}{\alpha}\left(\frac{N_{\rm on}}{N_{\rm on}+N_{\rm off}}\right)\right] + N_{\rm off}\ln\left[(1+\alpha)\left(\frac{N_{\rm off}}{N_{\rm on}+N_{\rm off}}\right)\right]\right\}^{1/2} \tag{3.4}$$

となる [10]

この有意度を用いて表した例が (図 3.9) である。銀河中心の観測結果で、左の θ^2 plot では、観測方向を 0 度とした時に、そこから 0.1 度以内、つまり $\theta^2 = 0.01$ 以内を on region としている。グラフの中で灰色に塗られた領域が off source のデータを使って求めた領域で、on source では明らかなイベントの超過が見られる。これを Li&Ma の式により有意度を求めると 10.87 である。右のスカイマップでは、有意度によってコントラストを付けて表示することで、ガンマ線天体が画像として見ることが可能である。

現在バックグラウンド観測の方法はいくつかあり、off source の選択の仕方が異なる。一般的な off source の観測方法は wobble 法と言い、on source を視野中心からずらして観測し、視野中心を挟んで on source と反対側を off source とするものである。また、reflected background model では、視野内で数箇所のポイントを決めておき、一定時間ごとに on source にするポイントを変えていく。その他に、on source の周りを囲むように off source を選ぶ ring background model という方法もある。それぞれ利点と欠点があるので、状況に合わせて利用されている。[3]

3.2.5　エネルギーの見積

(図 3.11) のように、コアポジションが望遠鏡から約 100 ~120 m 以内の位置であれば降り注ぐ光子数はほぼ一定である。そのため、検出された光子数はエネルギーに依存することになる。これを利用して、イメージと

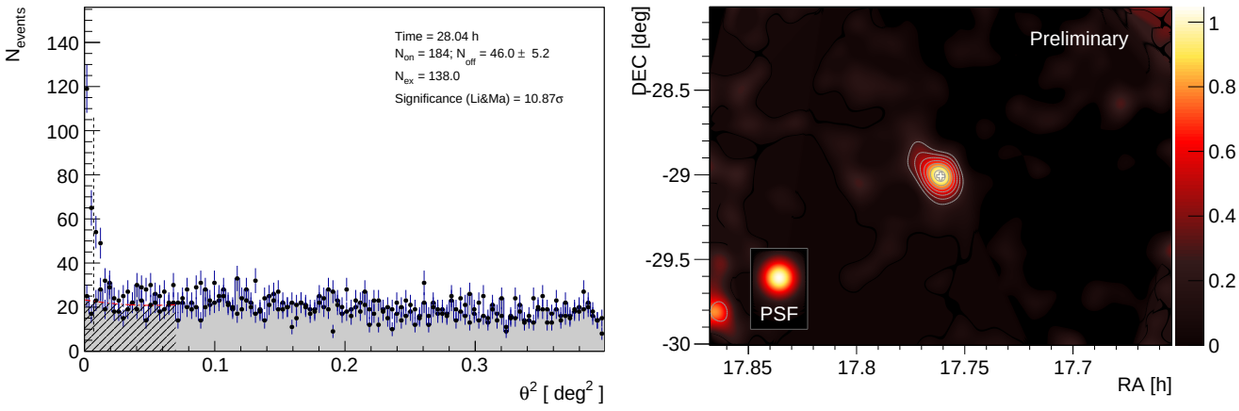


図 3.9　バックグラウンドを利用して銀河中心の Sgr A*を観測した例。左：θ^2 plot 、右：スカイマップ。

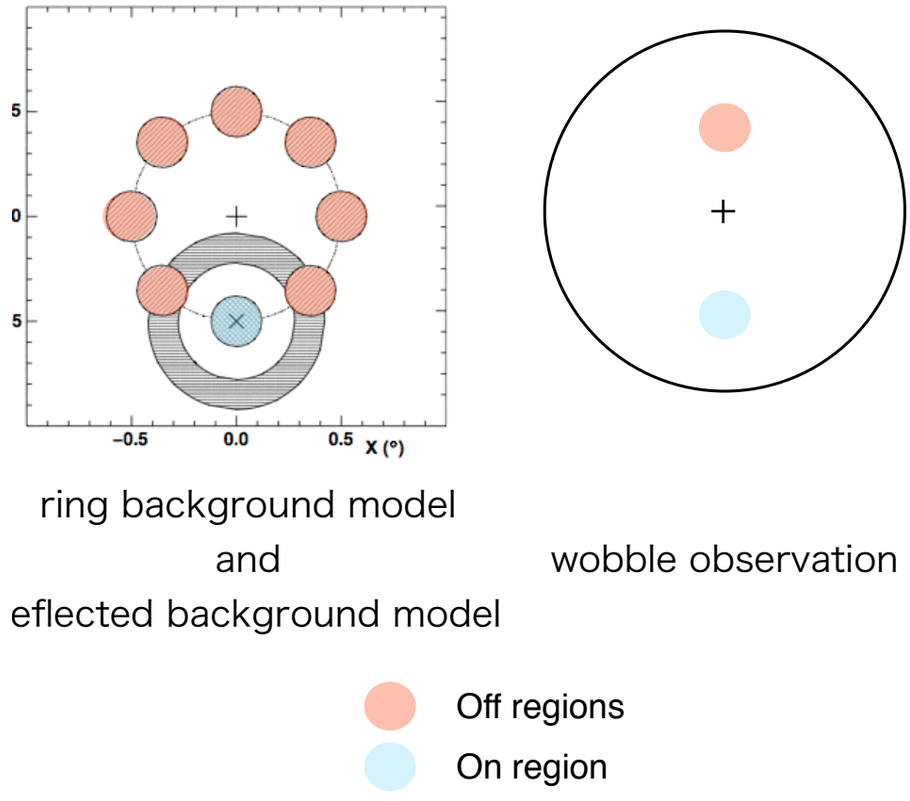


図 3.10　バックグラウンド観測の種類。左：ring background model（横線領域）と reflected background model（斜線赤の領域）[3]、右：wobble model。

して選択されたピクセルで検出された光子数 N_i を積分し、

$$Size = \sum_{i}^{k} N_i \tag{3.5}$$

これをサイズパラメタとよび、シミュレーションと合わせて LUT（look up table）を作ることで到来ガンマ線のエネルギー決定に大きな寄与をしている。[14] ガンマ線のエネルギースペクトルはここから求められる。

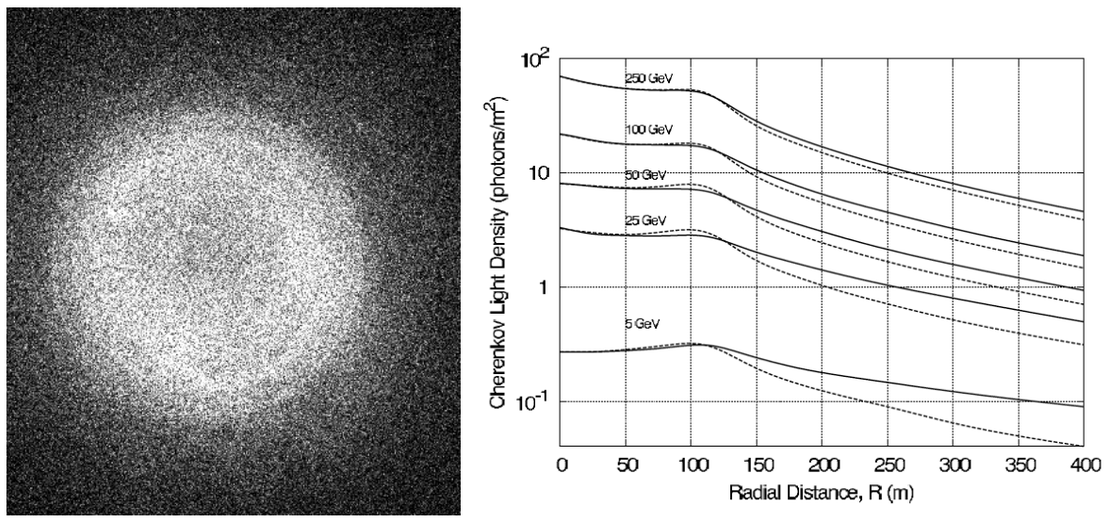


図 3.11　左：地上に到達したチェレンコフ光の分布のシミュレーション。円形に白い部分がチェレンコフ光で、半径は 150m ほどである。右：コアポジションからの距離に対するチェレンコフ光子の密度。[14]

第 4 章

Cherenkov Telescope Array 計画

現在、次世代の IACT の計画（Cherenkov Telescope Array、CTA）が進行中であり、地上ガンマ線望遠鏡初の公開天文台となる予定である。世界 28 カ国から約 1200 人の研究者が参加しており、日本からは 100 名以上の研究者が参加している。現在の IACT の 10 倍の感度を、数十 GeV から百 TeV という 5 桁に渡り達成することを目標としていて、1000 を超える天体の観測が期待されている。三種類のエネルギー領域に特化した設計思想で望遠鏡が計画されている。LST（Large Size Telescopes）は低エネルギー側を、MST（Mid Size Telescopes）は百 GeV から数十 TeV を、SST（Small Size Telescopes）は高エネルギー側を狙う。とくに日本は全体の 15% を貢献し、LST においては中心的な役割を果たすことが期待されている。LST は 2016 年には 1 台目、2020 年までに南北に 4 台ずつ、計 8 台の建設を予定している。[17] [5]

4.1 目標性能

- 観測可能エネルギー
  20GeV から 100TeV の広いエネルギー範囲での観測を目指す。これを実現するために、エネルギー領域を 3 領域に分け、それぞれに特化したデザインの望遠鏡を開発する。
- ガンマ線強度に対する感度
  現在稼働中の IACT よりも 10 倍向上させる。これを実現するために低エネルギー領域は口径の大きい望遠鏡を、高エネルギー領域は多くの望遠鏡からなるアレイを建設する。とくにエネルギー領域 300 GeV − 3 TeV は 50 時間の観測で 1mCrab (10^{-14}ergcm^{-2}s^{-1}) の感度を達成する。
- 角度分解能
  現在稼働中の IACT よりも 3 倍向上させる。現在の角度分解能は 0.1 度 (6 分角) 度程度を 1-2 分角まで向上させる。これを実現するために、多数の望遠鏡で一つのシャワーを捉えられるよう配置して、平均で 6 台の望遠鏡によりイメージをシャワーの再構成を行えるようにする。
- 全天観測
  北半球と南半球それぞれに観測ステーションを持つことで全天を観測する。

CTA の目指す感度は、最も明るい定常ガンマ線天体であるかに星雲の 1000 分の 1 の明るさ、また活動銀河核の最も明るいフレアーの 10000 分の 1 に相当する。これにより銀河内の Crab(距離 1kpc) 程度の明るさをもつ定常天体は 30kpc まで観測可能となり、銀河系全体が見渡せることにな る。

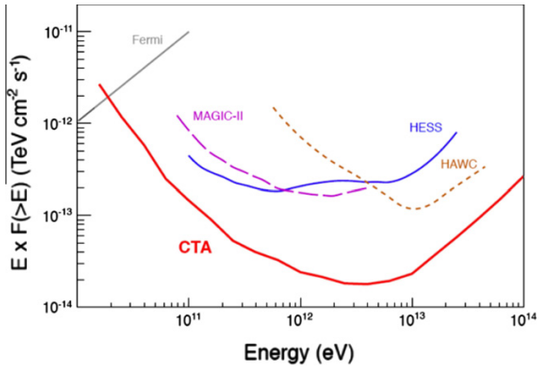


図 4.1 CTA と現在の主なガンマ線望遠鏡との積分感度比較。Fermi と HAWC は 1 年間、その他の望遠鏡は 50 時間の観測時間を仮定。[17]

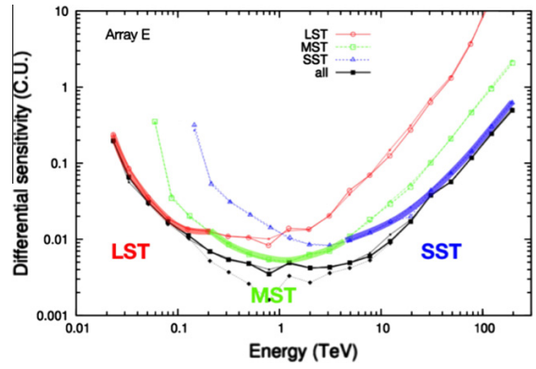


図 4.2 LST、MST、SST の微分感度。それぞれ低、中、高エネルギー領域を得意とする。50 時間の観測時間を仮定。[17]

図 4.3 サイトの候補として挙げられている地域。[5]

4.2 サイト

CTA は全天を観測するため、北半球、南半球二つのステーションからなる。20GeV-100TeV のガンマ線観測に最適な高度 1000m-3000m で、空気が乾燥し、風速が低く、夜間の晴天率が 60-80% と高く、かつ、人口光が夜光に比べ十分に低いことが必要である。これらの条件と、アクセス、インフラ、政治的安定性等の条件を考え、北半球の 候補サイトは、カナリア諸島 (La Palma、Tenerife、北緯 28 度、高度 2200m)、メキシ コ (San Pedro Martir, Baja California、北緯 31 度、高度 2800m)、アリゾナ (Yavapai, Meteo Crator 北緯 35 度、高度 1200m) でサイト調査が継続中である。南半球の候補サイトは、 ナミビア (Aar、南緯 26 度、高度 1600m)、チリ (Armazones, 南緯 24 度、高度 2400m) が選択され 、2015 年には両国と交渉がなされる。

表 4.1 南半球の望遠鏡台数。左：南半球、右：北半球。

望遠鏡サイズ	台数
大	4
中	25
小	70

望遠鏡サイズ	台数
大	4
中	15

4.3 望遠鏡

20GeV-100TeV という広いエネルギー領域でのガンマ線観測を実現するために、このエネルギー領域の中の高、中、低エネルギー領域それぞれに特化した望遠鏡をデザインしている。

- 大口径望遠鏡
  20GeV から 1TeV の低エネルギー領域をカバーする。鏡の口径は 23m である。低エネルギー領域のガンマ線は、シャワーにより発生するチェレンコフ光の光量が小さいため、集光面積を特に大きく取ってあり、さらに高反射率・高集光効率のの鏡面、高光検出効率の検出器を揃える対策を施す。また、夜光の影響を抑えてかすかなシャワーでもはっきり捉えるために、集光される時の光の同時性を重要視した光学設計になっている。
- 中口径望遠鏡
  100GeV から 10TeV の中エネルギー領域をカバーする。鏡の口径は 12m であり、HESS、VERITAS 望遠鏡と同サイズで、これを 20-30 台配置する。銀河面スキャン、全天スキャンを行うため、視野を出来るだけ広くする、Davis-Cotton Optics デザインを採用し一様なイメージを得られるようにしてある。
- 小口径望遠鏡
  1TeV から 100TeV の高エネルギー領域をカバーする。鏡の口径は 4m であり、高エネルギー領域のガンマ線のシャワーが生成するチェレンコフ光は明るいため、集光面積を大きく取らない代わりに、到来頻度の低さを補うために多数配置して感度を上げる。ガンマ線の最高エネルギー領域を観測することで、宇宙線のスペクトルが変化する 1000TeV 領域までの宇宙線加速源と加速機構の研究をねらう。高エネルギー領域は伝播中に宇宙赤外線背景放射を受けて減衰をするため、銀河系内のガンマ線源がターゲットとなる。

4.4 アレイ配置

銀河面をひろく観測できる南半球のステーションでは、広いエネルギー範囲を覆うことが必要のため、大 (23m)、中 (12m)、小 (4m) 口径の三種類の口径の望遠鏡を設置する。そのため南半球のステーションは約 $10\mathrm{km}^2$ の広さになる。

一方銀河面の観測が限られる北半球のステーションにおいては、低 いエネルギーレンジ (20GeV ～10TeV) に観測の重きがおかれ、大 (23m)、中 (12m) の二種類の望遠鏡群から構成され、約 $1\mathrm{km}^2$ の広さになる。

(図 4.7) は候補となっているアレイの配置図である。

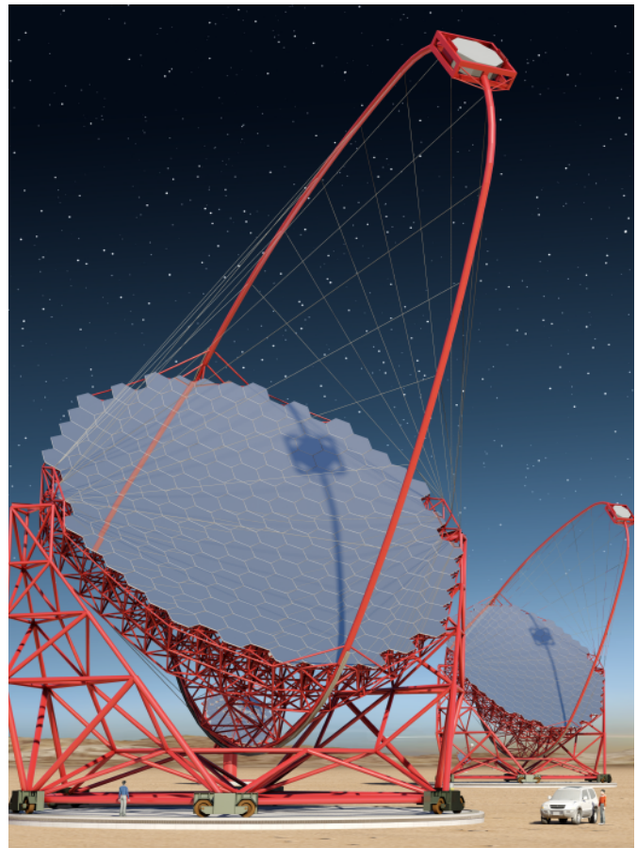


図 4.4 LST[5]

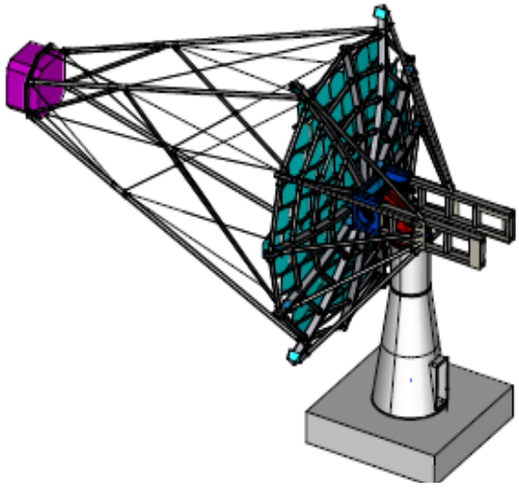


図 4.5 MST[5]

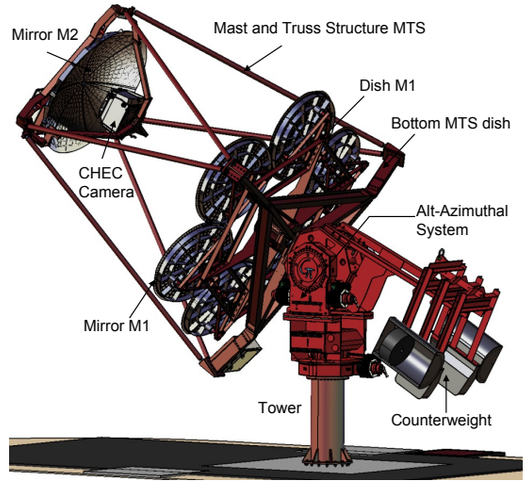


図 4.6 SST の一つ、日本が開発参加している GCT(Gamma-ray Compact Telescope)[5]。他にも数種類のデザインが考えられており、複数種類が配置される予定である。

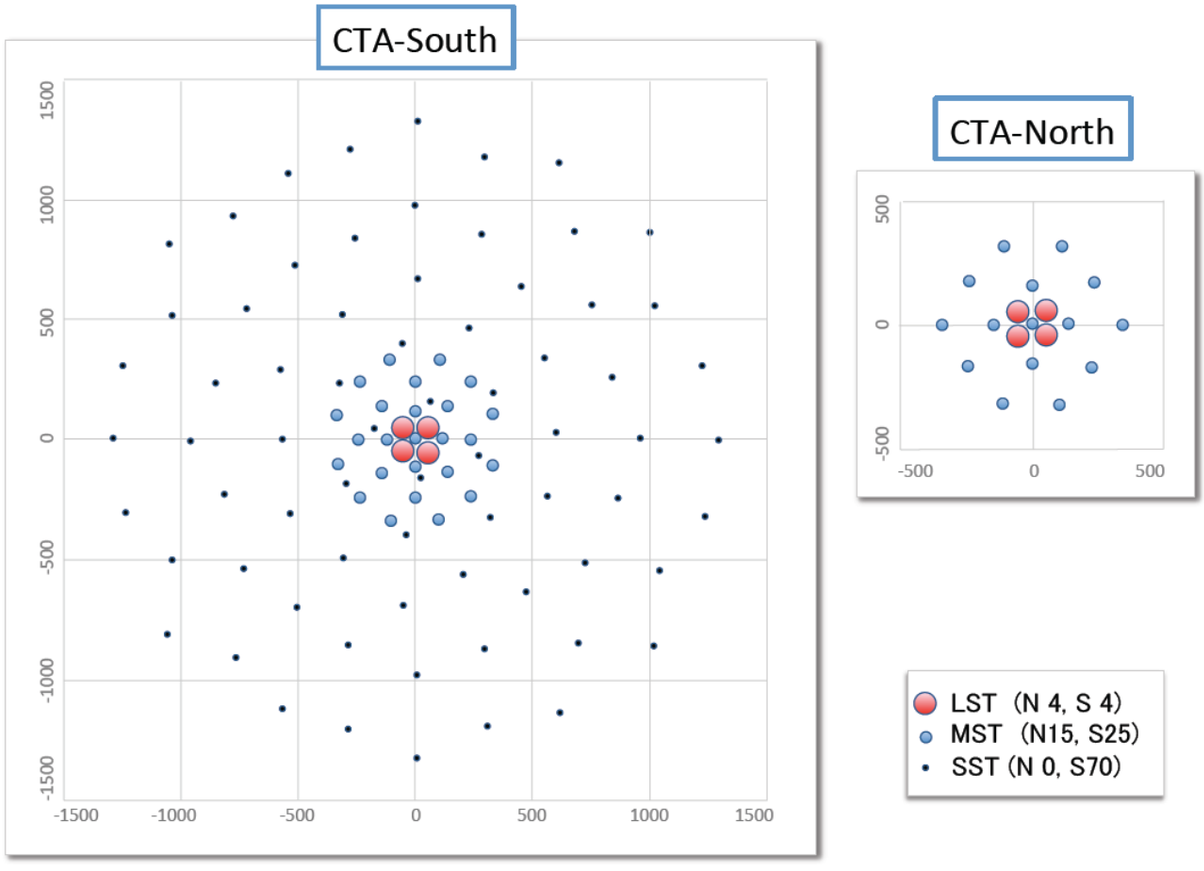


図 4.7　候補となっているアレイ配置。縦軸、横軸共に単位は m。赤丸が LST、青丸が MST、黒丸が SST の位置を表している [5]

第 5 章

LST データ収集システム

LST は、CTA の狙うガンマ線エネルギー領域の中でも低エネルギー側のガンマ線検出に特化した望遠鏡である。ガンマ線が低エネルギーであるほど大気中で発生させるチェレンコフ光量が少なくなるので、検出感度を最大限に確保するためにはトリガ閾値をを可能な限り下げなければならない。閾値を下げると、シャワー事象がより多くトリガされるだけでなく、夜光によるノイズの寄与も無視できなくなり、取得するデータ量は膨大になる。そのため、これに対応できるデータ収集システムが必要となる。

この章では LST データ収集システムの概要、データ収集システムに関わる各コンポーネント、データ収集システムに対する要求性能、本実験でのデータ収集プログラム開発方針について述べる。

5.1 データ収集システムの概要

データ収集システムは、カメラにトリガがかかり各 PMT から取り出された波形情報を収集し、ひとつのイベントデータとして結合して保存するシステムである。LST は各サイトで 4 台ずつ建設される予定で、そこからのデータ収集は CTA のアレイ全体での運用も視野に入れつつ、LST のエネルギー領域に合わせた独自の運用が考えられている。そのため LST からデータ収集はまず LST データ収集システムを通して行われ、その後にアレイ全体のデータ収集と連携される (図 5.1)。データ収集はおおまかに以下の各段階に分かれる。

- データの取得
  データの取得段階では、トリガをかけることによって望遠鏡焦点面のカメラを構成する全ピクセル、つまり全ての PMT から信号波形が読出され、ネットワークでの転送を経てカメラサーバー（DAQ を担当するサーバー）に集められる。トリガはハードウェアベースで行われ、各望遠鏡のみで判断をするローカルトリガモードと、複数の LST でコインシデンスの判断を行うステレオトリガモードがある。また、LST 以外とも連携したトリガモードもあり、アレイトリガと呼ばれる。アレイトリガはソフトウェアベースのトリガで、トリガ情報が望遠鏡からアレイトリガシステムに送信されて判定される。これらのトリガについて詳細は後述する。
- データの結合
  データの結合段階では、集めたデータを同一のトリガタイミングに基づくデータ同士で結合することによって、ひとつのイベントデータを作る。
- 低レベル解析
  低レベル解析で行われる手続きは大きく分けて 2 つある。
  1. イベントのプロセッシングとフィルタリング

2. アレイトリガを用いたステレオイベントの選別

これによりピクセルごとのデータ選別とイベント毎のデータ選別を施しデータ量を低減することで、後段の処理の負荷を抑える。また、監視システムに解析結果を提供する。

- 監視システムへの送信

  データの結合と低レベル解析を経て生成された生データは監視システムに送られる。監視システムではシステムの健全性を監視するだけでなく、リアルタイム解析を行って大まかな観測結果をその場で出しガンマ線の兆候を即時に見ることで、外部天文台へのアラート送信や観測スケジュールの判断に利用する。

- データの保存

  生データはオンサイトアーカイブにも送られる。オンサイトアーカイブはデータの一旦現地保管を行う。

以上の様に、LST データ収集システムは望遠鏡のカメラ部分からのデータを受け取り、アレイトリガからの情報も得てデータを選別し、結果を監視システムとオンサイトアーカイブに送信するという役割を担っている。これらはすべてアレイコントロールシステムによって統合制御される。

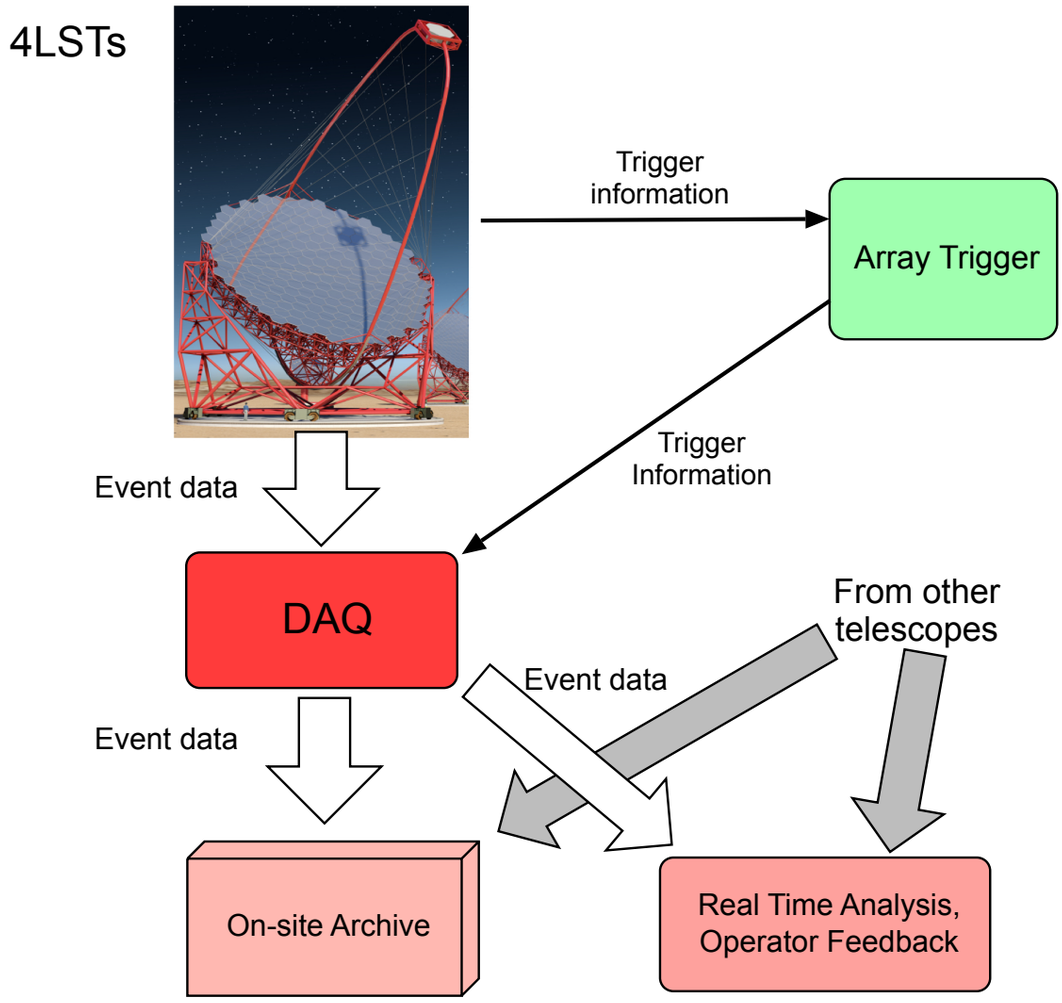


図 5.1 データ収集システムを中心にしたブロック図。

5.2 データの読出し（カメラ）

LST の望遠鏡焦点面には 1855 本の光電子増倍管（Photo Multiplier Tube, PMT）が並び、1 本が 1 ピクセルに相当する望遠鏡のカメラを構成している。この PMT は 7 本ずつ束ねられ、”Dragon”と呼ばれる読出ボード (Front End Board , 以下 Dragon FEB) に接続されている。Dragon FEB は PMT の電圧情報をサンプリングし、トリガがかかると波形情報をデジタル化して TCP/IP 通信によって送信する。ここでは PMT に入射した光がデジタルデータとなって取り出され、送信されるまでの「データの読出し」について説明する。

5.2.1 読み出し回路 ”Dragon” FEB

PMT から信号波形は、読出回路「Dragon FEB」により (図 5.2) のような処理でデータが取り出される。以下にその流れを簡単に説明する。

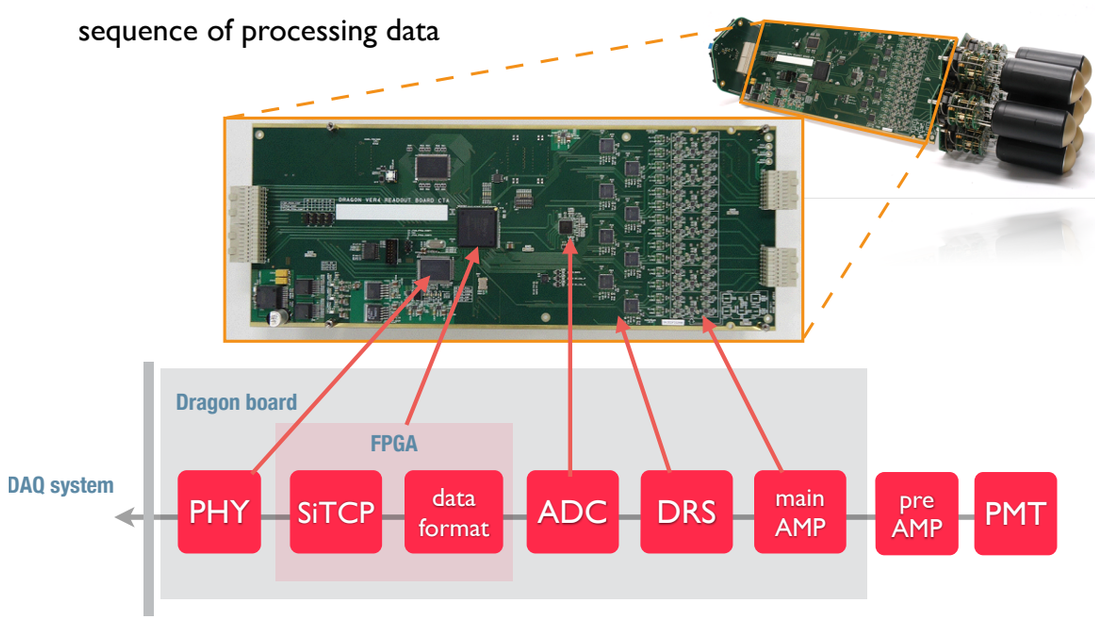


図 5.2　Dragon FEB によって信号波形がデータとして読み出されるまでの流れ。

PMT7 本からの信号はそれぞれプリアンプを通して 1 枚の Dragon FEB に入力される。Dragon FEB ではそれをトリガ回路に渡す一方で、メインアンプによって High gain と Low gain の 2 種類の増幅を行って取り出した後、それぞれを DRS4 というキャパシタアレイで構成された RingBuffer にてサンプリングし、一定時間保存される。DRS4 は 1 ピクセル当たり、High gain と Low gain それぞれの 4096 time slice ぶんが一時的に保存される。現段階では 1GHz のサンプリングレートのため、4 μs の深さが保存できる。

トリガ回路に分岐して送られた電圧情報はトリガ判定に使われ、閾値を超える光子の塊が隣り合うピクセル同士で広がって存在するとトリガをかける。

トリガ生成回路によってトリガが判断されると、FPGA がそのトリガ信号を受け、DRS4 のキャパシタに対し、サンプリングを停止させた上でデータ読出しを開始する。この際 FPGA はトリガに該当する時刻にさかのぼって読出し開始位置と読み出す time slice の深さを指定し、DRS4 からは指定された各キャパシタの電圧値が順次読み出され、ADC によってデジタル変換が行われる。

その後、FEB 上にてデータ整形された上で Ethernet 通信により送信される。送信は PHY チップが担当する。Ethernet 通信技術は FPGA 内に KEK により開発された SiTCP 技術を埋め込むことで実現されている。

5.2.2　データのサイズ

光子到来によるパルスは 2 - 3 ns の時間幅となり、これを含む信号波形を取り出すために 30 ns に相当するデータを読み出す。データ構造の詳細は未決定だが、現段階では 1GHz サンプリングで 30 time slice（30 ns の長さに相当）の情報をデジタル化して取り出した場合、FEB1 枚が 1 回のトリガで取り出すデータの大きさは、976 Byte になる (図 5.3)。

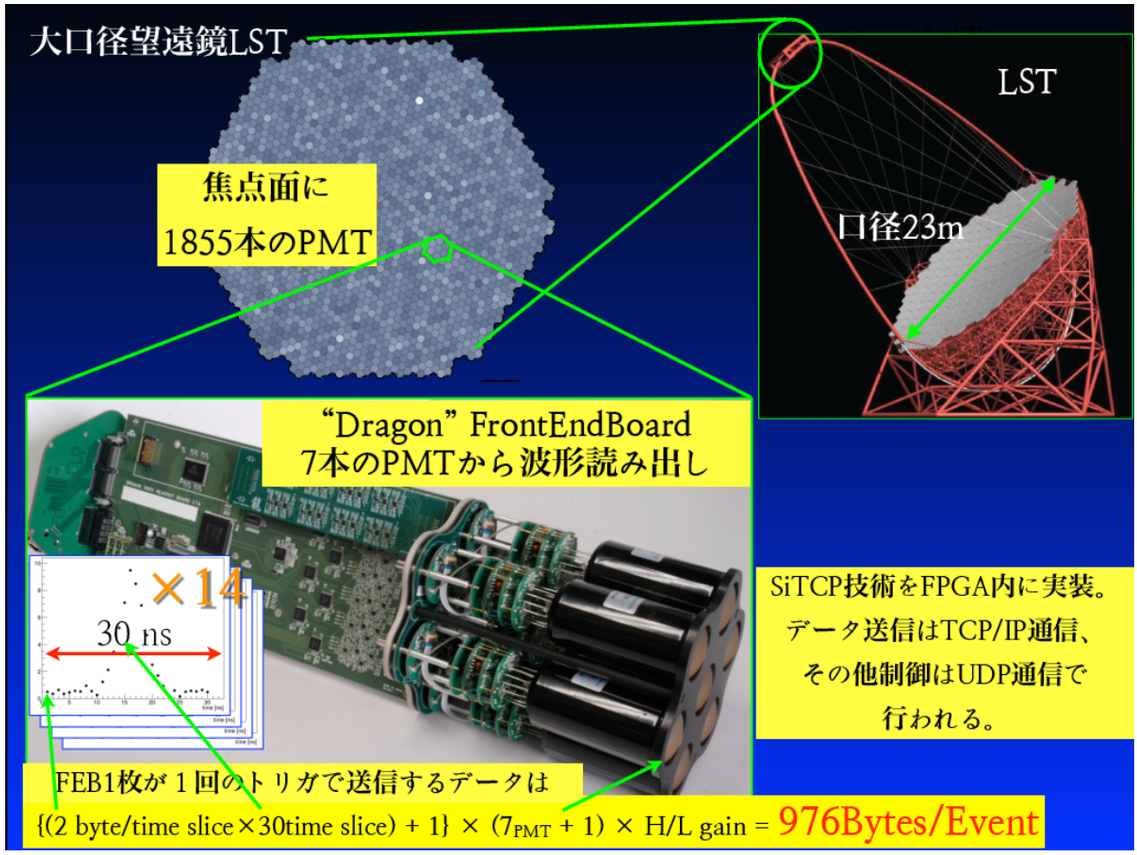


図 5.3　観測するチェレンコフ光とデータの関係

5.3　ローカルトリガとステレオトリガ

先述の通り、LST でのトリガは単体の LST でトリガをかけるローカルトリガと複数台の LST でコインシデンスを取るステレオトリガの 2 種類のトリガモードがある。

5.3.1　ローカルトリガ

望遠鏡単位でのトリガ生成回路は level0（L0）と level1（L1）の 2 段階に分かれている (図 5.4)。L0 では FEB でサンプリング回路と分岐して渡された 7 本の PMT の電圧情報を足しあわせ、L1 に分配される[*1]。L1 では Analog Trigger Backplane(図 5.5) を通じて隣同士の FEB で L0 信号を交換しあい、隣からの L0 信号と自分の L0 信号を組み合わせて信号を足し、閾値からローカルトリガを判定する。組み合わせは 2 クラスタモードから 4 クラスタモードまで 3 種類があり、観測条件に応じた組み合わせ方を設定できるようにする (図 5.6)。

発生したトリガ信号は、一旦 Trigger Interface Board（TIB）(図 5.8) に送られた後にカメラ全体に配布され、各 FEB からデータが読み出される (図 5.7)。

[*1] ここでは各 PMT の信号に対して上限の電圧値を設けることでアフターパルスの効果を抑える。クリッピングという。

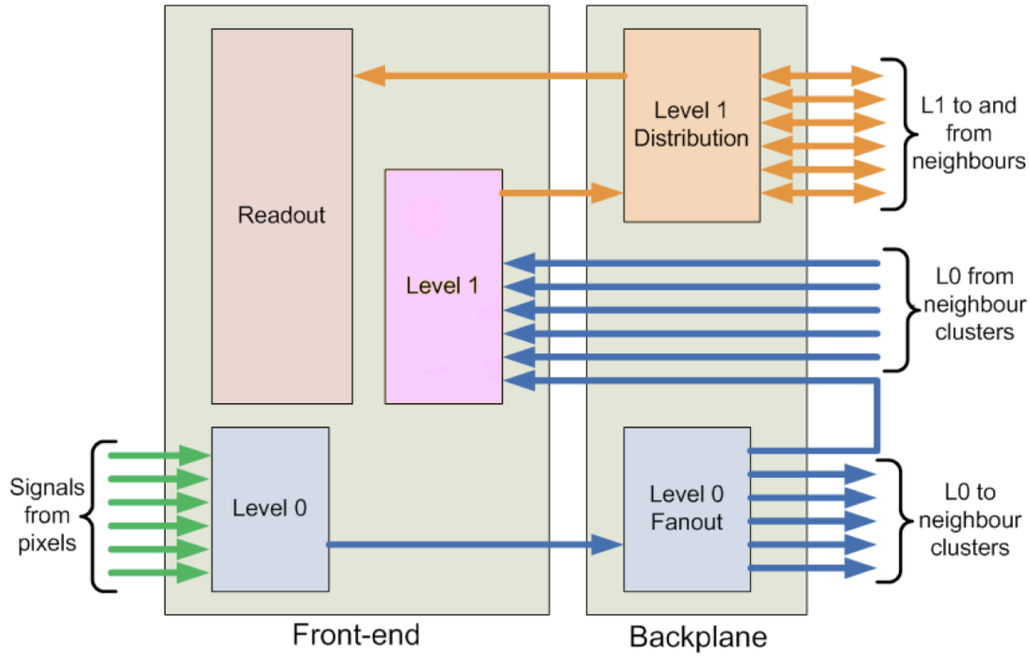


図 5.4　ローカルトリガ生成のダイアグラム。L0、L1 共に FEB に実装され、Backplane によって両隣と信号を交換する。また、L1 でトリガが判定されるとカメラ全体にトリガ信号が配布される。緑矢印は PMT からの信号、青矢印が L0 を経た信号、橙矢印が L1 を経た信号（トリガ）。

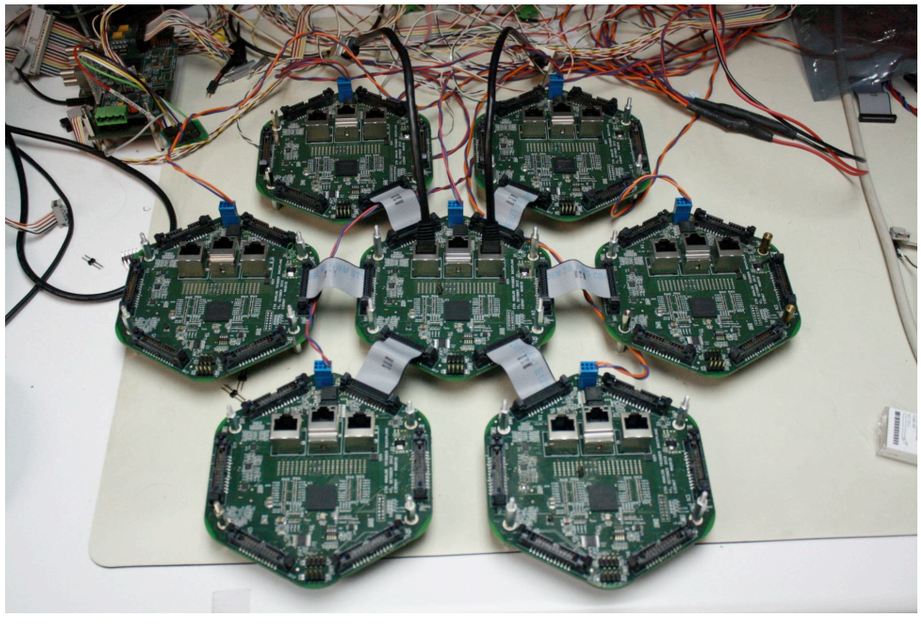


図 5.5　Analog Trigger Backplane

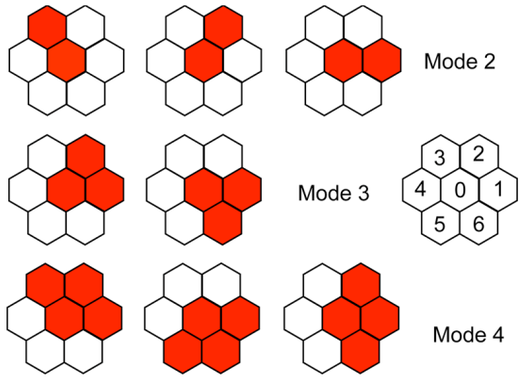
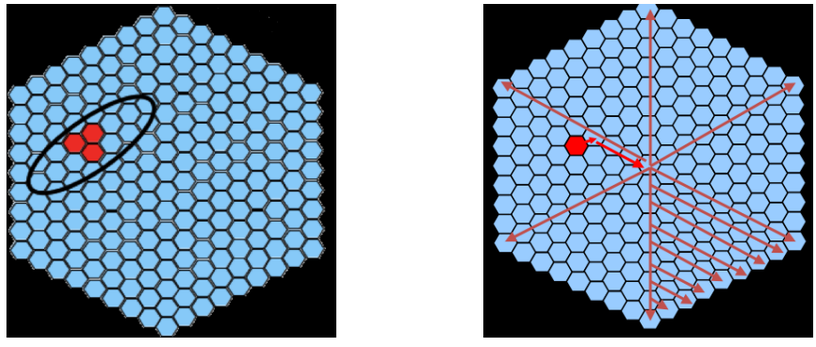


図 5.6 L1 トリガの決定方法。それぞれの六角形が FEB に接続された Analog Trigger Backplane を表している。右端の図の六角形の中に表示された数字は 0 が自分の FEB、1 から 6 が L1 のために利用する隣の FEB。



(a) Trigger decision (b) Trigger distribution scheme

図 5.7 L1 トリガの配布。小さい六角形が FEB に接続された ATB を表しており、全体で一つのカメラを表す。(a) のような FEB の組み合わせでトリガが判定された場合、(b) のようにトリガの配布がカメラ全体に行われる。

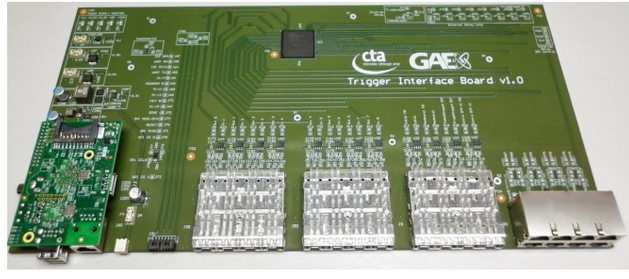


図 5.8 Trigger Interface Board

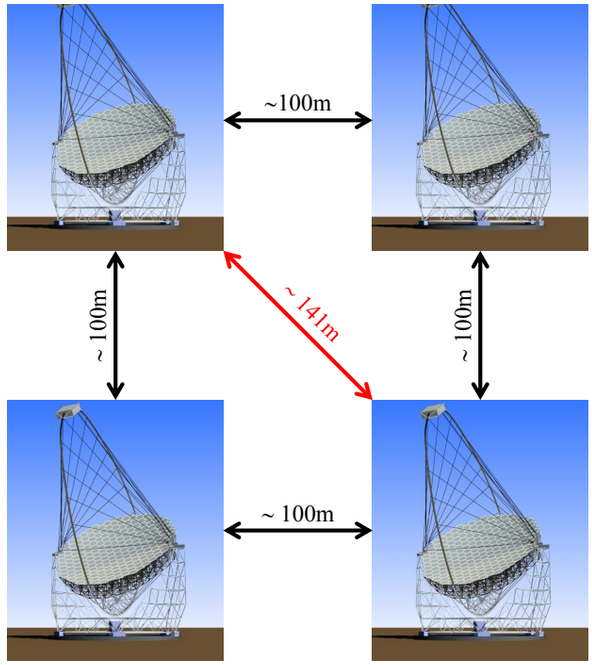


図 5.9　ステレオトリガの配布

5.3.2　ステレオトリガ

ステレオトリガはローカルトリガを土台にして機能する。TIB に送られたトリガ信号は LST の間で相互に送信し合い、コインシデンスを判定する (図 5.9)。コインシデンスの判定後はローカルトリガ同様に TIB から各 FEB にトリガ信号が配布され、データが読み出される。

5.4　トリガレート

予想されるトリガレートは、空気シャワーのシミュレーションツール「Corsika」と、望遠鏡のシミュレーションツール「SimTelarray」を用いて見積もられている。その結果、20GeV 程度のエネルギー閾値を目指す場合、トリガレートは 15kHz 程度になると見込まれており、この値が要求値に設定されている。より低エネルギーのガンマ線を捉えるにはトリガの閾値を下げなければならないが、閾値を下げると夜光によるトリガが支配的になってしまうため下限がある。夜光によるトリガと宇宙線のシャワーによるトリガが同程度になる閾値が、適切な下限値である。ただしこのような低閾値に設定すると、実際の夜光は変動が激しいためにトリガレートも大きく変動してしまう (図 5.10)。

5.5　データ転送（ネットワーク）

20kHz のトリガレートを仮定した場合、データ転送レートは FEB1 枚からは 156Mbps に なる。望遠鏡 1 台は 1855 本の PMT からのデータを 265 枚の FEB によってまとめているため、望遠鏡 1 台から のデータは、40Gbps という超高速転送が要求される。

FEB から 150Mbps の送信レート、また、受信側では FEB265 枚分、40Gbps の転送レートを実現するため、

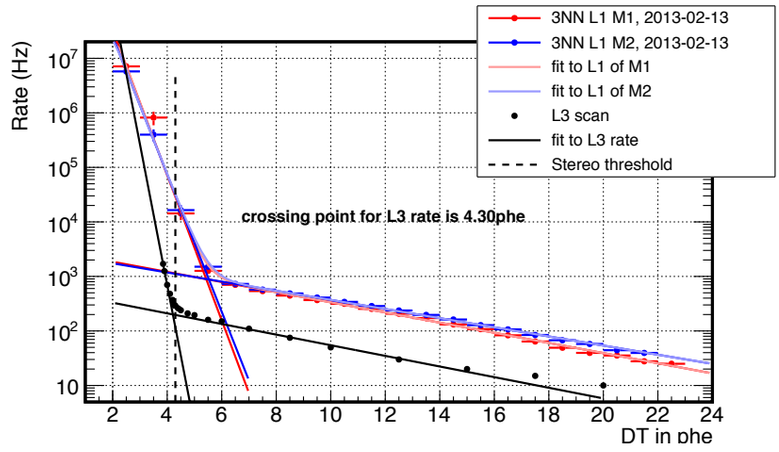


図 5.10　トリガ閾値に対するトリガレート変化の例 [4]。MAGIC 望遠鏡による観測結果で、3NN L1 は隣り合ったピクセルを判断してトリガをかける方法を、赤と青で色分けされた M1 と M2 はそれぞれ MAGIC I 望遠鏡と MAGIC II 望遠鏡を表している。直線は解析的なフィット。左側の直線は夜光由来のトリガ、右側の直線は宇宙線由来のトリガ。黒の点はステレオ法を採用した場合のもの。

(図 5.11) の様なシステム構成を予定している。

FEB からスイッチ[*2]に 1Gbps イーサネット通信で接続する。スイッチは、1 台あたり約 45 本の接続を束ねる。 スイッチからは 10Gbps の SFP+ 通信によってカメラサーバーまで接続する。よってカメラサーバーは 265 枚 の FEB からのデータ全てを受信し、データの結合処理などを行ったあと、監視データ、保存用の観測データとしてそれぞれ送信される。カメラサーバはデータ受信のために相当の負荷がかかることが予想されるため、複数台のカメラサーバーによって受信を分担することも考えられている。

5.6　データの結合・低レベル解析（カメラサーバー）

カメラサーバーは、各望遠鏡毎に用意されるサーバーである。望遠鏡に搭載されたカメラからのデータを集め、データをまとめ、低レベル解析を行う。

5.6.1　データの結合

1 台の望遠鏡は 265 枚の Dragon FEB によってカメラからの読出しを構成している。そのため望遠鏡内でトリガがかかりデータを取り出すとき、受信側のカメラサーバーは 265 接続から到着したデータを結合する必要がある。これを可能にするために望遠鏡側ではトリガに番号を付番することになっている。各 Dragon FEB から届くデータに埋め込まれたトリガ番号を読み取り、同一のトリガ番号で結合することで 1 イベント分のカメラ全体のデータにまとめる。以降、この手続をイベント結合と呼ぶ。

[*2] スイッチングハブ、またはレイヤー 2 スイッチとも呼ばれる。その名前の通り、OSI 参照モデルの第 2 層「データリンク層」での情報転送を行う装置であり、インターネットの MAC フレームを MAC アドレスによって転送先を決める。

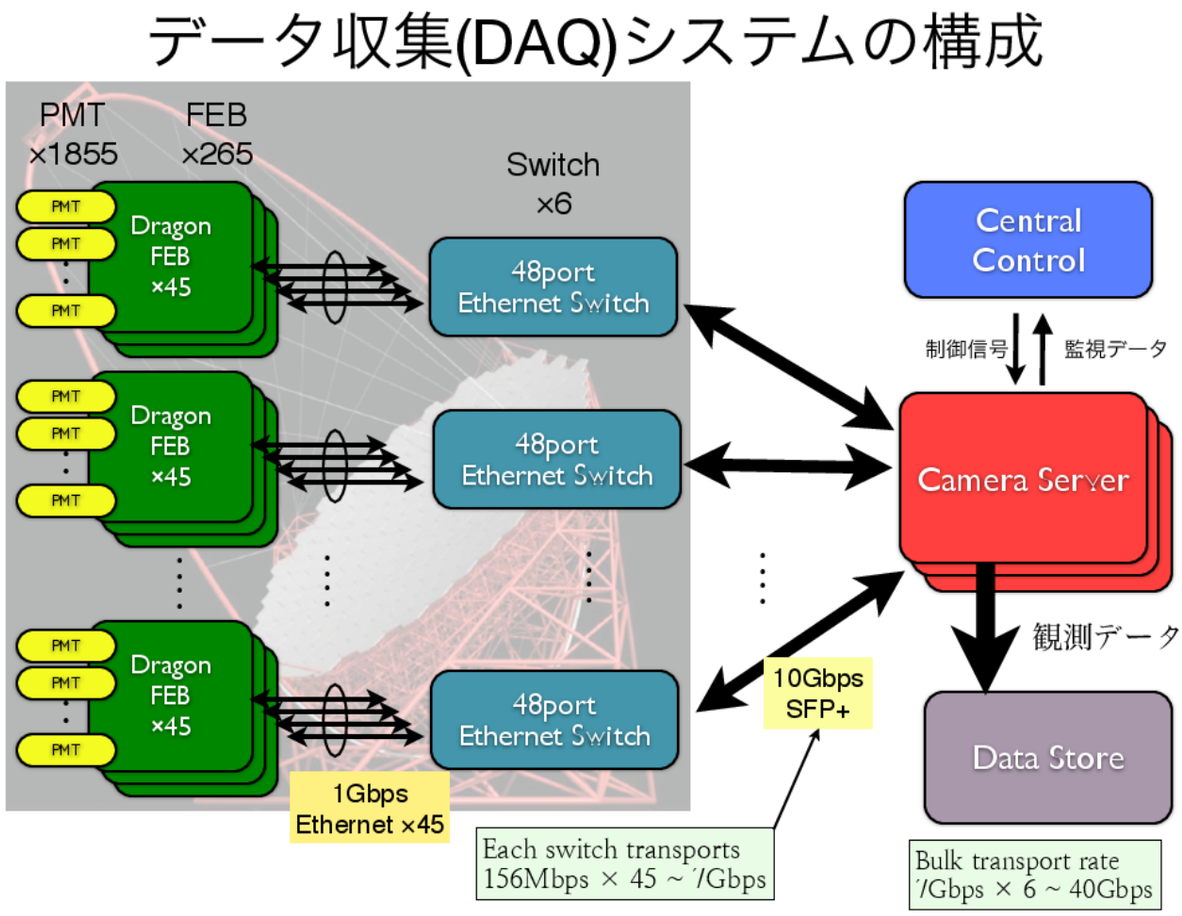


図 5.11　データ収集システム構成

5.6.2　低レベル解析

イベント結合を経たデータは、その後の処理の前に低レベル解析を必要とする。全てのカメラとそのサーバーとの間で転送されるトータルのデータレートは CTA South の場合、330 GB/s になる。このレートは、ATLAS 検出器のレベル 1 トリガの後段での読出しレートより、少なくとも 3 倍高い。そのため、CTA 全体のデータを扱うリアルタイム解析やオンサイトアーカイヴに転送することは大変な負担となる。これを回避するために、低レベル解析によるデータ量の低減が検討されている。

先述の通り、このための処理はプロセッシング、フィルタリング、ステレオイベントの選別がある。

プロセッシング

プロセッシングは、2 つの処理を行う。各ピクセルに信号波形の積分を行って相当する電荷を求めることにより、到着光子数を求めることと、波形のピークとなる時刻を求めることである。これらを元にフィルタリングを行う。

フィルタリング

フィルタリングにおいて、プロセッシングの結果得られた検出光子数がバックグラウンドレベルに対し

て有意に大きくない、またはタイミングが他のピクセルと合わないと判断したピクセルのデータは、波形データを削除して基本データのみにする。

これによって、波形情報は選別された重要なピクセルのみで保持され、一方で他のピクセルは基本情報（電荷積分、最大の時刻など）のみだけ保持される。これまでのモンテカルロシミュレーションの結果から、

1. 波形全体が保存されるのはカメラピクセルの 3% 程度
2. 97% のピクセルについては、基本電荷と、信号波形の特性情報として 15 バイトのみになる。

と予想されている。よって、データのサイズを大幅に減らすことができる。

ステレオイベントの選別

ステレオイベントの選別はステレオトリガを有効にした場合行う処理で、この場合はアレイトリガシステムでステレオトリガの判断が行われたイベントのみを選別する。

データ量低減効果

以上の方法を用いたデータ低減効果は次のとおりになると予想されている。ステレオ法を課すことでデータレートは CTA South 全体では 55GB/s になる。またさらにフィルタリングによって全波形のうち 3% のみが保存されるとき、このレートは 8.5Gb/s にまで低減できる[*3]。

5.7 アレイトリガシステム

アレイトリガシステムは、アレイ単位でステレオ選別を行うためのトリガシステムである。

5.7.1 アレイトリガシステムの必要性

アレイ全体でステレオ法を実現するためにトリガをかけようとする場合、1 箇所でトリガを判断して各望遠鏡に送信する方法は最良とは言えない。アレイの広がりは信号のラウンドトリップタイム 10 μs を超えるため、カメラ内の読出し回路側でそれだけの深さのバッファを用意しなければならず、仮にそれを用意したとしても、モンテカルロシミュレーションの結果から、ステレオ法によってカメラからカメラサーバーまでのデータ転送レートを低減する効果は高々 2 分の 1 から 3 分の 1 程度と予想されているからである。そのため、LST が他の望遠鏡と連携して観測を行う際は、一旦 LST で独立してトリガを掛けてデータを取得した後に、カメラサーバー側でさらなるステレオ選別を行う。

5.7.2 アレイトリガシステムの機能

アレイトリガシステムは、全ての望遠鏡からトリガ情報を収集し、そのトリガ時刻によってコインシデンスを取り、結果を各カメラサーバーに分配し、ステレオ選別を依頼する。

トリガ情報を各望遠鏡から得た際には、まず観測モードがサブアレイ構成の運用になっている場合はサブアレイ毎にトリガ情報をグループ分けする。そして望遠鏡の標高に合わせて時刻情報の補正を行い、時刻順にデータを並べ替え、コインシデンスを取る。この結果コインシデンスが検出された場合、該当するカメラサーバーにステレオ選別のトリガを送信する。

カメラサーバー側では、これらの処理が行われている間、データを保持する必要がある。現段階で要求値は 1

[*3] ただし、このレートは ATLAS 実験が永久保存するデータの書き込みレートより少なくとも 10 倍高い。

秒とされている。

5.8　時刻同期とトリガのタイムスタンプ

望遠鏡を始め、一つのイベントデータを構成するためのデータ送信元となる施設は広い範囲に分散して建設されるため、目的に応じた精度の時刻同期のメカニズムが必要になる。気候情報などはミリ秒の精度で十分だが、チェレンコフ光のイベントデータは、望遠鏡間で同一のトリガによるデータを結合させ、また、コインシデントしていないイベントを排除するためには最低でも 10 ナノ秒の精度で望遠鏡間を同期させ無くてはならない。さらに、到来方向を解析したりバックグラウンドの低減するために波面の到着時刻を利用するならば、1 ナノ秒から 2 ナノ秒の精度での望遠鏡間同期が望ましい。このような時刻同期のためのシステムを時刻配布システム（Clock Distribution & Trigger timestamping System）という。

CTA では CDTS のために White Rabbit という技術が採用されることになる。この技術は CERN、GSI、DESY と言った大規模な物理学実験で採用されている技術で、広い範囲にわたって分散しているシステム間でナノ秒以下の精度の時刻同期を提供する。

この White Rabbit 技術を用いた同期時刻配布・トリガタイムスタンプボード（Unified Clock Distribution and Trigger Timestamping board, UCTS）が CTA 全体の共有のコンポーネントとして使われることになる。UCTS を用いた CDTS は望遠鏡群の中で精密な時刻情報を必要とする全てのデバイスに配布することになっているが、データ収集に関しては以下の様な機能を提供する。

- カメラのクロック・カウンターに高精度な時刻情報を提供する。
- 各望遠鏡のカメラからのトリガ信号を受け取り、他の望遠鏡に対してナノ秒の精度でタイムスタンプをつける。
- このトリガ信号のタイムスタンプ情報はソフトウェアトリガに送られ、LIDAR[*4]使用時など VETO をかけるべき状況に対処する。
- 各望遠鏡のカメラでの 1 イベント分のデータ結合は、トリガ番号の付番によって行われて、そこに高精度時刻情報のタイムスタンプを与える。この番号は UCTS 側とカメラ側とでそれぞれ独立にカウントされる 2 種類の番号になる。

[*4] レーザーを上空に照射して散乱光を測定することで大気の状態を判断するための設備。

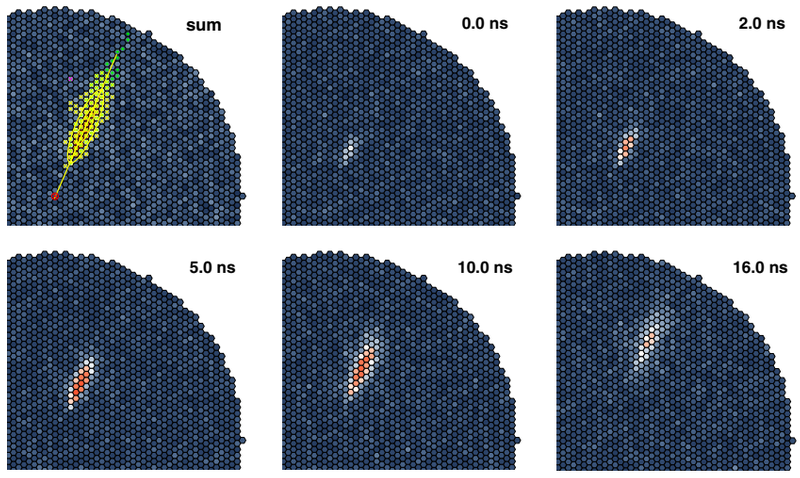


図 5.12　ナノ秒の精度が実現した場合のピクセルイメージの時間変化のシミュレーション。[2]

5.9　データの保存

CTA 天文台全体では 1 年当たり 1300 時間の観測時間が予定されている。もし観測中に CTA 全体で 1GB/s のレートでデータを保存した場合、1 年当たり 4.5PB のデータ量になる。ステレオ選別とフィルタリングを施したデータを保存したとしても 8.5GB/s のレートが予想されるので、40PB の全ボリュームが必要になる。

イベントフィルタリングとデータ圧縮のアルゴリズムを CTA の運用中に改善することが検討されている。

5.10　アレイコントロールシステム

データ収集システム（DAQ）はアレイコントロールシステム（ACTL）によって管理される。アレイコントロールシステムは南北両サイトの CTA 全体の運用を制御するシステムで、サイト外の遠隔操作によって行うことができる。オペレータによる現地でのスケジューリングは可能とするが、通常はスケジューリングソフトウェアの制御で、観測、キャリブレーション、メンテナンスなどの運用全てが行われる。

観測はスケジュールに合わせて行い、そのスケジューラは、データベースに登録されたプランに基づく。このプランでは観測における様々な設定があらかじめ用意され、観測時間、記録するデータの量と質についての情報、大気の状態、外部天文台などからのアラートの存在などの基本について以外に、様々な観測モード（サーベイ、ポイント観測、多波長観測）、外部へのアラート（CTA データの level-0, 簡単なリアルタイム解析）の生成、観測機会のターゲットまたは他の天文台からくるアラートに対して迅速で適切な反応、そして複数のサブアレイを形成させた並行な運転など、それぞれの観測に最適化された運転ができるようにする。

また、外部天文台からのアラートを受け取る場合は、トリガ（ToO、ガンマ線バーストアラートなど）をフィルタし、受信されるとすぐにスケジューリングを変更する。

そのため、このアレイコントロールシステムがデータ収集システムの準備、開始、終了、エラー処理を行う。

5.11　LST データ収集システムに対する要求

LST（大口径望遠鏡）のデータ収集システムに対する仕様と要求をまとめると、以下のようになる。

- 標準・既製品のネットワーク技術とコンピューティング技術のハードウェアコンポーネントを手頃な価格で利用する。それによって、性能の向上した新製品への交換を可能にする。つまり、カメラデータの転送にイーサネット技術と実証されたネットワークプロトコルを利用する。
- 明確に定義されたソフトウェアのフレームワークを使用すること。また、可能ならば要求性能が似ている望遠鏡で現在運用で実証されているコンポーネントを再利用する。
- カメラでの波形サンプリングは 1GHz とする。
- サンプリングの 1 ピクセル、1 time slice 当たりのデータサイズは 4 バイトとする[*5]。
- 波形情報として取り出すタイムウィンドウの深さは 30 ナノ秒とする。
- カメラからカメラサーバーに転送されるトリガイベントは LST（大口径望遠鏡）1 台では 15kHz を要求値とする。つまり、1855 ピクセルのカメラを構成している場合、26Gbps の転送レートを実現すること。
- アレイトリガシステムのために、1 秒程度データを保持するバッファを用意する。

この他に必要なことは、夜光の変動からくるトリガレートの変動に耐え得るデータ収集性能と、自然環境に耐え得る動作性能である。データ収集システムのうち、スイッチなどのネットワーク機器がカメラ部に設置される。望遠鏡は標高 2000m ほどの砂漠地帯に設置され、気温や湿度に対して堅牢性を持たなければならない。落雷の発生時の対策として、スイッチからカメラサーバーは光ファイバーによる接続を行って、過電流を防がなくてはならない。また LST は、外部天文台からのガンマ線バーストアラートの受信時に望遠鏡が高速で回頭を行うため、この衝撃に耐えなければならない。

5.12　本実験でのデータ収集プログラム開発方針

本実験では、この要求性能を元にデータ収集プログラムの開発を行った。以下がその開発方針である。

- Dragon FEB 側には、SiTCP 技術により TCP/IP 通信によるデータ転送ができるようになっている。これにイーサネット接続を行ってネットワークによるデータ取得を実現する。
- データ取得プログラムは、15kHz でのトリガレートによる 265 台の Dragon FEB からのデータ取得を可能にする。
- 同一トリガ番号によってイベント結合を行い、データを後段の処理に渡せるようにする。

なお、データプロセッシングは、実際の Dragon FEB を用いた開発が必要になるため、今後の課題とする。また、ステレオ選別の機能も、アレイトリガシステムの仕様決定に合わせて今後行うことにする。

[*5] これは High gain/ Low gain の合算値である

第 6 章

データ収集プログラム開発

本研究では、LST のデータ収集プログラムの開発を行った。LST のデータ収集において最も重要な点は、超高速のデータ収集性能を求められることである。これは前述のように LST では CTA が対象とするエネルギー領域において低エネルギー側の高感度達成を目標にするためである。また、CTA はコミュニティーに開かれた天文台として 30 年間程度運用されると見積もられている。安価で安定した信頼性の高いシステムの構築は必須であり、さらにコンピューティング技術やネットワーク技術は日進月歩の進化を続けており、運用期間において最先端の機器を柔軟に導入できることも必要である。このような観点から通信技術は TCP/IP とイーサネットを用い、プログラムは C++ 言語を用いてデータ収集プログラムの実装を行った。

本章では開発範囲と実験装置について紹介した後、それを用いて実装したプログラムの説明に入る。プログラム開発はデータ取得機能、データのイベント結合機能の順に説明を行い、それぞれで性能試験の結果を示す。データ取得は読出し速度が重要であり、この点に重点をおいた性能試験を行う。データのイベント結合機能はデータ到着の非同期性を吸収するためのバッファが前段に必要になる。また、そのバッファにデータを TCP バッファからコピーしなければならず、このオーバーヘッドのためにデータの取得に負荷がかかる。これらの問題を解決するためにリングバッファとマルチスレッドという 2 つの技術を用いた。それぞれの機能に焦点を当ててイベント結合機能の説明を行った後、性能試験の結果を示す。データ結合は確実なデータの結合と安定した処理が重要であり、この点に重点をおいた性能試験を行う。

6.1 開発範囲と前提条件

データ収集システムのネットワーク構成 (図 5.11) は、スイッチ 1 台が担当する FEB 約 45 接続ずつに分割することができる。開発の費用やリスクを最小限に抑えるために段階的な開発を考え、本研究ではこの 1 スイッチ単位の接続に開発範囲を絞ることにする (図 6.1)。さらに、この単位の中でも主要な開発範囲は TCP/IP 通信による収集とイベント結合とする。

この範囲の後段の他システムへの送信とストレージへの書込は独立して開発が可能であり、対象外とした。他システムへの送信は仕様が明らかになった段階で開発することとする。ストレージへの書込もハードウェアの調査、入手、試験、調整が必要であり、開発費用の観点から次の段階とすることが適切である。

また、前段である Dragon FEB は現在量産前であるため、台数が揃わない。そこで擬似データを送信するエミュレーターを開発して対応した。エミュレータはトリガ受信機能と TCP/IP 通信によるデータ送信機能をもたせているが、FEB でトリガによって発生したデータがネットワークで送信されるまでの部分は確認ができない。そこで、FEB の dead time は 3.9 μs であることと送信バッファのサイズから、簡単なシミュレーション

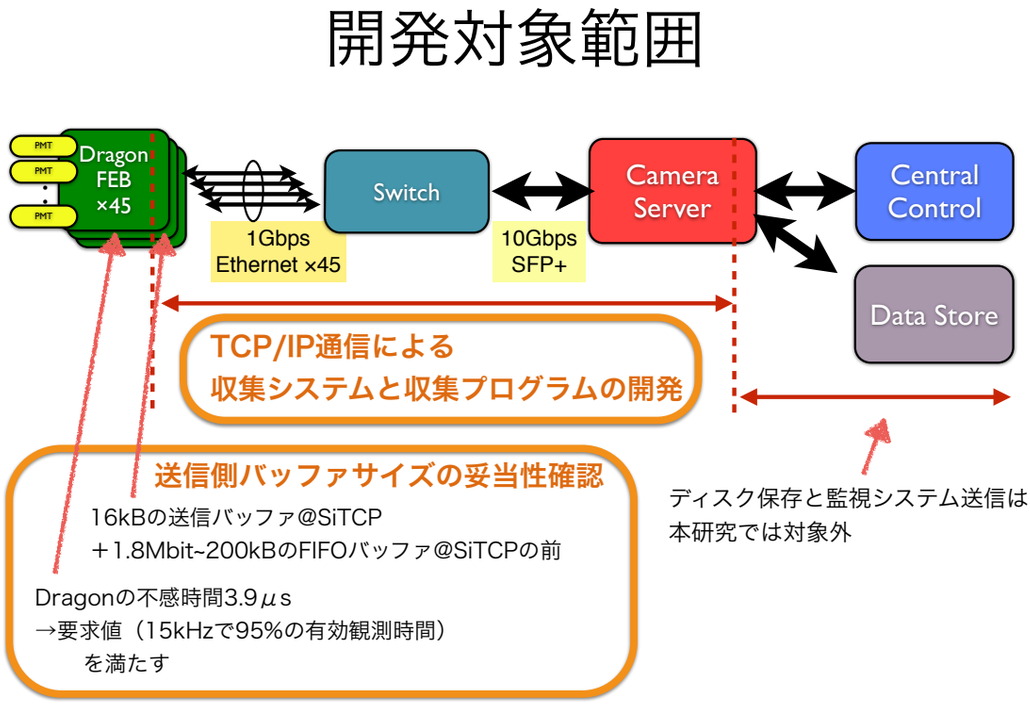


図 6.1　本研究での開発対象範囲。

を行って安全性を確認した。これについては本節の 6.1.3 で述べる。

以下、本研究での開発と試験の前提条件について述べる。

6.1.1　データサイズとトリガ

データサイズは今後変更がおこる可能性があるが、本実験では、現在予定されている仕様と同様に 1 GHz サンプリングで 1 イベント当たり 30 ns 分のタイムウィンドウに相当するデータを取得することにした。この時のデータサイズは、現段階では 976 Bytes になる。

トリガは本実験では定間隔でかけることとする。本来トリガはランダムにかかるが、トリガによって読み出されたデータはそのまま送信されるわけではなく、TCP の送信バッファに入り送信を待つことになる。送信のタイミングは、受信側から送られる確認応答を受け取った時であり、トリガのタイミングとは関係がない。さらに送信サイズも元のデータと同一ではなく、フラグメンテーション化を受けてデータ送信される。以上の効果により、送信のタイミングはほぼ平滑化されていると言え、データ収集システム開発のテストにおいては、定間隔トリガで十分近似的に試験を行えると判断した。

6.1.2　ネットワークの設定

ネットワークの設定値は、なるべく一般的な設定を心がける。中継機器は一般的な設定に最適化されて設計されているはずであり、特段の問題が無い限り変更することは望ましくないはずである。今後様々な症状が出た場合に必要に応じて変更することとする。

MTU

MTU を大きくすれば、1 回のパケット転送でより多くのデータを送信できるため、同じデータサイズでも少ないパケット数で転送できる。よってパケット 1 つあたり必要なヘッダやトレーラによるオーバーヘッドを少なくすることができる。しかし本実験のようなネットワーク構成の場合、一回のトリガによって全ての接続先からデータを同時に集約するため、スイッチでは各接続先からパケットを同時に受け取らなければならない。パケットサイズが大きければ、スイッチのパケットバッファへの負担を増やしてしまい、パケットロスの危険性が大きくなる。

今回は SiTCP で MTU を最大 2048 までしか上げることができないことの得失、また MTU を 1500 とする設定は非常に一般的であり、ネットワーク機器もこの値に最適化されて設計されている可能性が高いことも考え、本実験では特に大きな MTU の値とせず、1500 として行う。

PauseFrame

Pause Frame は IEEE802.3x 規格によるフロー制御である。スイッチには、異なるポートに複数のパケットが同時に届くことがあり得る。その場合でも送信ポートの処理に届けられるように、受け取ったパケットをスイッチのバッファに貯めている。しかし送信処理が間に合わなくなりバッファの空きメモリ容量がなくなった場合は新たにパケットを受け取ることができなくなるため、バッファの空き容量がが少なくなった時にスイッチに送信してくるホストに対して送信の停止を要求している。この送信停止要求のパケットが Pause Frame である。

TCP/IP 通信におけるウィンドウ制御でもゼロウィンドウを指定することで送信を停止させることはできるが、Pause Frame はこれとは独立しており、イーサネットヘッダを用いたパケットで送信停止を要求する。

SiTCP ではこの Pause Frame に対応しておらず、スイッチからの送信停止要求が効かない。本研究では終盤までこのことに気づかなかったため、スイッチ側で Pause Frame は有効になっている。そのため Pause Frame を解釈するエミュレータと解釈しない Dragon FEB では、特に高いトリがレートにおいて差が現れる可能性がある。

SiTCP の設定値と制約

Dragon FEB との接続を行った際、SiTCP 側では (表 6.1) の設定値を用いた。これらは SiTCP での default 値である。また、SiTCP は FPGA 上に TCP/IP 通信の機能を限定的に実装されているため、(表 6.2) のような制約が存在する。

表 6.1 SiTCP の設定値

項目	値
Nagle Buffering	on
Keep Alive	off
MSS	1460
再送タイムアウト	500 msec

表 6.2 SiTCP による TCP/IP 通信上の制約

項目	値
受信 Window size	65535 Bytes で固定
送信バッファ	16 kB
IEEE 802.3x フロー制御	なし
MSS	最大 2048 まで変更可（default は 1460）

6.1.3　FEB 内の TCP 送信バッファ検証

TCP/IP のネットワークにおいてはデータ到達の信頼性が保証されている。しかし FEB 内で TCP/IP の送信バッファにデータ届く前までの段階は検証が必要であり、特にトリガのタイミングや頻度によっては、送信バッファにデータが届く手前でデータ損失が起こる可能性がある。ここでは、データ損失の起こる可能性と現在の対策の妥当性についてシミュレーションを行って考える。

トリガが頻繁にかかった場合は当然、送信バッファに多くの未送信データが滞留することになる。また、TCP/IP 通信ではウィンドウ制御により送受信を管理している。送信バッファから送信されたデータは、受信側に届いたことによる確認応答が戻ってこない限り削除されないため、送信処理が追いつかない場合もやはりデータが滞留し、送信バッファに空きがなくなってしまう。このような状況では SiTCP にデータを受け渡すことができず、トリガによって取り出されたデータは捨てられてしまうことでデータ損失が起こる。

よって、シミュレーションでは FEB 側の送信バッファ滞留量を求め、現在 FEB に用意されているバッファのサイズに余裕があることを確かめた。

トリガによるデータ追加

空気シャワーの到来はランダムであり、当然トリガもランダムにかかることになる。ここでは安全のためトリガ頻度を大きく取り、20 kHz の Poisson 分布でトリガがかかるとした。Poisson 分布では、ある事象が起こった後に次の事象が起こるまでの時間間隔 t が τ になる確率密度は

$$P(t=\tau) = Ae^{-m\tau} \tag{6.1}$$

で表される（A は定数）。この確率密度分布のもとで生成した時間間隔でイベントを発生させる。その時に 1 イベント=976 Byte ずつ送信バッファにデータを追加する。不感時間は 3.9μs とし、追加後この不感時間の長さは新たなデータ追加を行わない。

送信によるデータ削除

一方、FEB から帯域 220 Mbps を使用できたとする。これは約 56 μs 間隔で 1460 B ずつパケット送信できることに相当する[*1]。この送信により送信バッファ内に滞留しているデータはクリアされるとして、56 μs 間隔で 1460 Byte ずつデータを減らす。

計測タイミング

56 μs 毎に滞留量を計測する。（つまり送信により 1460 Byte 減る直前のバッファ内滞留量）送信バッファのシミュレーションを 20 分間の状態について行った。

結果

(図 6.2) のようになった。最大滞留量は 16 kB 程度となった。Dragon FEB は SiTCP の前段で 200 kB の FIFO バッファを持っている。シミュレーション結果より、ネットワークにおいて安定した帯域が確保できればイベントロスを起こさずデータ転送が行えるはずである。

[*1] 220 Mbps=27.5 MB/s、またパケットサイズはヘッダなどを足しあわせて 1538 B。定常的に送り続けることができて 1 秒間に $\frac{27.5\,\mathrm{M}}{1538}$ パケットずつ送信できると仮定して、大まかな値を求めた。

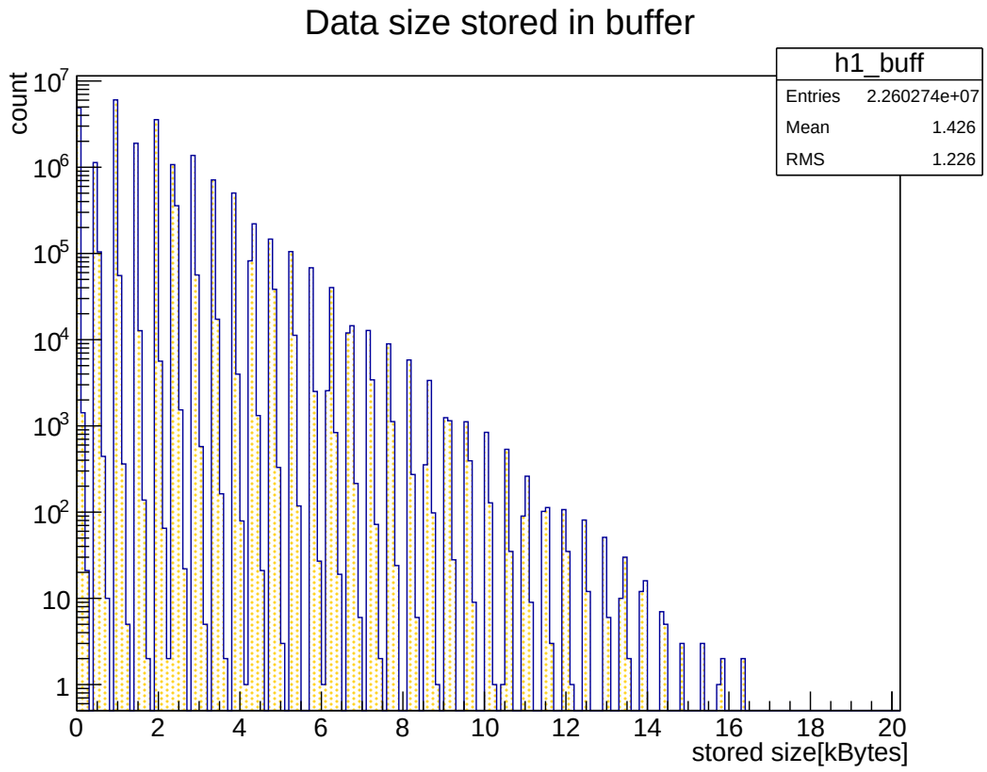


図 6.2　FEB の TCP 送信バッファ内滞留量のシミュレーション

6.2　実験装置

本実験に用いた各コンポーネントの説明をする。

6.2.1　データ送信機器

DragonBoard 用外部トリガ

パルスジェネレータは Hewlett-Packard 社製 33120A を用いた。Single End 信号のアウトプットだが、Dragon FEB は外部トリガとして LVDS を受け付けるため、シグナルコンバータを作成した。コンバータのチップは Texas Instruments 社製 SN65LVDS100D を用いた。(図 6.3)

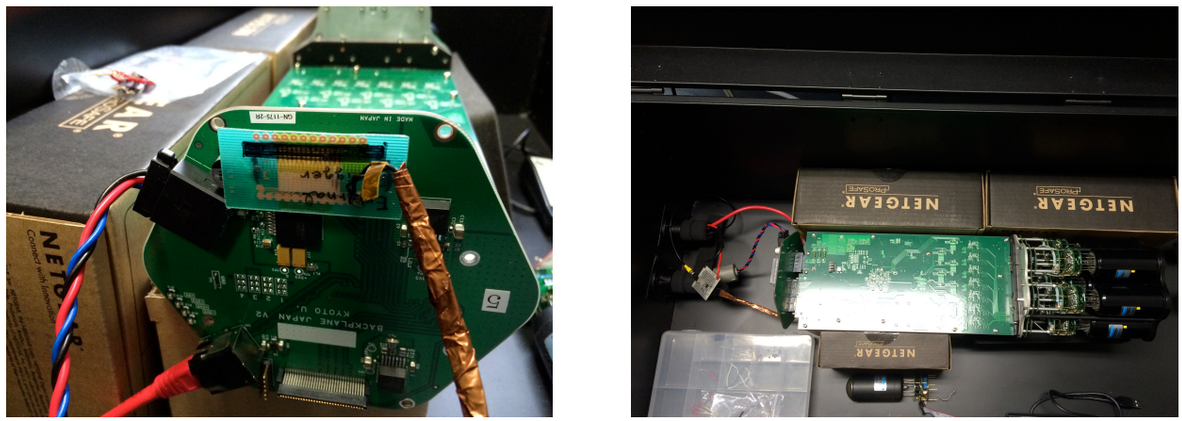


図 6.3　外部トリガを入力している状態の Dragon FEB。左：バックプレーン側から眺めた図。銅テープで保護されているケーブルがトリガ用ケーブル。右：上から眺めた図。左側にシグナルコンバータが見える。

PC（エミュレータ用）

Dragon FEB のデータ送信機能をエミュレートするため、iMac を計 24 台[*2]用いた。2008 年製 2013 年製と製造年度が揃っていないので、性能は異なる。エミュレータの性能試験の項にて検証結果を述べる。

6.2.2　データ収集機器

データ収集プログラムを稼働させるカメラサーバー用 PC(図 6.4) は (表 6.3) の部品を用いている。これらの部品のうち、特にデータ収集性能に関係のある各項目について以下で説明する。

[*2] 14 台が IPMU から借用、ほかは研究グループ内からの借用

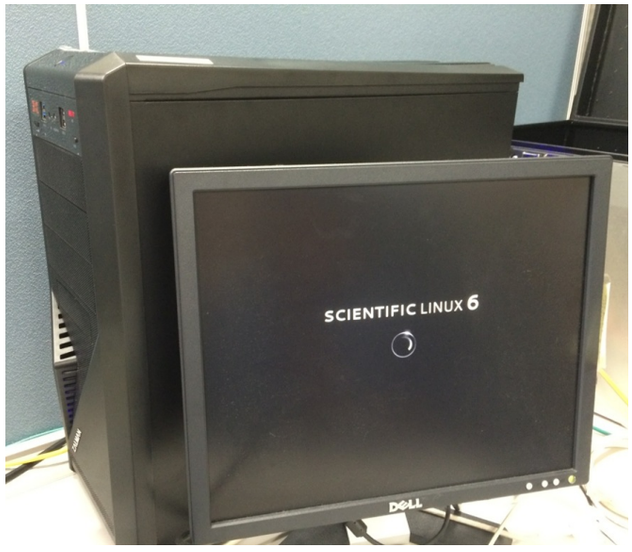


図 6.4 データ収集用 PC

表 6.3 DAQ 用コンピュータの部品とその性能

部品	型番	性能
CPU	Core i7-4930K BOX	3.40 GHz /ターボブースト時 3.90 GHz / 6 Core 12 Thread / L2 256 kB×6 / L3 12 MB / TDP 130 W
Memory	KS14731821	DDR3 2400 MHz CL11 DIMM 4 GB × 2 枚
Disk	Transcend TS256GSSD320	256 GB / SSD / 2.5 inch / MLC / SATA 3.0 (6 Gb/s) × 4 枚
Mother Board	X79 Extreme9	Intel X79 Chipset, Intel Core i7 対応, SATA3 6.0 Gb/s ×8
NIC	Intel X520DA2	PCI-Express 2.0 対応 5.0 GT/s × 8 Lane intelligent offload
NIC	Chelsio T420CR	PCI-Express 2.0 対応 5.0 GT/s × 8 Lane 32 K offloaded connections

CPU とチップセット

(図 6.5) は、X79 チップセットのブロックダイアグラムである。Core i7 CPU の対応機種のため、ダイアグラムは Core i7 と共に記載されている。CPU の Intel 社製 Core i7 は 2008 年に発表されたシリーズだが 4930K はそこから改良が進んだもので、CPU と直接接続されているバスが、メモリだけでなく PCIe 2.0 も 40 レーンある。これにより、理論上は片方向当たり 320 Gbps のデータ伝送が可能であり、この値は継続的なデータ伝送を仮定した値であることを考慮しても、今回のデータ収集には十分の能力がある。

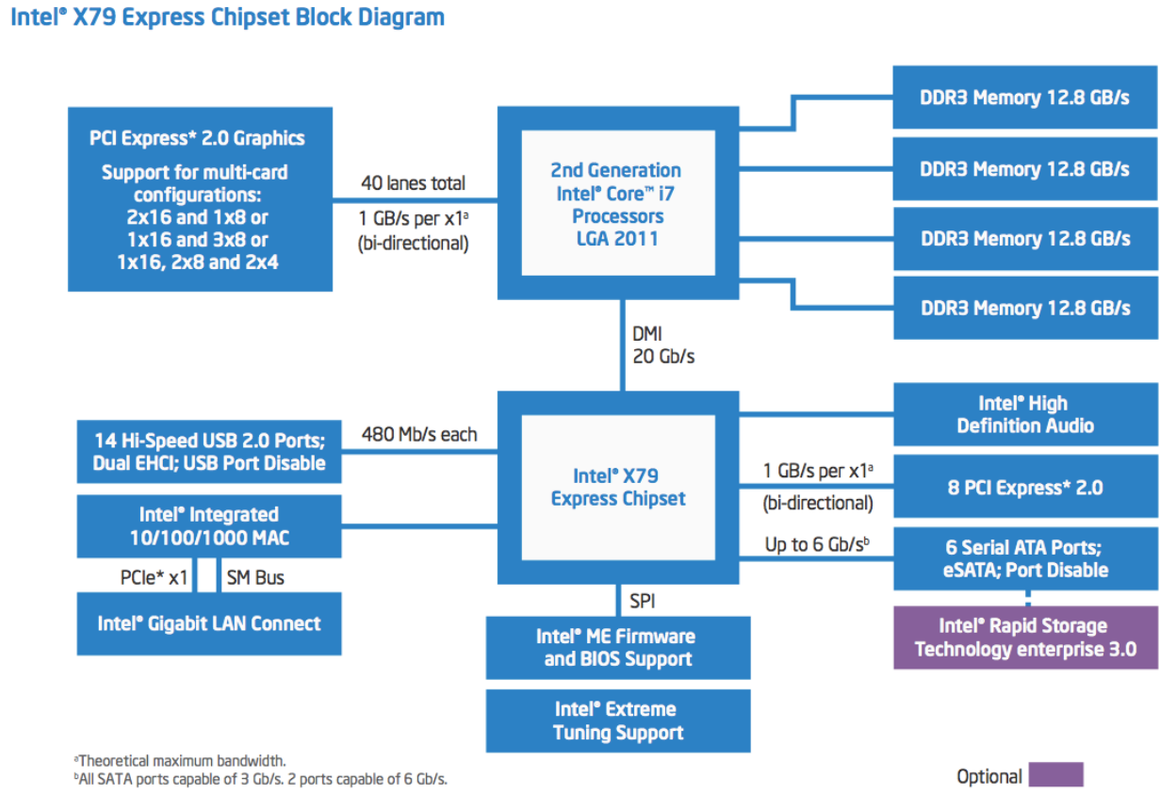


図 6.5　Intel Core i7 プロセッサと Intel X79 チップセットのブロックダイアグラム

NIC

NIC（ネットワークインタフェースカード）[*3]は 10 Gbps SFP+ ポートを 2 ポート持つカードを 2 種類用意した (図 6.6)。一つは Intel 社製 X520DA2、もう一つは Chelsio 社製 T420CR であり、共にマザーボード上の PCIe スロットに挿して使用する (図 6.7)。これらの性能は、PCIe2.0 対応で 8 レーンであることから CPU とは 32 Gbps のバンド幅で結ばれていることになり、10 Gbps ポート 2 つぶんの 20 Gbps に対する制限なく CPU とデータ伝送が出来ると思われる。

[*3] ネットワークインターフェースとは、コンピュータをネットワークに接続するために必要な機器である。このインターフェースは NIC(Network Interface Card) やネットワークアダプタ、ネットワークカード、LAN カードとも呼ばれる。コンピュータのマザーボードに組み込まれていたり、NIC を拡張スロットなどに増設することもある。

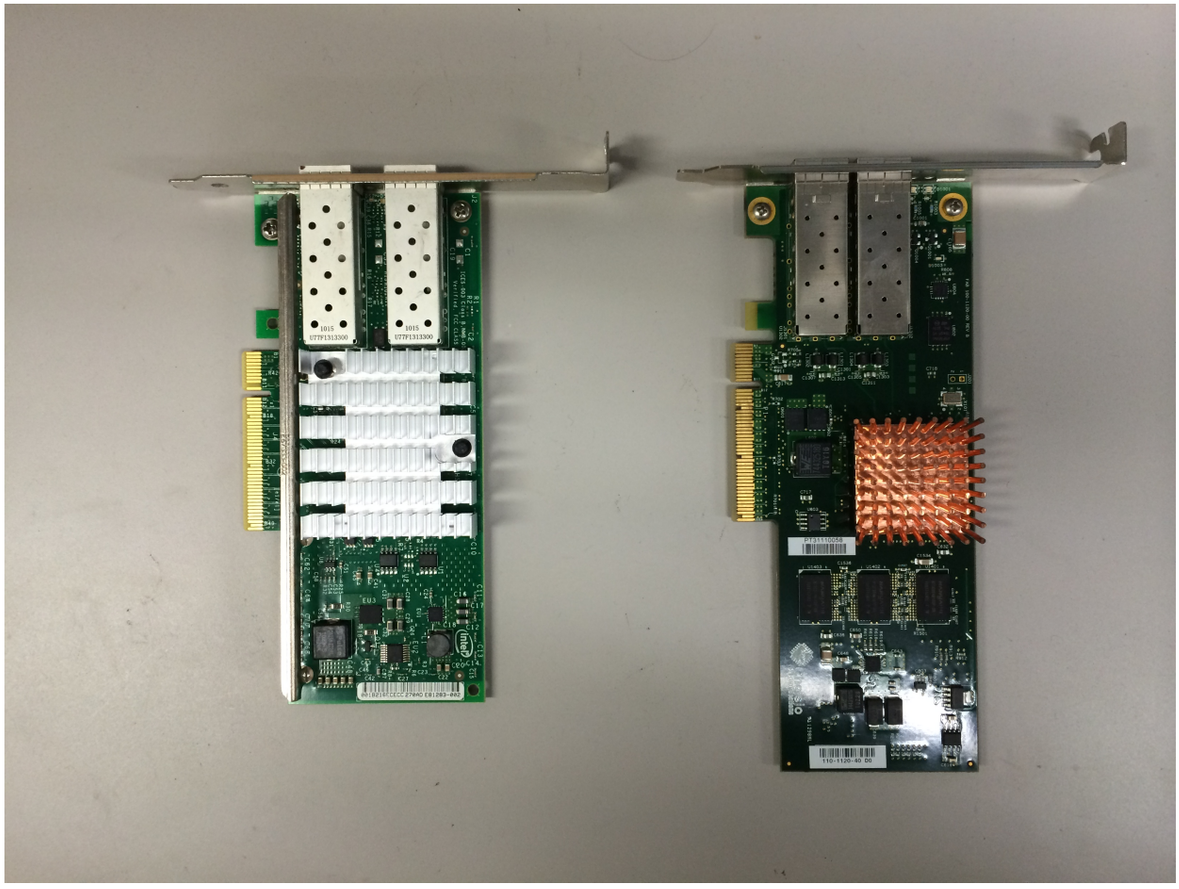


図 6.6 NIC。左：Intel 社製 X520DA2、右：Chelsio 社製 T420CR

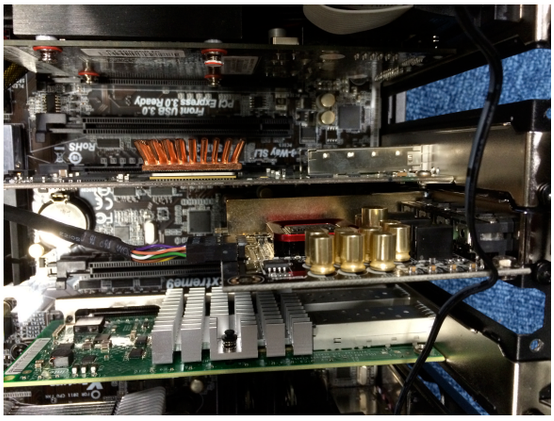


図 6.7 マザーボードに NIC を挿している様子。左は筐体内部を、右は筐体の外部から見たところで、共に上から DVI などのグラフィック入出力ボード、Chelsio T420CR、オーディオ入出力ボード、Intel X520DA2。

図 6.8　ネットワークスイッチ GS752TXS

6.2.3　中継機器

スイッチ

スイッチは Netgear 社製 GS752TXS を用いた。Dragon FEB を 45 接続行い 100 kHz のトリガがかかった場合、約 1 kB のデータが 1 つのパケットによって送られると仮定すると、Ack パケットの返送も含めて単純計算で 9Mpps は必要になるが、パケットフォワーディングの値はそれを満たしている。パケットバッファメモリが 2 MB あり、こちらも単純計算で 1 kB のデータの 2000 倍あるため余裕があるように見えるが、TCP incast[*4]はこのクラスのスイッチでも起こるとされるため、注意が必要である。

表 6.4　Netgear 社製 GS752TXS スイッチの性能

項目	性能
ポート	RJ45 48 ポート、10 Gbps SFP+ 4 ポート
転送方式	store & forward
システムメモリ	128 MB
パケットバッファメモリ	2 MB
フラッシュメモリ	32 MB
スイッチングファブリック	176 Gbps
パケットフォワーディング	130.9 Mpps
レイテンシ	20 μs 以下

[*4] パケットが同時に到着した場合にパケットロスが発生する現象。マイクロバーストとも言う。

図 6.9 Cat6 のイーサネットケーブル

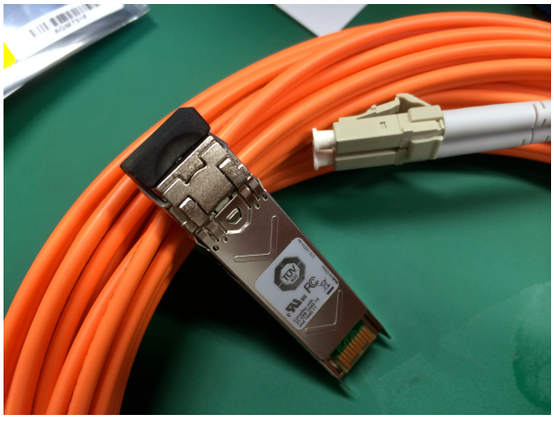


図 6.10 光ファイバーケーブルと SFP+ アダプタモジュール

図 6.11 DA ケーブル

イーサネットケーブル

- 1 Gbps イーサネットケーブル
  Cat6 ケーブルを主に用い (図 6.9)、一部で Cat5e ケーブルを使用した。どちらも RJ45 規格のポートを両端に持つ 1 Gbps 全二重通信の帯域のケーブルで、使用上特に区別する必要は無い。
- SFP+ ケーブル
  SFP+ ポートを持つ 10 Gbps ケーブルは、DA ケーブル（ダイレクトアタッチケーブル）を用いた (図 6.11)。これはスイッチとデータ収集マシンの間の接続に使用する。本来ならばこの部分は望遠鏡への落雷の影響を抑えるために光ケーブルで敷設しなければならない。しかし光ケーブルの両端にはアダプタモジュールが必要であり (図 6.10)、このアダプタモジュールはベンダーロックのため接続先のネットワーク機器のベンダーに合わせなければ機能しない。さらに高額なため性能評価では見送るのが妥当と考えた。また、DA ケーブルは電気信号を伝搬する。光ケーブルのほうが信頼性が高いことを考えると、DA ケーブルで性能が確認できれば光ケーブルでも性能は十分に見込める。

6.3　データ取得機能の開発

TCP または UDP によりデータ取得を行う場合、Linux を含む UNIX 系 OS コマンドラインアプリケーションには nc コマンド（Netcat コマンド）という、ネットワーク通信を行うためのユーティリティが備わっており、通信状態の確認やポートスキャンなどに使用できる。これを用いれば、以下のように簡単なコマンドだけでも FEB からデータが取得できる。

```
% nc 192.168.10.16 24 > datafile
```

しかし、複数の接続を行う場合はプロセスが複数となり、オーバーヘッドが大きい。また、イベントを取得後、、データはイベントの区切りなく数珠つなぎになった状態であり、各イベント毎のデータとして読み取った上で、イベント結合や低レベル解析など、後段の処理の負荷も考慮しなければならない。これらの事を踏まえ、複数接続からなるべく低負荷で高速にデータ取得ができるプログラムがよい。このため、データ取得の際には select() コールを用いた FD 参照型読み出しを行った。TCP/IP 通信により複数の接続からデータ収集を行う場合、非同期通信であるために各接続からのデータ到着は一様性が保証されない。このとき、データが到着している接続を recv もしくは read といったシステムコールを発行する前に判別できる select() を用いれば、データ収集プログラム側ではより効率的なデータ取得ができる。

この節ではこれらの関数の説明をしながら、データ取得プログラムで用いた主な関数の機能を流れを追って説明する。

6.3.1　主要な関数の機能

TCP/IP 通信によりデータを送受信するためには以下の関数が用意されている。本プログラムでは、socket によるインタフェースの定義、connect による通信の確立、read による受信バッファの読込、close による通信の終了を行った。また、select コールも複数接続での受信のために用いた。

通信端点（socket）

ソケットは、アプリケーションプログラムからネットワーク通信サービスを受け取るときの API[*5]である。プログラム側で通信を行うときのインタフェースとして機能し、TCP とアプリケーション間の通信路を作る。ぉファイル入出力と同様に、ソケットをオープンするとファイル記述子[*6]と同様の整数値が返され、ファイル記述子を通したファイル入出力と同様に、send システムコール[*7] による送信、recv システムコールによる受信を行える。

また、以下の bind、connect、close は、全てこの socket インタフェースのディスクリプタを渡して行う。

ポートの対応付け (bind) と接続要求待ち

bind システムコールは、socket とポートの対応を行うことで自分のホスト内のアプリケーション層とトランスポート層の通信路を確保する。ポート番号を指定して bind システムコールを呼び出す。しかし、オペレー

[*5] アプリケーションプログラミングインターフェース

[*6] ファイル記述子はファイルディスクリプタ (file descriptor, fd) ともいう。 0 以上の整数である。

[*7] システムコールはユーザプログラムからシステムが持っている機能を呼び出す方法。プログラマから見た場合は C の関数として呼び出されているが、システムコールではユーザーモードで引数のテーブルを用意して、ソフトウェア割り込みによってカーネルモードに移行しカーネルモードで実際の処理を行う。終了すると再度ユーザーモードに戻って値を返す。

ティングシステムに自動で割り当ててもらうことができるので、使用しない[*8]。

接続の開始・終了（connect、close）

connect システムコールは、通信相手の IP アドレスとポート番号を指定し、TCP のコネクション確立要求を行う。コネクションを切断し終了するときは close システムコールを使用する。

データ送受信

先述のように送受信には send、recv というシステムコールが用意されている。さらに、低水準ファイル入出力同様に read,fread 関数による受信、write や fwrite 関数による送信も行える。

select

ひとつのプロセスで複数の接続からデータを取得するには、各接続の確立のために一つずつ socket を accept し、それぞれのソケットを読み込む必要がある。データの読み込みのためには recv、read 関数を呼び出すが、同時にひとつの socket にしか recv、read を発行できず、その入出力が完了するまで次の socket を見に行くことができない。この読み込みを順番に続ける方法で並列にデータの取り出しが行えるが、socket にデータが到着していない場合は、このためにオーバーヘッドが余分に取られてしまう。

これを避けるために、select システムコールが用意されている。select はファイル記述子を用いて参照を行うことで recv、read を発行する前にどのソケットにデータが到着しているかを調べることができる。これを用いて以下のプログラムの流れにすることで、処理を多重化できる。

select は、プログラムが入出力を行う際のファイル記述子のアクセス先を fd 表[*9] という参照テーブルとして持っている。ここに接続先の socket ファイル記述子を代入して select を呼び出すことにより、データの到着している接続先の一覧が得られることになる。あとは FD_ISSET 関数により fd 表を参照して、データ到着済みの socket だけを読み込む処理を行う。

[*8] 後述の FakeFEB プログラムは、DragonFEB と同じポート番号で接続要求を待ち受ける必要があったため、このコールを用いている。

[*9] file descriptor table, ファイル記述子表ともいう。

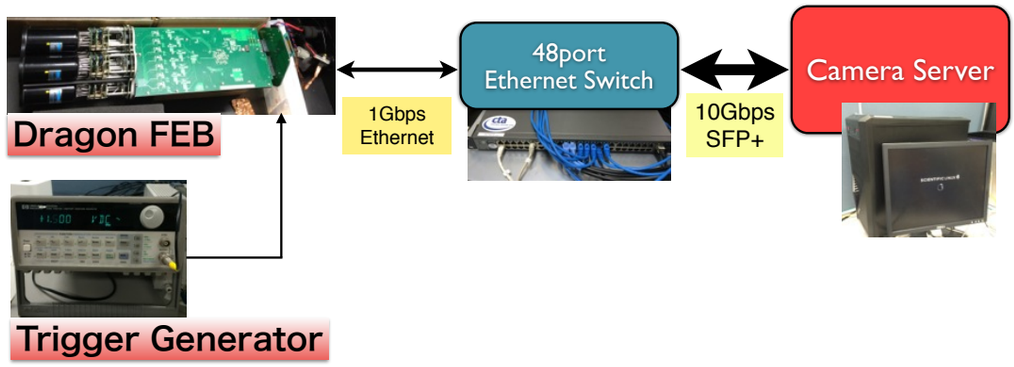


図 6.12　1 対 1 試験の接続構成

6.4　1 対 1 試験（FEB の性能確認）

この節では、カメラサーバーと Dragon FEB を 1 対 1 で接続してデータ取得を行った時の性能について述べる。Dragon FEB からは 1Gpbs のイーサネット通信が可能だが、その帯域に近い 900Mbps 以上でのデータ取得が達成でき、また転送レートの上限はネットワークに依存している事を確かめた。まず試験方法と通常のデータサイズでの試験結果を示す。その後、1 イベント当たりのデータサイズを変化させてデータ取得試験を行い、FEB から読み出せるイベントレートがネットワーク依存であることが分かった。

6.4.1　Dragon からのデータ取得

カメラサーバーと Dragon FEB をスイッチ経由で接続し、カメラサーバー上でデータ取得プログラムを稼働させ FEB に接続し、FEB には外部トリガによりトリガをかけてデータ取得テストを行った (図 6.12)。

Dragon FEB は、読出しボード内の FPGA に SiTCP 技術が実装されており、TCP/IP 通信でデータを送信するために個々の IP アドレスを持っている。電源を投入した Dragon FEB とカメラサーバーをイーサネットで接続し、カメラサーバーは Dragon と同じサブネットマスクの配下になるように IP アドレスを設定する。これだけで FEB との通信は既に可能な状態にある。スローコントロール[*10]やデータ取得のための接続は FEB の IP アドレスを指定して実行する。

まずスローコントロールを行い、トリガモードを外部トリガに、トリガ毎に読出すデータの time slice を 30 にした。PMT の波形をとる必要があるならば HV コントロールの電源投入コマンドを送ったあと各 PMT の HV 値を設定するが、データ取得のみが目的なので行わない。

その後データ取得プログラムで FEB の IP アドレスを指定して接続を確立させる。Dragon に (図 6.3) の通り外部トリガ回路を接続し、パルスジェネレータから外部トリガをかけると、パルスジェネレータからのトリガ信号が 1 回届くたびにデータが一件送信されてくることが確認できる。このセットアップでデータ収集を行い、10^8 Byte を超えた時点で、イベント数の整数倍となるように残りを読込み終了することとして、取得した

[*10] ここでのスローコントロールは、Dragon FEB と PMT に関する設定である。各 PMT の電圧値やトリガモード、データを読み出す際の time slice の深さなど、実験装置のレジスタや FPGA 内部メモリなどへアクセスして設定する。SiTCP には、このスローコントロールを行うために、RBCP（Remote Bus Control Protocol）というプロトコルが実装されている。ユーザーは PC からこの RBCP によるメッセージを UDP パケットとして送り、スローコントロールを行う。

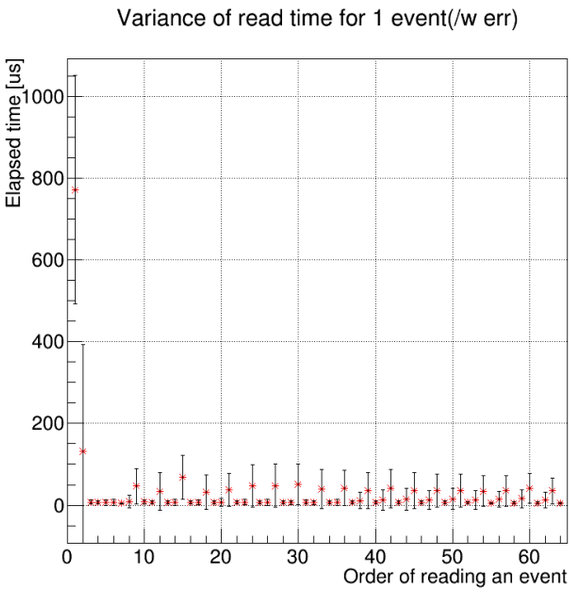


図 6.13　読込一回あたりにかかる時間の変化

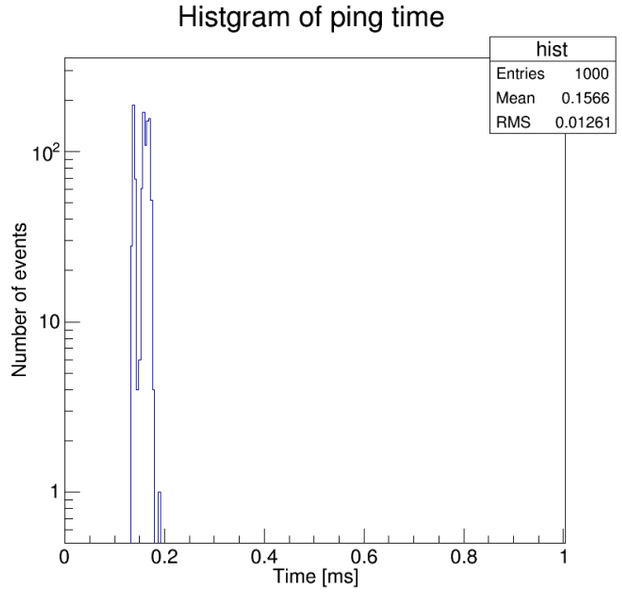


図 6.14　DAQ マシンから FEB までの RTT 測定結果

件数と所要時間からデータ取得レートの計算を行った。

6.4.2　計測開始時間

データ取得プログラムは Connect 実行後 read 発行を繰り返すことによりデータを取得していく。しかし Connect 実行の後を計測開始時間とした場合、最初の 2 件ほどは読み込みに時間がかかり計測の精度が良くない。(図 6.13) は Dragon FEB から一回あたりの読み込みにかかる時間の変化を計測した結果である。Connect 実行後最初のデータ読み込みが完了するまで約 800 μs 所要している。その後は数 μ 秒から数十 μ 秒で周期的に変化している様子が見える。

最初に時間がかかっているのは 3 ウェイハンドシェイクにより通信が確立するまでの時間 (図 6.15) で、RTT を 0.2 ms とした場合、RTT × 2=0.4 ms が予想される時間だが、これと同程度の時間になっている。実際に RTT を計測したところ、平均値 0.15 ms 程度となった (図 6.14)。その後の周期的に読み込み時間が変化している部分は、バッファにパケットが到着する頻度よりも到着したデータを読み込む頻度が速いためにデータ量がゼロになり、待ち時間ができているためと考えられる。

この結果から、通信開始時の時間の影響を低く抑えるため、接続開始後データ読み出し開始までに 500 μs のスリープを入れた後に計測開始時刻を記録することにした[*11]。

6.4.3　性能測定結果

以上を元に 1 枚の Dragon FEB に対してデータ取得を行った結果が (図 6.16) である。FEB に入力する外部トリガーのレートを 10 kHz ずつ変えながら 20 回ずつデータ収集を行い、各入力トリガーレートについてデータ取得レートを求めた。横軸は入力したトリガレート、縦軸はデータの取得レートの平均値である。

[*11] 6.6 以降は計測法を改良し、通信開始とトリガ送信開始を分離した上で計測開始時刻を 1 件目のデータ取得終了時に設定している。この実験ではさらにトリガ信号にカウンタを用意した方が良い。

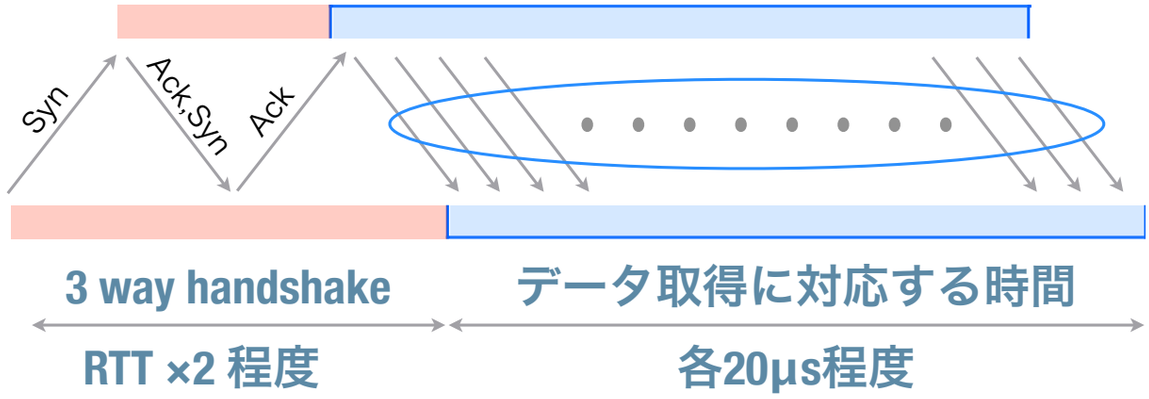


図 6.15　データ取得のための通信の流れ。上はデータ送信側である FEB、下はデータ要求側の PC を表している。

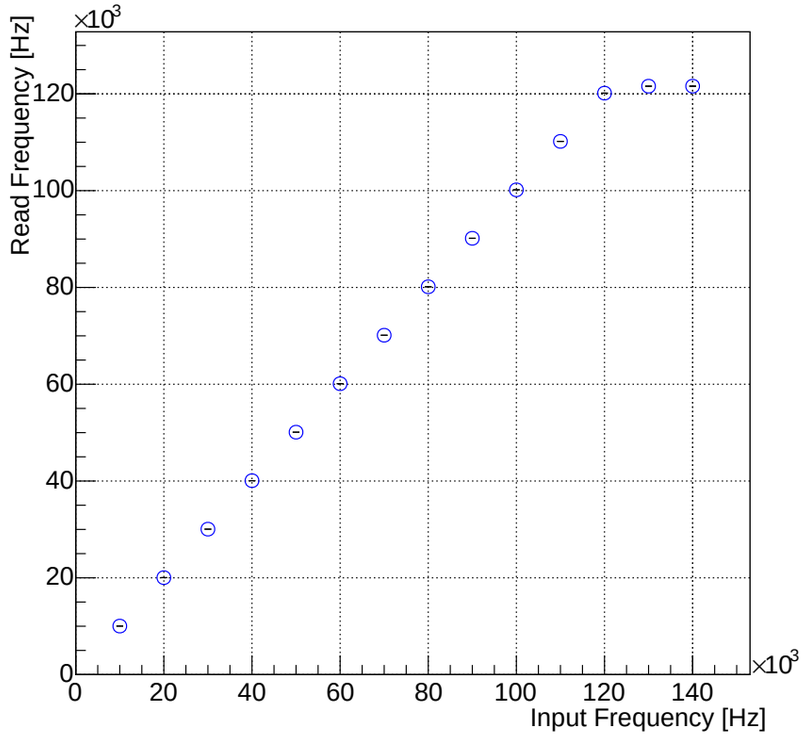


図 6.16　Dragon FEB からのデータ取得結果。

120 kHz までデータの取得レートは入力トリガレートに追随してほぼ同じ値になり、比例して増え続けた。しかし追随できる入力トリガレートに限度があり、データ取得レートは 121kHz 以上にはならなかった。これは転送レートに換算すると 904 Mbps である。Dragon からの接続は 1 Gbps ケーブルなので、FEB はネットワーク帯域の限界までデータを送出できていると考えて良い。

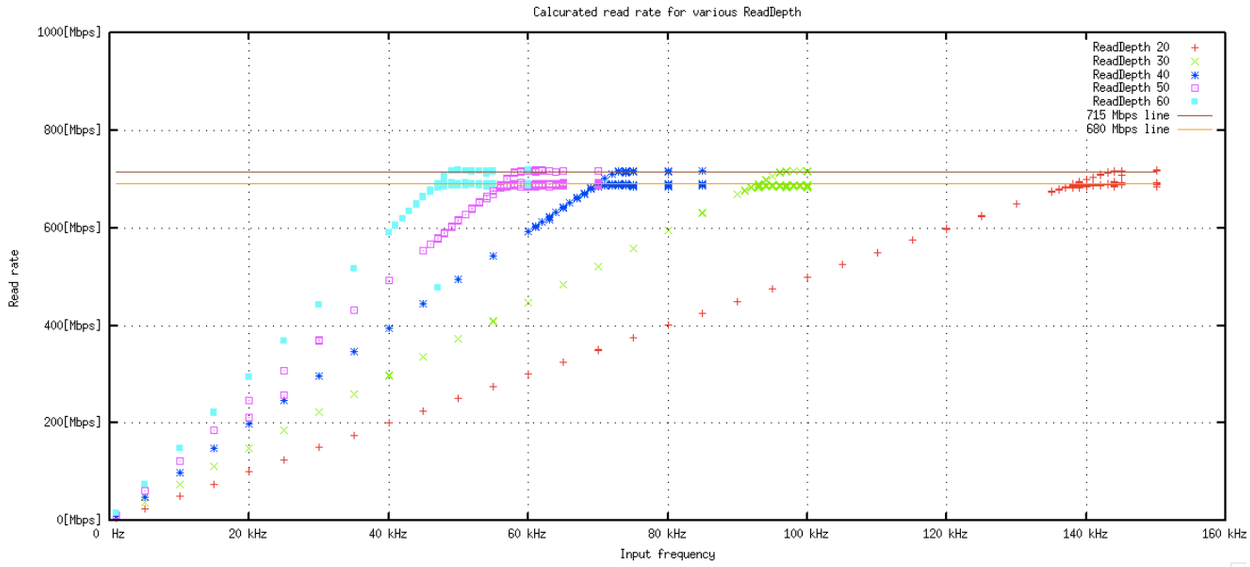


図 6.17　time slice の深さを変化させた時の入力トリガレートに対するスループット

time slice を変化させた性能測定

データの送出の限界が SiTCP が組み込まれた Dragon 内部の FPGA の性能ではなく、ケーブルの帯域に制限されている事を、time slice を変化させた性能測定によって確かめた。

イベントサイズは、１イベント当たり読み込む time slice の深さによって変わってくる。現段階では、読み出す time slice とイベントサイズ X_e の関係は以下の式で表せる。

$$X_e = [(2 \times \text{timeslice}) + 1] \times 16[Byte] \tag{6.2}$$

さまざまな大きさの time slice について、入力トリガレートを変化させてデータ取得プログラムによりデータ取得を行い、データ転送レートを計測した。結果を (図 6.17) に示す。

各 time slice の長さにおいて、はじめは入力トリガレートに比例して転送レートが増加しているが、saturation を起こすポイントは入力トリガレートによらず、一定の転送レートに達したところになっている。このため FPGA の処理能力は限界に達しておらず、Dragon FEB のデータ送信能力は接続するネットワークの帯域に制限されていることがわかる。

なお、この測定では上限のスループットが 718 Mbps 程度となっているが、使用スイッチとデータ取得プログラムを稼働させた PC がデータ収集用マシンと異なっているため[*12]である。

[*12] データ収集用 PC は Ubuntu を直接起動させた iMac No.5 を用い、スイッチは Netgear 社製 GS110TP を 2 つ挟んでいる。

6.5　エミュレータ開発

2014 年現在、Dragon FEB は開発段階に有り、多数の接続によるデータ取得試験、つまりスケールテストができない。そこで、DragonFEB の代用として、ダミーデータ送信機能をエミュレートしたプログラムを作成した。データ取得機能の 1 対 N 試験を始め、以降の項ではこのエミュレータを用いたデータ収集を行ってデータ収集プログラムの機能と性能を評価する。本節ではこのエミュレータプログラムの構造と性能評価を説明する。

6.5.1　FakeFEB と TriggerGenerator

1 台の PC は Dragon FEB と同様に RJ45 ポートを持っており、1 Gbps の通信ができる。これを用いれば、(図 6.26) のように、Dragon FEB の代わりに PC を用いてデータ取得試験ができる。

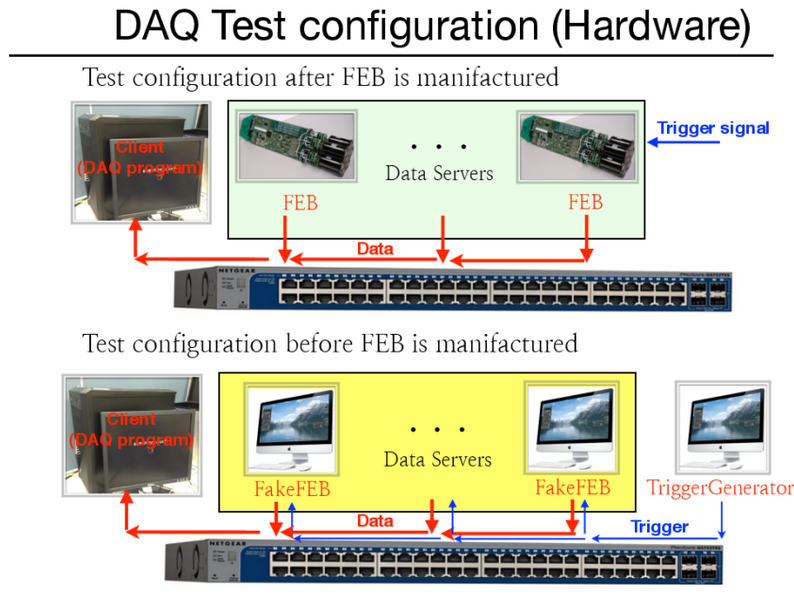


図 6.18　DAQ システム開発環境。FEB 量産後は上のように複数の FEB を利用できる予定。現在は量産前なので下のように PC で機能をエミュレートすることにより対応する。

そこで、Dragon FEB のデータ送信機能をエミュレートするプログラム「FakeFEB」を作成した。Dragon FEB と同様、FakeFEB は TCP/IP の接続要求を受け付け、接続を確立すると、トリガを受信する毎に 1 件ずつ擬似データを接続先に送信し続ける。データのサイズは 976Bytes にしてある。(図 6.19) はプログラムのフローダイアグラムを示している。プログラムを立ち上げると、TCP のサーバーソケットを作成し、データ取得プログラムが接続要求をしてくるポート番号を bind する。このソケットで接続要求を待ち受け、ポートに接続要求が到着すると fork 関数によりプロセスのクローンを生成し、子プロセスがデータ取得プログラムの稼働している IP アドレスにデータを送信し始める一方、親プロセスは子プロセスの終了を待ち受け、プロセス終了を受け取ると接続要求待ちに戻る。

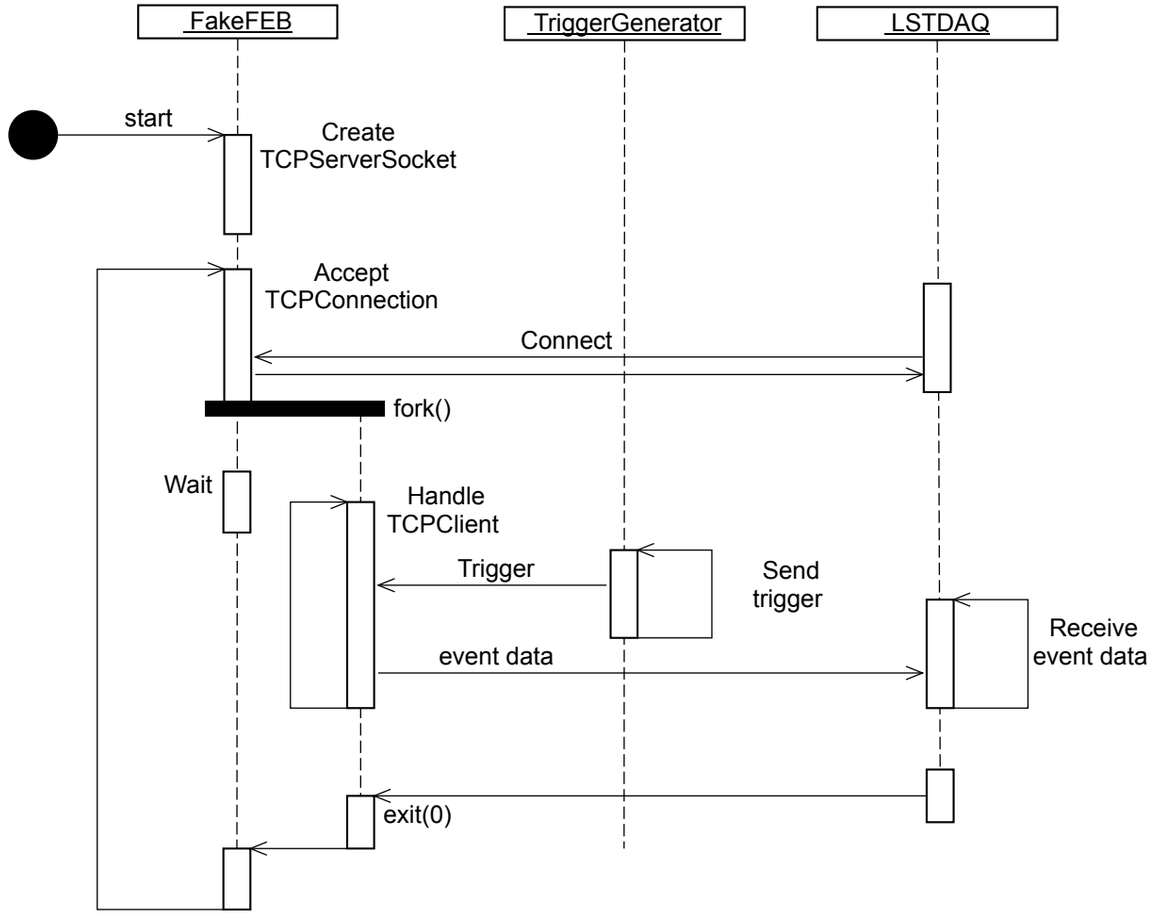


図 6.19　FakeFEB のフローダイアグラム。

Dragon FEB はハードウェアトリガだが、Fake FEB では代わりにブロードキャストパケットをトリガに用いた。これは、全 FakeFEB へ一斉にトリガを配布できるからであり、さらにブロードキャストパケットは UDP パケットのためリアルタイム性があるからである。ただし UDP 通信は到達性の保証がないため、必然的にトリガ受信の信頼性は低い。

FakeFEB は、トリガの待受としてブロードキャストパケットの受け付けを行う。このトリガパケット送信のためのプログラムが TriggerGenerator である。TriggerGenerator は、引数によって与えられた周波数でブロードキャストパケットを送信し続けるプログラムである。(図 6.20) に、TriggerGenerator が送信するトリガパケットの流れと、それを受けて FakeFEB からイベントデータが送られる様子を示した。

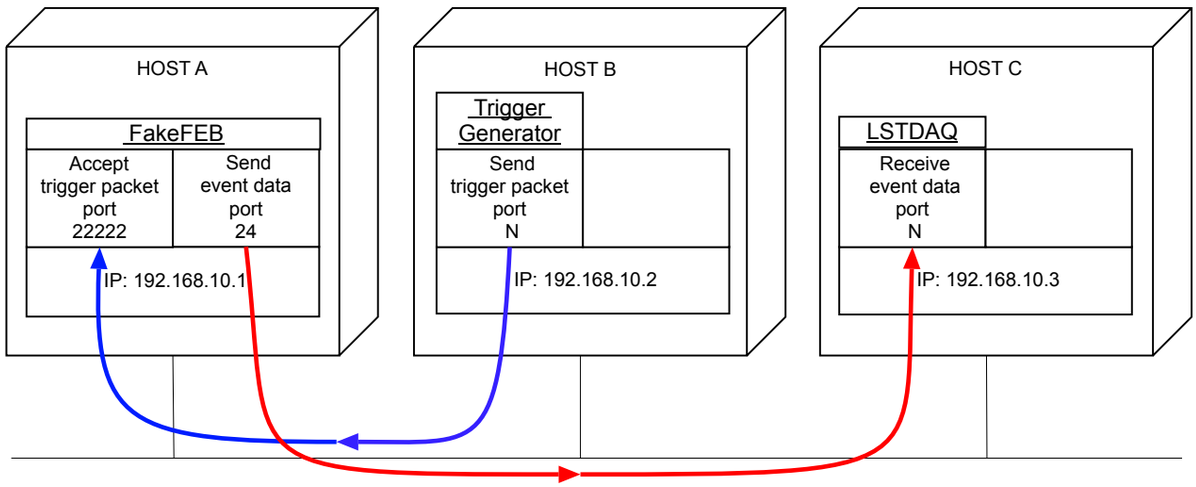


図 6.20　FakeFEB と TriggerGenerator を用いた時の、ネットワークにおけるトリガとパケットの流れ。青矢印はトリガパケット、赤矢印はイベントデータパケットの流れを示す。

6.5.2　トリガ番号とイベント番号

トリガ番号とイベント番号の 2 つのシーケンシャル番号を擬似データの中に埋め込むようにした (図 6.21)。この機能は後述のイベント結合機能のために利用する。これは本研究の機能実装のために仮に設定したもので、現段階ではトリガに対するラベルのフォーマットは決定しておらず、CTA での正式なラベルでは無い。

TriggerGenerator はトリガパケットを送信するが、データ部分には 4 Byte のシーケンシャル番号を埋め込んでいる。この番号は、TriggerGenerator が起動して最初に送信するトリガパケットを 1 として、送信のたびにカウントアップされていく値である。FakeFEB はトリガパケットを受け取った際に、この番号をそのままコピーして擬似イベントデータに埋め込む。

一方でイベント番号は、FakeFEB が送信する一件ごとのデータに付番される 4Byte のシーケンシャル番号で、データ取得プログラムから接続要求を受けたあと、最初にトリガパケットを受け取って送信するイベントデータを 1 として、送信のたびにカウントアップしていく。

トリガパケットは UDP パケットのため到達の信頼性は低く、FakeFEB が受け取ることができなかった場合、トリガ番号の飛びが発生する。

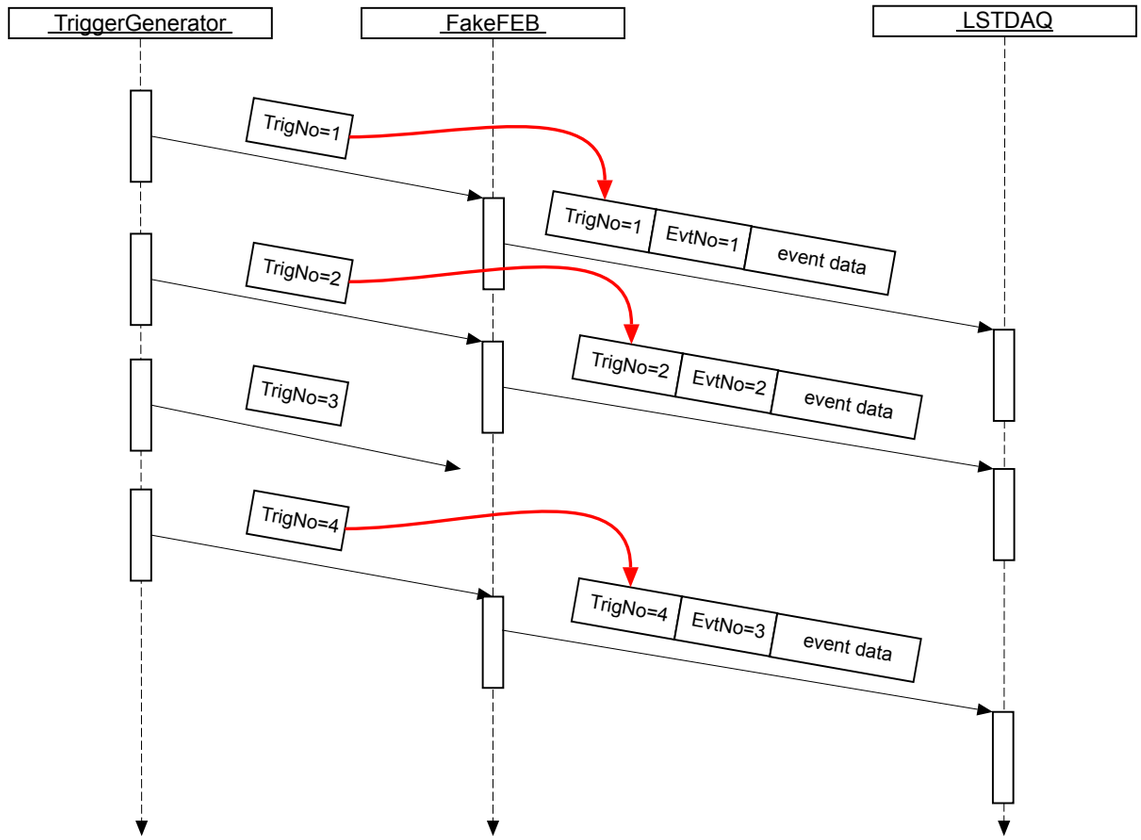


図 6.21 トリガ番号とイベント番号。

6.5.3 プラットフォーム OS の選定

Fake FEB を稼働させるプラットフォームはリアルタイム性を重視して決定した。iMac No.5 で数種類の OS を稼働させ、それぞれで Dragon FEB と ping コマンドで通信させることにより RTT の測定を行った。

この結果 (表 6.5) から、OSX や、OSX 上でバーチャル OS として稼働する状態では送受信の処理が比較的遅く、ばらつきも大きいことがわかる[*13]。よって FakeFEB は、Ubuntu を直接起動させてその上で稼働させることにした。

6.5.4 トリガ送信の負荷

Trigger Generator プログラムはデータ収集プログラムが稼働しているマシン上で共に稼働させることとした。このマシンでは CPU が 6 コアであり余裕があるためだが、確認のために CPU の負荷を測定した。また、

[*13] これは他のアプリケーションのアプリケーションとの競合によって処理が後回しにされていることもある。マウス操作を行うと ping の RTT が遅くなるという状況も確認した。

表 6.5　RTT 測定結果。

OS	平均値 [ms]	標準偏差 [ms]
OSX	0.216	0.0408
Linux on VMware	0.337	0.0979
Linux on VirtualBox	0.342	0.0851
Ubuntu direct boot	0.157	0.0216

TriggerGenerator プログラムは UDP パケットをトリガ信号として送信するため、ネットワークへの負荷も考慮した。

トリガ送信によるシステムへの負荷

TriggerGenerator プログラムをデータ収集マシン上で稼働させ、CPU 使用率を測定した。10 kHz から 120 kHz までの 10 kHz 毎について、プログラム稼働中の CPU 使用率を 100 回計測した結果が (図 6.22) である。この通り、1 コアでの使用のため、残りの 5 コアがデータ収集プログラムに使用でき、問題ない。

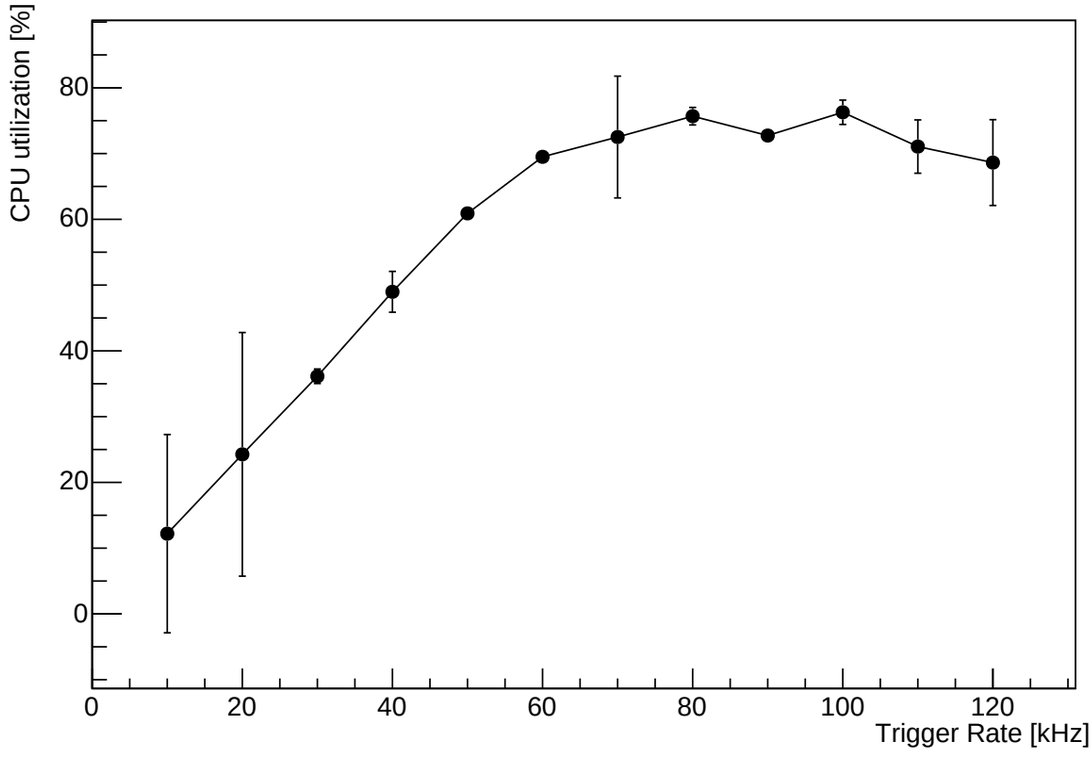


図 6.22　トリガレートを上げた時の CPU 使用率

トリガパケットのネットワークへの負荷

トリガパケットはデータ収集マシンから FakeFEB が稼働する各 PC へブロードキャストパケットとして分配される。反対にイベントデータは FakeFEB の稼働する各 PC よりデータ収集マシンに送られる。ケーブルは全て全二重通信であり、スイッチも十分な能力を持っているため、イベントデータのパケットとトリガパケットがお互いに干渉することはない。

ただし、イベントデータの転送は TCP/IP 通信によって行っている。そのため、各パケットがデータ収集マシンに到着した際に返送する確認応答のパケットがトリガパケットと同一経路となる。そこで、この確認応答パケットへの負担を考慮する。

トリガパケットと確認応答パケット（Acknowledgement packet）の構造を (図 6.23) に示した。トリガパケットの unsigned int 型の 4 バイトのデータを運んでおり、サイズはヘッダを含めて 50 バイトとなる。たとえば 120 kHz の非常に高いトリガレートで送信した場合、パケットのスループットに対する占有率は 1 Gbps 通信の経路で最も高く、

$$\frac{\big(50\mathrm{Bytes} + 20\mathrm{Bytes}(\mathrm{Preamble} + \mathrm{SFD} + \mathrm{IFG})\big) \times 120\mathrm{kHz}}{1\mathrm{Gbps}} = 6.72 \times 10^{-2} \sim 7\% \tag{6.3}$$

一方、確認応答パケットのサイズは 58 バイトである。確認応答は通常、受信したパケット全てに対しては行わないが、MTU1500 バイトの状況下で全てのパケットに確認応答をしたと考えると、1Gbps 通信の経路において最も負荷が高い状況は 120kHz のトリガレートでの確認応答であり、この状況下でのパケットのスループットに対する占有率は、

$$\frac{\big(58\,\mathrm{Bytes} + 20\,\mathrm{Bytes}(\mathrm{Preamble} + \mathrm{SFD} + \mathrm{IFG})\big) \times 120\,\mathrm{kHz}}{1\,\mathrm{Gbps}} = 7.488 \times 10^{-2} \sim 7\,\% \tag{6.4}$$

同様に、10Gbps 通信の経路において最も負荷が高い状況は 10 Gbps の帯域限界までイベントデータが送られた場合の確認応答であり、この状況下でのパケットのスループットに対する占有率は、

$$\frac{\big(58\,\mathrm{Bytes} + 20\,\mathrm{Bytes}(\mathrm{Preamble} + \mathrm{SFD} + \mathrm{IFG})\big) \times \frac{10\,\mathrm{Gbps}}{1518\,\mathrm{Bytes}+20\,\mathrm{Bytes}(\mathrm{Preamble}+\mathrm{SFD}+\mathrm{IFG})}}{10\mathrm{Gbps}} \sim 5.07 \times 10^{-2} \sim 5\% \tag{6.5}$$

よってそれぞれの帯域はトリガパケットによって負担がかかることはない。

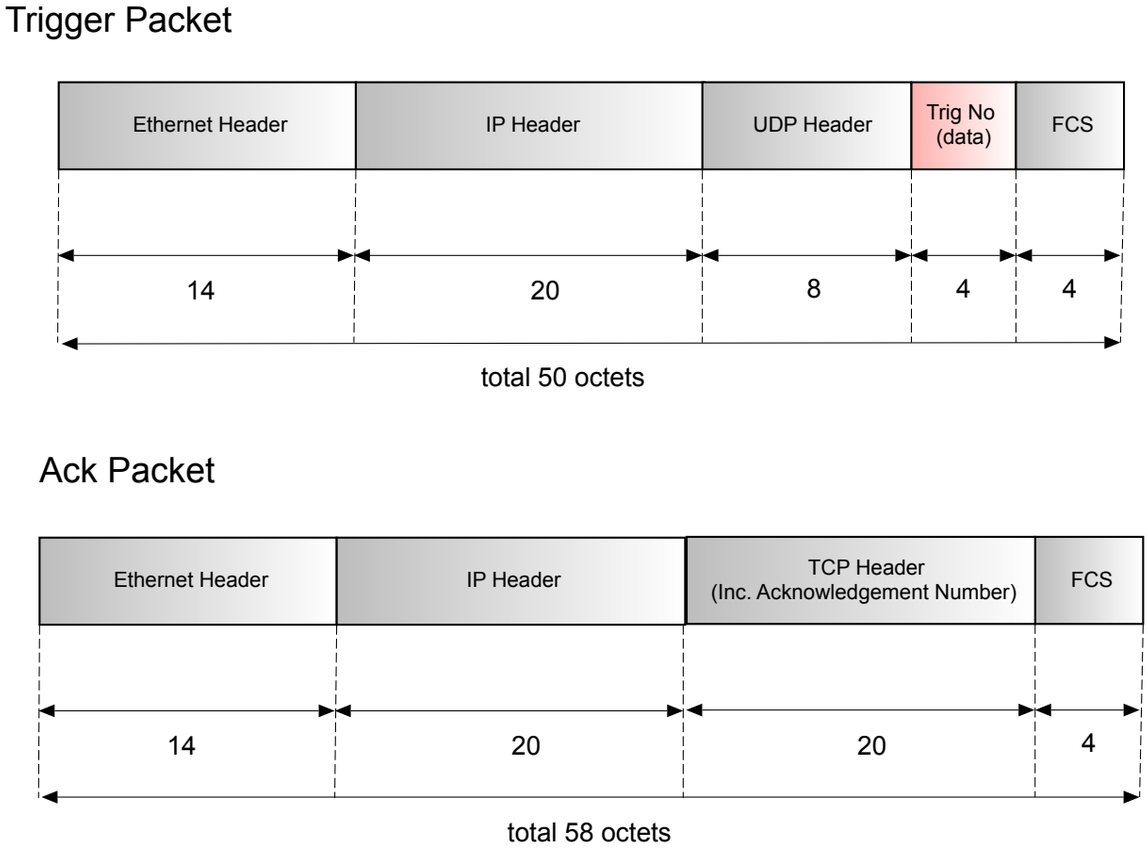


図 6.23　トリガパケットと確認応答パケットの構造。

6.5.5　エミュレータ用ホストの性能差測定

前節の 1 対 1 試験では、Dragon FEB の送信能力は通信ケーブルの帯域に制限を受けていることが確認できた。エミュレータでも同様の制限となることを確認するため、スケールテストで用いた iMac それぞれについて、入力トリガレートを変化させて追随能力を測定した。用いた iMac は (表 6.6) で、OS は OSX を使わず Ubuntu を直接起動させ、その上で FakeFEB プログラムを稼働させている。ONo.1 から No.12 までと No.20 以降とでは性能の差が大きいため、付番はわざと飛ばしてある。また、IP192.168.10.31 で付番する予定だった PC は起動ができなかったため、実験では使用しない。入力トリガレートは 10kHz から 130kHz まで、10kHz 毎に変化させて、データ取得プログラム側でトリガレート ×10 のデータを取得させた。つまり 10 秒ぶんのデータ収集に相当する。

表 6.6 スケールテストで用いた iMac の一覧。Ubuntu13.10 はハードディスクにインストールしたものから、Ubuntu14.04 は USB または DVD に保存した disk image から起動させている。

No.	IP address	Type	CPU	Memory	OS
1	192.168.10.1	iMac Mid 2010 (27inch)	2.8GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
2	192.168.10.2	iMac Late2012 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu13.10 installed
3	192.168.10.3	iMac Late2012 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
4	192.168.10.4	iMac Late2012 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
5	192.168.10.5	iMac Late2013 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu13.10 installed
6	192.168.10.6	iMac Late2013 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
7	192.168.10.7	iMac Late2014 (21.5inch)	3.1GHz Intel Core i7 × 8	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
8	192.168.10.8	iMac Late2014 (21.5inch)	3.1GHz Intel Core i7 × 8	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
9	192.168.10.9	iMac Late2012 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
10	192.168.10.10	iMac Late2013 (21.5inch)	2.7GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
11	192.168.10.11	iMac Late2012 (27inch)	2.9GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
12	192.168.10.12	iMac Late2012 (27inch)	2.9GHz Intel Core i5 × 4	16 GB 1600 MHz DDR3	Ubuntu14.04 USB
20 -	192.168.10.20 -	iMac 9,1 または iMac 8.1	2.66GHz Intel Core 2 Duo	4 GB または 2 GB 800 MHz DDR2	Ubuntu14.04 USB または DVD

性能測定結果

(図 6.24) の通りの結果となった。110kHz までは入力トリガレートに追随してデータ送信を行うが、No.20 以降である IP192.168.10.20 以降と、No.1 である IP192.168.10.1 は 120kHz の入力トリガレートの時に大きく性能が下がっている。よって、以降の実験では基本的に 110kHz までの入力トリガレートで試験を行う。また、大半が通常使用しているものの借用のため、(図 6.26) のように、各回で確保できた PC は台数も個体も様々である。以降、特に個体差に触れる必要がある場合には、(表 6.6) を元に述べる。

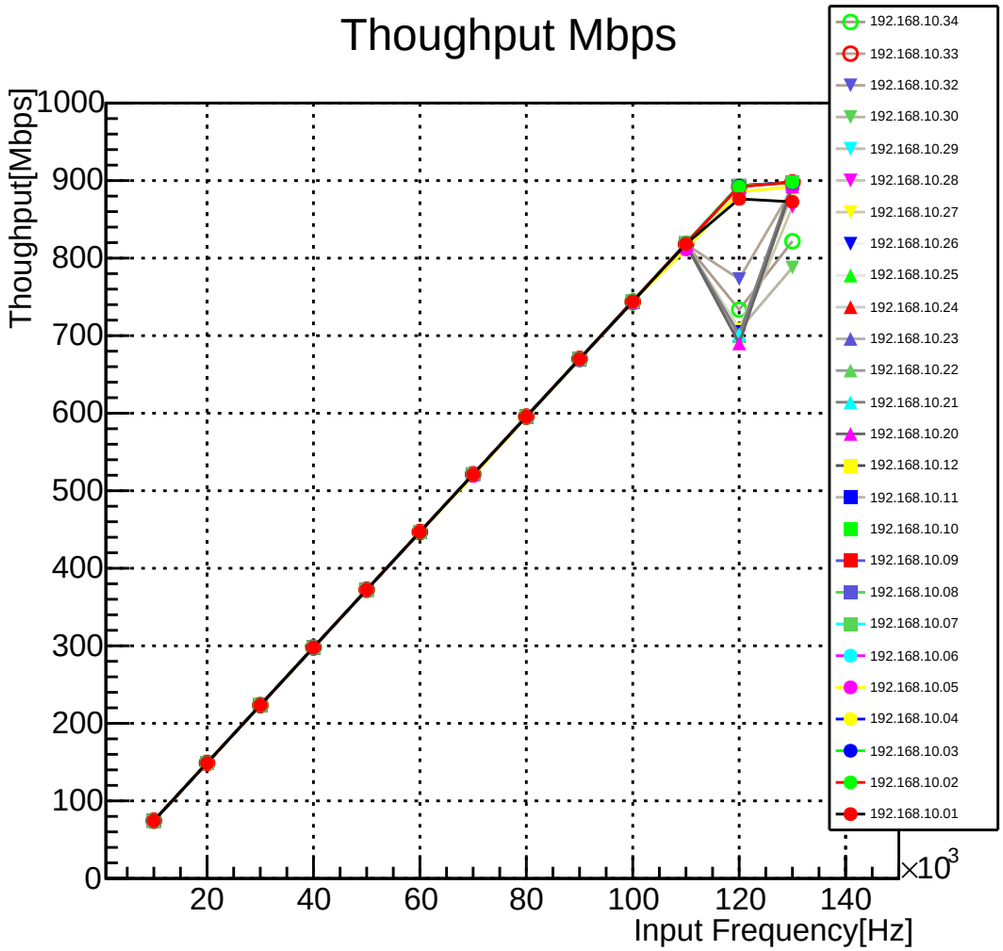


図 6.24　それぞれの PC で稼働する FakeFEB の性能試験結果

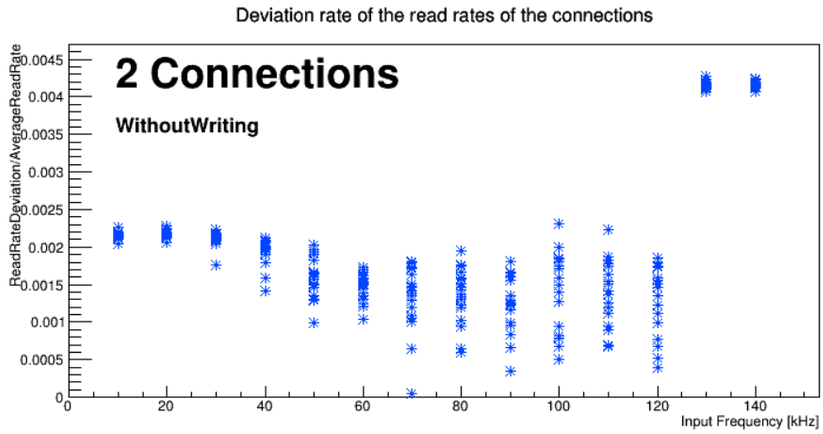


図 6.25　FEB と FakeFEB の性能差測定結果

6.5.6　Dragon FEB との性能差

SiTCP は、FPGA にコードとして実装された TCP/IP 技術のため、(表 6.2) の様な制約が存在する。このことによる PC とのネットワーク性能差は主にネットワークが混雑している状況で大きくなるが、ここでは簡単に入力トリガレートに対する追随の差を測定した。この試験は後述する 1 対 N 試験の一部として行ったので、詳しい試験方法 6.6.2 の Dragon FEB を組み合わせた 1 対 N 試験に記載してある。トリガレートは 10kHz から 140kHz で 10kHz 毎に変化させて、各トリガーレー トについて 20 回ずつ計測した。

得られた結果から、各計測ごとに、読込レートの標準偏差／平均読込レートを計算した。つまり、各計測では時間 t の間に Dragon FEB からのデータは N_d 件、Fake FEB からのデータは N_f 件取得できたとして、平均読み込みレート $< N > /t$ とすると、

$$\sqrt{\frac{(N_d/t - < N > /t)^2 + (N_f/t - < N > /t)^2}{2}} \Big/ \frac{< N >}{t}$$

である。

(図 6.25) にその結果を表した。2 つの取得レートの差は 0.25% 以内に収まっていると言える。

6.6　1 対 N 試験（データ取得性能確認）

この節では、接続する FEB の数を増やし複数の FEB からのデータ取得を同時に行った結果について述べる。1 台のコンピュータ上のデータ収集プロセス一つで複数の FEB からのデータ収集を行うため、1 対 N 試験と呼ぶことにする。試験の目的は、エミュレーターの接続台数やトリガレートを変化させてデータの取得性能を評価することである。この試験では最大で 22 台の PC を確保できた。また、Dragon FEB を 1 台確保できている時に 7 台の PC を同時に確保できたので、Dragon FEB と PC の組み合わせでも試験を行った (図 6.26)。

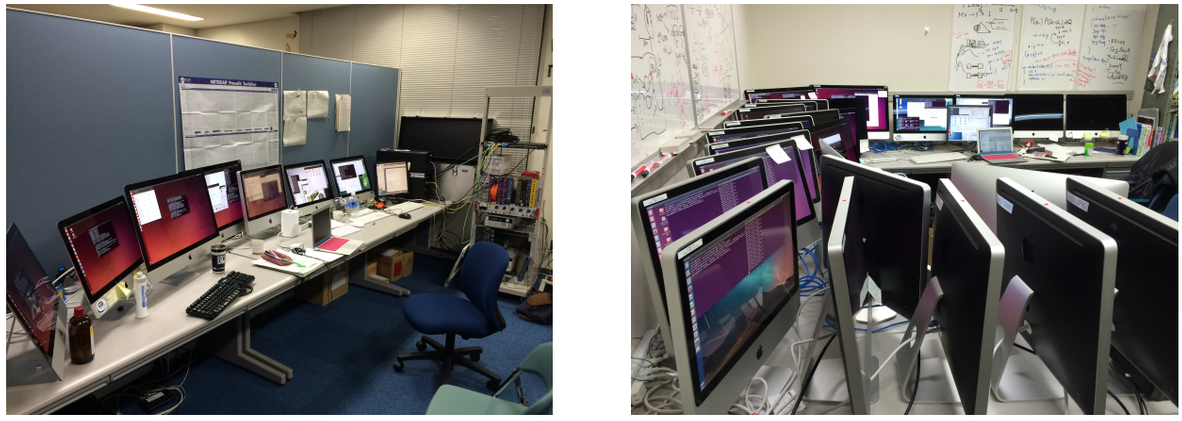


図 6.26　スケールテストの様子。右：エミュレータのみの試験、左：Dragon FEB を組み合わせた試験。DragonFEB は写真奥の箱のなかに設置されている。

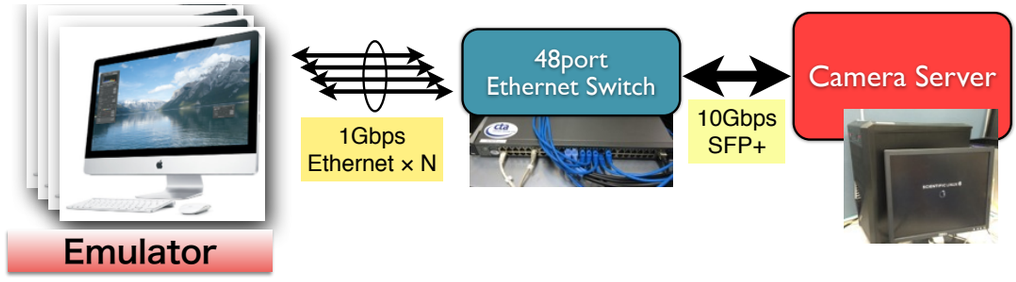


図 6.27　1 対 N 試験の接続構成

6.6.1　エミュレータのみの試験

試験方法

接続構成は (図 6.27) に示す通り、各エミュレータ用 PC とスイッチを 1Gbps ケーブルでつなぎ、カメラサーバーとスイッチを 10Gbps の DA ケーブルでつないである。トリガはカメラサーバーより分配させた。プログラムは今回作成したデータ取得プログラムを使用したが、select 関数の効果を検証するために順次読込型のデータ取得も行い、結果を比較した。

1 対 1 から 1 対 22 接続のそれぞれで、10kHz から 110kHz まで 10kHz 刻みでトリガレートを上げていき、各トリガレートについて 5 回ずつ測定を行った。各測定は、10 秒間に相当するデータ取得を要求させ、かかった時間を記録する。

性能測定結果（select 使用 FD 参照型読込）

結果は (図 6.28) のようになった。10 台程度の接続までは 110kHz のトリガレートに追随してデータを取ることができる。しかしそれより接続を多くすると、高いトリガ頻度に追随できず saturation を起こす。このことをわかりやすくするため、(図 6.29) のように表し直した。この図は、横軸は接続数、縦軸はある入力レートでトリガをかけてデータ取得を行った時のデータ取得頻度を表している。それぞれのトリガ頻度に対し、接続数を増やしていくと、ある台数を超えたところで入力トリガ頻度に追随できず取得頻度が下がっていくことが見える。また、スループットが一番高かったのは 12 接続で 110kHz をのトリガレートを入れた時で約 9.4Gbps

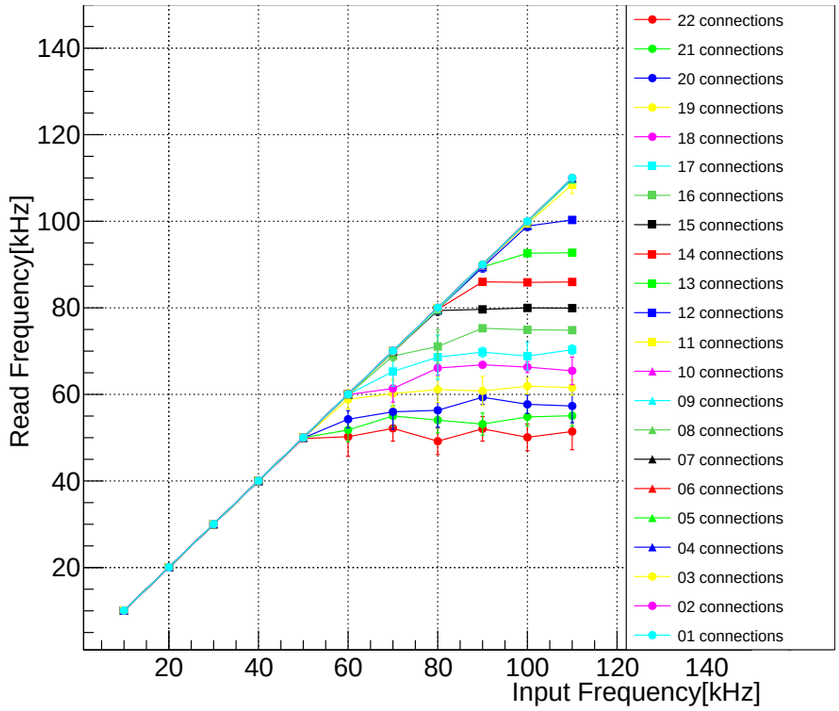


図 6.28　Select を用いた複数接続データ取得結果。各接続台数において、トリガ頻度を上げていった時のデータ取得頻度

となった。この値は 10Gbps のほぼ上限と言える。

このようにデータ取得能力は伝送路の帯域に大きく依存している。本実験での接続構成から 1 接続当たりの帯域を考えると、10 接続までは 1Gbps で、以降 N 接続では $\frac{10\mathrm{Gbps}}{\mathrm{N}}$ となるはずである (図 6.30)。実際に追随できなくなって以降の各点を定数＋反比例の形をした関数でフィットを行うとほぼ全て一つの曲線にのり、接続数 N に対して、

$$f(N) = a + b/N = \frac{aN + b}{N} \tag{6.6}$$
$$a = -5.315 \times 10^3, \quad b = 1.276 \times 10^6$$

の曲線でフィットできた。

a の値は負になっていること、b に比べて 3 桁も小さいことから、接続が多くなることによるオーバーヘッドが発生していると考えられる。しかし、b に比べて微小なので、この影響については考えない。

CTA におけるトリガレートの要求値は 15kHz である。カメラサーバーを複数に分担しスイッチ 1 台分の接続を受け持った場合、45 接続を担当することになる。フィットした関数の延長上は、この 15kHz よりも高くなるので、要求値は満たすと考えられる。

性能測定結果（select 不使用順次読込）

試験結果は (図 6.31) となった。select() を用いた FD 参照型読み出しでは、ファイル記述子を参照することでソケットにデータが到着しているかどうかを知ることができ、それによって余分な read システムコールを避ける事ができる。単純な順次読込型読み出しでは、データが到着しているかどうかにかかわらず、各接続のソ

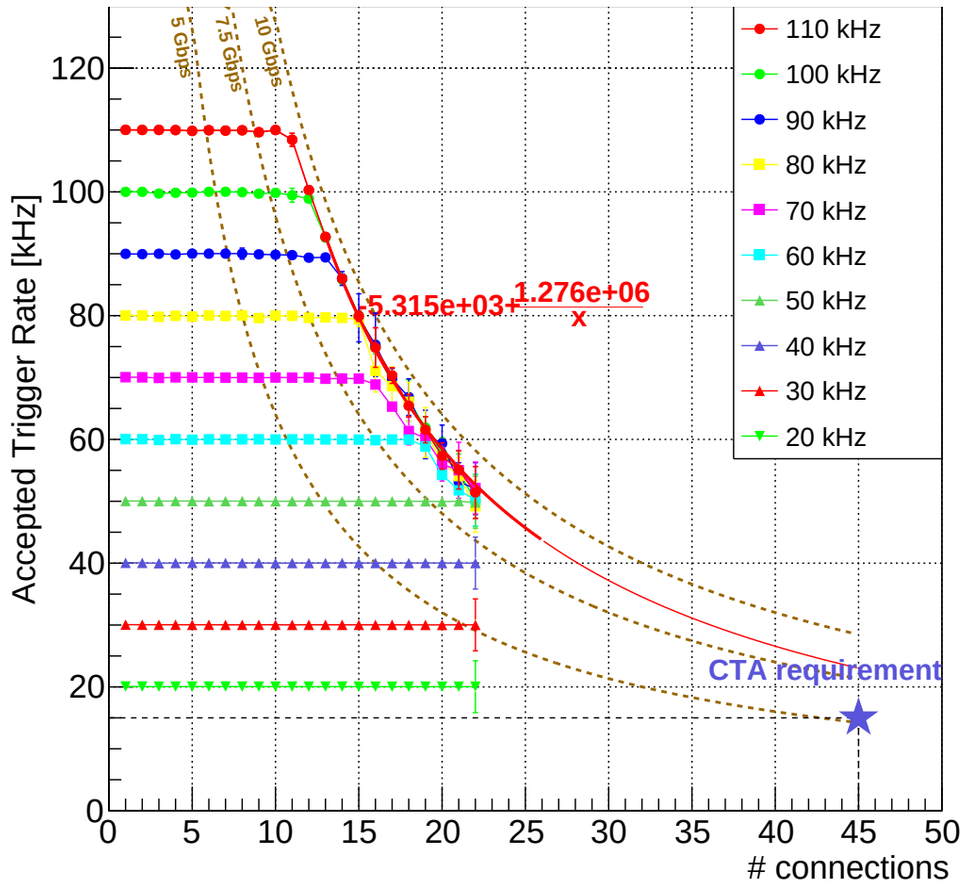


図 6.29　Select を用いた複数接続データ取得結果。形と色で分類された点を結ぶ各線は、一定の入力トリガレートで接続台数を増やした時のデータ取得頻度の変化を表す。茶色の各破線は、それぞれ付記された帯域を仮定した場合のデータ取得頻度。

ケットを順番に読み込み続ける。

この違いが表れていると思われるのが、50kHz から 110kHz での入力トリガレートについて、6 台程度以上、追随できなくなるまでの台数の接続をした部分である。ある程度の接続数で高頻度のトリガがかかると、空のソケットを読んでしまったことが原因で追随できなくなるようである。

ただし、多数の接続になった場合は select() 関数での読み込みとほぼ変わらない様子が見えた。多数の接続では、到着したデータを全接続で読み込んで回っているうちに次のデータが到着しているものと思われる。

また、入力トリガレートに追随できなくなった後の取得レートの変化をフィットした結果は、反比例の項はほぼ同じだが、定数のオフセット項は 67% 程度の小さな値になった。ループの各回でファイル記述子の代入や参照を行う select 関数より、順次読込型のデータ取得ではオーバーヘッドが少ないことが原因と思われる。しかし、反比例の項に比べて小さい上、実際のトリガはランダムにかかるため、以降のデータ結合機能の実装においては、select 関数を実装したデータ取得を行うことにする。

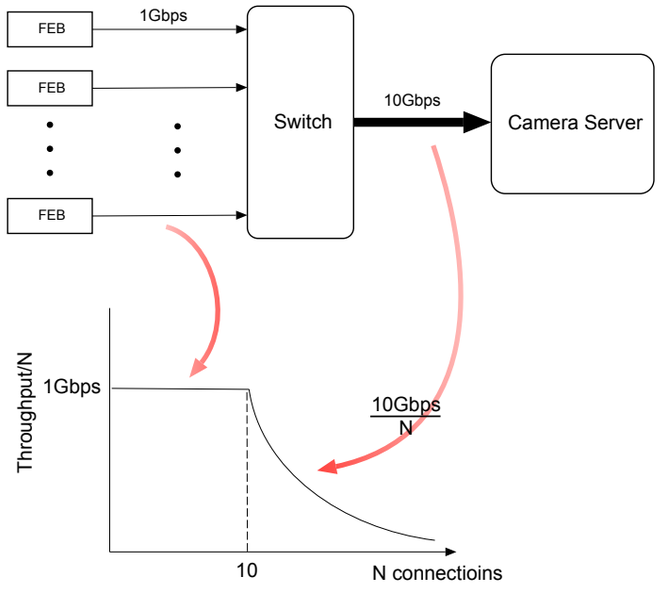


図 6.30　ケーブルの帯域に制限された場合の 1 接続当たりの転送レート。

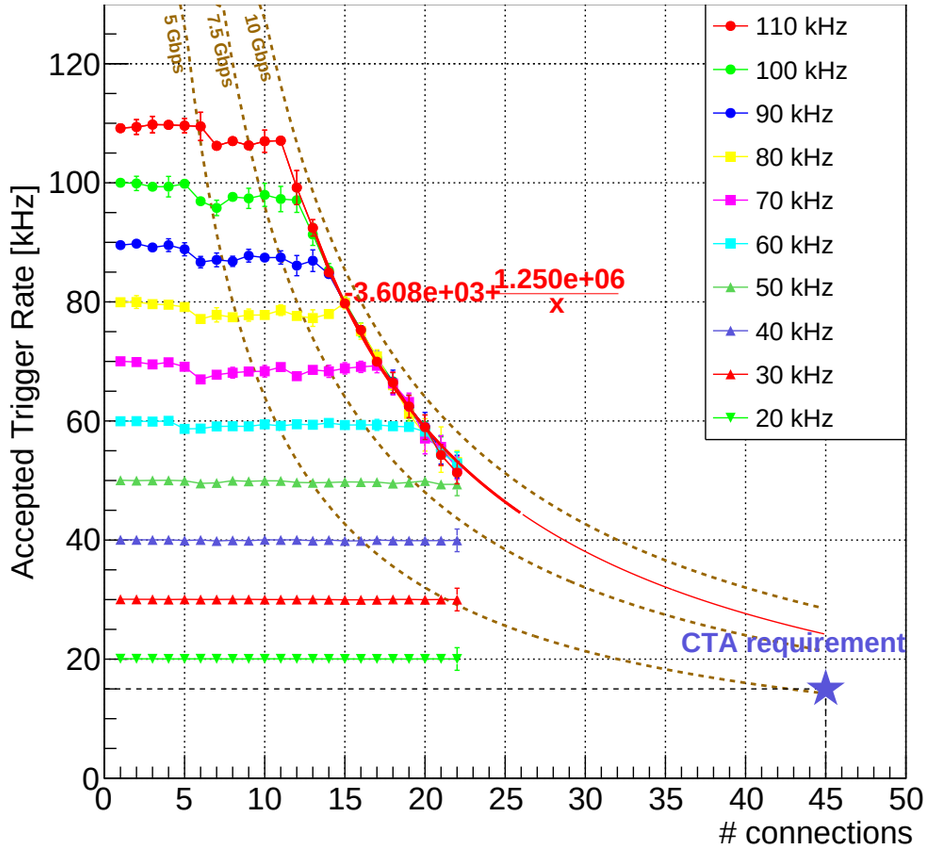


図 6.31　順次読込による複数接続データ取得結果。形と色で分類された点を結ぶ各線は、一定の入力トリガレートで接続台数を増やした時のデータ取得頻度の変化を表す。

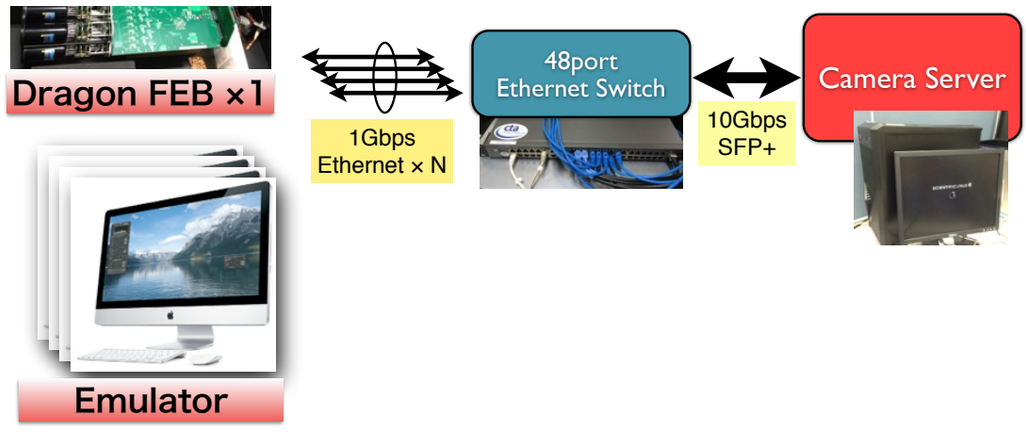


図 6.32　Dragon FEB を組み合わせた 1 対 N 試験の接続構成。Emulator はトリガパケットによるトリガだが、Dragon FEB には外部トリガをかけている。

6.6.2　Dragon FEB を組み合わせた試験

試験方法

Dragon FEB 1 枚と PC 7 台の計 8 接続を (図 6.32) のように構成して試験を行った。各 PC または FEB とスイッチを 1Gbps ケーブルでつなぎ、データ収集マシンとスイッチを 10Gbps の SFP+ 通信 DA ケーブルでつないである。

カメラサーバー上でデータ取得プログラムを稼働させて FEB または各エミュレータのプログラムと接続し、トリガを掛けて並列にデータを取得した。測定は 1 対 1 試験と同様に、Connect 関数実行後 500μs の sleep を挟んで計測開始とし、10^8 バイトを超えた時点でイベントサイズの整数倍になるように残りのデータを読んで測定終了とし、プログラムを終了する。この計測を使用し、読み込んだデータ件数とその所要時間からデータ取得頻度を計算する。

エミュレータに送信するトリガはエミュレータのみの試験同様にデータ収集マシンより分配させたが、Dragon FEB はパルスジェネレータのアナログ信号によるハードウェアトリガが必要であり、トリガの完全な同期ができない。そのためトリガレートがなるべく同じになるように、パルスジェネレータと TriggerGenerator プログラム双方に同じトリガレートを指定してトリガを発生させてから計測を行った。

試験項目は接続については 1 対 1 から 1 対 8 までで、それぞれの接続について 10kHz から 120kHz まで 10kHz 刻みである。各接続とトリガレートについて測定は 20 回ずつ行った。1 対 1 では Dragon FEB からのデータ取得とし、それ以降は 1 対 8 接続まで PC の接続を増やしていった。

試験結果

結果は (図 6.33) のようになった。前述のようなフィットはできないが、以下の考察を行った。

- 110kHz の入力トリガレートは、接続数 4 台程度で既に追随できず下がり始めている。接続台数 8 台では 70kHz まで低下している。これはエミュレータのみで試験した場合の 60% であり、ネットワークで解決するべき問題があると思われる。パケット解析を含めた詳細な調査が必要である。

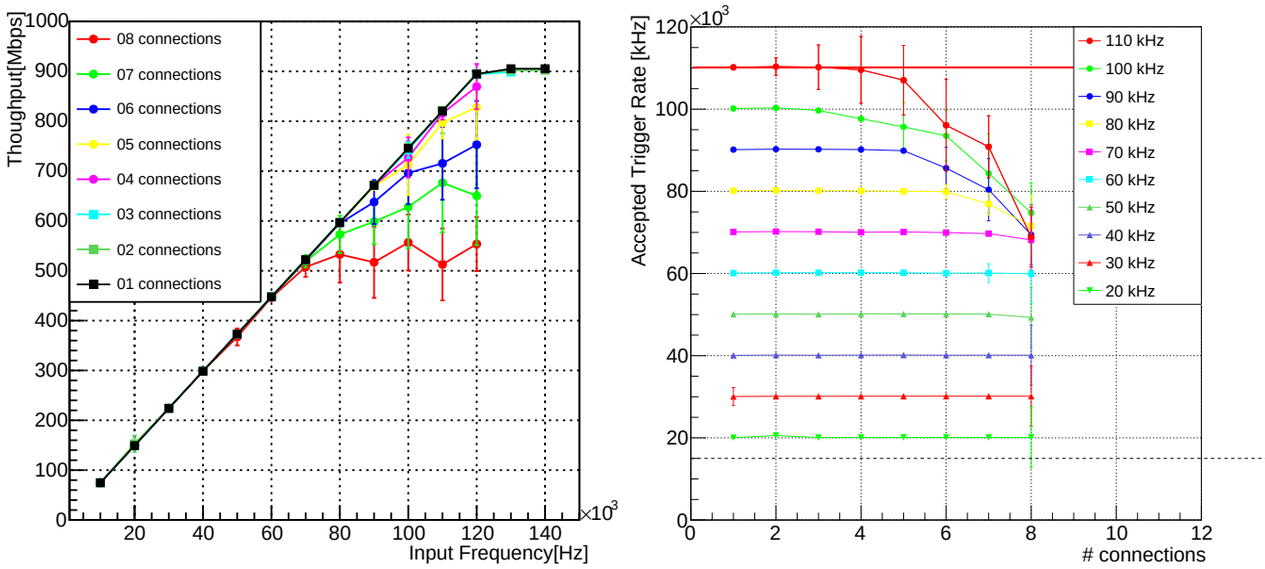


図 6.33　Dragon FEB 1 枚を含んだ 8 接続での試験結果

- データ取得プログラム側でのイベントの総読み込み回数/s を比較することで、接続数を 45 台にスケールさせた場合に要求値 (15kHz) まで追随可能かを簡単に評価した。つまり、総読み込み回数/s = (FEB1 台あたりから達成できたデータ取得レート) × (接続台数) を定義すると、15kHz × 45 台 = 675kHz の総読み込み回数/s まで追随していれば、45 台接続させた時にも同等の性能を発揮できる可能性がある。7 台との接続では約 93% を達成しているが、8 台の接続では約 89% に低下してい る。接続数が増えるほど、負荷がかかってしまっている。このため、このまま 45 台までスケールさせて予測することはできない。

よって、再度 Dragon FEB を用いたネットワークの調査が必要である。

6.7　イベント結合機能の開発

カメラサーバーは多数の FEB からデータを取得するが、物理的な接続はスイッチによってまとめられ、一本ないし数本程度の 10 Gbps ケーブルで受け取る。そのためデータ到着は非同期であり、同一のトリガによって各 FEB から取り出されたデータを結合するためには、非同期から発生するデータ到着時間のずれを吸収する必要がある。これをリングバッファを実装することで実現した。

また、各接続から到着したデータをリングバッファに格納するという作業が追加されるため、データ取得プログラムにはより多くの負荷がかかり、収集性能が低下する。それだけではなく、同一トリガによって送信された全接続からのデータを結合し、ひとつのイベントとして保存、監視、解析を行わなければならない。

処理能力を低下させず実現する効果的な方法はマルチスレッドプログラミングである。この方法により、それぞれの処理に CPU を割り当てることによって、並行処理を可能にする。この節では、マルチスレッドの効果とその応用法、そしてリングバッファの機能について述べた後、これらを用いて実装を行ったイベント結合機能について説明する。

6.7.1　マルチスレッド

イベント結合機能は受信バッファからデータを読み出す処理に負担を与えずに並行に働く必要がある。また受信バッファからデータを読みだす処理は、接続先が多数になり負担が大きくなる場合は並列化による負担の軽減を検討する必要がある。マルチスレッドはそれぞれの処理をスレッドに分割することで、各処理にコアを割り当てて並行処理を実現させることが可能である。この節では並列処理と並行処理の概念、それを実現するマルチプロセスとマルチスレッド、そしてマルチスレッドの中でも一般的な規格である Pthreads について説明する。

並列可能性

プログラムが複数の処理を行うとき、順序に関係なく処理を実行できる部分を含んでいることがある。このような部分は「並列可能性 (potential parallelism) と呼ばれている。このような並列可能性のある部分は、マルチプロセッサシステム上において、プログラムの実行を高速化することができる。しかし、並列可能性はこれだけにとどまらない。たとえば I/O がオーバーラップを起こしている時である。I/O システムコールが完了するまでブロックされるタスクがある場合、これを待ち続ける一方で CPU 集約型のタスクを独立して実行し続けられればパフォーマンスの向上が見込める。また、非同期イベントも並列可能性を持つ。不確定なイベント、つまりネットワーク通信のように間隔も頻度もわからないイベントの発生を仮定しているタスクを含む場合、そのタスクの進行状態にかかわらず別のタスクを続行するようにすれば効率がより向上する。

並列プログラミングと並行プログラミング

並行プログラミングは、特に異なるプロセッサ上で並行タスクが同時に実行されることをいう。並列プログラミングは、「順序に関係なく処理を実行できる」という並列可能性を実現するプログラミングであり、同じプロセッサ上で処理を切り替えることで処理を実現する、「並列処理」も指す。したがって、並行プログラミングは常に並列プログラミングであるが、並列プログラミングがすべて並行プログラミングとなるわけではない。

並行プログラミング環境

並行プログラミング環境には大きく分けてマルチプロセスとマルチスレッドがあるが、複数のスレッドと複数のプロセスで比べると、プログラムの実行に伴うシステムのオーバーヘッドはスレッドのほうが少ない。つまり、マルチスレッドプログラムのためのオペレーティングシステムの作業量は、マルチプロセスプログラムのための作業量より少ない。マルチプロセスにおいて余分に発生する作業は以下のような点があげられる。

- プロセスの作成は、オペレーティングシステムの呼び出し必要であり、さらにプロセスの作成がプロセスの再スケジューリングを引き起こす場合、オペレーティングシステムのコンテキスト切り替え機構も関与してくるために、時間がかかる。
- マルチプロセスは、プロセスを生成する際、生成元のプロセス全体を複製して新しいプロセスを作成する fork() コールが必要となる。そのため時間だけでなくメモリ資源も多く消費する。
- プロセス同士でデータを送受信する手段は、パイプ、共有メモリといった煩雑なプロセス間通信が必要になる。また、これらの同期処理にシステムコール（オペレーティングシステムの呼び出し）が関与するため、リソースを多く消費する。

これに対し、スレッドは、プロセス全体を複製することなく作成できる。また、スレッドの作成作業はカーネルが関与せずユーザー空間だけで行えることが多い。さらにスレッド間の同期は、単に変数を監視することで行えるため、ユーザーアドレス空間にとどまったままでいられる。よって、並行プログラミング環境を実現する技術は、マルチスレッドによるプログラム実装が望ましい。

Pthreads

Pthreads は、プログラムをサブタスクに分割し、各サブタスクを並列に、あるいはインターリーブして実行するという仕組みの標準化されたモデルである。Pthreads の P は POSIX(Portable Operating System Interface) から来ている。 POSIX は、IEEE のオペレーティングシステムインターフェース規格のファミリで、Pthreads もここで「POSIX Section 1003.1c」として定義されている。ベンダーはプログラムにインクルードするヘッダファイルと、プログラムにリンクするライブラリという形で Pthreads を実装している。[21][12][18]

(図 6.34) は Pthread を用いた単純なマルチスレッド処理のシーケンス図である。Pthreads においてスレッドの生成は pthread_create によって行う。生成元のスレッドは pthread_join によって生成したスレッドの待ち状態に入り、各スレッドの処理が終わると処理が続行される。

また、スレッド間での同期のために、pthread_mutex という相互排除変数と、pthread_cond という条件変数が用意されている。(図 6.35) は、相互排除変数 mutex を用いた排他制御のシーケンス図である。スレッド 1 とスレッド 2 が同じ mutex を持っている状況を仮定している。スレッド 1 が mutex に対して lock をかけると、他のスレッドは lock をかけてもサスペンドされるようになっている。スレッド 1 が何らかの処理を終えて mutex の lock を外すと、他のスレッドは lock を掛けられるようになっている。この仕組は、リングバッファの上書きを伴う場面で使用する。

(図 6.36) は、条件変数 cond を用いた処理待ちのシーケンス図である。スレッド 1 が mutex の lock を獲得して処理を行っている際に、スレッド 2 によって lock を掛けてなされる処理が必要になった場合、スレッド 1 は pthread_cond_wait 関数を呼び出す。この関数は一旦 mutex の lock を引き取り、スレッド 1 に代わって開放する。スレッド 2 はこのおかげで mutex に対して lock を掛けられるようになる。スレッド 2 が lock を外した際に、cond_signal 関数を呼び出しておけば、cond 条件変数は mutex の lock とともにそのシグナルを受け

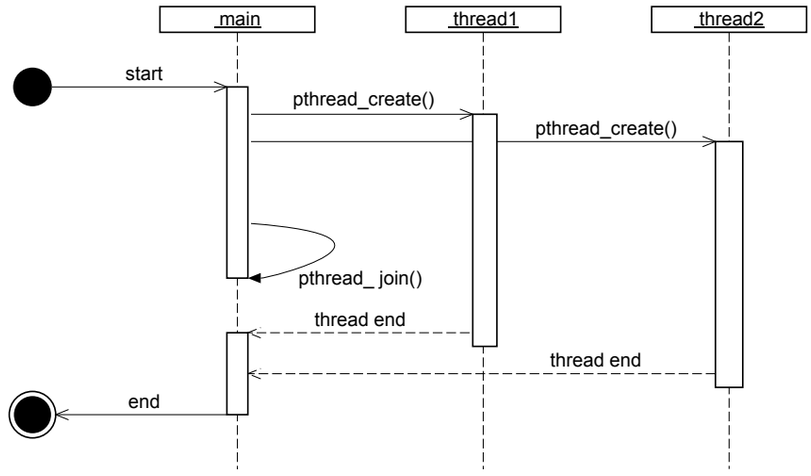


図 6.34　Pthreads におけるスレッドの生成と終了のシーケンス図。

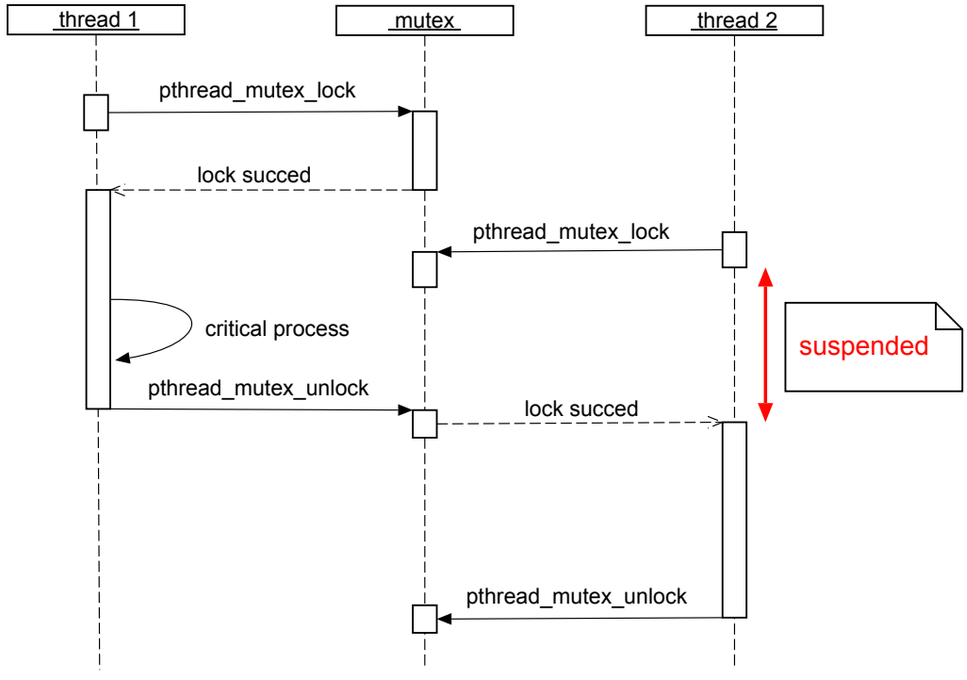


図 6.35　mutex を用いた相互排除の仕組み。

取り、mutex の lock をスレッド 1 に再び明け渡す。この仕組は、全スレッドのデータ収集処理スタート同期と、RingBuffer の上書き防止処理で使用する。

6.7.2　RingBuffer

循環バッファ（circular buffer）とも呼ばれ、配列の先頭と末尾を概念的に連結した環のようなデータ構造をもつ。First In First Out (FIFO) のデータ受け渡し場面では必須のデータ構造であり、Dragon FEB においても DRS4 チップ内でこの実装が取り入れられている。バッファを物理的にリング状に配置することはできな

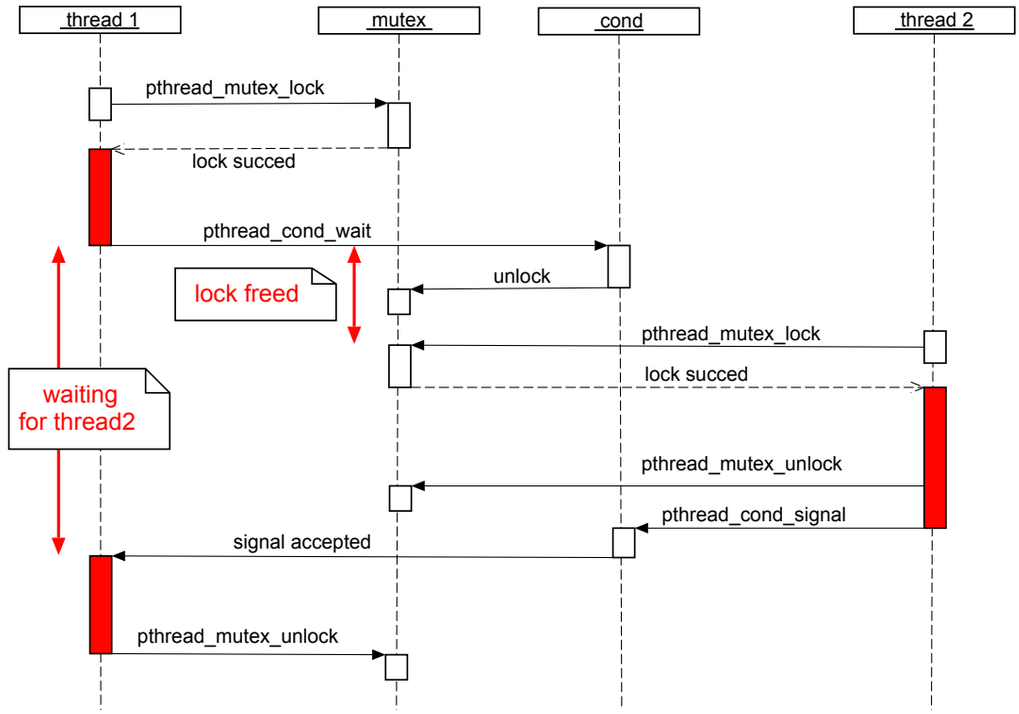


図 6.36　Pthreads における条件変数を用いた制御のシーケンス図。

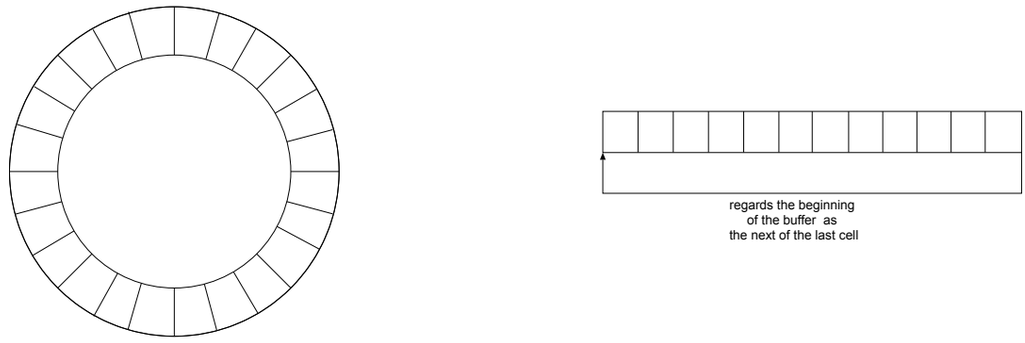


図 6.37　左：リングバッファの概念図。右：リングバッファを実現するメモリの構造。

いので、実際にプログラムで再現するには、配列を (図 6.37) のように論理的につなげて使用する。

書込と読出

リングバッファ構造を実装した配列は、(図 6.38) のような流れで書込と読出の処理を受け入れていく。データが書き込まれていく。書込済のイベント番号を、簡単のため 1 から順番に付番してある。読出し側は、書込が終わったデータが発生すると、それを追うようにして読出しを行う。図中では読出しが終わったイベントの数字を薄くしてある。

これらの処理の循環が続き、バッファを 1 周分使いきってしまった時、書込のポインタは配列の始点にもど

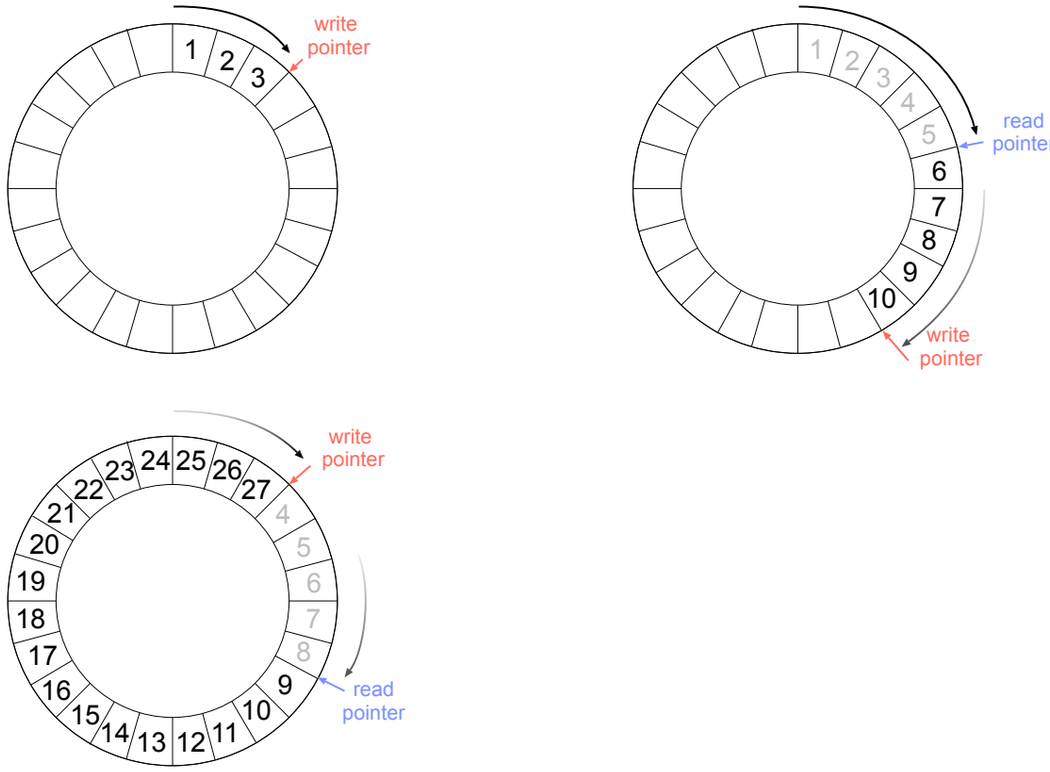


図 6.38　上左：書込処理スタート時。上右：書込済のイベントを読み取っていく処理。下：書込処理による読込済データの上書き。

り処理を続行する。このとき、自分が書き込んだデータの上書きを行うため、読出しが行われていないイベントの上書きを防がなくては行けない。

6.7.3　プログラムへの実装

以上の機能を用いて、「Collector スレッド」（データ取得スレッド）と「Builder スレッド」（データ結合スレッド）を作成した。Collector スレッドから Builder スレッドへのデータの受け渡しのためにリングバッファを作成した。(図 6.39) はプログラムの全体構造を表している。Collector スレッドが各 Dragon FEB と TCP/IP 接続を行い、データを RingBuffer に格納していく。Builder スレッドは各 RingBuffer から 1 件ずつデータを読み、同一トリガのイベントデータを結合して後段の処理のために用意した RingBuffer に格納する。その RingBuffer に書き込まれている結合済みデータを on-site ストレージに保存するスレッドと低レベル解析を行って監視データに転送するスレッドが読み込む。本実験ではこの内 Collector スレッドと Builder スレッド、そしてその間のデータの受け渡しのバッファであるリングバッファの構築を行った。ここでは、これらの機能を説明する。

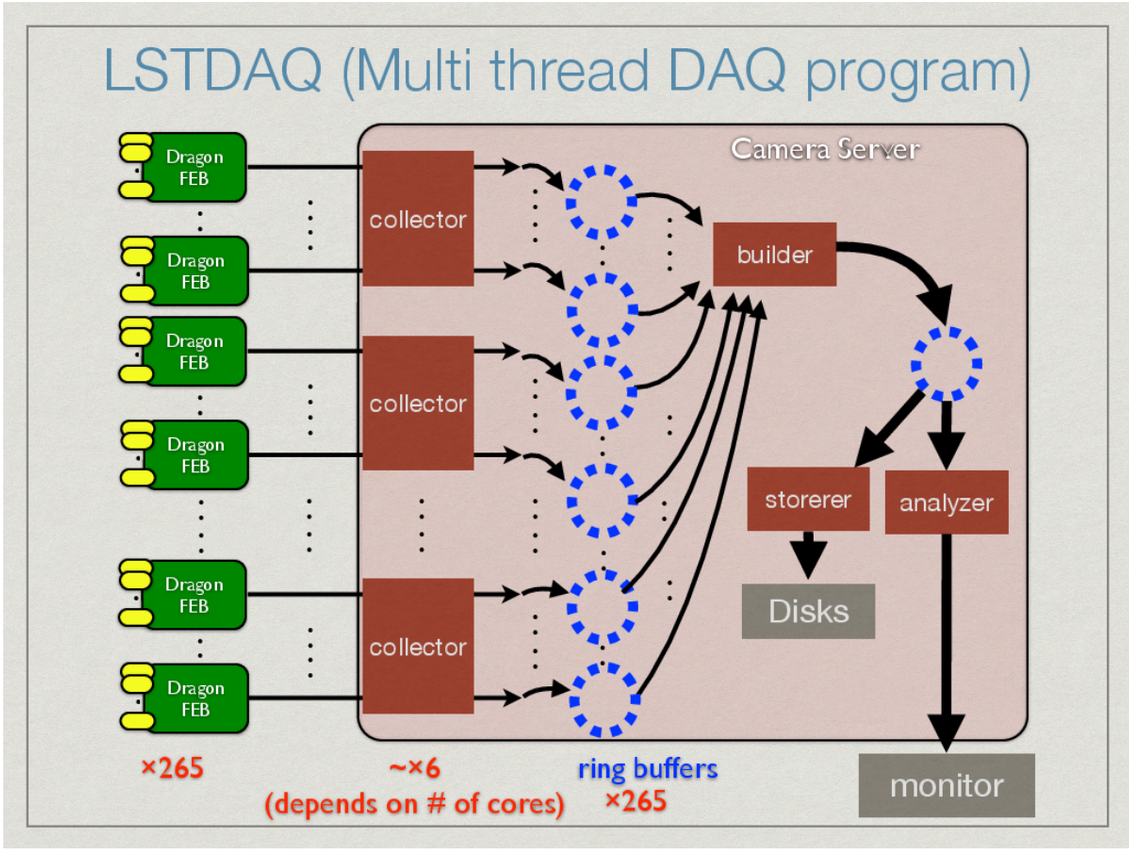


図 6.39　LSTDAQ のプログラム構造

Collector スレッド

Collector スレッドは Dragon FEB と TCP/IP 通信で接続を行い、ソケット通信でデータを取得するスレッドである。一つのスレッドで複数の接続を担当し、各接続からソケットに到着するデータをま並列に読み出すために、前述の select 関数を用いている。また、到着したデータを各接続毎に用意したリングバッファに格納する。この作業が追加されるため、Collector スレッドは単純なデータ取得プログラムより多くの負荷がかかり、収集性能が低下する。この問題を補うため、Collector スレッドは複数立ち上がるようになっている。生成するべき適切なスレッド数は接続数によって異なる。接続する Dragon FEB の台数に対して適切な Collector スレッド数を見積もるため、プログラムの立ち上げ時に設定ファイルを読み込んで、設定ファイルで指定された通りのスレッド数が生成されるようにしてある。このスレッド数の見積もりのための性能測定と評価を行ったが、これは性能測定の項にて後述する。

Collector スレッドのフローチャートを (図 6.40) に示した。まず自スレッドへの CPU の割り当てを行った後、担当する接続先と、それぞれの接続先のために使用するリングバッファがを把握する。その後、各接続先へ接続の確率を行った後、データの取得に入る。各回、select() の発行により、データが到着しているソケットを判断する。データが到着していればソケットを読みに行き、一回の読み込みで得られたデータのサイズがイベントのサイズに満たなければ、達するまで残りのデータの読み込みを繰り返す。

1 イベント分のデータが得られたところでリングバッファへの書き込みを行い、次のデータ到着ソケットへ移る。

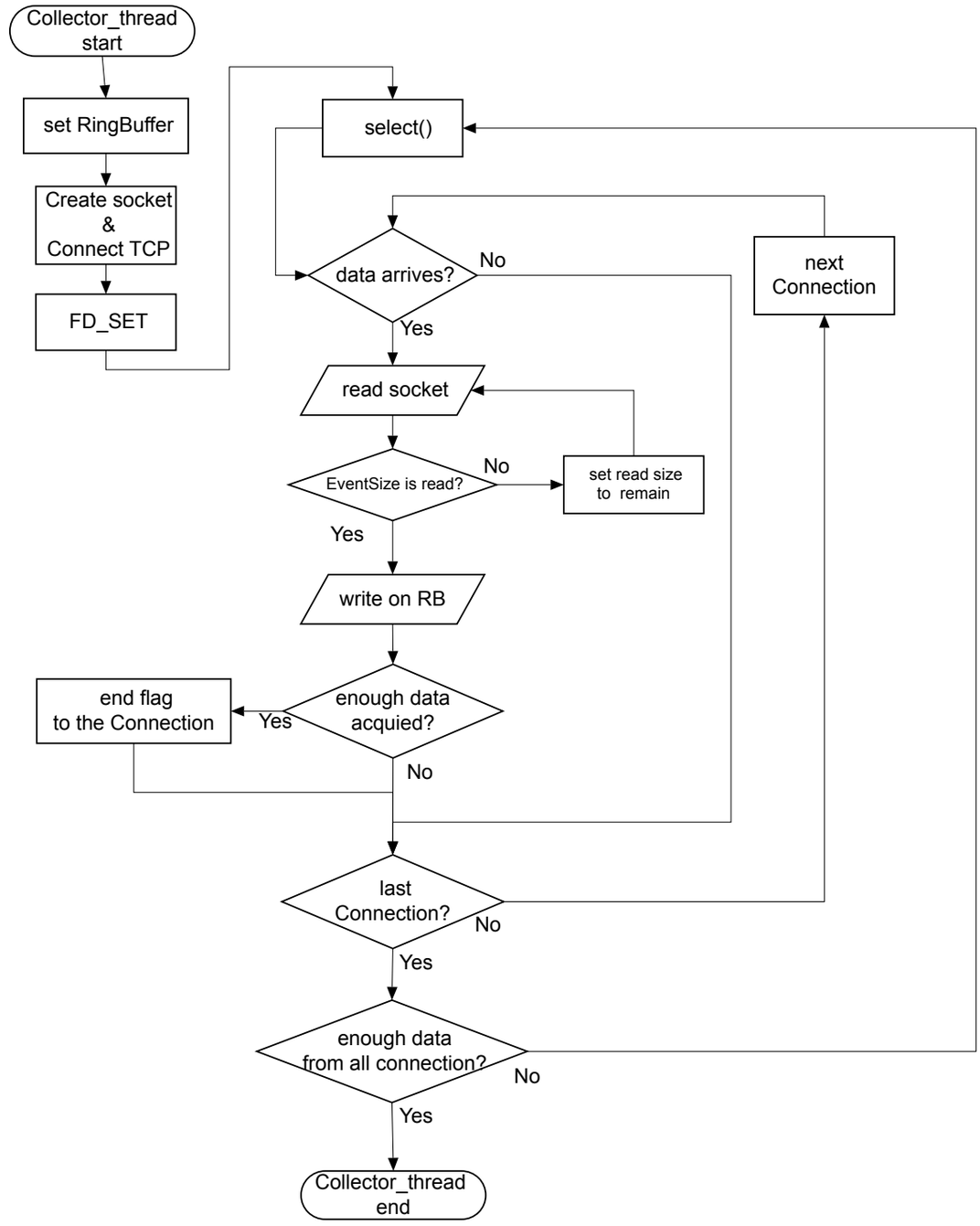


図 6.40　Collector スレッドのフローチャート

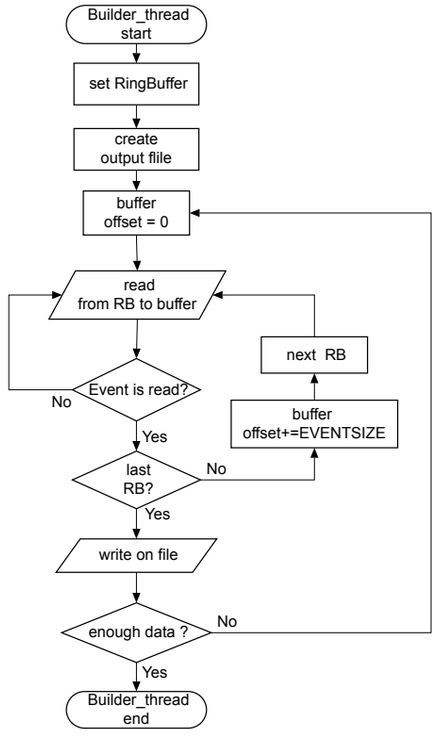


図 6.41　Builder スレッド概要のフローチャート

Builder スレッド

Builder スレッドは各リングバッファからデータを読み込んでデータ結合を行うスレッドである。自スレッドへの CPU を割り当て、全ての接続のためのリングバッファ一覧を把握した後、各リングバッファからのデータ読み込みを始める。

イベント結合なしの Builder

イベント結合機能の実装を行う前に Builder の前段階の処理の性能測定が必要のため、イベント結合機能のない Builder スレッドを用意した (図**??**)。全段階の処理の性能測定とは、Collector スレッドによる取得性能の評価を純粋に行うことと、リングバッファのサイズの見積もりを行うことである。このフローチャートの通り、Builder スレッドはイベント結合機能を停止させ、単純に全リングバッファのデータを読み取っては捨てるようにしてある。

このような対策をする主な理由は、イベント結合がある場合にデータ処理のリスクが大幅に増大するためである。高速なデータ転送の場合、非同期通信のためノード間に転送レートのばらつきが出る。そのため、RingBuffer のサイズを超えた差が起こると、あるノードは Collector スレッドが上書き防止の為 Builder を呼び出して wait に入り、一方で Builder が別のノードに対してデータ取得をするものの、データが届かず Polling を続けるということが起こる。

イベント結合有りの Builder

Builder に結合機能を実装した場合の、リングバッファからの読み取りのフローチャートを (図 6.42) に示した。これはスレッドの性能評価とリングバッファのサイズの見積もりを行って、危険でないスレッド数とリングバッファサイズに設定した後、機能試験を実施する。

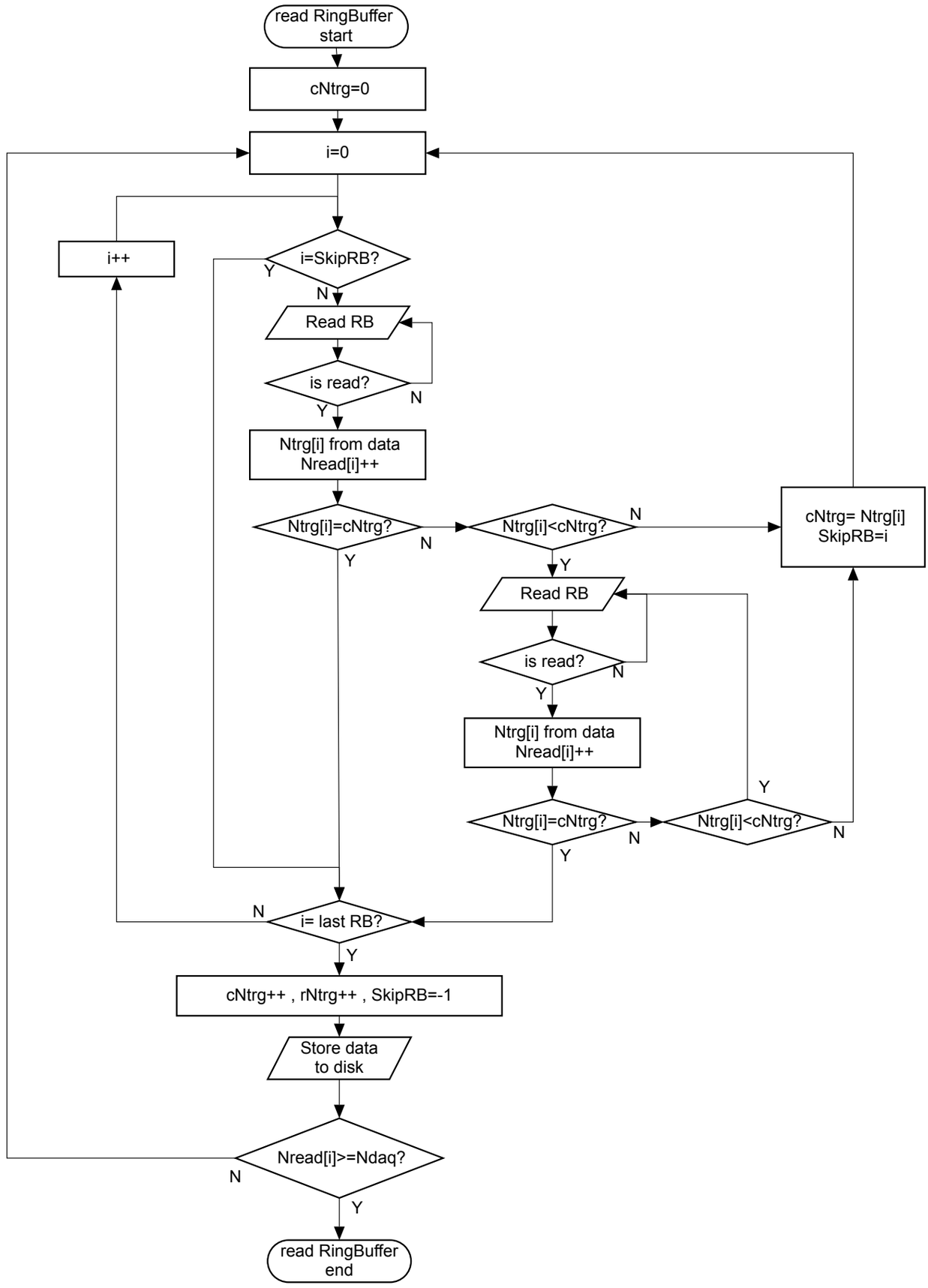


図 6.42　Builder のイベント結合機能のフローチャート。

Collector スレッドと Builder スレッドの関係

Collector スレッドと Builder スレッドは、共に main スレッドによって生成、終了を管理される。この様子をフローダイアグラム (図 6.43) に表した。

main スレッドは、外部からの開始命令を受けて最初に立ち上がるスレッドである。設定ファイルを読み込み、指定を受けた接続の個数分のリングバッファを生成する。同時にこれらのリングバッファを、指定通りの Collector スレッドに 1 つずつ割り当てる。その後、立ち上げるべき個数の Collector スレッドと、1 つの Builder を生成する。

Collector と Builder は、自分のスレッドに CPU を一つ割り当てたり、自分が担当するリングバッファを把握するなどの初期処理が必要である。また、Collector は各リングバッファが担当する接続先との接続を確立する。これらの処理が完了すると、データ収集を開始する。このデータ収集には全スレッド間でスタート同期を入れてある。もし Collector が先に接続を完了して RingBuffer へ書き込みを始める一方で Builder スレッドの初期化に時間がかかった場合、RingBuffer への圧迫が過剰になってしまう。そのため Builder スレッドが先に立ち上がり、データの到着を待ち受ける必要がある。

終了時は、main スレッドが pthread_join 関数にて各スレッドの終了を待ち受ける。全スレッドが終了すると、main スレッドが終了してプログラム全体が終了する。

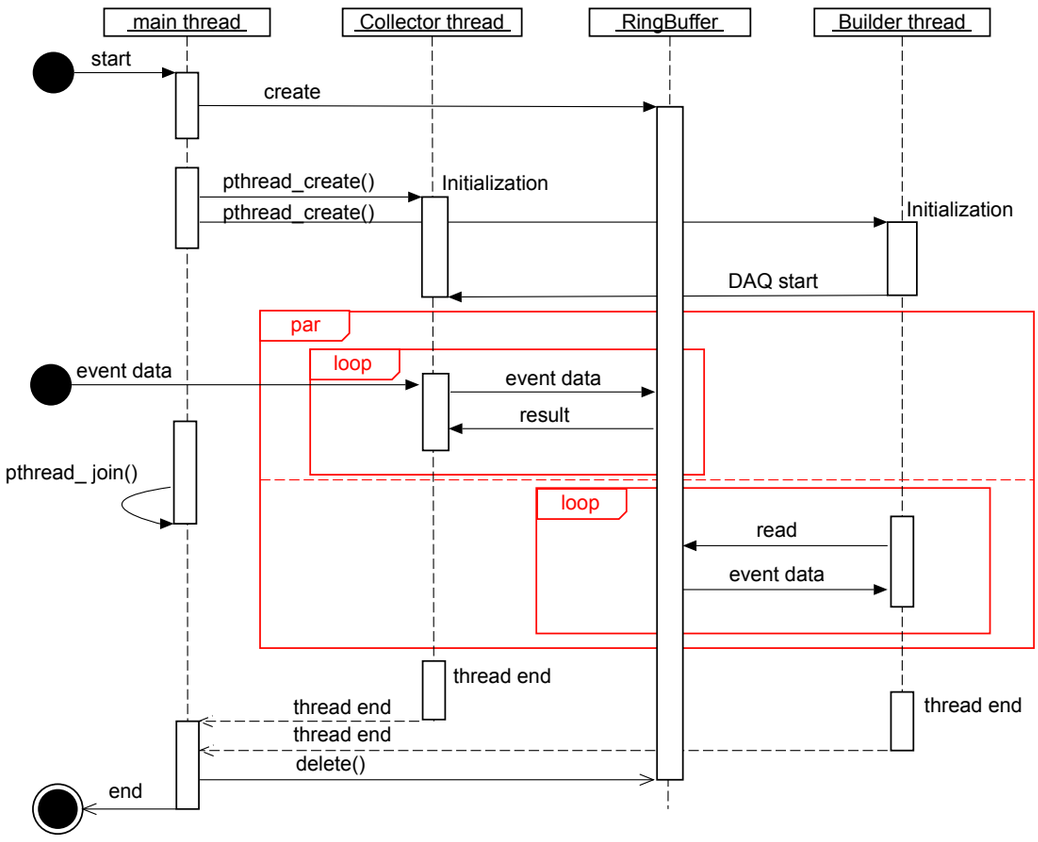


図 6.43　マルチスレッドの観点でのデータ収集プログラムのフローダイアグラム

RingBuffer の読み書き用関数

RingBuffer は、オブジェクト指向のクラスファイルで、各接続に 1 つずつオブジェクトとして生成され割り当てられる。この RingBuffer オブジェクトは主に write と read 関数を持っていて、オブジェクトが保持しているバッファに対する読み書きアクセスができるようになっている。

(図 6.44) はこれらの関数を簡単にまとめたフローチャートである。

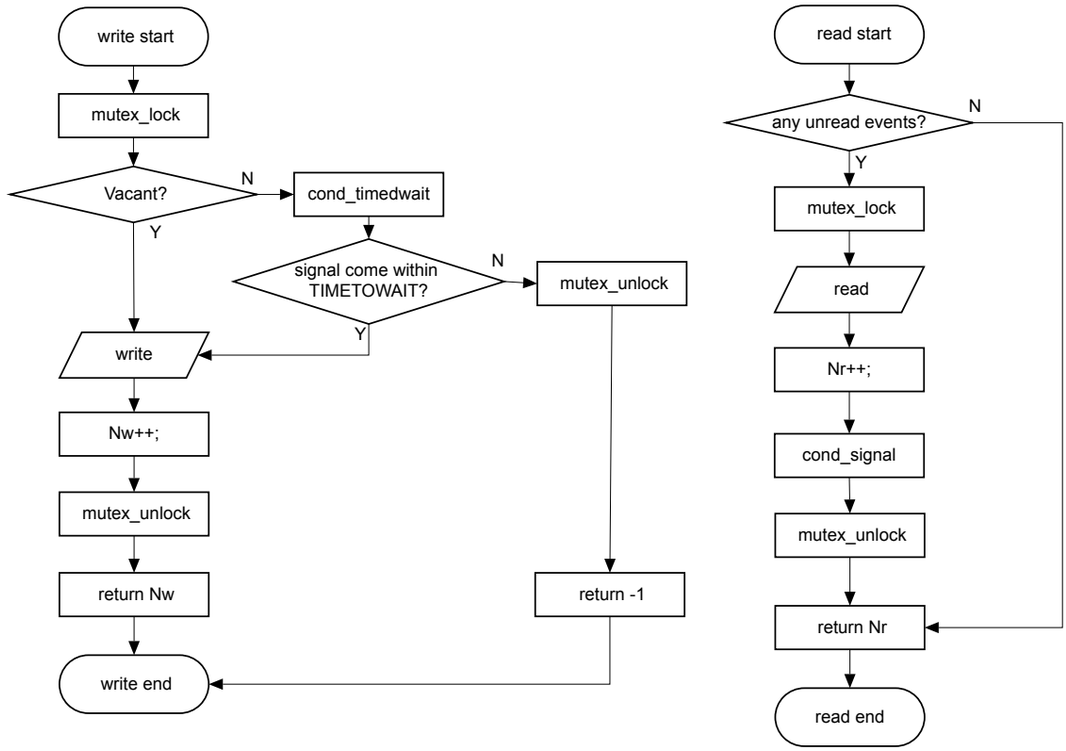


図 6.44　RingBuffer における write 関数と read 関数のフローチャート

write 関数（RingBuffer への書込）

RingBuffer への書込は Collector によって行われる。Collector は socket から読み込んだデータを RingBuffer に書き込むが、バッファに空きがない、つまり読み出しが周回遅れになっている場合、上書きを防ぐために cond_timedwait に入る。ここでは、一旦ロックを開放し、Builder スレッドがバッファから 1 件以上データを読み出し終わるまでプログラムを停止する。

この wait には待ち時間上限の設定が必要である。例えば、ある FEB にエラーが起こり、データ送信ができなくなったまま接続を確立し続けることが起こりえる。あるいは経路中のパケットロスにより、再送処理が完了するまでに通常よりも長いデータ未到達時間ができてしまうこともあり得る。その場合、それに対応する Collector は空の socket を読み続けることになるため、RingBuffer にはデータが書き込まれない。一

方、Builder はこの RingBuffer への新しいデータ到着を待ちポーリングを続けることになる。よって、他の RingBuffer への読込は行われず、RingBuffer は溢れてしまうことになる。待ち時間限度を超過した場合の処理は、呼び出し側にエラーコードを返す。待ち時間上限の値については後述する。

空きがあった場合、または待ち時間の間に新たに Builder による読み込みがなされて空きができた場合は、バッファに 1 イベント分のデータを書き込んで Nw を 1 だけカウントアップした後、ロックを開放して終了する。

read 関数（RingBuffer からの読出）

RingBuffer からの読出は Builder によって行われる。未読込のイベントが存在するかどうかを判定し、新たに読みだすイベントが無かった場合は何もせずに終了する。未読込のイベントが存在した場合は、ロックをかけ、データの読出しと Nr のカウントアップした後、自分の読出し終了のシグナルを送り、ロックを外して終了する。

バッファ溢れ防止と溢れ時の対策

バッファ溢れ防止のためには RingBuffer のサイズを大きく取る必要があるが、メモリのサイズには限度がある。また、読出しが遅れてバッファに空きがなくなった際に書き込みの待ち時間を長く取ってしまうと、TCP 受信バッファが溢れてしまう。これらのことを考慮して RingBuffer のサイズと cond_wait の際の待ち時間の上限を以下のように設定した。

RingBuffer の深さ

RingBuffer は接続先 FEB 間でのデータ到着のばらつきを吸収するために設定しているため、実際に試験を行ってばらつきを測定する必要がある。このため後述の RingBuffer 滞留量の見積もり試験によって RingBuffer の深さを決定する。

cond_wait の待ち時間上限

write 関数における、上書き防止の wait による待ち時間上限の値は、以下を基準に設定した。

- Builder スレッドによる新たな読込を待つが、待ち時間の間に 1 件以上新たにイベント結合が終了しなければ、新たな読込は行われないはずである。よって予想されるトリガ頻度に対して十分小さい値にするべきである。
- パケットロスが起こっているためにデータが届いていない接続があった場合、新たなデータ到着が遅れる。これを元に、パケットロスが発生してから再送処理が完了するまでに要する時間よりは長い時間待つべきである。

予想されるトリガ頻度は 15kHz なので、それより十分小さい値を 1 kHz とした場合、対応する値は 1 ms である。また、パケットロスが起こった場合、送信側は再送待ち時間を超えて確認応答が帰ってこなかったときに再送を行う。一般的にこの値は RTT の 5 倍程度に設定する。RTT は 0.2 ms 程度なので、やはり 1 ms が適切である。これを元に上限時間は 1 ms とした。

ロックの粒度

プログラムにおけるロックは、デッドロックの危険性があることと、ロックの開放待ちが発生すると性能が大きく落ちてしまうことから、極力避けるべきである。そのため、ロックをかける範囲は最小限にとどめた。

ロックが必要な部分は、Collector スレッドと Builder スレッドが共に参照・更新を行う変数である。この値

が常に適正な値でなければ、間違って上書きをしてしまう。また、Collector スレッドによる上書き防止の待ち処理の際は、Builder による新たな読込終了の信号を待ち受けるために cond_wait 関数を使用しているが、この関数は一旦ロックを取得してからロックを明け渡すという流れのため、ロックが必要である。

よって、write 関数では処理開始すぐにロックをかけ、Nw の更新が終了してからロックを解除することとした。また read 関数は、未読込データの存在を調べる処理にはロックが必要ないので、未読込データが存在しない場合はロックをかけないまま関数を終了させている。存在した場合は、その後でロックをかけてデータの読み取りと Nr のカウントアップを行う。その後の処理終了通知はロックを掛ける必要があるので、これが終わってからロックを解除している。

6.8　M 対 N 試験（収集のマルチスレッド効果の評価）

この節では、リングバッファを実装することで低下しているはずである Collector スレッドのデータ取得性能を測定する。合わせてこの Collector スレッドを複数スレッドで起動させ、データ取得性能が復帰していることを確かめる。これにより、最適なスレッド数の見積もりを行う。

6.8.1　試験方法

対象としたスレッド数

スレッド数はコア数程度までに抑えなければ、オーバーヘッドによりかえって非効率になる。また socket 関数による読み込みで複数の接続からの読み込みは十分可能である。一方、本プログラムを稼働させている開発用 PC は 6 コアである。1 コアを Builder スレッドに、もう 1 コアを TriggerGenerator に割り当てると、残り 4 コアが Collector スレッド用として使うことができる。よって稼働 Collector スレッド数は 1 スレッドから 4 スレッドまでについて試験を行った。

試験項目と測定方法

エミュレータ用に確保出来た PC は 26 台である。6.6.1 での (図 6.27) と同様の接続構成でテストを行った。26 台を上限として、各スレッド数において接続数をスレッド数の整数倍になるように増やしていき、それぞれの接続数でトリガレートは 10kHz から 110kHz で 10kHz おきで変化させ、各トリガレートで 5 回ずつ計測をした。1 回の計測では、トリガレートの 10 倍のイベント数を受信要求させて、約 10 秒程度のデータ収集をさせることにした。

エミュレータプログラム FakeFEB を稼働させた PC と、それに対しデータ取得を担当させた Collector スレッドの ID の対応を (表 6.7) に示す。この表において、接続は上行から順番に増やしている。

6.8.2　測定結果

結果は (図 6.45) のようになった。この 4 つのグラフは各スレッド数毎の結果で、横軸が入力したトリガレート、縦軸はそのときのデータ収集プログラム側での 1 接続当たりのスループットを表している。色と形で分類された点を結ぶ各線は接続数を表している。

またデータ取得プログラムの評価 (図 6.29) と同様に、一定の入力トリガレートにおいて、接続数を増やした時のデータ取得頻度を (図 6.46) に表した。どの結果も接続数が少ないうちは 110kHz まで追随してデータ取得ができるが、1 スレッドの場合 9 接続程度から、2 スレッド以降では 12 接続程度から、高いトリガレートに追

表 6.7　収集のマルチスレッド効果の評価試験で用いたエミュレータ iMac と、そこからデータ収集を担当した collector ID の一覧。エミュレータ No. は (表 6.6) と同一。

Emulator		Collector ID for each #of collectors			
No.	IP address	1	2	3	4
1	192.168.10.1	0	0	0	0
2	192.168.10.2	0	1	1	1
3	192.168.10.3	0	0	2	2
4	192.168.10.4	0	1	0	3
5	192.168.10.5	0	0	1	0
6	192.168.10.6	0	1	2	1
7	192.168.10.7	0	0	0	2
8	192.168.10.8	0	1	1	3
9	192.168.10.9	0	0	2	0
10	192.168.10.10	0	1	0	1
11	192.168.10.11	0	0	1	2
12	192.168.10.12	0	1	2	3
20	192.168.10.20	0	0	0	0
21	192.168.10.21	0	1	1	1
22	192.168.10.22	0	0	2	2
23	192.168.10.23	0	1	0	3
24	192.168.10.24	0	0	1	0
25	192.168.10.25	0	1	2	1
26	192.168.10.26	0	0	0	2
27	192.168.10.27	0	1	1	3
28	192.168.10.28	0	0	2	0
29	192.168.10.29	0	1	0	1
30	192.168.10.30	0	0	1	2
32	192.168.10.32	0	1	2	3
33	192.168.10.33	0	0		
34	192.168.10.34	0	1		

随しなくなっている様子がわかる。26 接続での上限スループットで比べると、1 スレッドでの収集に比べて 2 スレッドでは 1.6 倍程度となった。マルチスレッドによる効果がはっきり表れている。

入力トリガレートに追随しなくなった後の取得頻度はほぼ単一の曲線に乗っており、式 (6.6) の形でフィットした結果、1 スレッドでは CTA の要求値に到達しないことがわかる。しかし、2 スレッドでは性能が復活し、45 台接続では 30 kHz まで対応出来ると予想できる。この値をデータ取得プログラムと比較すると、接続が多くなった場合の取得頻度の落ち込みがデータ取得プログラムよりも少なく、13% 程度高い値である。よって、2 スレッド当たり 45 台からの収集を担当させれば、CTA の要求値の 2 倍程度を満たす。またスレッド数を 3、4 と増やした場合は、2 スレッドの場合とほぼ同じ結果となり、効果は無い。

結論

20kHz でのデータ取得を行う場合、30 接続ほどまでは 1 スレッドでも対応が可能である。スイッチ 1 台あたり 45 接続をまとめるが、これを 1 台のカメラサーバーで受け取る場合は 2 スレッドが必要である。

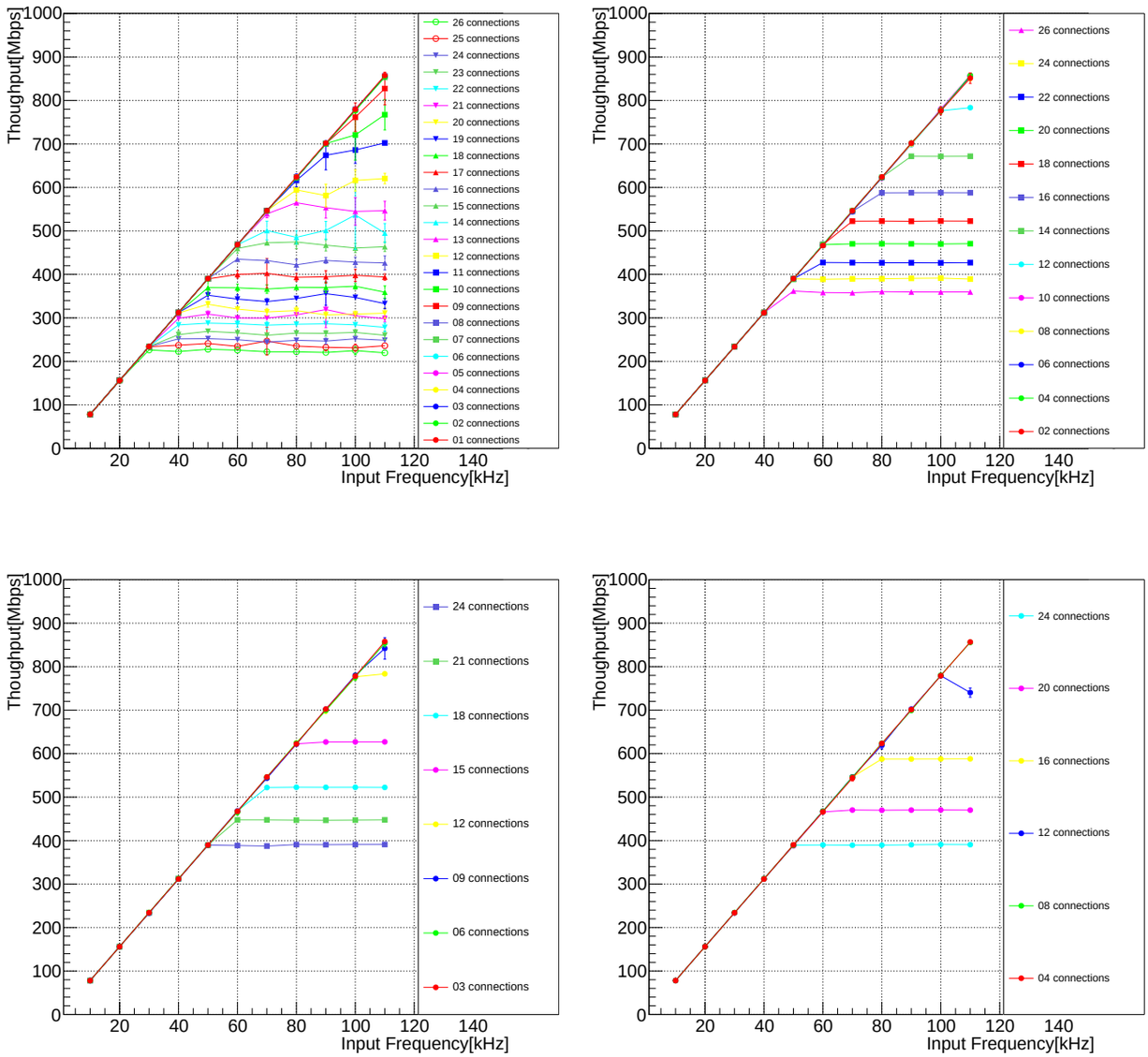
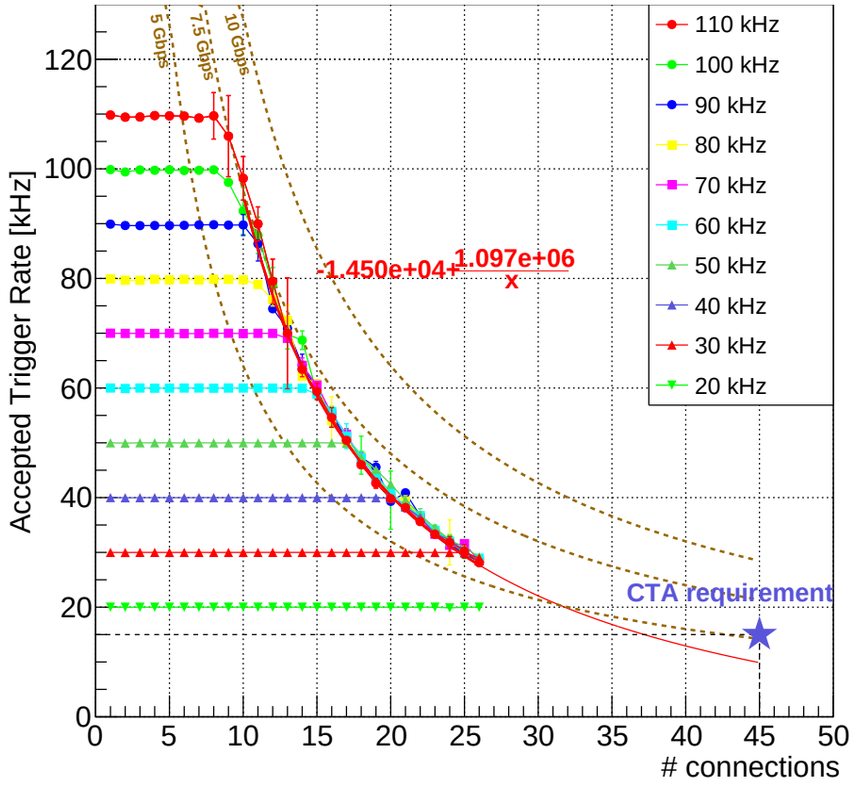
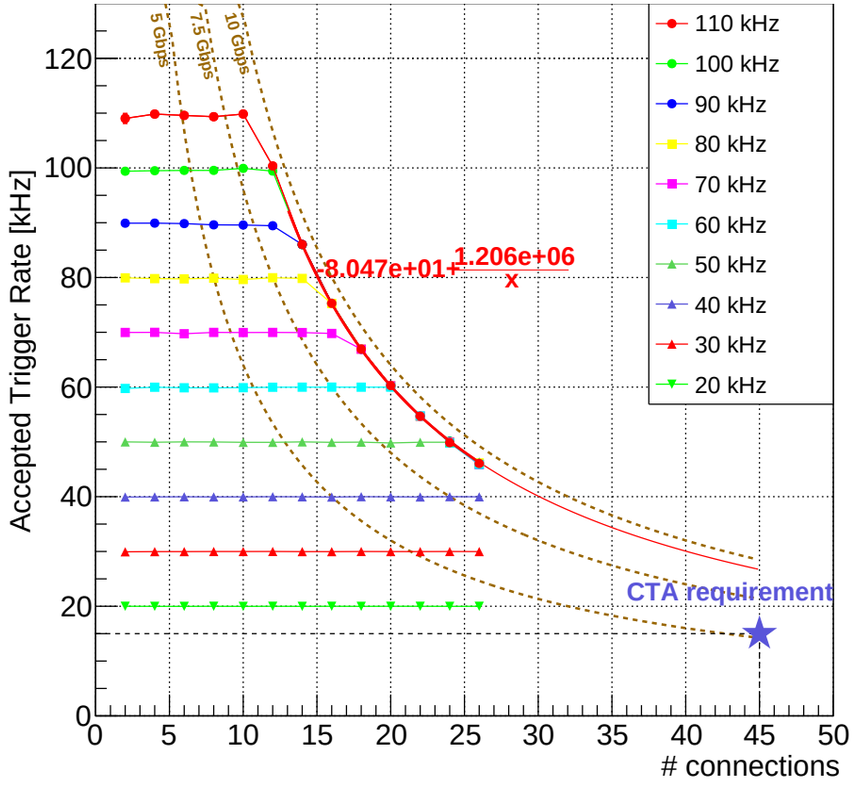


図 6.45　マルチスレッドによる収集性能評価。各接続数について、入力トリガレートを変化させた時の 1 接続当たりデータ取得レートの変化。Collector スレッド数を 1 スレッドから 4 スレッドまでの 4 種類について行った。上左：Collector が 1 スレッドの場合。上右：2 スレッドの場合。下左：3 スレッドの場合。下右：4 スレッドの場合。


5 Gbps
7.5 Gbps
10 Gbps
110 kHz
100 kHz
90 kHz
80 kHz
70 kHz
60 kHz
50 kHz
40 kHz
30 kHz
20 kHz
1.450e+04+1.097e+06
x
CTA requirement
Accepted Trigger Rate [kHz]
connections
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35
40
45
50


5 Gbps
7.5 Gbps
10 Gbps
110 kHz
100 kHz
90 kHz
80 kHz
70 kHz
60 kHz
50 kHz
40 kHz
30 kHz
20 kHz
8.047e+01+1.206e+06
x
CTA requirement
Accepted Trigger Rate [kHz]
connections
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35
40
45
50

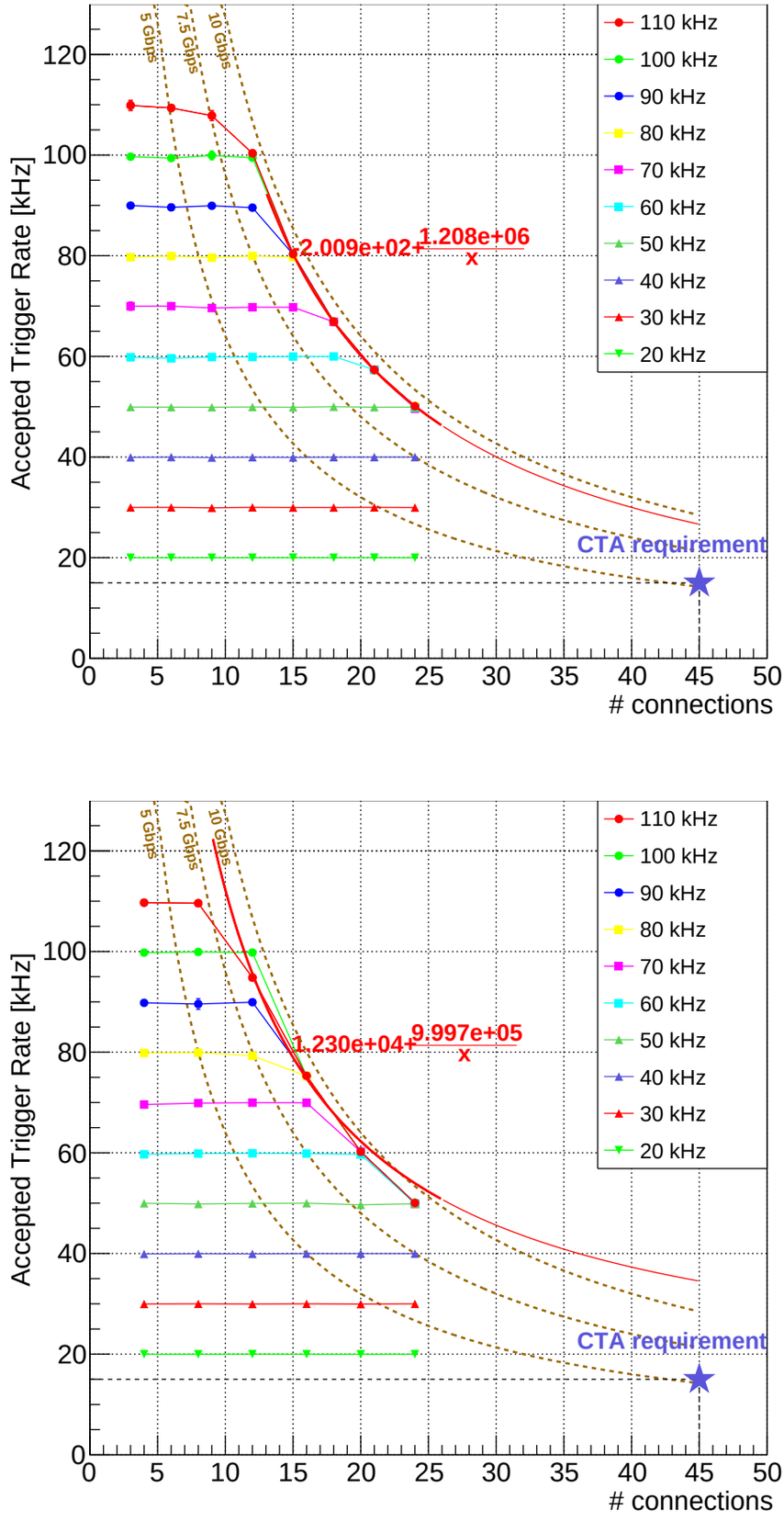


図 6.46　マルチスレッドによる収集性能評価。上から、スレッド数が 1 から 4 の順番。

6.9 RingBuffer 滞留量の見積もり

イベント結合機能の実装前に、結合機能なしで長時間データ収集を行い、測定結果を元に RingBuffer の適切な量を見積もった。イベント結合機能は同一トリガのイベントデータを結合ことで各接続での同期をとることになる。この前段階としてリングバッファは FIFO バッファの役割を持つことである程度の時間データを保持し、各 FEB との通信の非同期性により発生するデータ到着時間のすれを吸収する。イベントの保持能力は各リングバッファが確保するバッファのサイズによって決まるが、資源は無尽蔵では無いために予め上限値を見積もる必要がある。そこでイベント結合をせずデータ取得を行い、RingBuffer へ書き込まれたイベント数の時間変化を記録して RingBuffer 滞留量がどの程度になるのか評価を行う。

6.9.1 測定方法

チェレンコフ望遠鏡の典型的な 1 回の観測時間が 20 分間であることを参考に、各入力トリガレートに対してトリガレート ×1200 のイベントデータ取得を行わせた。また、トリガレートは 10kHz、20kHz、40kHz の 3 種類を選んだ。Collector スレッド数は 1 スレッドとし、接続数はなるべく多く取り、本実験では 19 接続を確保できた。(図 6.46) の 1 スレッドの結果より、19 接続においてはトリガレート 40kHz が帯域ぎりぎりの状況である。このような飽和寸前の状況では、帯域が足りず転送が一部行われない事がランダムに起こる可能性が高く、それによって各接続からの到着データに揺らぎができて RingBuffer に負担がかかりやすい状況ができる。

測定は、新たに計測用のスレッドを作成して稼働させ、1 秒に 1 回ずつ各 RingBuffer に書き込み済みのイベント数を記録した。本測定ではプログラムの負荷を最小限に抑えるため、トリガ番号などは記録させていない。データ送信を担当したエミュレータ iMac と、そこからのデータを保管する役目を担った各 RingBuffer の ID の対応を (表 6.8) に示す。

6.9.2 測定結果

結果は (図 6.47) から (図 6.49) に示した。

各図の上のグラフは、1 秒経過毎に新たに各 RingBuffer に書き込まれたイベント数を表している。つまり、ある時刻 t_k において i 番目の接続を担当するリングバッファに書き込み済みのイベント数 $N_{\mathrm{w},i}(t_k)$ として、

$$\left[\frac{\mathrm{d}N_i}{\mathrm{d}t}\right]_{t=t_k} = (N_{\mathrm{w},i}(t_{k+1}) - N_{\mathrm{w},i}(t_k))/(t_{k+1} - t_k)$$

の時間変化をプロットした。RingBuffer への書き込みは Collector がソケットから読み込んでそのまま書き込まれるため、これはネットワークにおいて各接続がデータ転送に消費している帯域と解釈できる。10kHz、20kHz では、RB1（2 番目のリングバッファ）において瞬間的な低下が見られ、それ以外はほぼ入力トリガレートとほぼ同じレートでデータがリングバッファに書き込まれていっていることがわかる。40kHz では、全てのリングバッファの一時的なレートの低下が 500 秒付近を始め数カ所で発生している。ネットワーク上で転送性能が一時的に追いつかなくなり、復帰するまで転送が低下していると思われる。

一方で、各図の下のグラフは、各接続のデータ取得処理を開始してから測定した時刻までに各 RingBuffer に書き込み終わった総イベント数同士を比べ、その時点で書き込み済みのイベント数が一番多い RingBuffer と一番少ない RingBuffer との差を表している。差は一定のまましばらく推移するが時々突然広がることで、階段状に差が増えていく。

表 6.8　RingBuffer 滞留量の見積もり試験で用いたエミュレータ iMac と、データ保管を担当した RingBuffer ID の一覧。エミュレータ No. は (表 6.6) と同一。

Emulator		RingBuffer ID
No.	IP address	
2	192.168.10.2	0
5	192.168.10.5	1
7	192.168.10.7	2
8	192.168.10.8	3
11	192.168.10.11	4
12	192.168.10.12	5
20	192.168.10.20	6
21	192.168.10.21	7
22	192.168.10.22	8
23	192.168.10.23	9
24	192.168.10.24	10
25	192.168.10.25	11
26	192.168.10.26	12
27	192.168.10.27	13
28	192.168.10.28	14
29	192.168.10.29	15
30	192.168.10.30	16
32	192.168.10.32	17
33	192.168.10.33	18

RB1 のレートが低下する瞬間と書き込み済みのイベント量の差が上昇する瞬間はほぼ同期している。これはリングバッファ RB1 が担当する接続先の Fake FEB においてトリガパケットの受信に失敗し、トリガ番号の抜けが起こったことで RingBuffer に書き込まれるイベントが少なくなった一方、他の FakeFEB からはトリガ番号の抜けがなく届き続け通常通りの割合で RingBuffer に書き込まれていたため差が広がったと思われる。[*14]

この RB1 の接続先の PC がトリガ受信を停止する詳しい原因は突き止められなかった[*15]が、USB から Ubuntu14.04 を起動させる対応をとったところ症状が消えることを確認した。この状態でのテストは、後述のイベント結合機能試験の長時間試験で実施する。また、この現象を利用して、Dragon FEB が同様にトリガ受信に失敗する場合を想定したデータ収集システムの対応を試験することができる。よって、この結果をリングバッファサイズの見積に使用する。

[*14] 後に実際スイッチの統計情報を確認したところ、この RB1 の接続先 PC との通信でスイッチがフォワードを行わなかったパケット数は 1 番高い値を示していた。これはトリガパケットの受信を拒否していると思われる。パケットロスは 0 であり、データの飛びも起こっていないためイベントデータのパケット転送は正常である。

[*15] この RB1 の接続を担当するエミュレータ No.5 は 2013 年製の iMac であり (表 6.6)、使用した iMac の中でも新しく性能の高い PC である。また約 1 分おきという周期性から Ubuntu の上で走るサービスプロセスが時折入出力またはネットワーク関連の処理をしてエミュレータプログラムが中断させられていると思われるが、Dragon FEB とはトリガ受信のメカニズムが全く違うため、これ以上の考慮はしない。

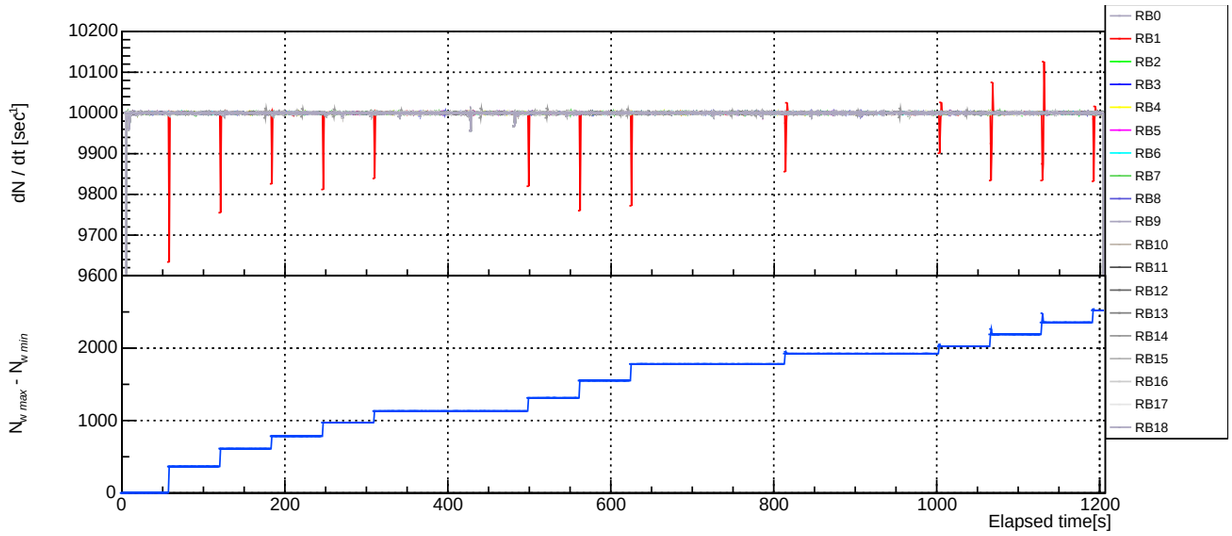


図 6.47 結合機能なし 1 対 19 接続 20 分間の DAQ 結果。トリガーレート 10kHz

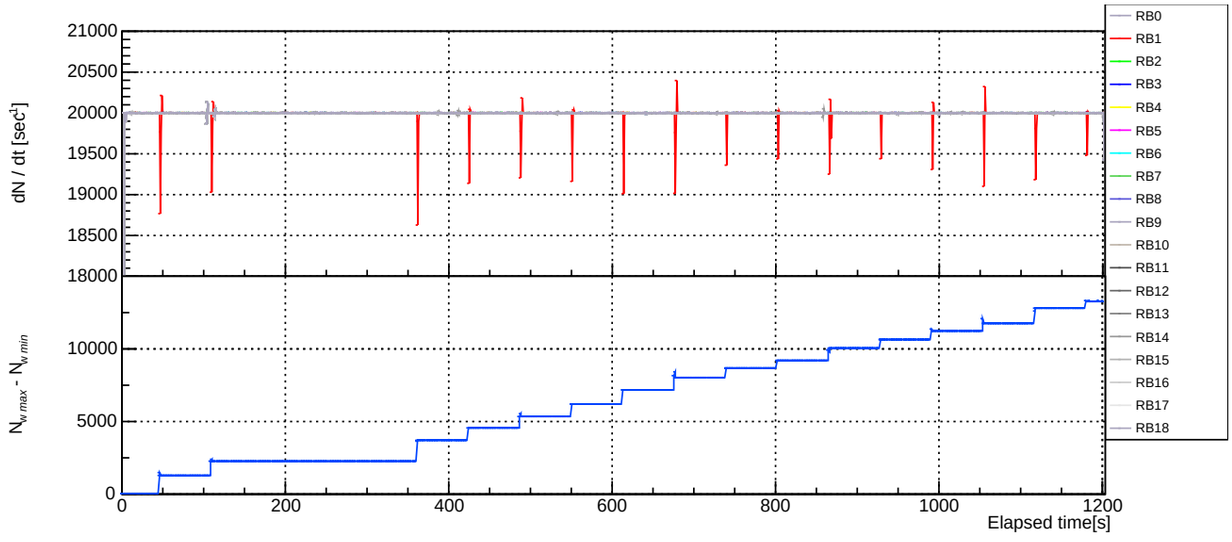


図 6.48 結合機能なし 1 対 19 接続 20 分間の DAQ 結果。トリガーレート 20kHz

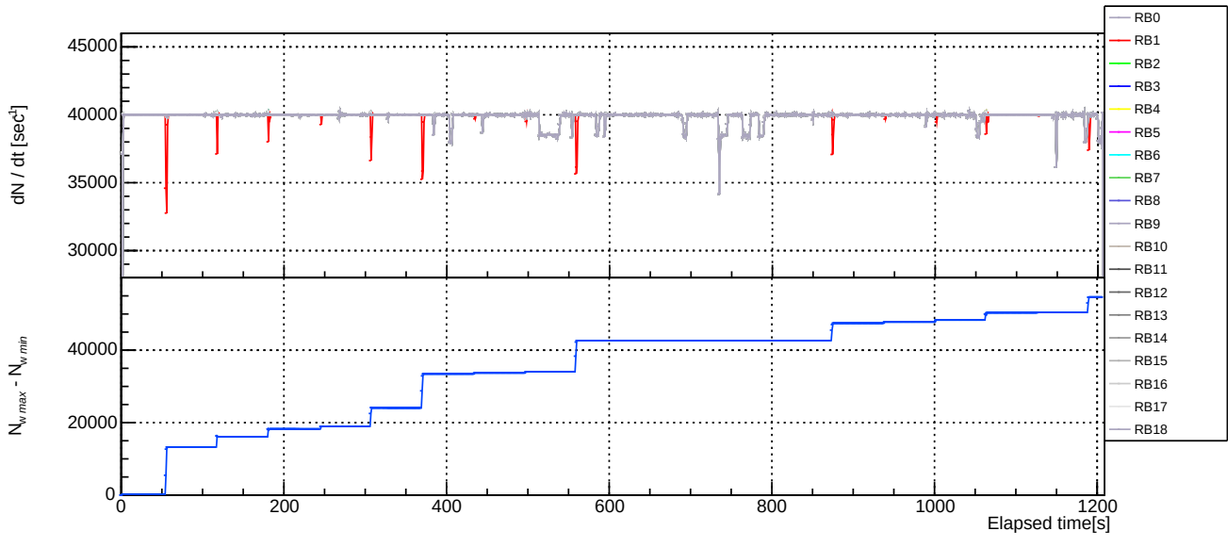


図 6.49　結合機能なし 1 対 19 接続 20 分間の DAQ 結果。トリガーレート 40kHz

6.9.3　リングバッファサイズの見積

トリガ受信に失敗したエミュレータは、その分送信するデータが少なくなる。一方でその他の健康にトリガを受信し続けているエミュレータはその分データを送信するため、リングバッファには滞留量の差ができることになる。仮に一件分ずつのイベント結合を何のチェックもなくそのまま行った場合は、リングバッファ間の滞留量の差をそのまま抱え続けてしまうことになる。そのため確実にバッファ溢れが発生する。

しかし実際のイベント結合では、同一トリガ番号のデータ同士を結合して一つのイベントデータを作成する。そのため Builder はリングバッファからのデータ読み込み時にトリガ番号をチェックし、トリガ番号が揃うまでデータを読み捨てるか、リスト形式の FIFO バッファにに保存するという対策をとることになる。このようにしてトリガが揃うまで他のリングバッファを読み込みつづけるため、より多くのデータが滞留した RingBuffer の滞留量は減少し、データ滞留量の差は解消される。また結合処理がデータ取得処理に追いついていれば、滞留量は常にゼロに戻る。

よってリングバッファは、トリガ受信に失敗した FEB が現れた場合の突然の到着データ量のばらつきと、Builder が一時的にトリガレートが高まり処理が追いつかなくなった場合とに対して余裕のあるバッファサイズを用意しておけば、安定に稼働ができることになる。データ到着量のばらつきや高処理負荷状態は、リングバッファに書き込まれるデータ量の差により大まかなみつもりができる。今回の測定の中で最大は 40kHz での測定で、15000 程度の差異が突然できている。後述のイベント結合機能のためのリングバッファのバッファサイズは安全を考慮して 50000 イベント分とした。

6.10　イベント結合機能試験

イベント結合機能を有効にしてイベント収集機能の試験を行い、イベント結合までを含めたデータ収集システムの耐久性能を確認した。リングバッファサイズは前述のとおり 50000 イベント分としてある。

試験は 20 分間耐久試験と 10 時間耐久試験の 2 種類を行った。

20 分間耐久試験では、Builder スレッドに負荷がかかる状況でもイベント結合が一定時間健全に稼働できることが目標である。Builder への負荷は、トリガ番号が揃わない状況に発生する。そのため、トリガ受信に失敗するエミュレータを含ませたままデータ収集を行ってトリガ番号の飛びを発生させることでリングバッファ内の滞留量を増加させ、Builder がリングバッファの滞留量を消化できる事を確認した。更に、トリガレートを上げることで全エミュレーターがトリガを様々なタイミングで受信失敗する状況を作り、より大きな負荷をかけて耐久性能を確認した。

10 時間耐久試験では、一晩の観測時間に耐久できることが目的である。全てのエミュレーターをトリガパケットの損失が少ない健全な状態で稼働させ、10 時間稼働し続ける事を確認した。

6.10.1　20 分間耐久試験

試験方法

イベント結合機能を有効にした上で、6.9 の RingBuffer 滞留量の見積もり試験同様の方法で試験を行った。この試験では Collector スレッド数は 1 スレッドとし、接続数はなるべく多く取り、20 接続を確保できた (表 6.9)。1 スレッド 20 接続では、トリガレート 40kHz 弱が取得性能ぎりぎりの状況である。余裕のある 30kHz から 40kHz の 6 種類について、各 20 分間に相当するイベント数（トリガレート ×1200）のデータ収集を行った。

計測は、データ収集中の各リングバッファ内の滞留量を監視した。計測用のスレッドがこれを行い、Collector スレッドによって各リングバッファに書き込まれたイベント数 N_{w} と、各リングバッファから Builder へ読み出されたイベント数 N_{r} を 1 秒ごとに記録する。

試験結果

結果は (図 6.50) から (図 6.55) のようになった。各図の上のグラフは、1 秒経過毎に新たに RingBuffer に書き込まれたイベント数の平均を表している。つまり、ある時刻 t_k において i 番目の接続を担当するリングバッファに書き込み済みのイベント数 $N_{\mathrm{w},i}(t_k)$ として、

$$\frac{\mathrm{d}\langle N\rangle}{\mathrm{d}t}(t=t_k)=\frac{1}{n_{RB}}\sum_{i}^{n_{RB}}(N_{\mathrm{w},i}(t_{k+1})-N_{\mathrm{w},i}(t_k))/(t_{k+1}-t_k)$$

である。真ん中のグラフは、$N_{\mathrm{w,max}}-N_{\mathrm{w,min}}$、つまり開始から測定時刻までにリングバッファに書き込まれたイベント数を比べて、最大の値と最小の値の差をとっている。下のグラフは i 番目のリングバッファについて $N_{\mathrm{w},i}-N_{\mathrm{r},i}$、つまり各リングバッファに未読込で滞留しているイベント数の時間変化である。

30kHz の場合、$<\frac{\mathrm{d}N}{\mathrm{d}t}>$ を見ると、書き込まれるイベント数は入力トリガレートに追随してほぼ一定で 30kHz であるが、時々レートが低下している。これは前節で取り上げたエミュレータ No.5（RB1）の影響である。RB1 はエミュレータ No.5 のデータ保管を担当するが、エミュレータ No.5 がトリガパケットを受信せずイベントが送信されない状況になると $<\frac{\mathrm{d}N}{\mathrm{d}t}>$ が一時的に低下し、それと同時に他のエミュレータからはデータが到着するために $N_{\mathrm{w,max}}-N_{\mathrm{w,min}}$ が上昇している事がわかる。その瞬間の下図 $N_{\mathrm{w},i}-N_{\mathrm{r},i}$ を見ると、リ

表 6.9　20 分間イベント結合試験で用いたエミュレータ iMac と、データ保管を担当した RingBuffer ID の一覧。エミュレータ No. は (表 6.6) と同一。

Emulator		RingBuffer ID
No.	IP address	
2	192.168.10.2	0
5	192.168.10.5	1
6	192.168.10.6	2
7	192.168.10.7	3
8	192.168.10.8	4
11	192.168.10.11	5
12	192.168.10.12	6
20	192.168.10.20	7
21	192.168.10.21	8
22	192.168.10.22	9
23	192.168.10.23	10
24	192.168.10.24	11
25	192.168.10.25	12
26	192.168.10.26	13
27	192.168.10.27	14
28	192.168.10.28	15
29	192.168.10.29	16
30	192.168.10.30	17
32	192.168.10.32	18
33	192.168.10.33	19

ングバッファの滞留量が一旦大きく上昇し、すぐに下がっていることが見える。滞留が多いリングバッファを読み捨てによって消化した様子がわかる。

一方で 40kHz の場合、$< \frac{\mathrm{d}N}{\mathrm{d}t} >$ を見ると、全てのリングバッファは入力トリガレートに追いつかず 36kHz 程度の書き込み速度になっている。$N_{\mathrm{w},i} - N_{\mathrm{r},i}$ を見ると、リングバッファの滞留量は階段状に徐々に上がっていく。この状況ではトリガの飛びが激しく、Builder スレッドが読み捨ての処理で対応しきれなくなっているものと思われる。

前述のとおり、トリガ番号が飛び読み捨てが発生した原因はトリガを FakeFEB が受信しなかったためだが、トリガは UDP 通信を用いているために到達性の保証がなく、トリガ受信失敗による少量の読み捨ての発生は期待通りである。実際の FEB がこのような状況になった場合は FEB の故障を強く疑うべきなので、20 接続 40kHz のトリガレートでも一定時間の耐久性能で十分と思われる。

またトリガ番号の飛びをなくすことができるかどうかは、FakeFEB がトリガパケットを高い信頼性で受信できるかどうかというデータ送信側の問題に依存し、データ収集システム側の問題ではない。読み捨てなく全件取れるかどうかは、実際の FEB のトリガ受信に対する信頼性も含めて考える必要が有るため、本研究の対象外とする。

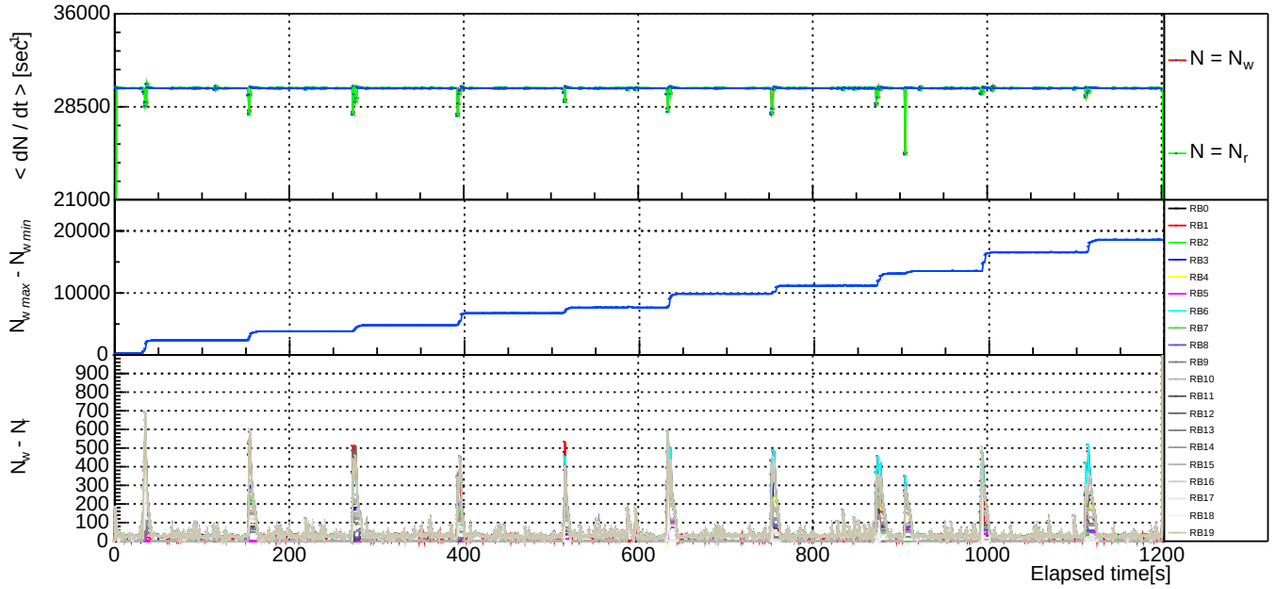


図 6.50　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 30kHz

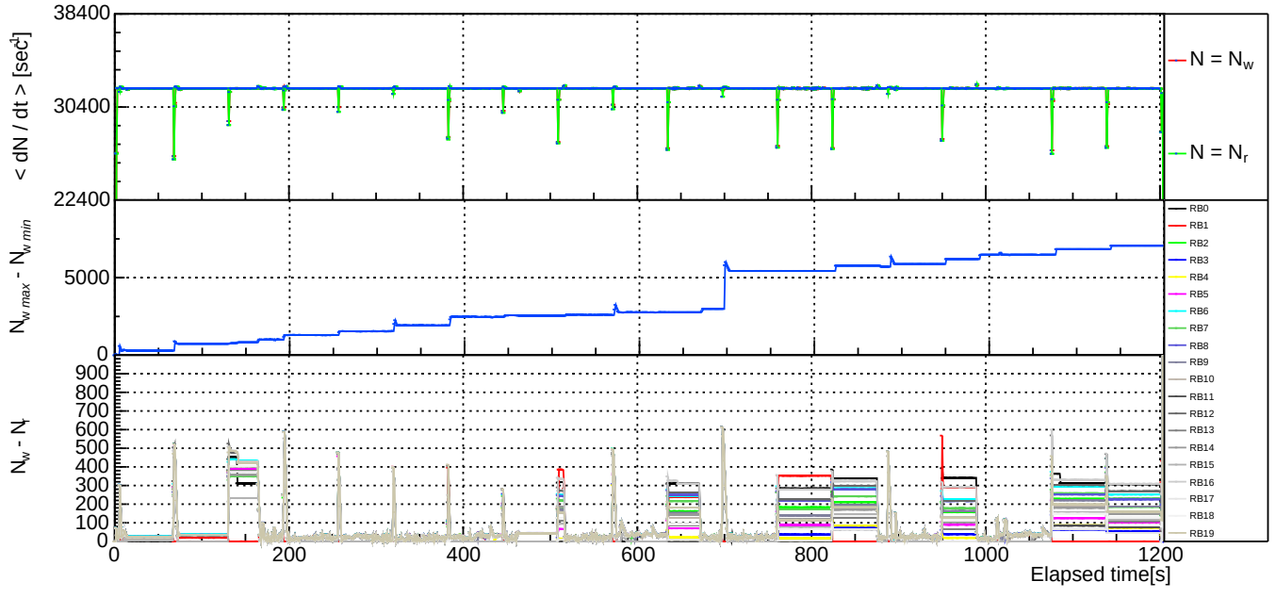


図 6.51　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 32kHz

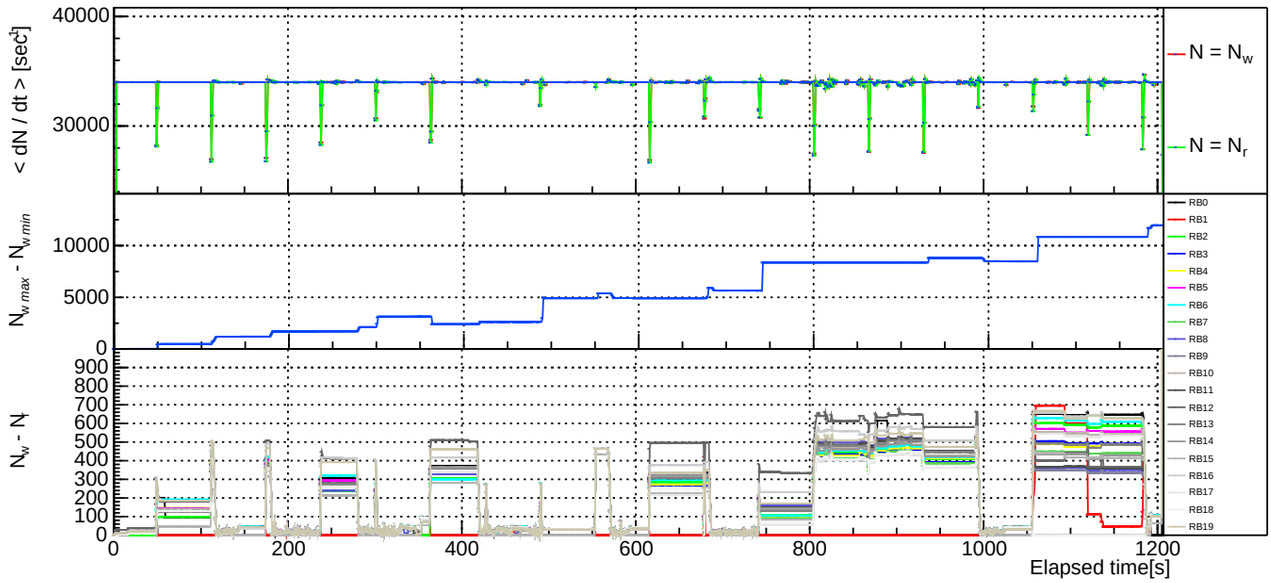


図 6.52　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 34kHz

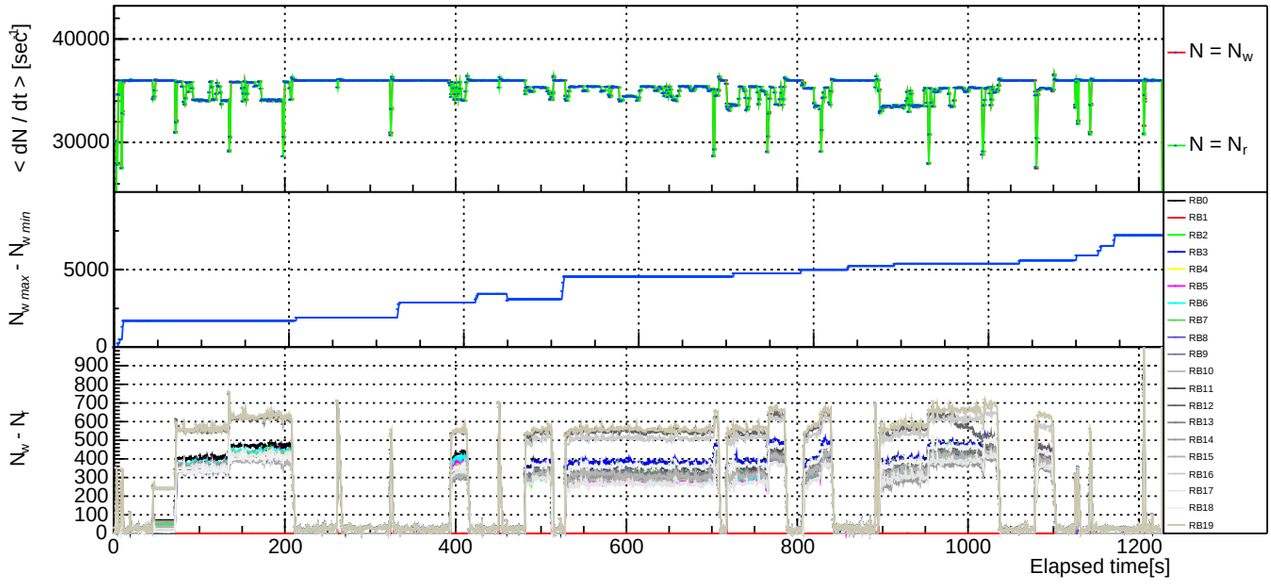


図 6.53　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 36kHz

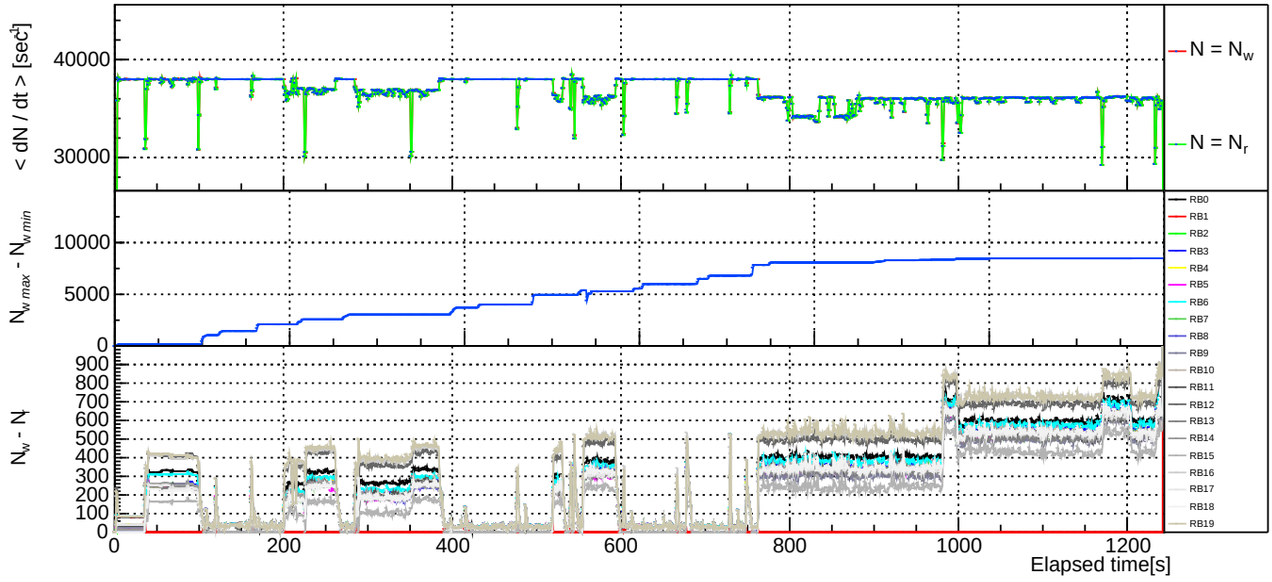


図 6.54　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 38kHz

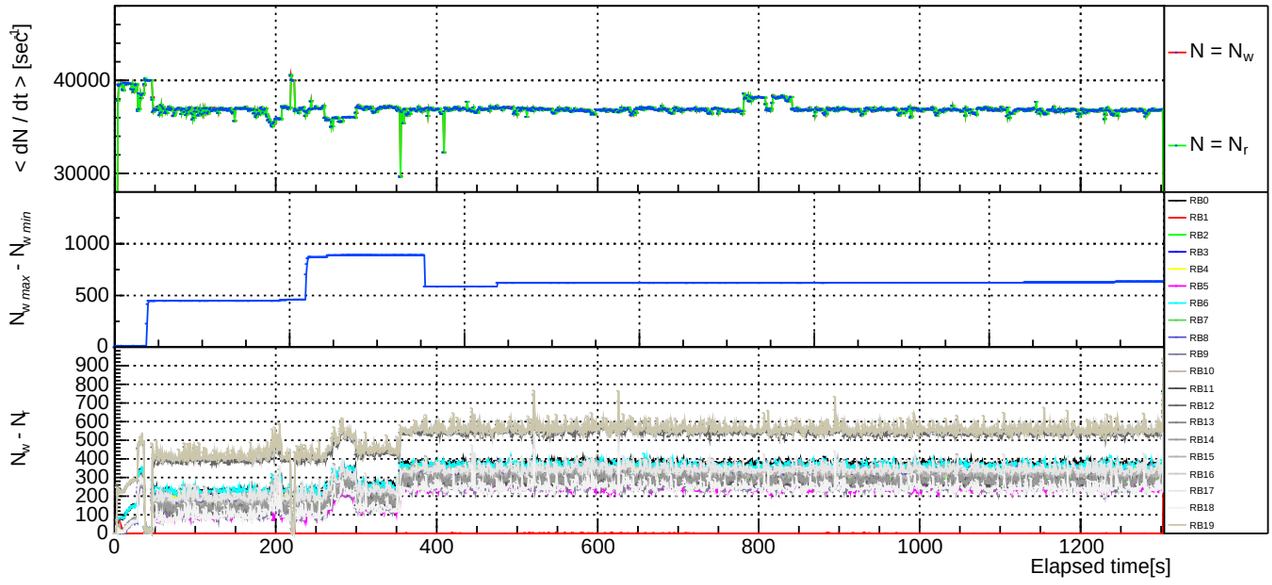


図 6.55　結合機能有り 1 対 20 接続 20 分間の DAQ 結果。トリガーレート 40kHz

6.10.2　10 時間耐久試験

トリガ受信不良エミュレータの改善

10 時間耐久試験に先立って、エミュレータ No.5 のトリガ受信失敗の解消を行った。この試験では 15 接続を確保できた (表 6.10)。この接続数で 30kHz のトリガをかけ、20 分に相当する件数（30kHz×1200s）を収集した。結果が (図 6.56) である。リングバッファへの書込レートは与えたトリがレートと常に同じ 30kHz でほぼ一定となり、20 分間耐久試験での結果 (図 6.50) と比べると、トリガ受信失敗の現象が大幅に改善したことがわかる。

表 6.10　エミュレータ No.5 健全性確認試験で用いたエミュレータ iMac と、データ保管を担当した RingBuffer ID の一覧。エミュレータ No. は (表 6.6) と同一。

Emulator		RingBuffer ID
No.	IP address	
2	192.168.10.2	0
5	192.168.10.5	1
20	192.168.10.20	2
21	192.168.10.21	3
22	192.168.10.22	4
23	192.168.10.23	5
24	192.168.10.24	6
25	192.168.10.25	7
26	192.168.10.26	8
27	192.168.10.27	9
28	192.168.10.28	10
29	192.168.10.29	11
30	192.168.10.30	12
32	192.168.10.32	13
33	192.168.10.33	14

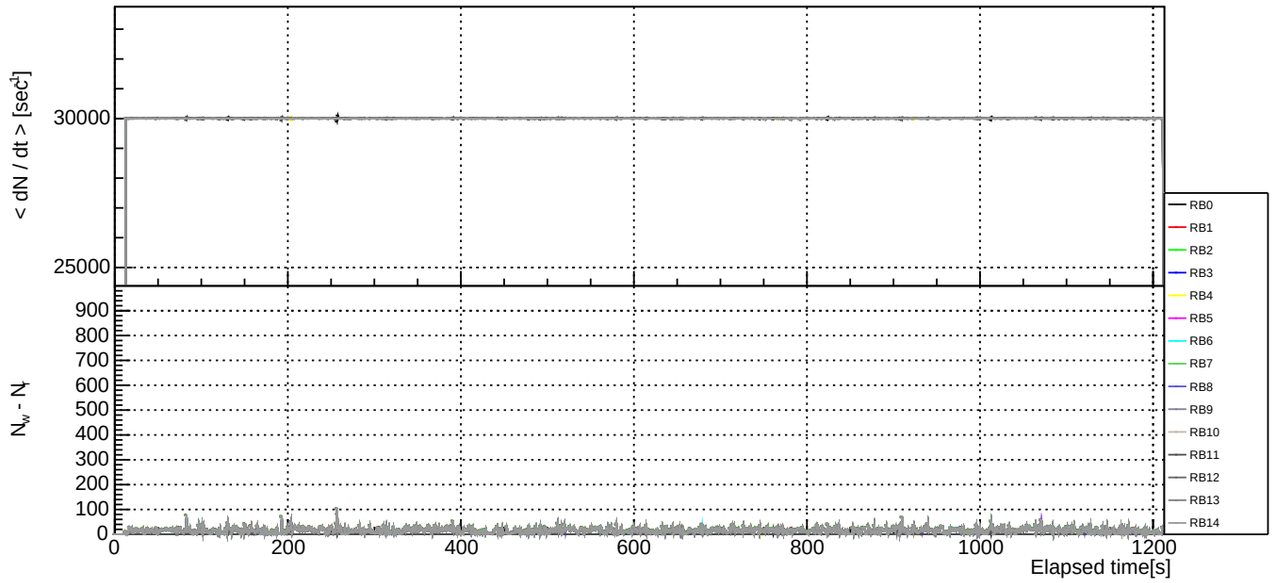


図 6.56　結合機能有り 1 対 15 接続 20 分間の DAQ 結果。トリガーレート 30kHz

10 時間耐久試験

これを元に、10 時間の耐久試験を行った。接続数は 16 接続を確保した (表 6.11)。安全のために Collector スレッド数は 2 スレッドとし、45kHz で 10 時間の稼働時間に相当する 1.62×10^{10} 件のイベント数を収集した。計測にはこれまでの実験同様計測用のスレッドをもちいて、Collector スレッドによって各リングバッファに書き込まれたイベント数 $N_{\rm w}$ と、各リングバッファから Builder へ読み出されたイベント数 $N_{\rm r}$ を記録し、データ収集中の各リングバッファ内の滞留量を監視した。記録の間隔は 10 秒ごととした。結果は (図 6.57) となり、正常にデータ収集を完了できた。

15 接続に対する 45kHz のトリガレートでのデータ収集の負担は、45 接続に対する 15kHz でのデータ収集の負担と同様になると考えられる。そのため CTA におけるトリガレートの要求値 15 kHz において、スイッチ単位の 45 接続に対応して長時間でのデータ収集が可能と予想ができる。

表 6.11　10 時間イベント結合試験で用いたエミュレータ iMac と、データ保管を担当した RingBuffer ID の一覧。エミュレータ No. は (表 6.6) と同一。

Emulator		Collector ID	RingBuffer ID
No.	IP address		
2	192.168.10.2	0	0
5	192.168.10.5	1	1
12	192.168.10.12	0	2
20	192.168.10.20	1	3
21	192.168.10.21	0	4
22	192.168.10.22	1	5
23	192.168.10.23	0	6
24	192.168.10.24	1	7
25	192.168.10.25	0	8
26	192.168.10.26	1	9
27	192.168.10.27	0	10
28	192.168.10.28	1	11
29	192.168.10.29	0	12
30	192.168.10.30	1	13
32	192.168.10.32	0	14
33	192.168.10.33	1	15

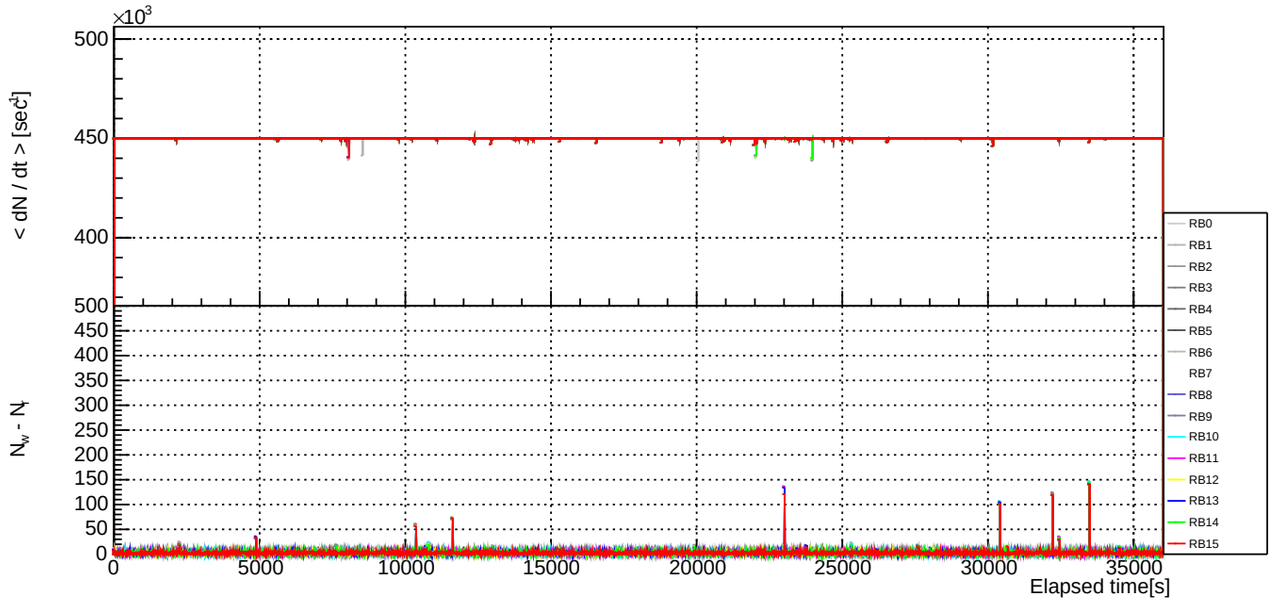


図 6.57　結合機能有り 2 対 16 接続 10 時間の DAQ 結果。トリガーレート 45kHz

第 7 章

まとめと今後の課題

まとめ

本論文では大口径望遠鏡（LST）のデータ収集システムのために、1 台の望遠鏡に搭載される 265 枚の FEB からデータを TCP/IP により取得し、イベント毎に結合し、後段の処理に回すプログラムの処理をマルチスレッドプログラミングで開発した。また、データ収集の性能評価のためにエミュレータの開発を行った。性能評価では、データ取得性能のマルチスレッド化効果、イベント結合機能の試験と性能評価を最大 26 接続で行い、それを元にサーバー分割最小単位の 45 接続、望遠鏡単位の 265 接続における性能を予測した。このスケールテストは Dragon FEB のデータ送信機能のエミュレーターを開発して行った。結果、データ取得性能は 2 スレッド 26 接続において 45kHz に追随し、45 接続における予測値 30kHz は要求値のほぼ 2 倍を満たした。また、イベント結合機能は、20 接続 40kHz で 20 分、16 接続 45kHz で 10 時間の機能試験に対して安全に稼働でき、一部の FEB がトリガ受信を失敗することによるトリガ番号の部分的な抜けにも柔軟に対応してイベント結合ができた。

課題

機能の拡張について

本実験では、データの取得とイベント結合の処理を開発した。イベント結合後は 1 秒以上の間はメモリに保持して、アレイトリガの受信と後段の処理のアクセスを可能にしなければいけない。よって、今後はイベント結合処理の後段にリングバッファを追加し、低レベル解析とオンサイトアーカイブへの送信処理の開発を計画している。また、FEB が量産された際は、データの低レベル解析処理を実装したいと考えている。

データ構造とトリガについて

データの仕様やトリガの仕様などは、まだ完全には固まっていない。データ収集プログラム側では、データの仕様やトリガの仕様をもとに、イベント結合の際に各 FEB からのデータにヘッダをつけて区別したり、トリガラベルは結合の際に取り除いて 1 箇所に纏めることでデータサイズを節約したり、あるいは低レベル解析の結果もヘッダに付加するなどの処理を実装したい。プロジェクトの関係グループと綿密な打ち合わせを行い、仕様確定や開発を続けていきたいと考えている。

ネットワークについて

今回は 26 接続までで性能の評価を行った。また、用いたデータ送信ノードは実際の FEB ではなくエミュレータである。実際の FEB265 接続から安定したデータ収集を行うためには更に負荷耐性の調査が必要である。注意しておくべき点は以下のとおりである。

エミュレーターの調整

エミュレータのトリガ受信率を上げるためには、UDP パケットの到達性を上げる必要がある。スイッチのパラメター調整でこれに寄与することができる。QoS という、受信側の負荷を判断して UDP パケットを落とす機能があり、今回の試験ではこれが有効になっていた。今後の試験では無効にしてトリガの到達率を更に上げることが望ましい。

また、データ送信ホストに Dragon FEB を 1 台加えたデータ収集試験では、エミュレータのみを使用した場合に比べてデータ取得性能が落ちることが判明した。SiTCP の制約によりネットワークに問題が起こっていることが予想され、ネットワーク部分の調査のために、エミュレータの性能を Dragon 同様の制約に調整した試験が必要である。現時点で要因として考えられる SiTCP の制約は、送信バッファが小さい点とフロー制御を受け付けない点である。

SiTCP での送信バッファサイズは 16kB だが、今回用いたエミュレータは PC 上に実装されたため、送信バッファサイズが大きい。sysctl コマンドにより net.ipv4.tcp_wmem の値を参照することで送信バッファサイズを見ると、(minimum, default, maximum) = (4096, 16384, 4194304) である。またカメラサーバーでの受信バッファサイズは net.ipv4.tcp_rmem の値を確認すると (4096, 87380, 4194304) であり、この受信バッファのウィンドウサイズのスケーリングも有効になっている[*1]。このためエミュレータからのデータ収集の場合、ウィンドウ制御によって受信ウィンドウが 16kB を大きく超える値になる一方で、エミュレータの送信バッファサイズが大きいために確認応答を待たずに多くのデータを送信することができ、高い帯域が可能になった可能性がある。

また、フロー制御が効かない場合、スイッチから送信される Pause Frame を無視してパケットがスイッチに送信されてしまい、パッファあふれを起こしパケットロスが発生する可能性がある。とくに実際想定している 45 接続の状況下では危険度がさらに増加する。これによって輻輳が起こり、ネットワーク全体の帯域が低下する可能性もある。

以上の可能性を踏まえて、エミュレーターの送信バッファサイズを 16kB に設定し、スイッチのフロー制御機能を無効にした上で、Dragon FEB と通信上の特性がより近づくことを確認したい。このために Dragon プロトタイプを再度入手し詳細なパケット解析を行うことで Dragon の通信上の特性を把握しなければならない。その後より Dragon に近い特性に調整したエミュレータにより試験を実施することで、スイッチやカメラサーバーのパラメータを調整して、ネットワーク上の問題を解決したいと考えている。

また近々量産される望遠鏡初号機搭載バージョンの Dragon Ver.4 を利用してミニカメラが製作され試験が行われるが、これを利用して Dragon からの大規模データ収集の試験を行って改良結果を確認したい。

[*1] これを決めるパラメターは net.ipv4.tcp_window_scaling=1

FEB の改良

SiTCP にフロー制御が実装されていないことへの対策法として、各 FEB からの送信タイミングをずらす方法が考えられる。トリガが配布されることで各 FEB が一斉にスイッチに向かってパケットを送信するが、送信タイミングを FEB 間でずらすことにより到着のタイミングもずれ、スイッチの受信負担が軽減される。パケットロスが観測された場合には、この手段を FEB に実装したいと考えている。

カメラサーバ内部の処理について

20 接続の結合機能のテストでは 38kHz 辺りからバッファ滞留量の上がっていく様子が見えた。Dragon 側でエラーが発生してデータの到来が非一様になった場合、同様に結合機能での対応が難しくなると予想される。トリガ損失率、データ転送時の輻輳制御など、エミュレータと実際の Dragon で異なる点がいくつかあるため、ミニカメラテストによる検証は必要だが、場合によってはイベント結合機能を 2 段階に分割することで Builder スレッドの負担を減らすことも必要となるだろう。

コア数

1 スレッドのみでデータ収集を行った場合、20 接続で 32kHz 程度までは 20 分の安定稼働に成功した。また 2 スレッドでは 16 接続で 45kHz のトリガレートに耐えて 10 時間の安定稼働が達成できた。これをスケールさせて必要最低限のコア数を見積もると、安全のためトリガレートは要求性能 15kHz の倍の 30kHz として考えた時、カメラサーバーの分割単位に適している 45 接続（スイッチ 1 台当たりの接続数）については、Collector が 2 スレッド、Builder が 1 スレッドないし 2 スレッド必要である。また望遠鏡単位でのデータ収集を考えると、接続数 265 はスイッチ 6 台ぶんに相当するために Collector スレッドと Builder スレッドはそれぞれ 6 倍、さらに Builder はこの場合イベント結合処理が分散されているため、それを更に統合するスレッドが追加で必要になる。

よって、イベント結合まででコア数は望遠鏡 1 台当たり 19 から 25 必要になると予想される。このコア数から、ノードは複数に分割することは適切である。

バッファサイズ調整

リングバッファのサイズと受信バッファのサイズは、今後の開発の進展に伴って調整が必要である。リングバッファの役目は、データ到着が非同期のために発生する到着データ量の各接続間での揺らぎをイベント結合処理という同期化のために吸収することと、後段の処理が行われている間にデータを保持することである。データ到着の揺らぎはトリガレートが低いほど小さく抑えられる。たとえば 1% 程度の揺らぎは、100kHz では 1 秒あたり 1000 件ほどの差を生むのに対し、10kHz では 100 件で抑えられる。一方で、今後後段の処理を増やしていくため、保持するべき時間は長くなる。これらのバランスを考えて調整が必要になると考えている。

また受信バッファは、プログラム内の処理負担に対して余裕をもったバッファサイズが望ましい。そのため、ソケットバッファと、更にその前段のカーネルバッファのサイズを大きめに取ることを考えている。カーネルバッファは 1 接続当たり 1MB 程度にすることが適切と言われている。

比較すべき関数機能

今回作成したデータ収集プログラムで用いた関数の幾つかは、代替候補がある。以下の関数において処理性能が不足する場合は比較を行ってより良い関数に変更することが望ましい。

- select
  epoll または poll という、ファイル記述子を用いずにソケットバッファ中のデータ到着を判断して読み込む関数がある。また、ioctl 関数で SIOCINQ を指定する方法でもバッファ内の滞留を判断できる。
- RingBuffer
  今回は Ring Buffer の機能をコーディングによって実装したが、Linux にはメモリ管理用のシステムコール群として QUEUE が用意されている。Linux によって用意されている機能のため保守性が高く、非常に高機能であるために拡張性も高い。
- Pthreads
  現在最新の C++ である C++11 では、boost ライブラリが利用できる。boost ライブラリにも Pthreads に代わる boost::thread が用意されている。実装がより簡便になっているほか、いくつかの点で処理速度の向上も見込まれる。

エラーハンドリング

データ収集プログラムに、現時点で以下の様なエラーハンドリングの実装が必要と考えている。

- トリガ番号エラー
  FEB がトリガの受信に失敗することでその FEB から到着するデータのトリガ番号に抜けが出た場合、またはハードウェア上のビット反転現象によりトリガ番号に異常が起こった場合の処理が必要である。現在は読み捨てで対応しているが、エラー部分やデータの抜けている部分にはダミーデータの埋込むなどの対策を行って、エラー内容の把握が容易なシステムにすることが望ましい。
- 受信エラー
  スイッチを長期間にわたって使用した場合、ファンや電源が故障することでスイッチとの通信が切断される場合がある。またポート単位での故障により、特定のノードからデータが届かなくなることもあり得る。スイッチ側でのデュラビリティ試験も必要だが、これらのエラーを感知する処理を実装するべきである。
- 処理能力超過エラー
  観測の障害となる強い光の影響や FEB の故障により異常に高いトリガレートが起こった場合の対処が必要である。システムの処理能力を超えた場合は、1 秒程度のデータを全部フラッシュさせリセットするといった処理を実装すべきである。

付録 A

Hillas parameter の導出

ガンマ線天体を期待する方向が視野中心とし、かつこれをカメラ内の直交座標 (x, y) における原点とする。カメラにおける検出器座標の各点 (x_i, y_i) には強度 s_i で光子が検出されたとする。カメライメージに対して、楕円に見立てたパラメータを定義する。

- 楕円の重心 (c.o.g , center of gravity) $(< x >, < y >)$
- 楕円の長軸 $y = ax + b$
- 楕円の長さ（$Length$）
- 楕円の幅（$Width$）
- 原点から重心までの距離 $(Distance)$
- 原点から長軸までの距離 $(Miss)$

重心の位置は、

$$< x >= \frac{\sum_i s_i x_i}{\sum_i s_i}, \qquad < y >= \frac{\sum_i s_i y_i}{\sum_i s_i} \tag{A.1}$$

長軸の式を導く。長軸はイメージの中心を走っている。したがって、長軸 $y - ax - b = 0$ は、イメージを構成する点 (x_i, y_i) までの距離 $l_i = \frac{|(y_i - x_i - b)|}{\sqrt{1+a^2}}$ の和を最小化する直線となるはずである。そこで、

$$F = \sum_i l_i^2 = \sum_i \frac{(y_i - x_i - b)^2}{1 + a^2} \tag{A.2}$$

最小二乗法によって、$F = F(a, b)$ として、a, b を動かした時の F の極小点を求め、a と b の条件を導く。極値の条件より、

$$\frac{\partial F}{\partial b} = 0 \quad \text{より、} \quad < y > -a < x > -b = 0 \tag{A.3}$$

$$\frac{\partial F}{\partial a} = 0 \quad \text{に (A.3) を代入して、} \quad S_{xy}a^2 - (S_x^2 - S_y^2)a - S_{xy} = 0 \tag{A.4}$$

ここで、

$$S_x^2 =< x^2 > - < x >^2, S_y^2 =< y^2 > - < y >^2, S_{xy} =< xy > - < x >< y > \tag{A.5}$$

この式は、a に関する 2 次方程式になっている。$S_x^2 - S_y^2 = d$ とおいて、

$$a^2 - \frac{d}{S_{xy}}a - 1 = 0 \tag{A.6}$$

ここから 2 つの解 $a_{\pm}$ は $a_+ \cdot a_- = -1$ の関係があり、直交する 2 つの軸であることがわかる。$a = a_+$ として、

$$a = \frac{d + \sqrt{d^2 + 4S_{xy}^2}}{2S_{xy}} \tag{A.7}$$

次に、Width, Length を求める。Width は、イメージを構成する各点 (x_i, y_i) の $y = ax + b$ からの距離の分散とすればよく、

$$\begin{aligned} Width =< distance^2 > =< \frac{(y_i - ax_i - b)^2}{1 + a^2} > \\ = \frac{S_y^2 + a^2 S_x^2 - 2aS_x y}{1 + a^2} \end{aligned} \tag{A.8}$$

同様に、Length は直交する直線からの距離の分散で求め、$a_+ \cdot a_- = -1$ に注意すると、

$$Length = \frac{S_y^2 + a^2 S_x^2 + 2aS_x y}{1 + a^2} \tag{A.9}$$

$(Miss)$ は原点から直線 $y = ax + b$ までの距離で求め、

$$Miss = \frac{|b|}{\sqrt{a^2 + 1}} \tag{A.10}$$

付録 B

Dragon FEB の読出し技術について

Dragon FEB の読出し技術で使用されているコンポーネントの詳細は以下のとおりである。

DRS4

DRS4 は Domino Ring Sampler version 4 の略で、スイス PSI 研究所が開発したアナログメモリの ASIC である。DRS は素粒子分野の MEG 実験や、IACT では MAGIC で使用実績がある。多数のキャパシタが並列に並んでいて、トランジスタによるスイッチによって切り替えられながら順番にアナログ信号が伝えられて電荷となって保管されていく、というやり方でサンプリングに使用される。

ADC

ADC は Analog Devices 社製 AD9222 を用いている。12bit, 8 チャンネルの ADC で、8 個の DRS4 チップに保存された電圧値のデジタル変換に使用されている。

エレクトロニクス

FPGA
: FPGA (Field Programmable Gate Array) とは、大規模な PLD (Programmable Logic Device) である。内部構造は無数のロジックセルによる集積回路であり、1 つのセルは組合せ回路を記述する LUT (SRAM の一種) と、順序回路を記述する FF (Flip Flop) で構成されている。LUT やロジックセル間をつなぐインターコネクトは RAM であり、これらに値を書き込む（コンフィギュレーション）ことでマクロセルをスイッチングマトリクスで接続することにより、目的の回路を形成することができる。ただしこのように RAM ベースのアーキテクチャなので、電源を落とすと回路は初期化される。電源投入時には、JTAG ケーブルにより PC から FPGA へプログラムをロードするか、または PROM へプログラムを書き込んでおき FPGA へ読み出されるように設定する。このようにユーザーが自由に内部回路を書き換えることができ、目的の回路を得るためのコストや時間が大幅に削減できるため、広範な用途で用いられている。

CPLD
: CPLD (Complex Programmable Logic Device) は FPGA と同様の集積回路である。しかし不揮発性のメモリをベースにしているので、FPGA と異なり電源を落としても回路が消えない。FPGA より規模

は小さいが、LSI 間を相互接続したり、不揮発性を生かしてブートローダに使用されたりする。

ASIC

ASIC (Application Specific Integrated Circuit) も集積回路だが、回路の設計図を作成しし、それを元に業者が製造をするものである。完成後にユーザーが回路を書き換えることはできない。大量生産する場合には FPGA よりも単価が安くなる。

SiTCP

SiTCP は FPGA 内にネットワーク処理回路を実装する技術である。1Gbps イーサネット通信が可能なため、安価な市販製品を用いて、高い接続性で高速な長距離通信が可能である。FPGA 内で同期 FIFO に書き込む手順でデータを SiTCP に書き込めばデータがイーサネットで転送される。ユーザー側は、ソケットプログラミングの標準関数のみのコーディングで TCP/IP 通信を利用してデータ取得ができる。使用リソース数は約 3000 Slice（全機能実装時）で、Xilinx 社製の FPGA 用のライブラリとして提供されている。イーサネットチップである PHY デバイスと共に用いる。[15]

付録 C

ネットワークの基礎

本研究で基礎となるネットワーク技術について、ここに補足としてまとめる。[19][20][13]

C.1 単位

伝送速度単位

本論文中で扱う数値について混同を避けるため、バイト、ビットの接頭辞について解説する。
コンピュータの内部表現では 2 進数を採用しているため、2^n で最も 1000 に近い 2^{10} が単位の接頭辞に用いられる。つまり、

- 1K = 1024
- 1M = 1024K
- 1G = 1024M

と表現する。これに対し伝送速度は伝送時に利用されるクロックの周波数により決まるため、

- 1K = 1000
- 1M = 1000K
- 1G = 1000M

と表現する。よって本論文で使用する単位の接頭辞は、特にコンピュータ内部表現については i を付加して KiB、MiB、GiB などと表現する。

オクテット

バイトという単位は、1 バイト=8 ビットという換算にならない場合も存在する。明示的に 8 ビットをまとめる単位としてオクテットが使われる。ネットワークの伝送でパケットの内部構造について説明する際はこのオクテットを使用する。

C.2 通信のプロトコル階層

プロトコルの階層化と OSI 参照モデル

プロトコルとは、「通信のための約束ごと」である。コンピュータ同士が互いに通信するためには、互いのコンピュータ同士の特定、通信の確立、データの確実な送受信、データフォーマットの判別など、実に様々な機能が必要であり、それぞれに「約束ごと」が必要になる。そのため、それに応じて様々なプロトコルが用意されている。また、上位層と下位層の間でサービスのやりとりをする時の約束事を「イ

ンターフェース」、通信相手の同じ階層とやり取りをするときの約束事を「プロトコル」と呼ぶ。
このようにプロトコルを階層化しておくことで、ある階層の内部を変更しても、その影響がシステム全体に波及しないため、拡張性や柔軟性に富んだシステムを構築できる。また、通信の機能分割が行われるため、それぞれの階層のプロトコルを実装することが容易になる。
OSI 参照モデル[*1](表 C.1) とは、複雑になりがちなネットワークプロトコルを単純化するために、通信に必要な機能を 7 つの階層に分割するモデルである。多くの通信プロトコルはこの OSI 参照モデルの 7 階層のうちのどれかの層に当てはめて考えることができる。本論文でも、この階層を参照しながら通信の各コンポーネントを説明していく。

表 C.1　OSI 参照モデル

	層	機能
7	アプリケーション層	特定のアプリケーションに特化したプロトコル。ファイル転送や電子メール、遠隔ログインなど、目的に応じたプロトコル。
6	プレゼンテーション層	ソフトウェアのための機器固有のデータフォーマットとネットワーク共通のデータフォーマットの交換。機器の違いが有ってもデータの表現形式の整合性をとるためのプロトコル。
5	セッション層	通信の管理。コネクション（データが流れる論理的な通信路）の確立／切断、データ転送タイミングの管理。トランスポート層以下の層の管理とも言え、実際に転送する機能は持っていない。
4	トランスポート層	両端ノード間のデータ転送の管理。データの到達を確認し、届かなかったデータを再送するなど、データ転送の信頼性を提供する（データを確実に相手に届ける役目)。
3	ネットワーク層	アドレスの管理（アドレス体系の決定）と経路の選択を行い、データを配達するためのプロトコル。
2	データリンク層	直接接続された機器間でのデータフレームの識別と転送。
1	物理層	“0”と”1”のビット列を、電圧の高低や光の点滅に変換する。コネクタやケーブルの形状の規定。

TCP/IP

TCP/IP という言葉は、TCP と IP という 2 つのプロトコルだけではなく、IP を利用したり、IP で通信したりするときに必要となる多くのプロトコル群の総称として使われる。インターネットを構築する上で必要なプロトコルのセットという意味でインターネット・プロトコル・スイートとも呼ばれる。現在、インターネットは ARPANET から発展し、全世界を接続しているコンピュータネットワークのことを指す。つまり、インターネットのプロトコルは TCP/IP のプロトコルである。IETF(Internet Engineering Task Force) によって標準化作業が行われている。

物理層：イーサネット

イーサネット[*2]はデータリンク層を支えるための物理的な規格である。現在最も普及しているだけでな

[*1] ISO（国際標準化機構）により提唱された。
[*2] 元々は Xerox 社と DEC が考案した通信方式だったが、IEEE802.3 委員会によって規格化され、この仕様を特に 802.3Ethernet

く、100Mbps、1Gbps、10Gbps、さらには 40Gbps ／ 100Gbps と高速ネットワークへの対応が進み、互換性と将来性も備えていると言える。当初のイーサネットは半二重通信が前提とされていたが、現在はスイッチ技術の進歩によりほとんどが全二重通信に移行し、半二重通信は利用されなくなっている。

データリンク層：MAC アドレス

MAC アドレスはデータリンクに接続しているノードを識別するための規格で、イーサネットでは IEEE802.3 で規格化された MAC アドレスが利用されている。MAC アドレスは 48 ビットの長さをもち、主にベンダ識別子とベンダ内での識別子からなる。上位層の IP アドレスと違い、MAC アドレスを使えば個々のネットワーク機器が判断でき、実際の配送先ノードの物理的特定に関与できる。そのため、このアドレスは各ネットワーク機器間の転送で使われる。

ネットワーク層：IP

IP(Internet Protocol) は、インターネットのためのプロトコルであり、IP アドレスを用いる。このインターネット層は OSI の第 3 層であるネットワーク層に相当する。ネットワーク層の下位に位置するデータリンク層はノード間のパケット転送を行うが、その経路の違いを気にすることなくパケットの配送を行えるようにすることによって終点ノード間の通信を実現する。これを経路制御という。

トランスポート層：TCP と UDP

TCP（Transmission Control Protocol）と UDP（User Datagram Protocol）は共に、ポート番号によってホスト内でのアプリケーションからデータを受け取り、そしてアプリケーションにデータを届ける役割を持つプロトコルである。

TCP は IP の機能を拡張し、終点ホスト間で信頼性のある通信を提供している。そのためにコネクション指向のプロトコルとなっており、コネクションの開始・終了により通信開始・終了が行われる。また、コネクション確立中は、送信データの順序番号制御とそれに対する確認応答制御を行い、データの紛失や食い違いを非常に高い信頼性で防いでいる。これらの機能を実現するためのプロトコルは TCP ヘッダの中に埋め込まれている。これについては後述する。

一方で UDP は IP の機能をそのままアプリケーションから利用するために作られたプロトコルである。コネクションの確立はなく、そのため到達の信頼性はないが、高速な転送を実現する。信頼性の低さは、UDP を利用する側のアプリケーション層プロトコルで実装することでまかなえるため、UDP を利用する多くのアプリケーションでエラー検出や再送信タイマーなどを独自に備えている。本研究ではエミュレータのトリガ信号として用いているが、Dragon FEB においてはスローコントロールで命令信号の送信に利用している。

C.3 通信データのカプセル化

OSI 参照モデルのようにネットワーク通信に対して機能毎に層を設けて役割分担を行った場合、データには各層のプロトコルが必要になる。このためデータには各層のプロトコルで書かれた情報が付加されてやり取りが行われる。また、受信側が各層に応じたプロトコルの情報を取り出せるように、各層では送信データに自分の層のプロトコルによる情報を先頭または末尾に付加した後、下位層に渡している。そうすれば受信側は、自分の層では先頭または末尾から必要な情報を取り出して取り除いた後に上位層に渡すことを下位層から上位層に向けて繰り返すだけで、効率的に目的の通信相手のアプリケーションにデータを受け渡すことができる (図

と呼ぶことがある。

C.1)。この方法をデータのカプセル化と呼び、それぞれの層で先頭、末尾に付加されるデータを「ヘッダ」「トレーラ」という。また、この送信される時のデータの単位をパケット、フレームなどと呼ぶ。

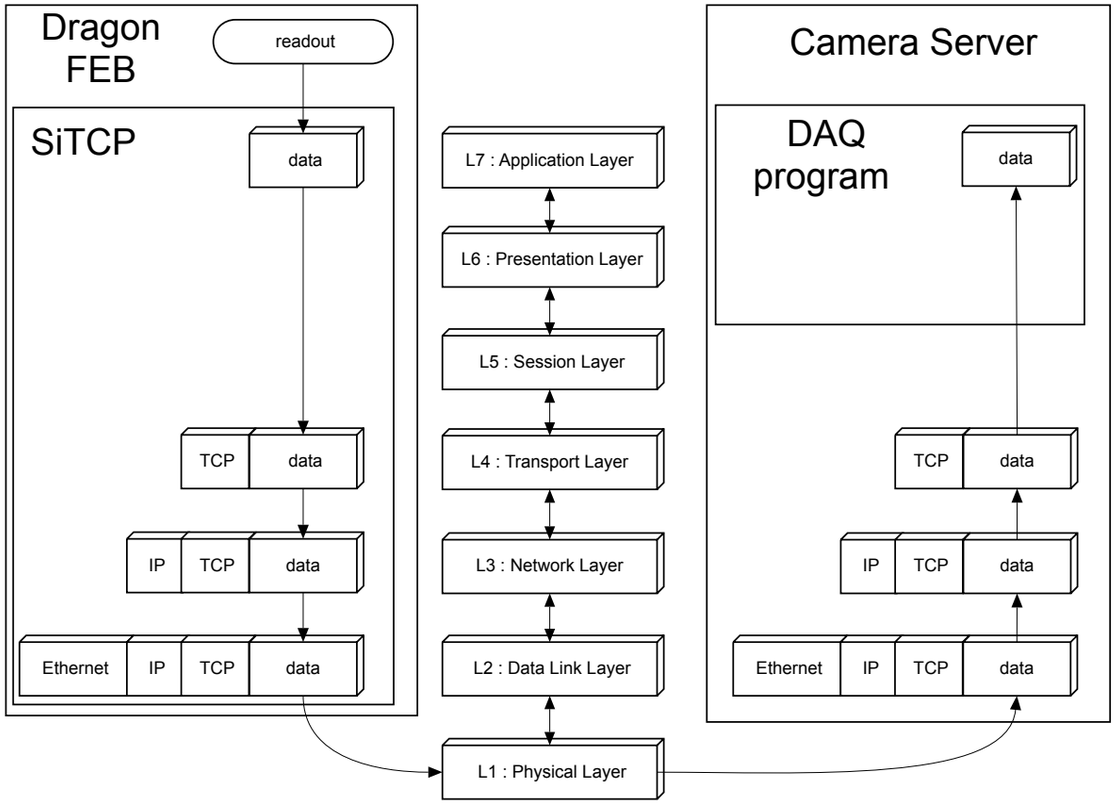


図 C.1　各プロトコル階層によりパケットヘッダが付加されることでカプセル化が行われる様子。

C.4　各プロトコル階層のヘッダ

パケットの構造はおおまかに (図 C.2) の様になっている。送信されるデータに様々な階層のヘッダ・トレーラが付加されたあとでデータは送信されるが、この付加されたヘッダ、トレーラそれぞれについて簡単に解説する。

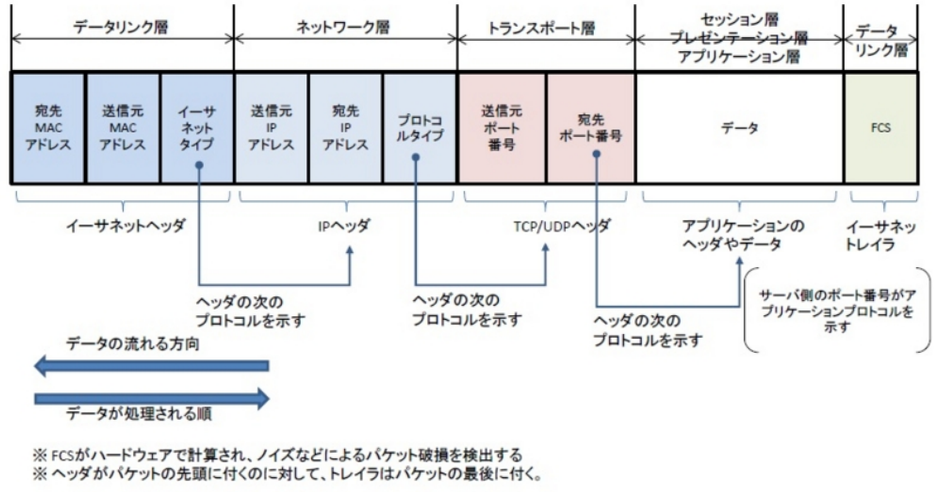


図 C.2　パケットの大まかな構造。

プリアンブル、SFD**、**IFG

まず、ヘッダ、トレーラを含めた、パケット全体を識別するため信号について説明する。

ネットワークでのデータの伝送は on、off の 2 種類の信号の組み合わせである。この信号の識別精度を上げるために、転送するフレームの先頭にはプリアンブルと SFD（start frame delimiter）という信号が先行する。プリアンブルは MAC フレーム[*3]伝送の同期確立に使う信号で，1 と 0 が 7 バイトにわたって繰り返される。SFD はプリアンブルに続いて送られる 1 バイトの信号である。プリアンブル同様 1 と 0 を繰り返すが、最後の 2 ビットのみ 11 となり、MAC フレームの始まりの部分を示すために使われれる。つまり、合計で 8 バイトの信号がパケットの先頭につく。

イーサネットでは、フレーム送出後に一旦通信を停止し、最低 12 バイトのフレーム間ギャップと呼ばれる感覚を空けてから次のフレームを送出する。これを IFG（Inter Frame Gap）と呼ぶ。これにより、小さいフレームが連続して伝送される際に発生する伝送効率の低下を改善する対策をとっている。

イーサネットヘッダと FCS

イーサネットヘッダは (図 C.3) の上図の構造を持ち、MAC アドレスが埋め込まれる。始点 MAC アドレスと終点 MAC アドレスがそれぞれ 6 バイト、送信タイプに 2 バイトの合計 14 バイトでイーサネットヘッダは構成されるが、スイッチ間の転送などで VLAN を設定している場合などは、送信タイプの部分の構造と長さが変わる。また、FCS（Frame Check Sequence）という 4 バイトの誤り訂正用トレーラが付加される。

IP **ヘッダ**

IP ヘッダは (図 C.3) の下図の構造を持ち、基本は全部で 20 バイトの長さで、主に始点 IP アドレスと終点 IP アドレスをもつ。この図は IPv4 の場合のヘッダ構造で、IPv6 の場合は、先頭の 4 ビットで違

[*3] データリンク層以上の全てのヘッダが付加された状態のフレームを言う。つまり、イーサネットヘッダ、IP ヘッダ、TCP/UDP ヘッダ全てが付加された状態である。

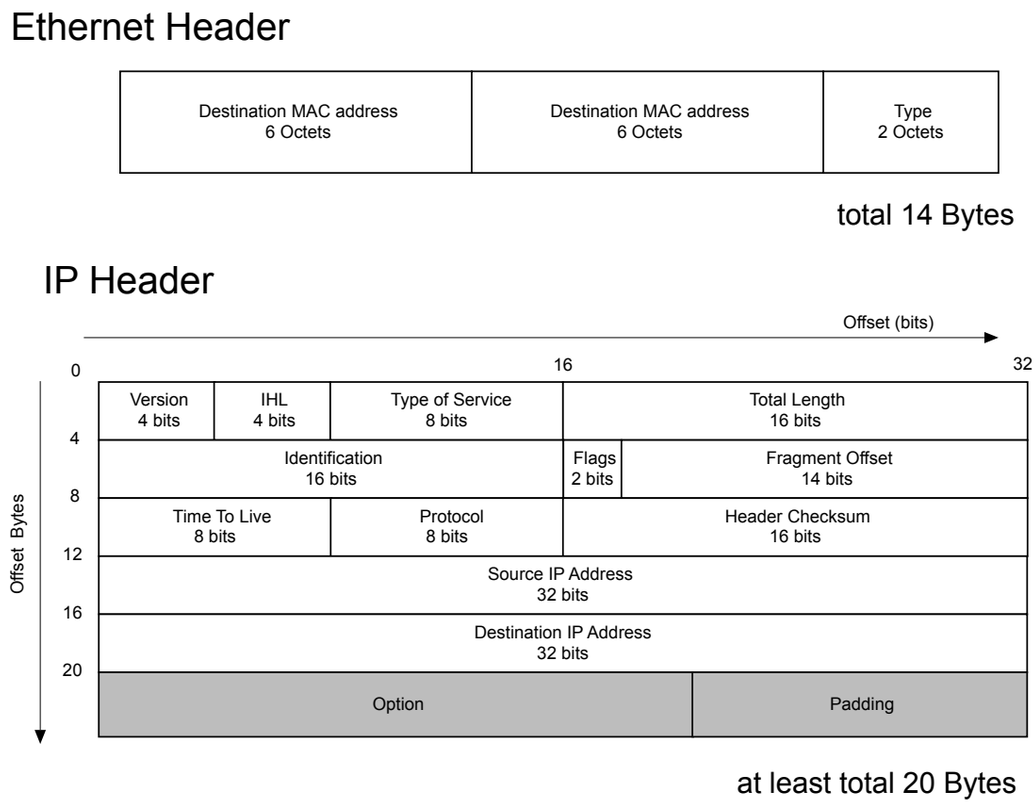


図 C.3　イーサネットヘッダと IP ヘッダの構造。

う値が指定され、構造と長さも異なる。

TCP ヘッダと UDP ヘッダ

主に始点ポート番号と終点ポート番号を持つ。TCP ヘッダでは通信確立、確認応答、ウィンドウ制御など様々な機能を実現するために、ヘッダの中に様々な情報を埋め込めるようになっている。そのため 20 バイトが最小のサイズになる。オプション部分は通常は使われない。

UDP ヘッダでは、始点ポート番号と終点ポート番号の他にはパケット長、チェックサムのみでわずか 8 バイトの非常にシンプルな構造をしている。

C.5　通信データの断片化と MTU、MSS

1 回で送ることのできるデータのサイズには上限が設けられている。これはデータリンク毎に決まっていて、「MTU（Maximum Transmission Unit）」と呼ぶ。イーサネットでは 1500 オクテットが一般的である[*4]。

一つ上の IP 層がデータの長さを管理しており、アプリケーションが IP 層に送信要求を行ったデータが連続している場合やサイズが大きい場合は、IP 層はデータリンク層にデータを渡す前に、この MTU に IP データ

[*4] これを変更することができ、大きく設定する場合はジャンボフレームと呼ばれる。ジャンボフレームでは 9000 オクテットに設定することが一般的である。変更する場合はデータリンク内の全てのネットワーク機器を対応させる必要がある。

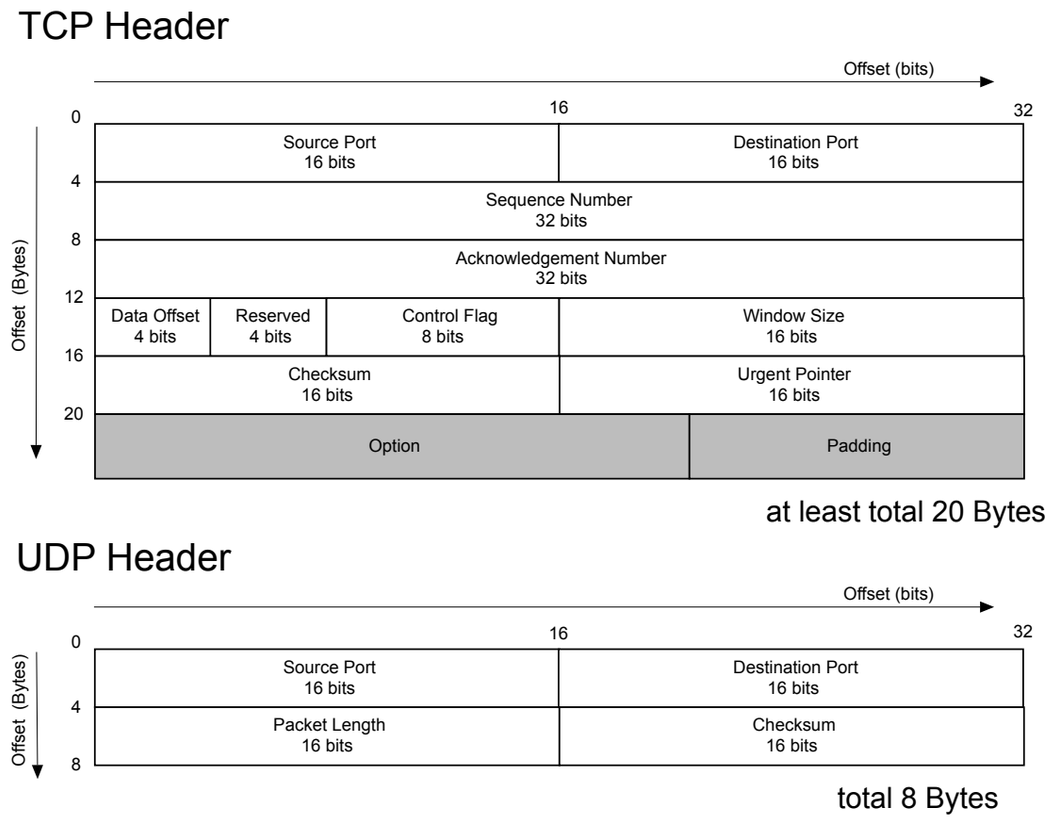


図 C.4　TCP ヘッダと UDP ヘッダの構造。

グラム[*5]が収まるようにデータを再分割する。これを IP フラグメンテーション（断片化）という。TCP/IP 通信の場合、IP データグラムは TCP ヘッダ 20 バイトと IP ヘッダ 20 バイトを付加された状態なので、データ自体は IP データグラムより 40 バイト短い長さである。そのためアプリケーション側で見ると、MTU が 1500 オクテットの場合は、データが再分割されない最大長は 1460 オクテットになる。

アプリケーション側で見た最大転送長を MSS（Max Segment Size）という。

C.6　TCP 通信制御

Dragon FEB からの観測データ転送は TCP/IP 通信である。これにより信頼性の高い通信を実現しているが、ここではその信頼性を実現するための TCP 通信の機能を概観する。

3 way handshake による通信の確立

コネクション指向の TCP 通信では、通信の開始時にお互いが通信の確立を認識することと送受信データの順序性を保証するために、通信開始専用のパケットのやり取りを 3 回行う。

1 回目は通信開始要求元から要求先への「SYN」パケット、2 回目は 1 回目の応答として要求先から要求元への「SYN/ACK」パケット、3 回目は 2 回目の応答として要求元から要求先へ送られる「ACK」パ

[*5] IP 層以上のヘッダが付加された状態のフレーム。

ケットである。

シーケンス番号と確認応答によるデータ到達の保証

TCP 通信におけるデータのやり取りには、必ずシーケンス番号が付番されている。送信されるデータには通信開始から数えて何バイト目から始まるデータかという情報が含まれている。受信側はそれを読み取ることにより、データ到着が前後しても影響を受けないようになっている。また受信側はこの情報を元に、何バイト目まで受信が完了しているという情報を ACK パケットとして送信側に返送している。

ウィンドウと輻輳制御

データの送受信のために、送信側は送信待機用のバッファを、受信側は受信後アプリケーションによる処理待ちのバッファを持っている。しかし、データ送信の効率化・ネットワーク混雑の回避のために、ウィンドウサイズを導入して送受信可能量を制御している。

FlowControl

TCP は、パケット消失が起こりそうになると、データ転送レートを調整してこれを防ぐ「スライディングウィンドウ」という機構を実装している。スライディングウィンドウは、受信側のウィンドウサイズを使ってデータフローを制御していて、例えばウィンドウサイズを小さくすると、小さくなったウィンドウサイズの情報は受信側から送信側への ACK パケットに載せられて送信側へ届き、送信側はそれを元に送るデータのサイズを絞る。

TCP **再送**

データ転送の際のパケットの消失が起こった場合、考えられる原因は様々である。しかしこのようなエラーは一時的な場合がほとんどのため、「TCP 再送」という、消失をリカバリすることで健全性を保つ機構が備わっている。TCP 再送はパケットの再送を行う機能で、送信側は各パケット送信後に再送タイマーをスタートし、予め設定された時間以上 ACK パケットの返信がない場合、再送タイムアウトとしてパケットの再送を判断する。

パケット送信から ACK が返信されるまでの時間は RTT（ラウンドトリップタイム）と呼ばれ、この値の平均値が再送タイムアウトまでの時間に利用される。

Nagle buffering

データ送信時、各パケットには TCP/IP ヘッダが付加されるため、小さなパケットを連続して送るよりは MSS のサイズにまでデータを各パケットに詰め込んで送る方がオーバーヘッドが少なくなる分効率的な転送が行える。これを実現するため、確認応答の返信が来るか、もしくは MSS に到達するまで送信データをバッファする機能である。

謝辞

本論文に直接関係した研究がスタートしたのは 10 ヶ月ほど前ですが、このように纏めてみれば本論文は修士課程の初めから積み上げた知識が裏で礎の大部分として機能している事を感じます。眺めている景色は 2 年間で大きく変化しました。

まずこの 2 年間の修士課程生活で指導教官としてお世話になった吉越先生に感謝させていただきます。このグループに来た当初、チェレンコフ望遠鏡での観測法は想像を絶する複雑さで大変苦労しましたが、複雑に絡み合う条件を根気強く教えて頂きました。注意すべきポイントが自分の頭で考えられるよう、様々な議論をしていただきました。プログラミングは特に初歩の段階から手ほどきして頂き、自分でコードを読みこなすきっかけをいただけたと思います。また、路線変更をしデータ収集システムをやりたいと言うわがままを承認して頂き、さらに五里霧中の状況の私に様々な助言をくださり、多数の書籍を貸して下さり、外部との折衝なども助けていただきました。このように新しく開けた視野の礎の部分を作っていただいたように思います。

また、数えきれないほど多くの方々のお世話になりました。垣根を超えて多くの方々と出会い、導いていただきました。以下に挙げる方々の協力抜きでは、特に終盤にかけて立て続けに起こった出会いの奇跡がなければ、ここまで走り切ることはできませんでした。まったく違う道を歩んでいたと思います。

中嶋さんには大変お世話になりました。DAQ のプロジェクトをスタートすることになった去年の暮れに真っ先に誘って下さり、4 月の本格スタートまで忘れずに居てくださったことがこの修士論文の始まりです。右も左もわからない私を根気よく、そして猛スピードで導いて頂き、わからないことが有って停滞している時にまるで手品の様に解決をして下さり、そこには置き土産として、発展させて使える核を作ってくださっていました。どうにか自分でできることを増やしましたが、終盤に再び大変お世話になってしまいました。この 1 年間は、中嶋さんという順化ケージの中に居たような気分です。

また手嶋先生からも DAQ への参加を薦めて頂き、そして進展のための重要なきっかけとなったセミナーも、手嶋先生が探してきて薦めてくださりました。このようにして、いつも先回りして道を切り開いてくださりました。様々な問題にいつも鋭い指摘をくださりました。普段から宇宙物理についての幅広い視点で縦横無尽に質問や指摘を頂き、考えを大きく変えていただきました。また必要なカメラサーバー開発環境用 PC を始めとした実験用のハードウェアを不足なく用意してくださり、非常に助かりました。

手嶋先生に紹介して頂いて赴いた 7 月の勉強会がきっかけで KEK 山形先生と出会うことが出来ました。初めて KEK 山形先生に相談をした折、すぐに私の窮状に気づいて頂き、お会いするたびに何時間も相談に付き合っていただいただけでなく、国立天文台大江先生をご紹介して頂き、NIC と DA ケーブルの貸与までして下さっています。頂いた助言は ATLAS 実験での豊富な経験に基づいているため、大きなスケールならではの問題点、注意点という大変貴重な情報をハードウェア、ソフトウェア両面にわたり知ることが出来ました。また、山形先生から教えていただいた timerfd が、エミュレータのトリガ機構開発のヒントになりました。山形先生の協力抜きでは実験のスタートは非常に困難を極めたことと思います。

また 11 月に山形先生に紹介していただいた国立天文台大江先生にも大変お世話になりました。日本最高速級

のデータ転送、書込の現場を初めて目の当たりに出来ただけでなく、ネットワーク・ストレージ・FPGA・OSなどについての、ベンダー側からも頼られるほどの知識を元にした助言を頂いております。また、非常に気にかけて頂き、内部勉強会である高速 PC ルータ研究会にお呼びいただき、発表の場を設けていただき、多くの専門家の方々から沢山の助言を頂きました。言葉に表せない程有難く思うとともに、自分のレベルが最先端に届いていないために制約が多く、非常にもどかしい気持ちです。一日も早く、頂いた助言を全て活かしたいと思います。

ATLAS グループ坂本研の浦野くんには、大変お世話になりました。9 月の物理学会で突然声をかけたにも拘らず 11 月に意見交換に時間を割いて頂きました。そこで頂いた情報が、RingBuffer の概念、mutex や cond などの同期制御など、本論文で作成したプログラムの設計構想具体化の始まりになりました。

11 月にサバティカルで伊から来日された Riccardo Rando 先生にも大変お世話になりました。マルチスレッドとマルチプロセスでの負荷の違いについての議論、最初にマルチスレッドで組んだ際のデバッグなど、とても有益なヒントをたくさん頂きました。議論にはいつも論理的思考があり、それを通して見える解決の近道があり、問題点の究明方法を根本から教わることができたと思います。

11 月に独より赴任で来日された Daniel Mazin 先生にも大変お世話になりました。現在の MAGIC における DAQ を元に、スレッドやリングバッファなどで注意すべき点など、たくさんの助言を頂きました。

カメラサーバー用の PC は、深見くんの作品です。この PC のお陰で実験が出来ました。良い性能を持った PC を使わせて頂き、非常に感謝しています。また、PC については甲南大の猪目くんに大変お世話になりました。PC の組み立て、その後の調整を手伝って頂き、特に不足部品の洗い出しと選定には助言により難を逃れることが出来ました。

スケールテストではご迷惑にもかかわらず、手嶋先生、IPMU 松田君、荻野さん、小島くん、高橋くん、林田さん、花畑さん、Daniel さん、Daniela さん、菅原さん、多くの方々に PC 提供の協力を頂きました。さらに IPMU の大林様からは、事務所有の 13 台もの PC を 1 ヶ月も貸与してくださいました。まさに直前の 12 月中旬で、これがなければテスト環境が整わず、滑り込みセーフとは絶対になりませんでした。

技術職員の大岡さんには Dragon Board や Slow Control Board の細かな背景をたくさん教えていただきました。理解が遅いにもかかわらず忍耐強く付き合って頂き、今後 FPGA のプログラムを組んでいく上での礎になったと思います。また京大の今野さんに Dragon Board の全体像と動かし方を非常に丁寧にわかりやすく教えて頂きました。最初にアクセスする上での障壁が大きく取り除かれ、大変助かりました。京大の窪先生には、プロジェクトスタートにあたって必要な文献を、リクエストに応えて何冊も購入して送付していただきました。今でも大切に利用させて頂いています。

大石先生にはモンテカルロシミュレーションの扱い方で助言をたくさんいただきました。今後低レベル解析の影響評価では非常に重要になります。教えていただいたことを役立てて前進したいと思います。

IPMU の松田君には、日々実験ばかりで遠ざかる物理の世界に幾度と無く連れ戻していただきました。やはり物理を楽しみたいという気持ちが根底にあります。しかし一旦遠ざかると計算や思考が落ちてしまいます。それでもわざわざ忍耐強く付き合って頂き、わかりやすく教えていただけました。年末年始の追い込みにはずっと一緒に執筆をし、一緒にコンビニでおでんを分けあったことは一生の思い出です。

また、岩本さんご一家、市川さんご一家にはいつも応援をして頂き、そして視野が狭くなりがちな日々の中で、毎回頭の中をリセットする機会を与えていただきました。学問を分け隔てなく純粋に楽しむという作業を手伝わせていただき、また自分が関わっている事をわかりやすく楽しく説明するという試みをいつも頂いていたお陰で、自分が煮詰まっている状況でも再び開発に熱意を傾けなおす事ができ、気持ちの面で大変救われました。

これまで多方面から援助を下さった父、母、祖父、祖母に感謝いたします。とくにデータ収集が向いているのではという指摘は母によります。始める前は自分がここまで楽しめるとは全く想像もできませんでした。そればかりか物理という分野は自分で選びましたが、元々道に迷っていた状態の自分に転戦の提案もして下さいました。ようやくここから選択基準の見える道が広がり始めましたが、見渡す限り荒野でしかなかった状態に道を示していただいたことに、ただただ脱帽です。そして父には苦しい状況にもかかわらず快く金銭的な援助をしていただきました。本当に有難うございます。修士課程入学を喜んでくれた祖父にはこの修士論文は間に合いませんでしたが、気持ちは届いているのでは無いかと思います。そして祖母にはこれからもしばらく迷惑を掛けますが、もう少しこのまま頑張りたいと思います。

最後に、この道の一番のきっかけとなったジャズピアニスト若井優也さんにお礼を申し上げたいと思います。全国を飛び回る多忙なプロ活動の中、セッションで一緒に演奏して頂いていただけでなく、その度毎にいつも心に残る楽しい数学の話をして頂き、数学の力を鍛えていただきました。そして宇宙物理に道を踏み出す事をとても喜んで頂き、宇宙物理関連の数学の勉強も手伝って下さり、幅広い興味を元に様々な分野の数学を手ほどきして頂き、常に視野を広げてくださいました。今までで出会った中で一番感動的な数学の授業は若井さんです。そのおかげでこのような道に踏み出すことになりましたので、若井さん抜きにはここまでの道のり全てがありえませんでした。ありがとうございました。明日も勉強会、楽しみです。

参考文献

[1] B. S. Acharya, M. Actis, T. Aghajani, G. Agnetta, J. Aguilar, F. Aharonian, M. Ajello, A. Akhperjanian, M. Alcubierre, J. Aleksić, and et al. Introducing the CTA concept. *Astroparticle Physics*, 43:3–18, March 2013.

[2] M. Actis, G. Agnetta, F. Aharonian, A. Akhperjanian, J. Aleksić, E. Aliu, D. Allan, I. Allekotte, F. Antico, L. A. Antonelli, and et al. Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy. *Experimental Astronomy*, 32:193–316, December 2011.

[3] F. Aharonian, A. G. Akhperjanian, A. R. Bazer-Bachi, M. Beilicke, W. Benbow, D. Berge, K. Bernlöhr, C. Boisson, O. Bolz, V. Borrel, I. Braun, F. Breitling, A. M. Brown, R. Bühler, I. Büsching, S. Carrigan, P. M. Chadwick, L.-M. Chounet, R. Cornils, L. Costamante, B. Degrange, H. J. Dickinson, A. Djannati-Ataï, L. O'C. Drury, G. Dubus, K. Egberts, D. Emmanoulopoulos, P. Espigat, F. Feinstein, E. Ferrero, A. Fiasson, G. Fontaine, S. Funk, S. Funk, Y. A. Gallant, B. Giebels, J. F. Glicenstein, P. Goret, C. Hadjichristidis, D. Hauser, M. Hauser, G. Heinzelmann, G. Henri, G. Hermann, J. A. Hinton, W. Hofmann, M. Holleran, D. Horns, A. Jacholkowska, O. C. de Jager, B. Khélifi, N. Komin, A. Konopelko, K. Kosack, I. J. Latham, R. Le Gallou, A. Lemière, M. Lemoine-Goumard, T. Lohse, J. M. Martin, O. Martineau-Huynh, A. Marcowith, C. Masterson, T. J. L. McComb, M. de Naurois, D. Nedbal, S. J. Nolan, A. Noutsos, K. J. Orford, J. L. Osborne, M. Ouchrif, M. Panter, G. Pelletier, S. Pita, G. Pühlhofer, M. Punch, B. C. Raubenheimer, M. Raue, S. M. Rayner, A. Reimer, O. Reimer, J. Ripken, L. Rob, L. Rolland, G. Rowell, V. Sahakian, L. Saugé, S. Schlenker, R. Schlickeiser, U. Schwanke, H. Sol, D. Spangler, F. Spanier, R. Steenkamp, C. Stegmann, G. Superina, J.-P. Tavernet, R. Terrier, C. G. Théoret, M. Tluczykont, C. van Eldik, G. Vasileiadis, C. Venter, P. Vincent, H. J. Völk, S. J. Wagner, and M. Ward. Observations of the Crab nebula with HESS. *A&A*, 457:899–915, October 2006.

[4] J. Aleksic, S. Ansoldi, L. A. Antonelli, P. Antoranz, A. Babic, P. Bangale, M. Barcelo, J. A. Barrio, J. Becerra Gonzalez, W. Bednarek, E. Bernardini, B. Biasuzzi, A. Biland, M. Bitossi, O. Blanch, S. Bonnefoy, G. Bonnoli, F. Borracci, T. Bretz, E. Carmona, A. Carosi, R. Cecchi, P. Colin, E. Colombo, J. L. Contreras, D. Corti, J. Cortina, S. Covino, P. Da Vela, F. Dazzi, A. De Angelis, G. De Caneva, B. De Lotto, E. de Ona Wilhelmi, C. Delgado Mendez, A. Dettlaff, D. Dominis Prester, D. Dorner, M. Doro, S. Einecke, D. Eisenacher, D. Elsaesser, D. Fidalgo, D. Fink, M. V. Fonseca, L. Font, K. Frantzen, C. Fruck, D. Galindo, R. J. Garcia Lopez, M. Garczarczyk, D. Garrido Terrats, M. Gaug, G. Giavitto, N. Godinovic, A. Gonzalez Munoz, S. R. Gozzini, W. Haberer, D. Hadasch, Y. Hanabata, M. Hayashida, J. Herrera, D. Hildebrand, J. Hose, D. Hrupec, W. Idec,

J. M. Illa, V. Kadenius, H. Kellermann, M. L. Knoetig, K. Kodani, Y. Konno, J. Krause, H. Kubo, J. Kushida, A. La Barbera, D. Lelas, J. L. Lemus, N. Lewandowska, E. Lindfors, S. Lombardi, F. Longo, M. Lopez, R. Lopez-Coto, A. Lopez-Oramas, A. Lorca, E. Lorenz, I. Lozano, M. Makariev, K. Mallot, G. Maneva, N. Mankuzhiyil, K. Mannheim, L. Maraschi, B. Marcote, M. Mariotti, M. Martinez, D. Mazin, U. Menzel, J. M. Miranda, R. Mirzoyan, A. Moralejo, P. Munar-Adrover, D. Nakajima, M. Negrello, V. Neustroev, A. Niedzwiecki, K. Nilsson, K. Nishijima, K. Noda, R. Orito, A. Overkemping, S. Paiano, M. Palatiello, D. Paneque, R. Paoletti, J. M. Paredes, X. Paredes-Fortuny, M. Persic, J. Poutanen, P. G. Prada Moroni, E. Prandini, I. Puljak, R. Reinthal, W. Rhode, M. Ribo, J. Rico, J. Rodriguez Garcia, S. Rugamer, T. Saito, K. Saito, K. Satalecka, V. Scalzotto, V. Scapin, C. Schultz, J. Schlammer, S. Schmidl, T. Schweizer, A. Sillanpaa, J. Sitarek, I. Snidaric, D. Sobczynska, F. Spanier, A. Stamerra, T. Steinbring, J. Storz, M. Strzys, L. Takalo, H. Takami, F. Tavecchio, L. A. Tejedor, P. Temnikov, T. Terzic, D. Tescaro, M. Teshima, J. Thaele, O. Tibolla, D. F. Torres, T. Toyama, A. Treves, P. Vogler, H. Wetteskind, M. Will, and R. Zanin. The major upgrade of the MAGIC telescopes, Part I: The hardware improvements and the commissioning of the system. *ArXiv e-prints*, September 2014.

[5] CTA-Japan Consortium and 東京大学宇宙線研究所. Cherenkov telescope array 計画書. **パンフレット**, 2014.

[6] K.A. Olive et al. (Particle Data Group). Review of particle physics. *Chin. Phys. C, 38, 090001*, 2014.

[7] J. A. Gaidos, C. W. Akerlof, S. Biller, P. J. Boyle, A. C. Breslin, J. H. Buckley, D. A. Carter-Lewis, M. Catanese, M. F. Cawley, D. J. Fegan, J. P. Finley, J. B. Gordo, A. M. Hillas, F. Krennrich, R. C. Lamb, R. W. Lessard, J. E. McEnery, C. Masterson, G. Mohanty, P. Moriarty, J. Quinn, A. J. Rodgers, H. J. Rose, F. Samuelson, M. S. Schubnell, G. H. Sembroski, R. Srinivasan, T. C. Weekes, C. L. Wilson, and J. Zweerink. Extremely rapid bursts of TeV photons from the active galaxy Markarian 421. *Nature*, 383:319–320, September 1996.

[8] A. M. Hillas. Evolution of ground-based gamma-ray astronomy from the early days to the Cherenkov Telescope Arrays. *Astroparticle Physics*, 43:19–43, March 2013.

[9] http://tevcat.uchicago.edu/. Tevcat.

[10] T.-P. Li and Y.-Q. Ma. Analysis methods for results in gamma-ray astronomy. *ApJ*, 272:317–324, September 1983.

[11] M.S.Longair. High energy astrophysics 3rd ed. *Cambridge University Press*, 2011.

[12] B. Nichols and J. P. Farrell. Pthreads プログラミング. **オライリー・ジャパン**, 1998.

[13] Chris Sanders. 実践パケット解析 第 2 版. **オライリー・ジャパン**, 2012.

[14] Diego Tescaro. Tev γ-ray observations of nearby active galactic nuclei with the magic telescope: exploring the high energy region of the multiwavelength picture. *PhD thesis, Universitat Autonoma de Barcelona*, 2010.

[15] T. Uchida. Hardware-Based TCP Processor for Gigabit Ethernet. *IEEE Transactions on Nuclear Science*, 55:1631–1637, June 2008.

[16] T. C. Weekes, M. F. Cawley, D. J. Fegan, K. G. Gibbs, A. M. Hillas, P. W. Kowk, R. C. Lamb, D. A. Lewis, D. Macomb, N. A. Porter, P. T. Reynolds, and G. Vacanti. Observation of TeV gamma rays from the Crab nebula using the atmospheric Cerenkov imaging technique. *ApJ*, 342:379–395, July

1989.

[17] CTA-Japan コンソーシアム. Cherenkov telescope array 計画. **パンフレット**, 2010.

[18] 安田 絹子, 小林 林広, 飯塚 博道, 阿部 貴之, and 青柳 信吾. マルチコア cpu のための並列プログラミング. **秀和システム**, 2006.

[19] 村山公保. 基礎からわかる tcp/ip ネットワーク実験プログラミング. **オーム社**, 2001.

[20] 竹下隆史, 村山公保, 荒井透, and 苅田幸雄. マスタリング tcp/ip 入門編 第 5 版. **オーム社**, 2013.

[21] 猪平栄一 and 重松保弘. Linux による並行プログラミング入門. **共立出版**, 2014.