

HL-LHC ATLAS ピクセル検出器量産時の品質試験に向けた データベースシステムの構築

東京工業大学 理学院物理学系物理学コース 陣内研究室
奥山広貴 (19M00398)

2021 年 1 月 19 日

Abstract

The ATLAS Experiment is being conducted with the Large Hadron Colider at CERN. Its purposes are measurement of the Standard Model (SM) and searches for particles beyond the SM.

In order to acquire more statistics and to achieve more advanced measurements and searches, LHC is plannig to increase the luminosity, referred to as HL-LHC. The target luminosity and integrated luminosity is approximately seven times and ten times higher respectively.

Due to the upgrade, it is required for detectors to have more radiation tolerance and high granularity. It is planed to replace the ATLAS inner detector to the new one, referred to as the Inner Tracker(ITk). ITk entirely consists of silicon detectors and covers much more wider solid angle acceptance than the current Inner Detector.

For the production of ITk, we are planning to produce $O(10,000)$ modules and conduct a series of tests for quality control(QC tests) for individual modules. Those are carried out repeatedly in the production flow. All the QC tests should be stored to a central database, which is set up at Unicorn university in Czech Republic, to record the performance of modules itself.

“Local database” system have been developed in a previous study in order to manage data at local production sites and to synchronize informations to check the central DB. The system have been under test-use among production sites towards full production. There were items left to be developed for the module production. Particularly the devlopment of the funcions specialized for the module production and QC tests and the synchronizing tools between the central database and local database.

I have developed the database system in this study. I have implemented key functions for the production, for example searching results, synchronizing data between the local and the central database, and validated that we can use the whole functionalities of the system, including my developed tools, using devices at the laboratory. Additionally I confirmed that we can use the tools with the actual data at the production to measure the processing time of the services.

概要

欧州原子力研究機構 (CERN) に設置されている大型ハドロン衝突型加速器 (LHC) の 1 つの衝突点にて、ATLAS 実験が行われている。この実験は現在、素粒子物理学の基本理論となっている標準模型の精密測定や標準模型を超えた新粒子の探索を目的とし稼働している。

更なる測定、探索に向けて取得統計数の増加を狙い、LHC では 2025 年より加速器をアップグレードし、ルミノシティをあげる計画を予定しており、これを HL-LHC と呼ぶ。ルミノシティは現在の LHC の約 7 倍、積分ルミノシティは約 10 倍となる予定である。

HL-LHC において、検出器には高い放射線耐性や位置分解能の向上など現在のものよりも高い水準が要求される。そのため、ATLAS 実験では最内層に設置している内部飛跡検出器の総入れ替えを予定しており、新しく製造する検出器を Inner Tracker (ITk) と呼ぶ。ITk では全ての領域でシリコン検出器が搭載され、また現在の内部飛跡検出器よりも広い立体角をカバーする設計となっている。

ITk の製造に向けてピクセルモジュール約 10,000 台を生産し、各モジュールに対して品質試験を行う予定となっている。品質試験は項目が多く、モジュール組み立て工程の中で何度も行うものである。モジュール情報及び品質試験の結果は固体性能の保持を目的としてチェコに設置されている中央データベースに保存する必要がある。

また各組み立て機関においてモジュール及び品質試験の情報管理に使用することを目的とした「ローカルデータベースシステム」が先行研究で開発されており、いくつかの機関でシステムの試験運用が行われている。このシステムは、モジュールの量産及び品質試験での運用に向けては必要不可欠であるいくつかの開発課題が残されていた。特に品質試験に特化した機能開発や中央データベースとローカルデータベース間の同期機能開発は、量産時には必要となる機能であるが実装されていなかった。

本研究では、モジュール組み立てとその品質試験に向けたデータベースシステム構築を行なった。ローカルデータベースにおける品質試験管理機能の 1 つとして結果検索機能やデータベース間の同期機能を開発し、システムの拡張を行なった。また品質試験におけるデータベース操作の流れを確立し、データベースの機能が一連の流れの中で使用可能であることを確認した。量産時に想定されるデータを用いて開発機能の処理時間測定を行い、その機能性を評価した。

目次

概要	i
第 1 章 序論	1
1.1 素粒子標準模型とその問題点 [1]	1
1.2 LHC について	4
1.3 ATLAS 実験	4
1.4 HL-LHC 実験アップグレード計画	8
第 2 章 ピクセルモジュール	14
2.1 ピクセルモジュールの構造	14
2.2 ピクセルモジュールの構成部品	14
2.3 新型モジュールの種類	18
第 3 章 モジュール組み立てと品質試験	19
3.1 組み立て工程	19
3.2 品質試験	19
3.3 検出器量産におけるデータ管理	30
第 4 章 モジュール情報及び品質試験結果管理システム	31
4.1 中央データベース	31
4.2 ローカルデータベース	32
4.3 本研究における開発項目の詳細	39
第 5 章 品質試験のデモンストレーション	53
5.1 用いたソフトウェア	53
5.2 用いたハードウェア	55
5.3 デモンストレーションの流れ	58
5.4 機能確認	59
第 6 章 ローカルデータベースにおける検索機能とその処理時間	68
6.1 実装方法	68
6.2 処理時間測定	75
6.3 改善方法と処理時間測定	78

第 7 章	中央データベースとローカルデータベースの同期	83
7.1	サーバーの設置場所による処理時間の違い	83
7.2	モジュール情報のダウンロード	87
7.3	読み出し試験結果のアップロード	96
第 8 章	まとめ	103
8.1	本論文のまとめ	103
8.2	現状と今後の課題	104
付録 A	シリコン検出器の原理	106
A.1	半導体 [12]	106
A.2	検出原理	108
付録 B	RD53A の回路図とフレキシブル基板	109
B.1	アナログ回路	109
B.2	試験用電荷入射のイメージ	109
B.3	RD53A のデータフォーマット	109
B.4	フレキシブル基板	112
付録 C	ローカルデータベースのチュートリアルと普及状況	113
付録 D	モジュール生産状況の解析	115
付録 E	ファイル送信時におけるデータ容量と処理時間の関係について	116
参考文献		119
謝辞		122

1 第1章

2 序論

3 欧州原子力研究機構 (**CERN**) に設置されている大型ハドロン衝突型加速器 (**LHC**) では、現在、素
 4 粒子物理学の基礎となっている標準模型の精密測定や標準模型を超える物理現象の探索が行われている。
 5 ATLAS 実験は LHC 上にある 4 つの衝突点の 1 つで行われている実験であり、ATLAS 検出器を用いて
 6 崩壊粒子の測定が行われている。LHC では加速器のアップグレード (**HL-LHC**) を予定しており、これ
 7 に向けて ATLAS 検出器のアップグレードを行う。この章では LHC-ATLAS 実験とそのアップグレード
 8 計画について説明する。

9 1.1 素粒子標準模型とその問題点 [1]

10 1.1.1 素粒子標準模型

11 現在素粒子物理学で基礎となっており、多くの実験事実を説明している理論を「標準模型」と呼ぶ。標準
 12 模型はスピン 1/2 のフェルミオンである 6 種類の「クォーク」と 6 種類の「レプトン」、スピン 1 の 4
 13 種類の「ゲージボソン」及びスピン 0 の「ヒッグスボソン」から成る。表 1.1、1.2 に一覧を示す。

14 自然界には電磁相互作用、強い相互作用、弱い相互作用、重力相互作用の 4 種類の相互作用がある。こ
 15 の中で重力相互作用は標準理論では扱っていない。2012 年 7 月に CERN でヒッグスボソンが発見され、
 16 標準模型における全ての素粒子が実験的に確認された。

表 1.1 標準模型のフェルミオン。

世代	クォーク	電荷 [e]	質量 [MeV/c ²]	レプトン	電荷 [e]	質量 [MeV/c ²]
1	u	2/3	~2.3	e^-	-1	0.511
	d	-1/3	~4.8	ν_e	0	$< 2.2 \times 10^{-6}$
2	c	2/3	~1280	μ^-	-1	105.7
	s	-1/3	~95	ν_μ	0	< 0.19
3	t	2/3	$\sim 1.73 \times 10^5$	τ^-	-1	1777
	b	-1/3	~4200	ν_τ	0	< 18

表 1.2 標準模型のボソン。

記号	名称	相互作用	スピン	電荷 [e]	質量 [GeV/c^2]
γ	光子	電磁	1	0	0
W^\pm	荷電弱ボソン	弱	1	± 1	80.4
Z	中性弱ボソン	弱	1	0	91.2
g	グルーオン	強	1	0	0
H	ヒッグスボソン	-	0	0	125

1.1.2 ヒッグス機構

標準模型がゲージ不変性を保ちながら素粒子が質量を持つために、標準模型のラグランジアンにヒッグスセクターを導入する。ここで導入するヒッグスセクターは複素スカラー場 ϕ を用いて以下とする。

$$\mathcal{L}_{\text{Higgs}} = (\partial_\mu \phi)^\dagger (\partial^\mu \phi) - V(\phi) \quad (1.1)$$

$$(1.2)$$

ここでヒッグスポテンシャル $V(\phi)$ は、ゲージ対称性とくりこみ可能性の要請の下で、以下のようにする。

$$V(\phi) = \mu^2 \phi^\dagger \phi + \lambda (\phi^\dagger \phi)^2 \quad (1.3)$$

この導入により、素粒子が質量を持つ理由を記述できる。ここで導入したヒッグスポテンシャル $V(\phi)$ の特徴として、素粒子の質量とヒッグス粒子との結合定数が比例する。ヒッグス粒子と各粒子との結合定数は多くの実験で測定されており、線形性が見えている(図 1.1)。このヒッグスセクターの導入及び現在の標準模型は、実験事実と大きく矛盾していないことが分かる。

一方、ここで述べた標準模型におけるヒッグスセクターの導入は暫定的なものであり、任意性を持つ。例えば、現在は複素スカラー場 1 つを導入しているが、要請を満たせば複数の場を導入することも可能である。標準模型を超える新理論では、しばしばこのヒッグスセクターを変形し、標準模型の拡張や諸問題の説明を行なっている。

1.1.3 問題点と新理論

標準模型は、これまでの実験事実では説明できない以下の問題点を抱えている。

- 重力理論との統一.
- なぜ素粒子は 3 世代なのか.
- 階層性問題.
- 強い CP 問題.
- 暗黒物質, エネルギー.
- ニュートリノの質量.

現在上にあげた標準模型の諸問題を解くために、超対称性 [3] などの新しい理論が考案されている。

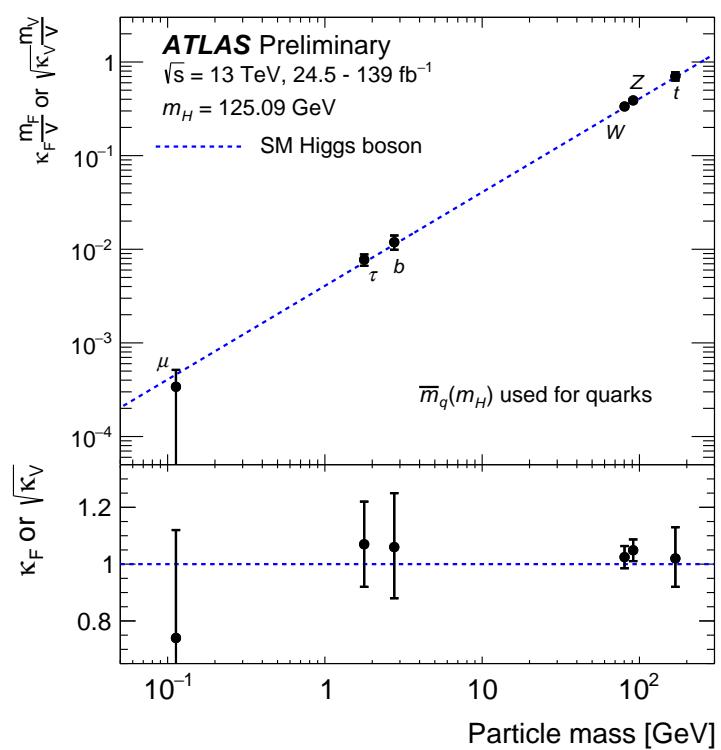


図 1.1 標準模型の素粒子の質量とヒッグス粒子との結合定数の関係 [2]。横軸は粒子の質量、縦軸は各粒子のヒッグス粒子との結合定数を示し、結合の強さを表す。各粒子の測定点が一直線状にのっており、線形性を示していることが分かる。

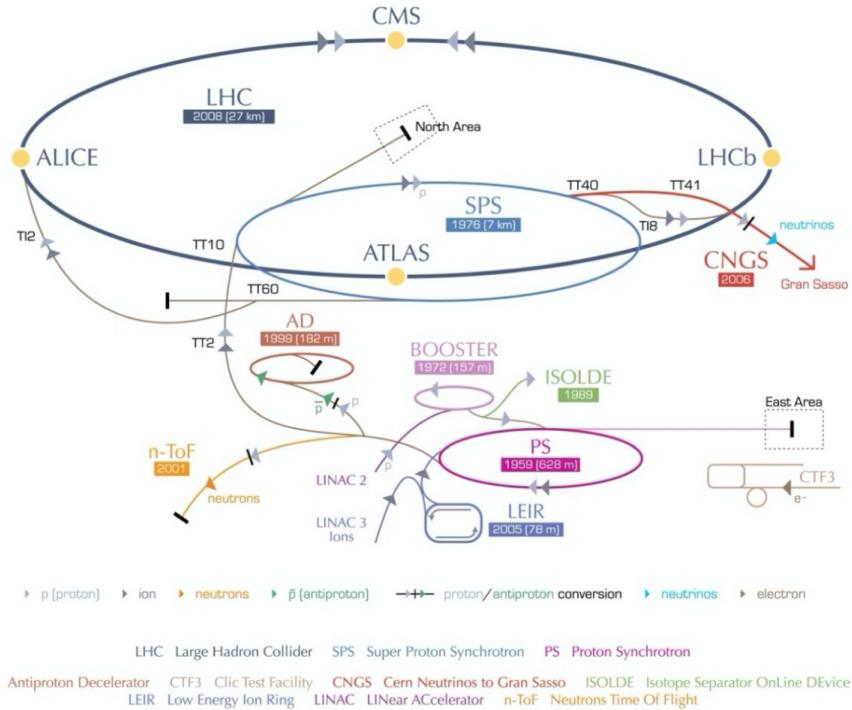


図 1.2 加速器の全体像 [4]。図は CERN に設置されている加速器の全体像を示す。陽子はいくつかの前段加速器で段階的に加速され LHC に入射する。LHC 上には 4 つの衝突点が存在し、それぞれ ALICE、LHCb、CMS、ATLAS 実験が行われている。

39 1.2 LHC について

40 LHC は CERN の地下およそ 100 m に設置されている周長 26.7 km の大型ハドロン衝突型加速器である。
 41 バンチと呼ばれる陽子のかたまりを 7 TeV まで加速し、衝突させる。世界最大エネルギーの加速器である。

43 陽子ビームの加速は 4 つの前段加速器を用いて行う。始めに水素ガス中の水素原子から電子を分離す
 44 ることで陽子を生成する。その後最初の線形加速器 (Linear Accelerator: LINAC)、陽子シンクロトロ
 45 ンブースター (Proton Synchrotron Booster: PSB)、陽子シンクロトロン (Proton Synchrotron: PS)、
 46 スーパー陽子シンクロトロン (Super Proton Synchrotron) で加速されたのち LHC に入射する。CERN
 47 にある加速器の概要を図 1.2 に示す。LHC には 4 つの衝突点があり、それぞれ ALICE(A Large Ion
 48 Collider Experiment)、LHCb、CMS(Compact Muon Solenoid)、ATLAS(A Troidal LHC Apparatus)
 49 実験が行われている。それぞれの衝突点には崩壊粒子の飛跡やエネルギーを測定するための検出器が設置
 50 されており、取得したデータを元に多様な物理解析が行われている。

51 1.3 ATLAS 実験

52 初めに ATLAS 実験に用いる座標系と用語について説明する。まず衝突点を原点として定義しており、
 53 ビーム軸を z 軸、これに対して垂直な平面を $x - y$ 平面とする。 x 軸方向は原点からみて LHC リングの
 54 中心に向かう方向であり、 y 軸は上に向かう方向である。方位角 ϕ は z 軸周りの角度であり、極角 θ は z
 55 軸とのなす角である。ATLAS 実験では、極角 θ は以下のように擬ラピディティ η で表される。また極角

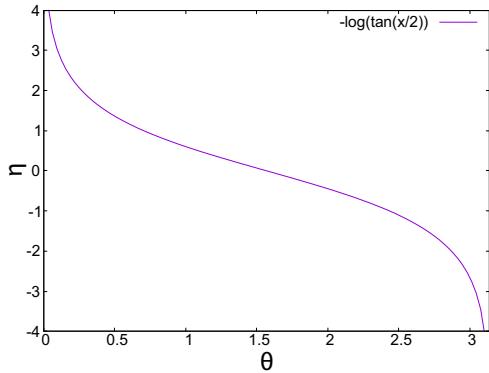


図 1.3 極角 θ と擬ラピディティ η の関係図。横軸は式 1.4 における極角 $\theta (0 \leq \theta \leq \pi)$ 、縦軸は擬ラピディティ η を表している。図のように、 η への変換により定義域が実数全体に広がる。

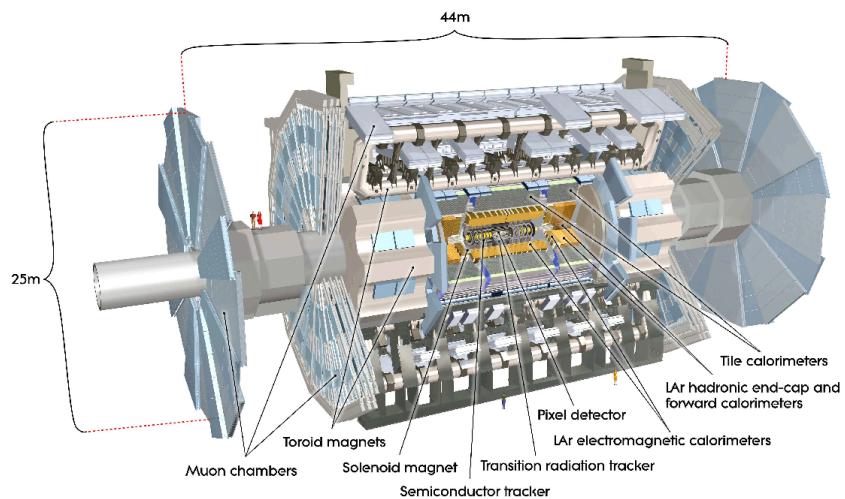


図 1.4 ATLAS 検出器の全体像 [5]。内側から内部飛跡検出器、ソレノイド磁石、カロリメータ、トロイド磁石、ミューオン検出器が設置されている。

⁵⁶ θ と擬ラピディティ η の関係図を図 1.3 に示す。

$$\eta = -\ln \left(\tan \left(\frac{\theta}{2} \right) \right) \quad (1.4)$$

⁵⁷ 1.3.1 ATLAS 検出器

⁵⁸ ATLAS 検出器は複数の検出器から構成され、陽子の衝突によって生成された粒子の運動量、エネルギーを測定することができる。⁵⁹ 最内層に内部飛跡検出器が設置されていて、次に超電導ソレノイド磁石、⁶⁰ カロリメータ、トロイド磁石、ミューオン検出器の順に設置されている。⁶¹ ビームパイプ以外をほとんど検出器で覆うような設計となっている。ATLAS 検出器の全体図を図 1.4 に示す。

⁶² 1.3.2 内部飛跡検出器

⁶³ ATLAS 検出器の最内層に位置する検出器である。内部飛跡検出器は粒子の飛跡測定をするための検出器であり、飛跡の曲率から粒子の運動量を計算できる。⁶⁴ 検出器の外側には超伝導ソレノイド磁石が設置されており、2 T の磁場が z 方向にかけられる。⁶⁵ これにより荷電粒子はローレンツ力を受け、軌跡が曲がる。

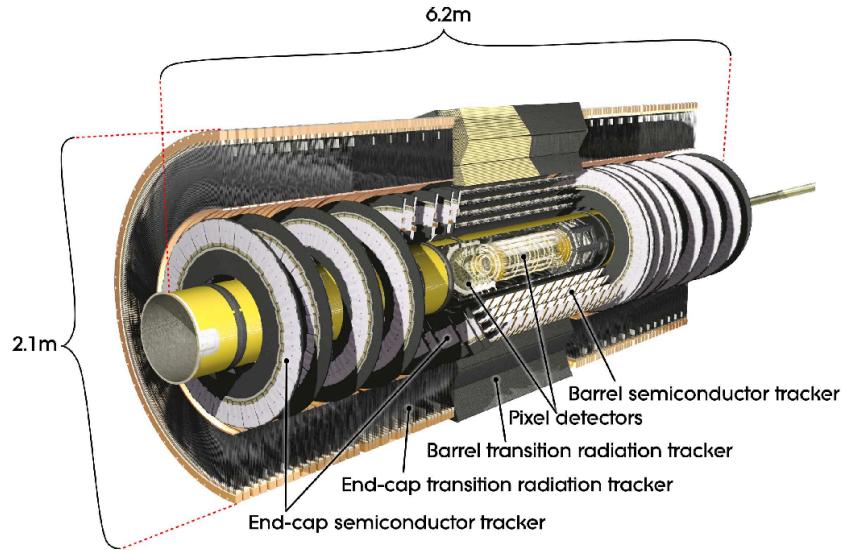


図 1.5 内部飛跡検出器の全体像 [5]。内側からピクセル検出器、ストリップ検出器、遷移放射検出器が設置されている。また円筒状のバレル部とディスク上のエンドキャップ部を構造として持つ。

66 内部飛跡検出器は 3 つの検出器で構成され、内側からピクセル検出器、ストリップ検出器、遷移放射検
67 出器の順に設置されている。ピクセル、ストリップ検出器は階層構造になっており、粒子は複数の検出器
68 を通過する。それぞれの検出器で取得した通過位置をつなぎ合わせることで粒子の軌跡を計算するこ
69 ことができる。

70 内部飛跡検出器の全体図を図 1.5 に示す。

71 ピクセル検出器

72 内部飛跡検出器の最内層に位置する検出器である。ピクセル検出器はバレル部が 4 層、エンドキャップ
73 部が 6 層で構成される。バレル部の最内層は IBL(Insertable B-Layer) と呼ばれ、順に B-Layer、Layer-1、
74 Layer-2 となっている。

75 ピクセル検出器の全体図を図 1.6 に示す。

76 ピクセル検出器の各層は、モジュールと呼ばれる最小単位の検出器をいくつも搭載している。ピクセル
77 モジュールを図 1.7 に示す。このピクセルモジュールの詳細については 2 章で述べる。

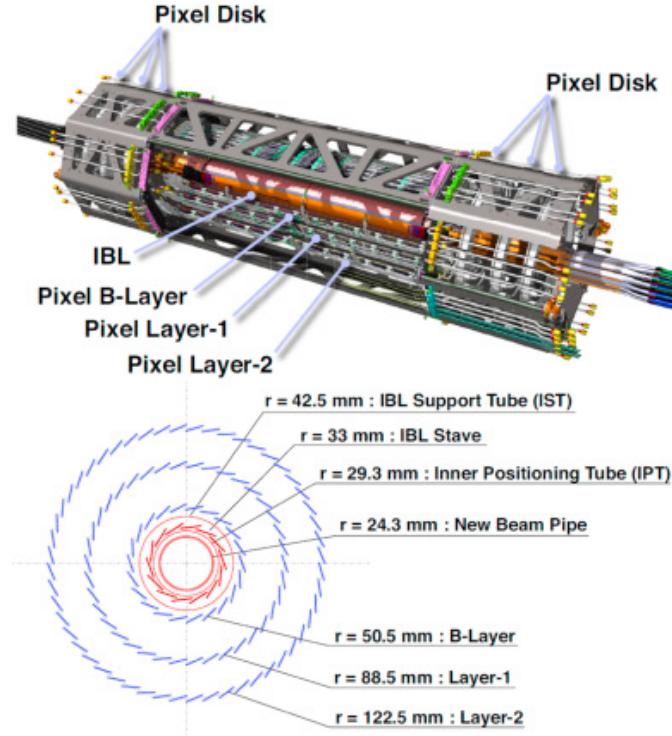


図 1.6 ピクセル検出器の全体像 [6]。上図はピクセル検出器の全体を模式的に表したものであり、下図はビーム軸方向から見たピクセル検出器の断面図である。バレル部の 4 層は内側から IBL、B-Layer、Layer-1、Layer-2 と呼ぶ。

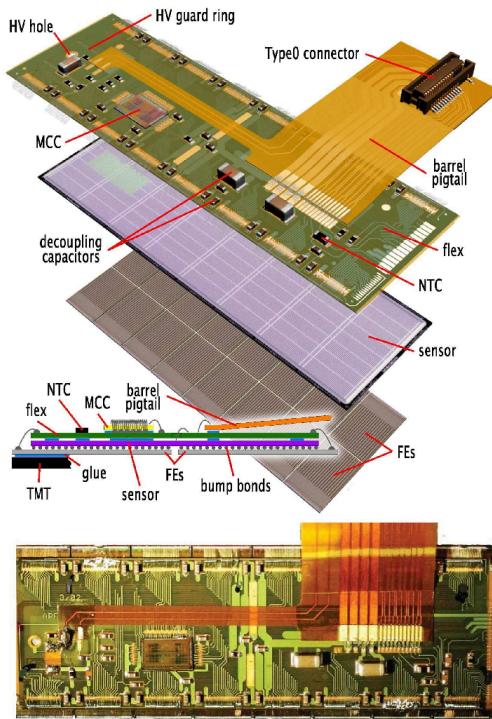


図 1.7 ピクセルモジュールの全体像 [5]。ピクセル検出器は、最小単位としてモジュールと呼ばれる構造を持ち、図にモジュールの全体像を示している。モジュールは、荷電粒子が通過し、信号を生成するセンサー部、AD 変換を行う FE チップ部、データ転送を行うフレキシブル基板から構成される。

表 1.3 現行 LHC と HL-LHC の比較 [7]。瞬間ルミノシティは 7 倍、積分ルミノシティは 10 倍になることが見積もられており、取得統計数の増加が期待できる。

	LHC	HL-LHC
重心系エネルギー	14	14
瞬間ルミノシティ [$\text{cm}^{-2}\text{s}^{-1}$]	1×10^{34}	7×10^{34}
積分ルミノシティ [fb^{-1}]	300	3,000
1 陽子衝突あたりのイベント数	27	140



図 1.8 HL-LHC 運転計画 [8]。2025 年の初めより HL-LHC の導入、2027 年の途中より本運転が始まる。

1.4 HL-LHC 実験アップグレード計画

LHC では加速器のアップグレードを予定しており、これを HL-LHC アップグレード計画と呼ぶ。詳細を以下に示す。

1.4.1 概要

HL-LHC ではルミノシティ呼ばれる陽子バンチの密度を上げることで、衝突頻度を大きくし、取得統計数を増やす目的がある。LHC と HL-LHC の比較を表 1.3 に示す。

LHC の運転計画を表 1.8 に示す。2025 年の初めより HL-LHC の導入が始まり、2027 年の途中から HL-LHC 運転開始の予定となっている。

1.4.2 素粒子物理に対するモチベーション

HL-LHC の素粒子物理に対するモチベーションに 1 つとして、ヒッグス粒子イベントの精密測定があげられる。ヒッグス粒子イベントの精密測定は、標準模型の検証だけでなく、新理論への制限や方向性の決定に対して重要な意味を持つ。

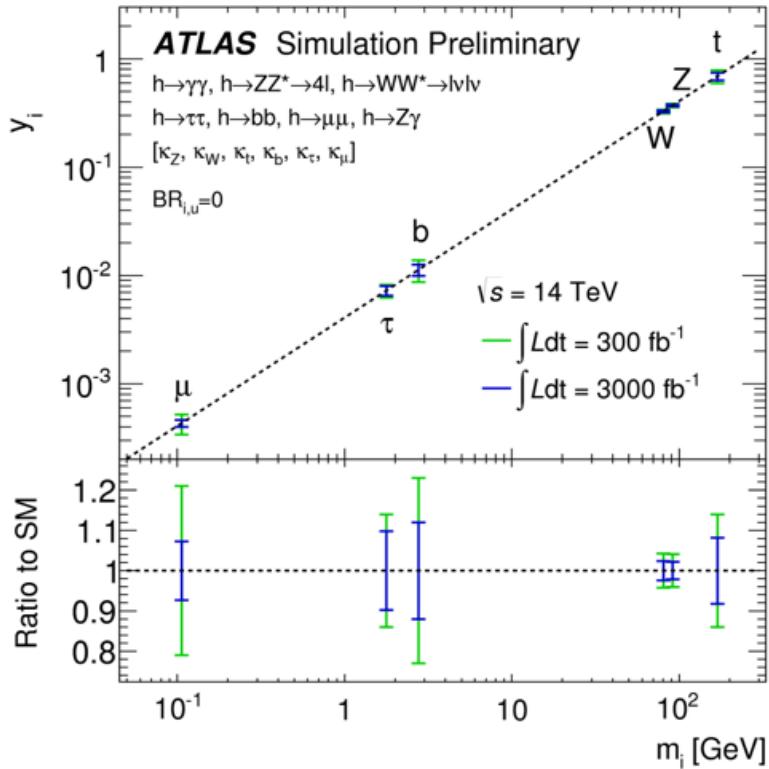


図 1.9 LHC、HL-LHC における粒子の質量とヒッグス粒子の結合定数の関係と統計誤差の見積もり [10]。図において青線、緑線がそれぞれ LHC、HL-LHC の測定における統計誤差の見積もり値を示している。統計誤差は向上する見込みであり、標準模型の精密検証をすることができる。

90 LHC、HL-LHC における粒子の質量とヒッグス粒子の結合定数の関係と統計誤差の見積もりをまとめ
91 たものを表 1.9 に示す。統計誤差が向上する見込みであり、より精密な標準模型の検証を行うことができる。
92

93 1.4.3 内部飛跡検出器のアップグレード

94 HL-LHC のアップグレードによりルミノシティが増加するそれに伴い、検出器には以下のような性能
95 が要求される。

- 96 • 放射線耐性の向上。
97 統計量に伴い、放射線量も増加するため高放射線耐性が要求される。
- 98 • 高速読み出し。
99 ルミノシティが増加するため、現行よりも高速な読み出しが要求される。
- 100 • 検出器の細密化。
101 1 バンチ衝突あたりのイベント数が表 1.3 より約 5 倍になる。これらの衝突点を区別するためには
102 より細密な検出器を使用する必要がある。図 1.10 にイメージを示す。

103 HL-LHC に向けて内部飛跡検出器はアップグレードを予定しており、検出器の総入れ替えを行う。アッ
104 プグレード後の検出器を **Inner Tracker(ITk)** と呼ぶ。模式図を図 1.11 に示す。

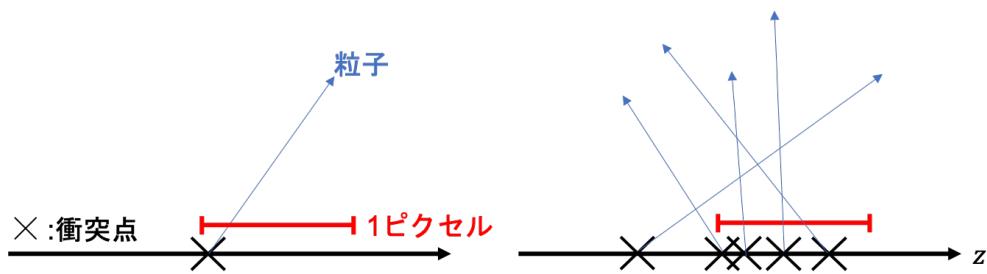


図 1.10 1 バンチ衝突あたりのイベント数増加のイメージ。図において黒線がビーム軸と衝突点、青線が生成粒子、赤線が 1 ピクセルを表す。左図が LHC、右図が HL-LHC の様子を表している。HL-LHC では 1 バンチ衝突あたりのイベント数が約 5 倍に増えるため、図のように衝突点の間隔が小さくなる。これらを区別するためには今までよりも細密な検出器を設置する必要がある。

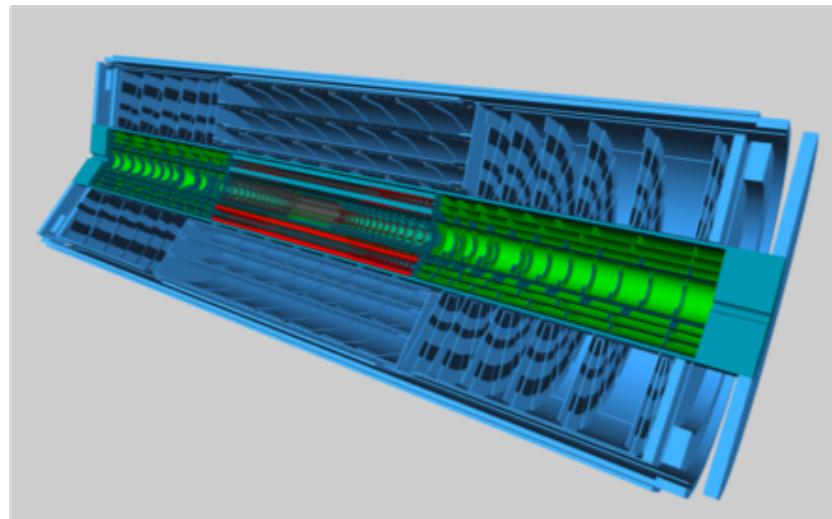


図 1.11 ITk の全体像 [9]。図は ITk 全体の模式図を示している。ITk はピクセル検出器（緑の領域）とストリップ検出器（青の領域）から構成される。

105 ITk の構成と現行ピクセル検出器との比較

106 図 1.12 に ITk のビーム軸方向の断面図を示す。ITk はピクセル検出器とストリップ検出器で構成され
107 る。ピクセル検出器はバレル、傾斜バレル、エンドキャップ部で構成され、バレル部は 5 層となっている。
108 ピクセル検出器の配置に関して、現行と ITk の比較を表 1.4 に示す。またモジュール数の比較を表 1.5
109 に示す。ITk では現行に比べ、 η が大きい領域まで検出器を設置するため、その分使用するモジュール数
110 も増える。

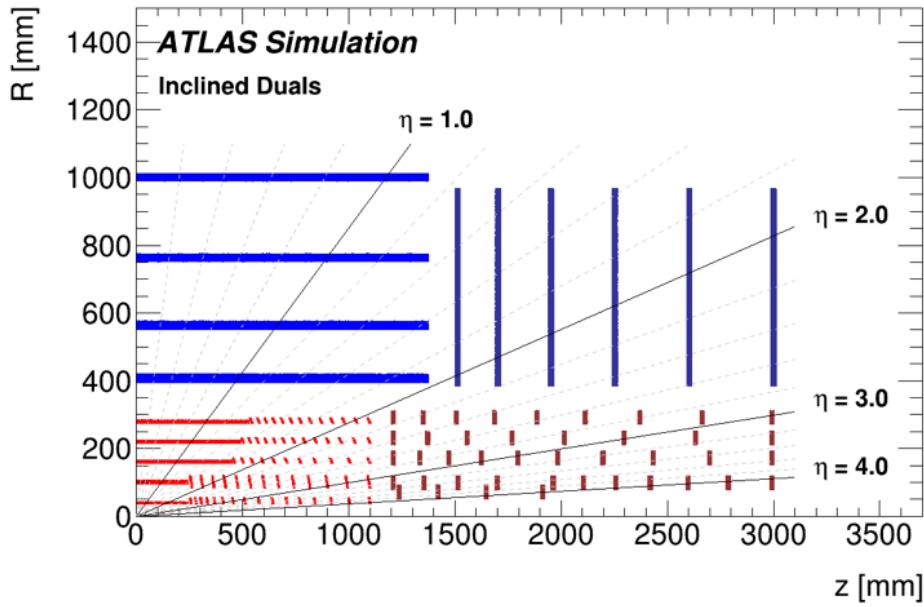


図 1.12 ITk の断面図 [9]。ITk はピクセル検出器とストリップ検出器から構成され、バレル部はそれぞれ 5、4 層の構造を持つ。ピクセル検出器はバレル部、傾斜バレル部、エンドキャップ部に分かれ、 $|\eta| < 4.0$ までの範囲をカバーし、粒子の飛跡測定をすることができる設計となっている。

表 1.4 ピクセル検出器設置領域の比較。表は現行と ITk におけるピクセル検出器設置領域の比較を示す。 η の範囲に関して、括弧内は度数法における対応する値を示す。ITk では遷移放射検出器を使用しないため、ピクセル検出器の半径方向のカバー領域が増え、層の数は 4 から 5 となる。また前方方向に生じる粒子測定を行うために、 η の領域も拡大している。

	現行	ITk
$r[\text{mm}]$	33~129	39~279
層の数 (バレル部)	4	5
$ \eta $	$< 2.5 (< 19^\circ)$	$< 4 (< 4.2^\circ)$

表 1.5 搭載するピクセルモジュール数の比較。表は現行と ITk のピクセル検出器において、各構造における搭載ピクセルモジュールの数を示している。ITk ではバレル部で現行の約 2 倍になっているのに加え、傾斜バレル部、エンドキャップ部に多くのモジュールを搭載する設計となっていることが分かる。

層	バレル部		傾斜バレル部		エンドキャップ部	
	現行	ITk	現行	ITk	現行	ITk
1	280	192	—	512	—	64
2	286	240	—	520	—	242
3	494	660	—	660	—	320
4	676	960	—	1040	288	352
5	—	1300	—	1300	—	468
合計	1736	3352	0	4032	288	1446

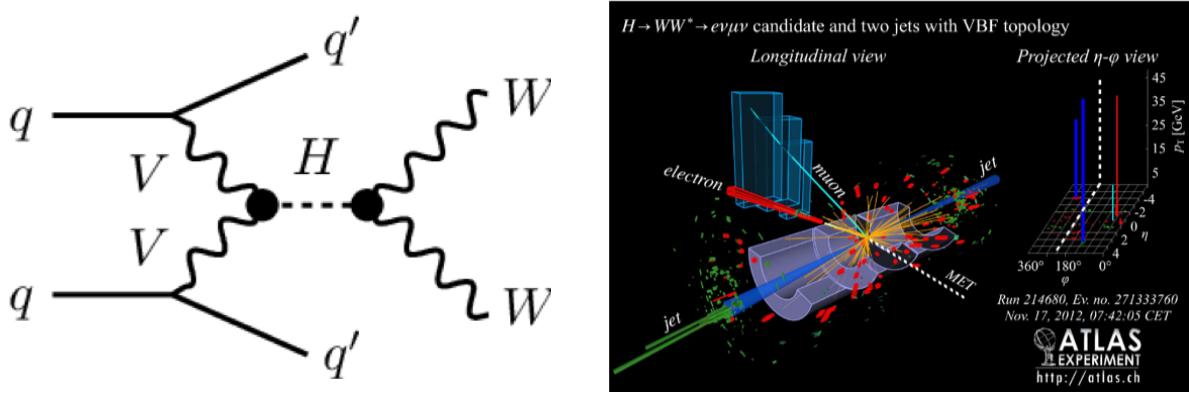


図 1.13 VBF イベントの図 [11]。図は VBF イベントの一つである $H \rightarrow WW^*$ 崩壊チャンネルを示している。左図はファインマン図、右図はそのイベントディスプレイを表す。このイベントにおいて、初めに相互作用したクオーク対は、それぞれ崩壊を繰り返しジェットと呼ばれる粒子群となる。このときこれらのジェットは前方方向に大きな運動量を持つ。ITk では $|\eta| < 4.0$ の範囲をカバーしており、このようなジェットを捉えることができるため、測定精度が向上する。

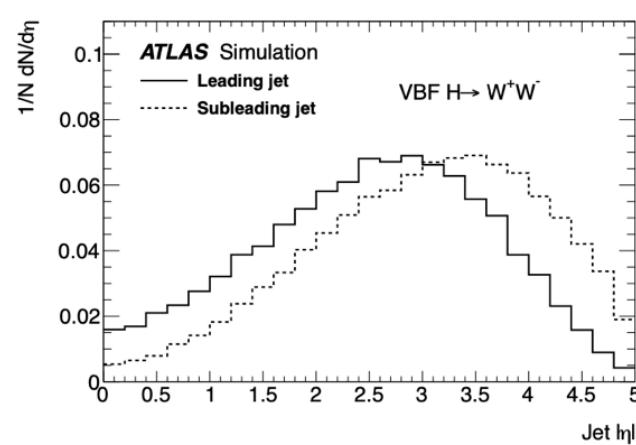


図 1.14 VBF イベントにおけるジェットの運動方向の η 分布 [9]。図はシミュレーションの結果を示している。横軸はジェットの運動方向の η 成分を示しており、縦軸はイベント数に比例する量である。“Leading jet”、“Subleading jet”は各イベントにおいて 1、2 番目に大きな運動量を持つジェットである。 $|\eta| = 3 - 3.5$ 付近にピークを持ち、多くのジェットは前方方向に大きい運動量を持つことが分かる。

1.4.4 物理測定に及ぼす影響

カバーする η の範囲が大きい特徴が生む利点として、前方方向に大きな運動量を持つ粒子を含む物理現象の測定精度向上があげられる。

この例として、ボゾン粒子結合によるヒッグス粒子生成過程 (Vector boson fusion higgs production, VBF) をあげる。VBF の 1 つのチャンネルに関するファインマン図及びイベントディスプレイを図 1.13 に示す。この衝突により生じる 2 つのクオークはジェットと呼ばれる粒子群となり、多くのジェットは図 1.14 に示すように前方方向に大きな運動量を持つ。

ITk ではこれらのジェットを正確に捉えることができ、VBF イベントの測定精度を向上することができる。この測定における系統誤差は表 1.6 のように見積もられる [9]。

表 1.6 VBF $H \rightarrow WW$ の測定における系統誤差の見積もり。表は統計数 3000 fb^{-1} において、 $|\eta| < 2.7$ 、 $|\eta| < 4.0$ の領域を使用した場合の VBF $H \rightarrow WW$ イベント測定における系統誤差の見積もりを示している。 $|\eta| < 4.0$ の領域を使用することで系統誤差が向上する見積もりであることが分かる。ここでは統計誤差は考慮にいれていない。

検出器の使用範囲	$ \eta < 2.7$	$ \eta < 4.0$
系統誤差	22%	12%

120 第2章

121 ピクセルモジュール

122 この章ではピクセルモジュールの構成と各構成部品について説明する。

123 2.1 ピクセルモジュールの構造

124 ピクセルモジュールはベアモジュールとフレキシブル基板より構成される。ベアモジュールは荷電粒子
125 の通過を検知し、信号を発生するシリコンセンサーと、AD 変換を行う FE チップで構成される。ベアモ
126 ジュールが持つ FE チップの数はモジュールの種類によって異なる。モジュールの構成を図 2.1 に示す。

127 2.2 ピクセルモジュールの構成部品

128 2.2.1 シリコンセンサー

129 ピクセルモジュールに搭載するセンサーはシリコン半導体を用いている。センサー内部構造として pn
130 接合を持ち、逆バイアス電圧をかけ空乏層を広げた状態で使用する [12]。この空乏層領域に荷電粒子が通
131 過すると、Bethe-Bloch の式 [13] に従い粒子はエネルギーを損失する。このエネルギー損失量に従い、電子
132 ・ホール対が生成、これを収集することで荷電粒子の通過情報をアナログ信号として取得することができる。
133



図 2.1 ピクセルモジュールの構成。図は Quad モジュールの構成を模式的に表したものである。モ
ジュールは「ベアモジュール」と呼ばれる心臓部と「フレキシブル基板」を貼り付けることで作られ
る。ベアモジュールはセンサーと FE チップから成る。Quad モジュールの場合、1 枚のセンサーに対
し FE チップは 4 枚である。センサーと FE チップは「パンプ」と呼ばれる構造で電気的に接続して
おり、1 ピクセルに対して 1 つのパンプ接合がなされている。FE チップとフレキシブル基板の接続に
は、ワイヤーが多数 ($O(100)$) 配線される。



図 2.2 新型ピクセルモジュールに搭載するシリコンセンサーの断面図。図は新型ピクセルモジュールに搭載するシリコンセンサー断面の模式図を示している。p 型半導体に n^+ 型電極を埋め込んだ $n^+ - in - p$ 型と呼ばれる構造を持つ。 p^+ 側にフレキシブル基板、 n^+ 側に FE チップが付く。逆バイアス電圧を印加すると n^+ 電極側から空乏層が広がる。

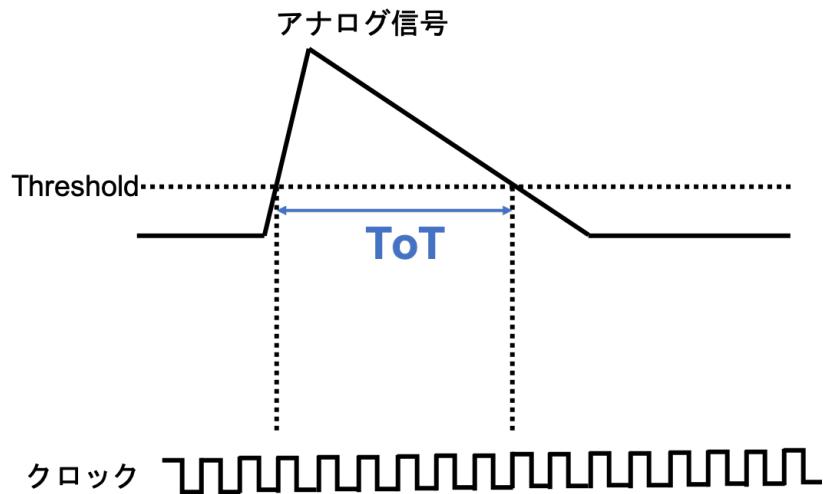


図 2.3 ToT の概念図。FE チップで生成されるデジタル信号は ToT と呼ばれる。図はその概念を模式的に表したものであり、FE チップで行われている AD 変換である。ToT はアナログ信号が Threshold 値を超えた時間幅に相当する。実際には ToT 間のクロック数 [b.c.] として取得される。ここで 1 b.c.(25 ns) は LHC における陽子の衝突間隔に相当する。

¹³⁴ 新型ピクセルモジュールに搭載するセンサーは、 $n^+ - in - p$ 型である。模式図を図 2.2 に示す。 n^+ 電
¹³⁵ 極で電子を収集し、信号を取得する。現行のセンサーに比べ、型変換を起こす可能性がなくなるため安定
¹³⁶ した運転が見込まれる [9]。

¹³⁷ 2.2.2 読み出し FE チップ

¹³⁸ 読み出し FE チップはシリコン半導体を用いて作られた集積回路である。読み出し FE チップの主な役
¹³⁹ 割は、シリコンセンサーで発生し受け取ったアナログ信号を整形、增幅したのち AD 変換し、後段に転送
¹⁴⁰ することである。AD 変換について、アナログ信号が Threshold を超えた時間幅を測定し、デジタル信号
¹⁴¹ に変換する。この信号の値を **Time over Threshold (ToT)** と呼ぶ。

¹⁴² ToT の概念図を図 2.3 に示す。

¹⁴³ RD53A

¹⁴⁴ RD53A[12] は、新型ピクセルモジュールの研究、開発のために作られたプロトタイプの読み出し FE
¹⁴⁵ チップである。RD53A の性能を表 2.1 に示す。チップサイズ、ピクセル数は、ITk に搭載予定のモジュー

表 2.1 RD53A のスペック。

チップサイズ [mm ²]	20.0 × 11.6
ピクセルサイズ [μm ²]	50.0 × 50.0
ピクセル数 [行 × 列]	400 × 192
トリガーレート [kHz]	1000
データレート [Mbps]	1280 × 4
ToT の範囲	0～15(4 ビット)
放射線耐性	500Mrad(トータルドーズ効果 [14])

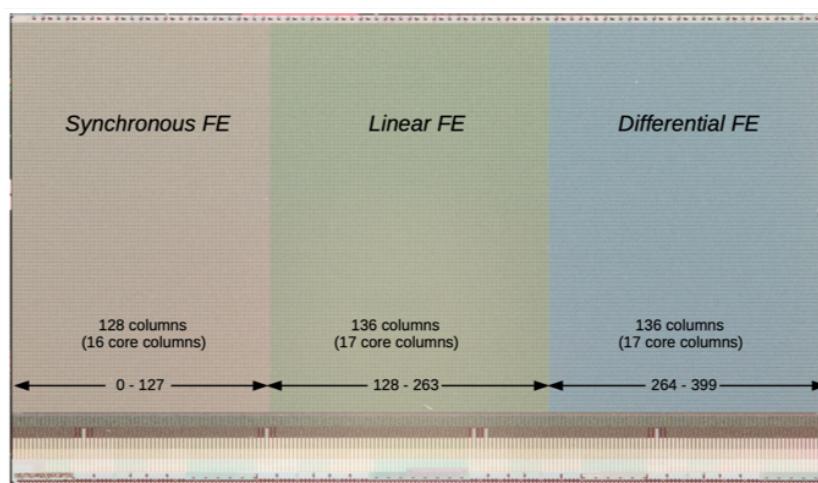


図 2.4 RD53A[12]。ピクセル数は 400 列 × 192 行となっている。図のように RD53A は 3 つの領域が設けられており、それぞれ Synchronous FE、Linear FE、Differential FE と呼ぶ。各領域のピクセルが持つアナログ回路、AD 変換回路が異なる。

146 ルが持つ FE チップの半分となっている。

147 FE チップ上の各ピクセルはアナログ回路部とデジタル回路部を持つ。RD53A では図 2.4 に示すよう
148 に、3 つの領域があり、それぞれの領域でピクセルのアナログ回路、AD 変換回路が異なる。左から順に
149 Synchronous FE、Linear FE、Differential FE と呼ぶ。研究、開発用に 3 つの領域が設けられている
150 が、性能比較の結果 ITk に搭載するモジュールには Differential FE を用いることが決定している。な
151 お、デジタル回路部は全てのピクセルにおいて共通である。それぞれの回路図は付録 B に示す。

152 2.2.3 フレキシブル基板

153 フレキシブル基板は、基板上に電子部品が搭載されたものである。FE チップからのデジタル信号を後
154 段の回路へ転送する他、FE チップ、センサーへの電圧印加制御の役割も担う。

155 2.2.4 信号伝達

156 モジュールの信号伝達の様子を模式的に表したものを図 2.5 に示す。センサーで生成した信号は FE
157 チップ、フレキシブル基板の順に送られ、PC でデータ取得できる。

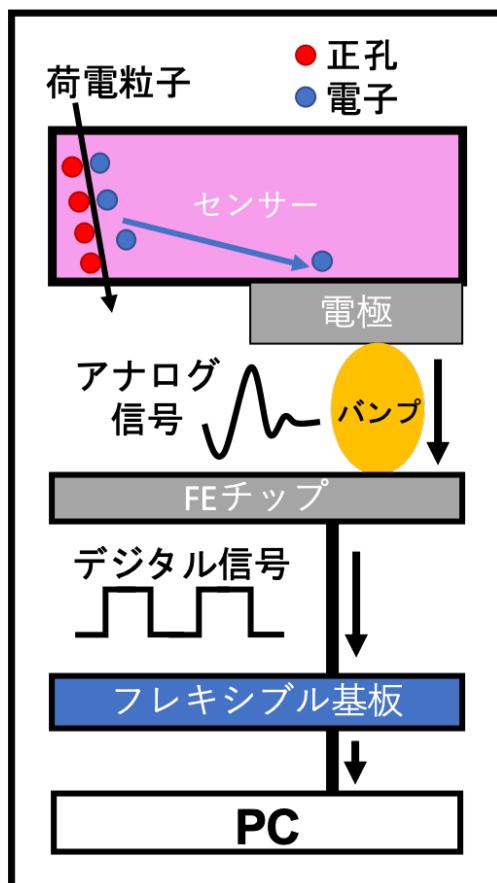


図 2.5 ピクセルモジュールにおける信号伝達の様子。図はピクセルモジュールにおける信号伝達の様子を模式的に表したものである。初めに、荷電粒子がセンサー部を通過し、エネルギー損失に応じた電子・ホール対を生成する。電子は電極により収集され、バンプを通してアナログ信号として FE チップに送られる。FE チップでは信号を整形、増幅後、デジタル信号に変換する。その後ワイヤーを通してフレキシブル基板に転送、そして PC に転送されデータを取得する。

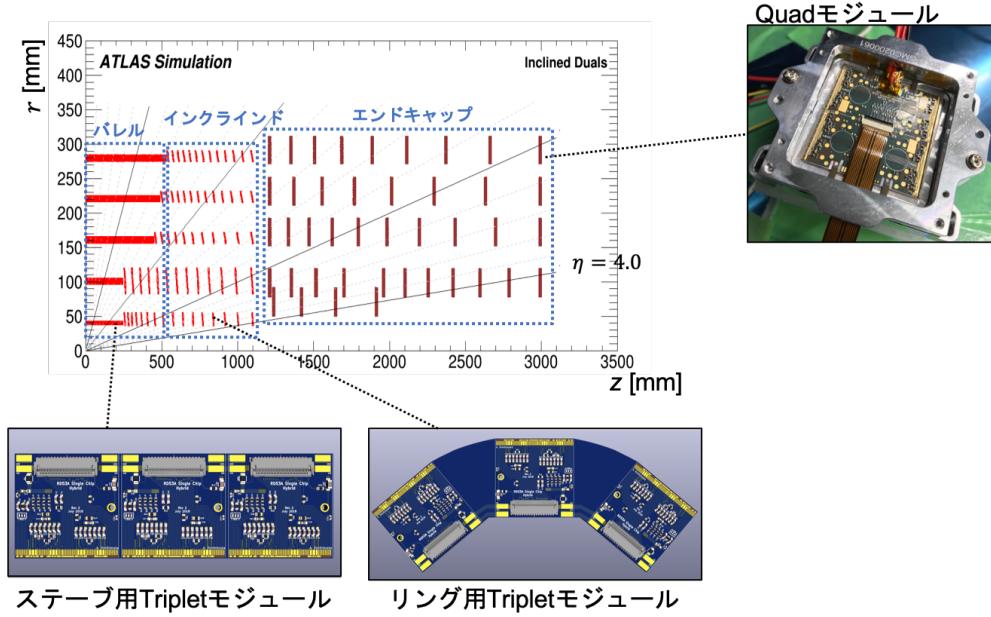


図 2.6 搭載するモジュールのプロトタイプと ITk における配置。2 種類の Triplet モジュールと Quad モジュールを搭載する予定となっている。Triplet モジュールは最内層のステープ構造、リング構造の領域に使われる。それ以外の領域には Quad モジュールが使われる。

158 2.3 新型モジュールの種類

159 現在、ITk に搭載するモジュールとして以下の 3 つのモジュールを予定している。

- 160 • ステープ用 Triplet モジュール
- 161 • リング用 Triplet モジュール
- 162 • Quad モジュール

163 Triplet、Quad モジュールはそれぞれ FE チップを 3、4 枚搭載するモジュールである。それぞれのモ
164 ジュールの図及び検出器上における位置の一例を図 2.6 に示す。

165 第3章

166 モジュール組み立てと品質試験

167 この章ではモジュールの組み立て工程と品質試験について説明する。

168 3.1 組み立て工程

169 モジュール組み立て機関は、初めにベアモジュールとフレキシブル基板を受け取る。組み立て工程とし
170 て以下が設定されている。

171 1. ベアモジュール・フレキシブル基板貼り付け.

- 172 ● 受け取ったベアモジュールとフレキシブル基板を接着剤を用いて貼り付ける。

173 2. ワイヤー配線.

- 174 ● ワイヤー配線を行い、FEチップとフレキシブル基板を電気的に接続する。

175 3. ワイヤー保護.

- 176 ● ワイヤーが損傷があり断線が起きると、そのワイヤーに接続されているピクセルの読み出しが
177 できなくなる。これを防ぐため、モジュールに屋根型の構造を取り付け、ワイヤーを物理的に
178 保護する。

179 4. パリレンコーティング.

- 180 ● モジュール読み出し部以外での電通や放電を防ぐため、パリレン高分子を用いてモジュールを
181 保護する。

182 5. 温度サイクル試験.

- 183 ● ITk運転時の環境温度は -45°C から 40°C まで変化しうる[15]。この温度変化に耐えうるか
184 を確認するため、温度サイクルを行う。

185 6. 低温耐久試験.

- 186 ● ITk運転時の典型的な環境温度は $-15\sim 0^{\circ}\text{C}$ 付近である。これに耐えうる性能を持つかを確
187 認するために、低温下にモジュールを長時間設置する耐久試験を行う。

188 流れと各組み立て工程のイメージを図3.1に示す。

189 3.2 品質試験

190 各組み立て工程に対して、いくつかの品質試験を行う。行う品質試験の代表的なものを以下に示す。

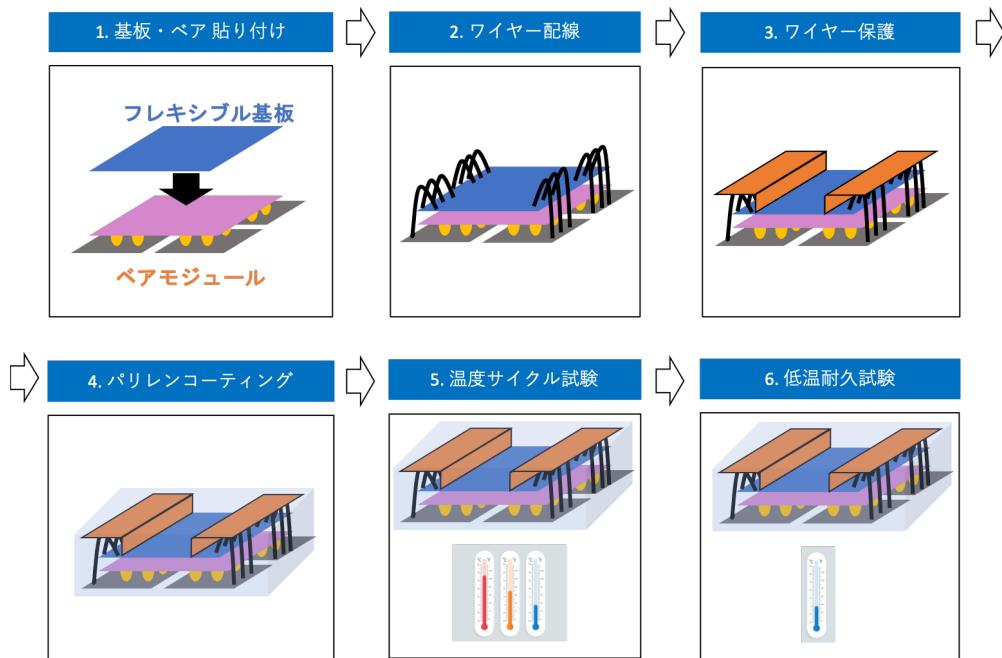


図 3.1 組み立て工程のイメージ図。モジュール組み立て機関はペアモジュール、フレキシブル基板を受け取り、それらの貼り付け、ワイヤー配線、ワイヤー保護、パリレンコーティング、温度サイクル試験、低温耐久試験の順に組み立てを行う。

191 3.2.1 外観検査

192 モジュールの外観写真を撮り、モジュールに以下のような欠陥がないかを確認する。また外観検査の様
193 子を図 3.2 に示す。

- 194 • 抵抗等取り付け部品の損傷.
- 195 • ワイヤーの接着位置確認.
- 196 • 基板上の回路やワイヤーの断線.
- 197 • 付着汚れ.

198 3.2.2 質量測定

199 モジュールの質量を測定する試験である。

200 3.2.3 平坦性測定

201 モジュール上の位置座標を何点か測定し、モジュールの平坦度、厚さ、歪み具合等を測定する。測定の
202 様子と解析の例を図 3.3 に示す。

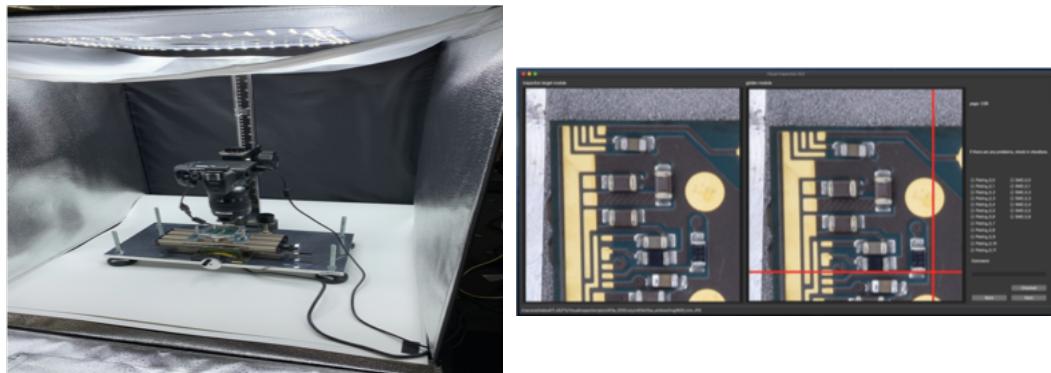


図 3.2 外観検査の様子。図は日本で実際に行われた外観検査における写真撮影の様子（左図）と撮影写真の確認画面（右図）である。左図のように、写真撮影は光量を制御するため暗箱の中で行っている。またモジュールに損傷や断線がないかを確認するために、右図のように撮影した写真と良好なモジュールの写真を見比べ、電気部品の損傷、回路の断線などの何らかの所見があった場合には記録をする。

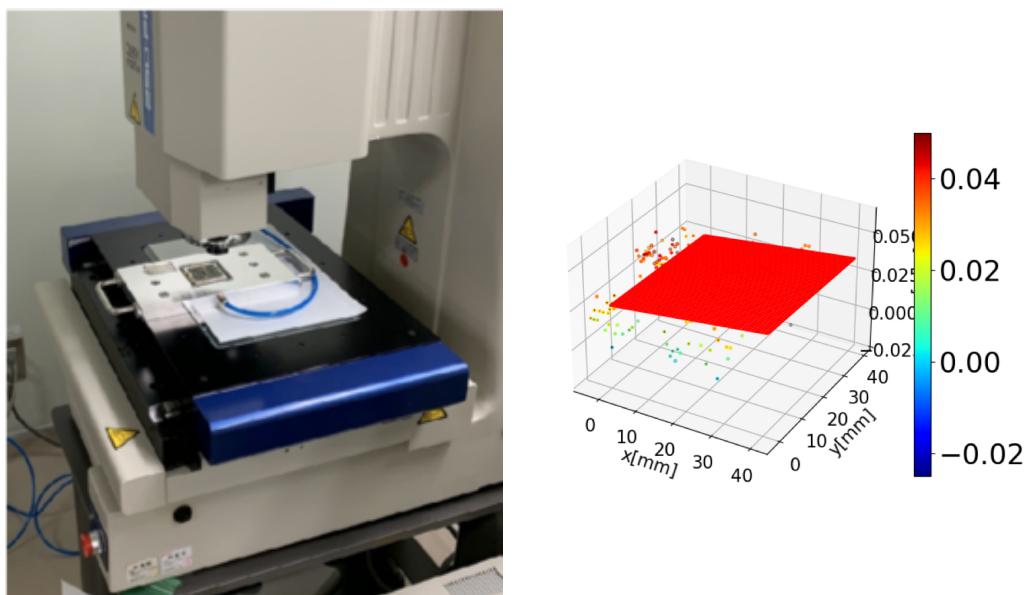


図 3.3 平坦性測定の様子。図は日本で実際に行われた平坦性測定における測定の様子（左図）と解析結果（右図）である。専用の装置を用いてモジュールの位置座標を何点か測定する。得られた測定点は右図のように図示し、平面でフィットティングを行う。フィット平面からのズレの分布や、モジュールの厚みを計算する。

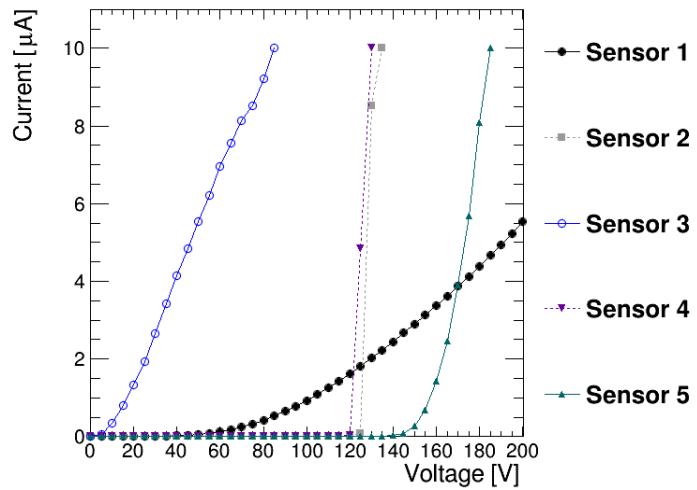


図 3.4 センサー電流-電圧特性結果の例 [16]。図は複数のセンサーに対する測定結果の例を表している。横軸にセンサーに対する印加電圧 [V]、縦軸に電流 [μA] を示す。逆バイアス電圧を印加しているため、電流がほとんど流れていらないが、ある地点から電流が急激に増加していることが分かる。この降伏電圧はセンサーにより個体差があるが、一般的に 100 – 200 V 付近となる。

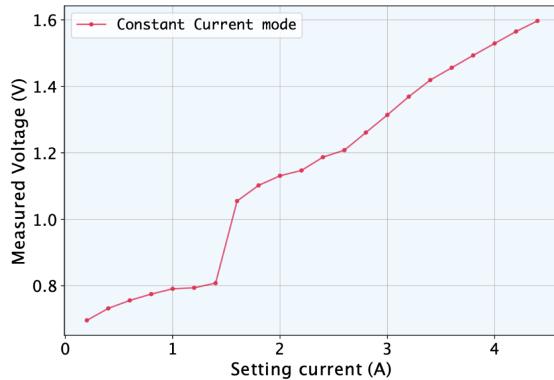


図 3.5 FE チップ電流-電圧特性試験結果の例 [17]。横軸に FE チップに対する設定電流、縦軸は測定した実際に FE チップにかかる電圧値を示している。1.5 V 付近に急激に電圧が増大しており、この段階で FE チップが機能するようになる。それ以降は線形になっているのが分かる。

203 3.2.4 センサー電流-電圧特性確認

204 モジュールのシリコンセンサーに逆バイアス電圧をかけ、電流-電圧特性をみる。印加電圧を段階的に
205 変化させて測定点をとり、電流と電圧の関係を確認する。この試験の結果の例を図 3.4 に示す。逆方向電
206 圧では電流はほとんど流れないが、降伏電圧に達すると急激に増大する。これは pn 接合の特性 [12] であ
207 り、正常であればこの振る舞いを確認することができる。

208 3.2.5 FE チップ電流-電圧特性確認

209 FE チップに対して電圧をかけ、電流-電圧特性をみる。センサーに対してと同様に、印加電圧を段階的
210 に変化させて測定点をとり、電流と電圧の関係を確認する。FE チップは抵抗として振る舞い、電流、電
211 圧間の関係は図 3.5 のように線形性を持つ。

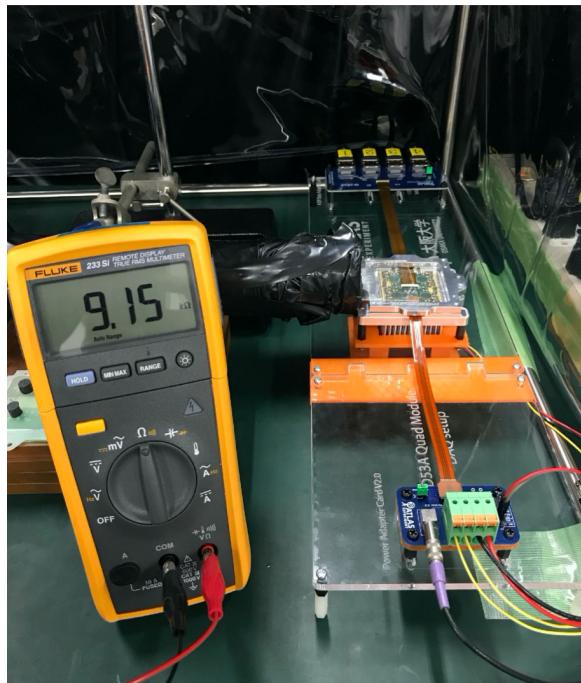


図 3.6 読み出し試験の様子 [17]。図は日本で実際に行われた読み出し試験の様子である。図の右側に見えるように、モジュールにケーブルを配線し、PCとの通信を行うことでデータ取得をする。また図の左側では抵抗値を測定している。これはモジュールに付属しているサーミスタの抵抗値を確認することで、温度情報を取得するためのものである。

212 3.2.6 読み出し試験

213 読み出し試験では、読み出し回路が正常に機能するのかを確認する。読み出し試験の様子を図 3.6 に示す。モジュールに読み出しケーブルを配線し、PC と通信を行い、データを取得する。また試験において 214 温度、FE チップやセンサーに与える電圧、電流といった検出器の操作、環境関わる情報取得を行って 215 行い、異常がないかを確認する。これらの情報は「Detector Control System, DCS」と呼ぶ。

217 汎用読み出しシステム YARR

218 YARR(Yet Another Rapid Readout) システム [18] は、ピクセルモジュール用に開発された、
219 PCI Express(PCIe) 接続を用いた読み出しシステムである。ファームウェア、ソフトウェアから構成さ
220 れる。YARR ではファームウェア上で行う処理はデータ通信やトリガー処理等の最低限に抑え、その他
221 多くの処理をソフトウェアで担うという特徴がある。

222 読み出し試験に使用するファイルと変数

223 YARR で扱う全てのファイルは JSON(JavaScript Object Notation) と呼ばれる形式で記述さ
224 れる。

225 YARR を用いた読み出し試験では以下の設定ファイルが要求される。

- 226 ● 試験設定ファイル。
 - 227 – 読み出し試験の初期設定や解析手法を記述する。
- 228 ● ハードウェア設定ファイル。

- 229 – 試験に用いるハードウェアの指定や設定を記述する。
- 230 ● 接続設定ファイル.
- 231 – 読み出しを行う FE チップの種類やチャンネルを記述する。
- 232 ● FE チップ設定ファイル.
- 233 – 各 FE チップ毎に出力され、全ピクセルに共通な試験の設定値、各ピクセル固有の設定値を記述する。

235 また試験 1 項目ごとに以下のファイルが、1 つのディレクトリに生成される。

- 236 ● 試験結果ファイル.
- 237 – 読み出し試験の結果値を記述する。
- 238 ● 試験設定ファイル.
- 239 ● FE チップ設定ファイル.
- 240 – 試験の中で変更が加えられるため、各 FE チップにつき試験前後の 2 つのファイルが出力される。
- 241 ● 試験ログ.
- 242 – 試験情報を記録する。

244 後述するローカルデータベースシステムにおいてはさらに以下の設定ファイルを用いる。

- 245 ● データベース設定ファイル.
- 246 ● 試験者設定ファイル.
- 247 ● 試験場所設定ファイル.

248 読み出し試験項目において以下の *Occupancy* という量が定義される。試験用電荷の入射数や発行トリー
249 ガーの数等、発行した信号数を n_i 回、取得信号数を n_0 としたとき、

$$\text{Occupancy} = \frac{n_0}{n_i} \times 100 [\%] \quad (3.1)$$

250 と定義する。

251 読み出し試験では各 FE チップに対して「レジスタ」と呼ばれる設定値が存在し、FE チップ上の全ピ
252 クセルに対して共通なものを「グローバルレジスタ」、各ピクセル固有のものを「ピクセルレジスタ」と
253 呼ぶ。

254 読み出し試験項目

255 以下に読み出し試験項目の一覧を示す。

256 レジスタの読み書き.

257 グローバル及びピクセルレジスタの読み書きが正常にできるのかを確認する試験.

258 デジタル回路読み出し.

259 各ピクセルのデジタル回路部に試験用電荷を入射し、信号の応答数を確認する試験. デジタル回路
260 部の性能確認に用いる.

261 アナログ回路読み出し.

262 各ピクセルのアナログ回路部に試験用電荷を入射し、信号の応答数を確認する試験. アナログ回路
263 部の性能確認に用いる.

264 Threshold 測定.

265 各ピクセルの Threshold 値を測定する試験.

266 Threshold グローバルレジスタ調整.

267 Threshold グローバルレジスタの変更、基準となる Threshold に近づけるための調整.

268 Threshold ピクセルレジスタ調整、再調整、精密調整.

269 Threshold ピクセルレジスタの変更、基準となる Threshold に近づけるための調整.

270 ToT グローバルレジスタ調整.

271 ToT グローバルレジスタの変更、基準となる ToT に近づけるための調整.

272 ノイズ占有率測定.

273 各ピクセルのノイズの頻度を確認する試験.

274 スタックピクセル測定.

275 入力電荷の有無にかかわらず、常に信号を出力するピクセルを確認する試験.

276 クロストーク測定.

277 各ピクセルのクロストークの有無を確認する試験. 隣接する Analog FE に試験用電荷を入射し、
278 応答を確認する.

279 バンプ接続確認測定.

280 各ピクセルのバンプ接合が正常かを確認する試験. 隣接する Analog FE に大きめの試験用電荷を
281 入射する. バンプが正常であればクロストークにより応答を確認できる.

282 外部トリガーを用いた測定.

283 外部トリガーを用いて信号の取得を行う試験. 放射線源を用いた測定に使用.

284 各試験における試験用電荷入射のイメージについて、付録 B に示す。

285 主な試験項目の詳細について以下で説明する。

286 デジタル回路読み出し

287 各ピクセルのデジタル回路部に試験用電荷を入射し、信号を取得する。Occupancy(式 3.1) は、入射電
288 荷数 n_i と取得信号数 n_0 で定義される。YARR のデフォルトでは $n_i = 100$ となっている。

289 デジタル回路読み出しにおける Occupancy の二次元分布の例を図 3.7 に示す。

290 この試験の結果として出力されるファイルを以下に示す。

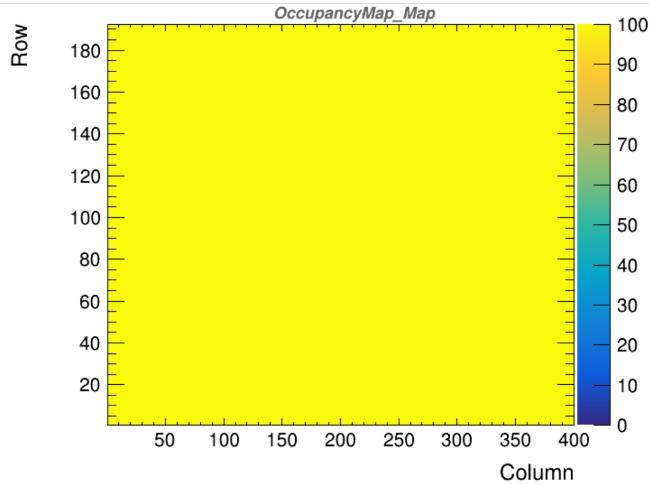


図 3.7 デジタル回路読み出しにおける *Occupancy* 分布の例。図は RD53A を用いたデジタル回路読み出しの結果であり、横軸、縦軸はそれぞれピクセルの列 (400)、行 (192) に対応する。 z 軸は各ピクセルにおける *Occupancy* の値を示している。正常なピクセルは 100 付近になることが期待され、図では全てのピクセルで 100 であることが分かる。

²⁹¹ OccupancyMap 各ピクセルの *Occupancy* を記す。

²⁹² Enmask $\text{Occupancy} = 100$ のピクセルを 1、それ以外を 0 とした値を記す。

²⁹³ L1Dist 信号取得タイミングの分布を記す。

²⁹⁴ アナログ回路読み出し

²⁹⁵ 各ピクセルのアナログ回路部に試験用電荷を入射し、信号を取得する。デジタル回路読み出しと同様、²⁹⁶ Occupancy (式 3.1) は、入射電荷数 n_i と取得信号数 n_0 で定義される。試験結果として出力されるファイルはデジタル回路読み出しと同じ形式である。²⁹⁷

²⁹⁸ Threshold 測定

²⁹⁹ 各ピクセルのアナログ回路部に試験用電荷を入射し、入射電荷数 n_i と取得信号数 n_0 より Occupancy ³⁰⁰ を測定する。これを入射電荷量を増加させて繰り返し行う。あるピクセルにおけるこの処理結果の例を図³⁰¹ 3.8 に示す。

³⁰² この分布を以下の式でフィッティングする。フィッティングの形に由来し、これを「S カーブフィッティ³⁰³ング」と呼ぶ。

$$f(x) = 0.5 \times \left[2 - \text{erfc} \left(\left\{ \frac{x - Q}{\sigma \times \sqrt{2}} \right\} \right) \right] \times p \quad (3.2)$$

$$\text{erfc}(x) = 1 - \text{erf}(x) \quad (3.3)$$

³⁰⁴ ここで $\text{erf}(x)$ はガウスの誤差関数である [19]。

³⁰⁵ 得られた Q 、 σ がそれぞれピクセルの Threshold 値、ノイズに相当する。ある FE チップにおける³⁰⁶ Threshold 値、ノイズの分布の例を図 3.9 に示す。この分布をガウス関数でフィットすることで、全ピク³⁰⁷セルに対する Threshold、ノイズの平均値と幅が得られる。

³⁰⁸ この試験において、出力される結果ファイルを以下に示す。

³⁰⁹ ThresholdMap 各ピクセルの Threshold 値を記す。

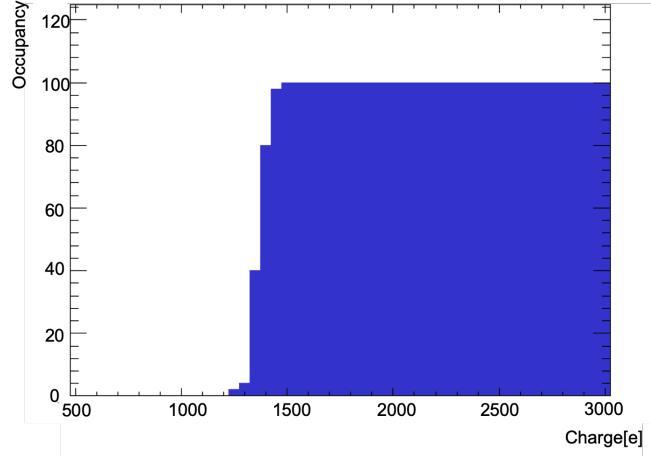


図 3.8 入射電荷量と *Occupancy* の関係。横軸はピクセルに対する入射電荷量 [e]、縦軸は *Occupancy* を示す。入射電荷量の単位 e は、電荷量 C を素電荷 $e = 1.6 \times 10^{-19}[\text{q}]$ で割ったものである。Threshold 測定の際にはこのように入射電荷量を増加させながら *Occupancy* の値を測定する。測定結果を後述する式 (3.2) でフィットし、Threshold とノイズの値を得る。

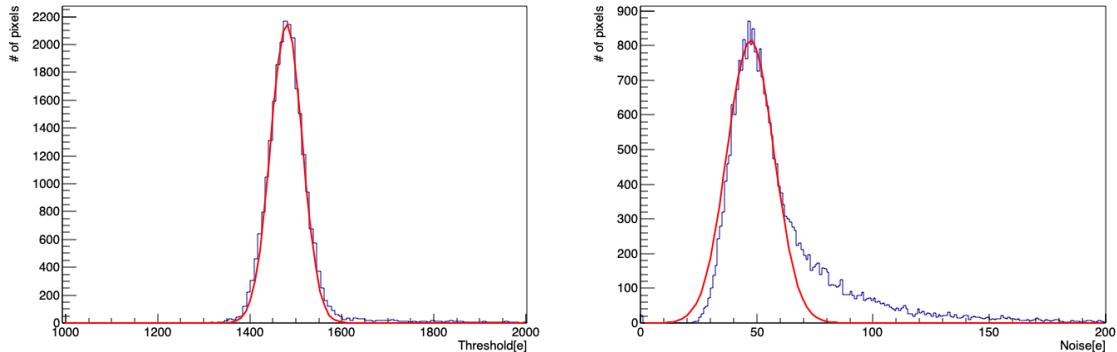


図 3.9 Threshold 値とノイズの分布。図はある FE チップ上のピクセルにおける Threshold 値(左図)とノイズ(右図)の分布を示している。また赤線はガウス関数によるフィット関数を示しており、これにより平均値と幅を取得する。

- 310 ThresholdDist Threshold 値の分布を記す.
- 311 NoiseMap 各ピクセルのノイズを記す.
- 312 NoiseDist ノイズの分布を記す.
- 313 Chi2Map 各ピクセルの S カーブフィッティングにおける χ^2 の値を記す.
- 314 Chi2Dist χ^2 の分布を記す.
- 315 sCurve あるピクセルの入射電荷量と *Occupancy* の関係を記す. いくつかのピクセルについて出力する.
- 316 sCurveMap 全てのピクセルの SCurve を重ね合わせた値を記す.

317 ToT 測定

各ピクセルのアナログ回路部に試験用電荷を入射し、ToT を測定する。この操作を複数回 (デフォルトでは 100 回) 行い、平均値と幅を求める。出力される結果ファイルを以下に示す。

- 320 MeanToTMap 各ピクセルの ToT の平均値を記す.

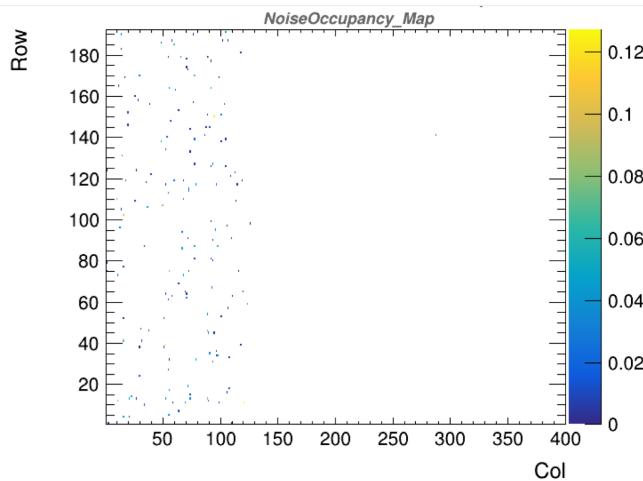


図 3.10 ノイズ占有率測定における *NoiseOccupancy* 分布の例。図は RD53A を用いたデジタル回路読み出しの結果であり、横軸、縦軸はそれぞれピクセルの列(400)、行(192)に対応する。z 軸は各ピクセルにおける *NoiseOccupancy* の値を示している。正常なピクセルでは限りなく 0 に近い値になることが期待され、図ではほとんどのピクセルが 0 となっているが、列番号が 0 から 120 付近の領域で値を大きく持つピクセルが存在する。

- 321 MeanToTDist ToT 平均値の分布を記す。
- 322 SigmaToTMap 各ピクセルの ToT の分散を記す。
- 323 SigmaToTDist ToT 分散の分布を記す。

324 ノイズ占有率測定

- 325 試験用電荷を入射せずに、周波数 f [Hz]、時間 t [sec] のトリガーをかけ、取得信号数 n_0 を測定する。
- 326 YARR のデフォルトでは $f = 5000$ 、 $t = 300$ である。 $Occupancy$ (式 3.1) は、発行したトリガー数 $n_i = f \times t$ と取得信号数 n_0 で定義される。
- 327 また *NoiseOccupancy* を以下で定義する。*NoiseOccupancy* の分布の例を図 3.10 に示す。

$$NoiseOccupancy = \frac{n_0}{t} \times (25 \times 10^{-9}) \quad (3.4)$$

- 329 25×10^{-9} [sec] = 25[nsec] は 1 b.c. である。
- 330 出力される結果ファイルを以下に示す。

- 331 OccupancyMap 各ピクセルの *Occupancy* を記す。
- 332 NoiseOccupancyMap 各ピクセルの *NoiseOccupancy* を記す。
- 333 NoiseMask $NoiseOccupancy < 10^{-6}$ のピクセルを 1、それ以外を 0 とした値を記す。

334 Threshold 調整とピクセル解析

- 335 今後この論文では、この試験を読み出し試験と呼ぶ。以下の流れで読み出しを行う。

- 336 ● デジタル回路読み出し。
- 337 ● アナログ回路読み出し。
- 338 ● Threshold 測定。
- 339 ● Threshold グローバルレジスタ調整。

表3.1 ピクセル解析の評価基準一覧 [20]。読み出し試験において各ピクセルが正常に機能しているかを判断するためにピクセル解析を行う必要があり、その判断基準が表のように定義されている。この基準一覧は不良評価であり、全ての基準に当てはまらないピクセルが良好と判断される。不良ピクセルの数や分布は、モジュールの品質を決定する1つの要素となり、ITkに搭載するモジュールの選択や配置の決定に用いられる。

評価名	読み出し項目	評価基準
Digital Dead	Digital scan	$Occupancy < 1$
Digital Bad	Digital scan	$Occupancy < 98 \text{ or } Occupancy > 102$
Merged Bump	Analog scan Crosstalk scan	$Occupancy < 98 \text{ or } Occupancy > 102$ High Crosstalk
Analog Dead	Analog scan	$Occupancy < 1$
Analog Bad	Analog scan	$Occupancy < 98 \text{ or } Occupancy > 102$
Tuning Failed	Threshold scan	Sカーブフィット失敗 (YARRでは $\chi^2 = 0$ となる)
Tuning Bad	Threshold scan ToT scan	$ Threshold - Threshold_{平均} > 5 \times Threshold_{幅}$ $ToT = 0 \text{ or } 15$
High ENC	Threshold scan	$ ノイズ - ノイズ_{平均} > 3 \times ノイズ_{幅}$
Noisy	Noise scan	$NoiseOccupancy > 10^{-6}$
Disconnected Bump	Disconnected bump scan Source scan	現段階では未決定 $Occupancy$ がFEチップ全体平均の1%
High Crosstalk	Crosstalk scan	$Occupancy > 0$ with 25ke (sync FE) $Occupancy > 0$ with 40ke (lin and diff FE)

- 340 • Threshold ピクセルレジスタ調整.
- 341 • ToT グローバルレジスタ調整.
- 342 • Threshold グローバルレジスタ再調整、精密調整.
- 343 • Threshold 測定.
- 344 • スタックピクセル測定.
- 345 • クロストーク測定.

346 測定後、試験結果の解析を行い、モジュール上の各ピクセルが正常かどうかを判断する。設定されてい
347 る評価基準を表3.1に示す。不良ピクセルには評価基準に応じた評価名が付けられる。

348 簡易読み出し試験

349 簡易読み出し試験では以下の項目を扱う。

- 350 • レジスタの読み書き.
- 351 • デジタル回路読み出し.
- 352 • アナログ回路読み出し.
- 353 • Threshold 読み出し.
- 354 • ToT 読み出し.
- 355 • バンプ接続確認読み出し.

356 バンプ接合確認試験

357 放射線源を用いてバンプ接合の確認を行う。以下の項目を扱う。

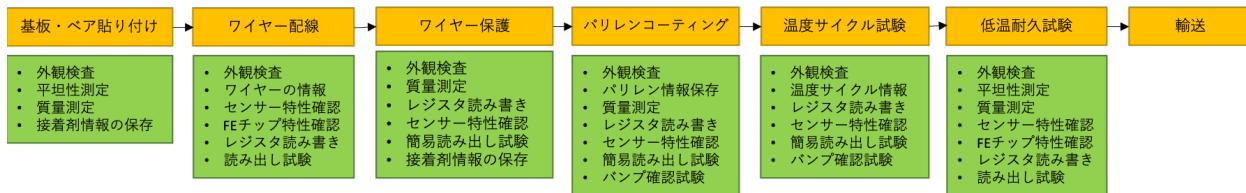


図 3.11 組み立て工程と対応する品質試験一覧。節 3.1 で述べた各組み立て工程において、複数の品質試験を行う必要がある。図は各組み立て工程と対応する品質試験一覧を示している。試験数は 30 程度存在する。

- 358 ● バンプ接合確認測定.
- 359 ● ノイズ占有率測定測定.
- 360 ● 外部トリガーを用いた測定.

361 3.2.7 各組み立て工程における品質試験

362 各組み立て工程と品質試験項目を図 3.11 に示す。図より、1 モジュールに対して行われる品質試験は
363 30 程度存在することが分かる。

364 3.3 検出器量産におけるデータ管理

365 各組み立て機関で $O(100) \sim O(1,000)$ のモジュールを作り、上述したように 1 モジュールに対して 30
366 程度の品質試験を行う。1,000 モジュール作る機関であれば 3,000 の試験結果が得られる。この数の品質
367 試験結果を正確に管理することが必要となる。特に読み出し試験については、1 試験につき更に細かい項目に細分化され、項目、結果ファイルも多様である。これらの情報は最終的に共通のデータベースに保存
368 する必要があり、各組み立て機関で適切に管理する必要がある。

369 本研究では各組み立て機関におけるモジュール情報及び品質試験のデータ管理を簡易化することを目的
370 として、データベースシステムの構築を行った。このシステムについて 4 章で記述する。

372 第4章

373 モジュール情報及び品質試験結果管理シ 374 ステム

375 前章で述べたように、モジュール生産及び品質試験を世界中で行う。これらの情報はデータベースシ
376 テムを用いて管理することが決定していて、現在この開発を行っている。システムは、ITk の全情報を保
377 存する中央データベースと、各組み立て機関に設置し、データ管理を行うローカルデータベースから成
378 る。本章ではこれらのデータベースについて説明する。また本研究における開発項目について詳細に説明
379 する。

380 4.1 中央データベース

381 4.1.1 中央データベースの概要

382 概要

383 中央データベースは、ITk の製造に関する全ての情報の保存を目的として開発されたデータベースであ
384 る。ユニコーン大学 [21] が開発、運用を行っていて、チェコにデータベースサーバーが設けられている。
385 これらを生産するにあたって、シリコンセンサーやフレキシブル基板といった小さな部品から製造を行
386 い、それらを用いたモジュールの組み立て、複数モジュールを搭載したステープやリングの組み立てを経
387 て検出器が完成する。また各組み立て段階において、動作確認等を目的とした品質試験を行う。これらの
388 過程における全ての構成部品の情報、及び品質試験結果を中央データベースに保存する。

389 意義

390 中央データベースを扱う一番の目的は、ITk に用いるモジュールの選別とその配置の決定に用いること
391 である。中央データベースに保存された品質試験結果を解析し、ITk に搭載するモジュールを選別を行
392 う。また、モジュール配置も品質を考慮して決定する。例として以下の 2 つをあげる。

- 393 1. $|\eta|$ が小さい領域には、不良ピクセルが少なく品質の良いモジュールを搭載する。
- 394 2. 不良ピクセルの領域や不良モジュールが固まらないようにする。

395 項目 1 に関して、ヒッグスなど物理解析で興味の対象となる粒子は相対的に大きい質量をもつため、ビー
396 ム軸方向に大きな運動量を持たず $|\eta|$ の小さい領域域に生成される。図 1.13 のイベントディスプレイで
397 も、ヒッグス粒子から生成される電子やミューオンが持つ $|\eta|$ は小さいことが分かる。そのため、 $|\eta|$ が小

398 さい領域には品質の良いモジュールを配置する。項目2に関して、不良ピクセルの領域がばらけるように
399 モジュールの配置を決定する。

400 次に中央データベースに保存された情報は、検出器運転時の参考値として扱われる。モジュールを例に
401 だと、品質試験で読み出し試験を行った際の最適な設定値を中央データベースに保存するため、実際の
402 運転時に参照することができる。また運転前後での検出器性能比較を行うことができる。

403 4.2 ローカルデータベース

404 4.2.1 ローカルデータベースの意義と概要

405 中央データベースでは、前述したようにモジュール情報のみならずITkの製造に関わるすべての情報
406 を管理する。データベースの機能としては、製造を通して汎用的に使えるものになっている。モジュール
407 の組み立て及びその品質試験に関しては3章で述べたように工程が複数に渡り、行う品質試験の数も多
408 い。1つの組み立て機関で多いところでは数千個のモジュールを作ることになるため、データ管理が簡単
409 にかつ円滑に進むようになっているのが好ましい。このような理由から、組み立て機関での生産性、利便
410 性に特化し、円滑な生産をサポートすることを目的としたデータベースシステムを開発しており、これを
411 「ローカルデータベース」と呼ぶ。システムの概要図を図4.1に示す。オープンソースのサービスである
412 MongoDB[22]と、ローカルデータベース用ウェブアプリケーションを併用することで、データ管理や中
413 央データベースとの同期を行うシステムとなっている。ここで開発しているアプリケーションはPython
414 のウェブフレームワークであるFlask[23]を使用している。

415 ローカルデータベースシステムは以下のような利点を持ち得る。これらの利点を持つシステムの開発、
416 実装を行うことが開発課題となっている。

- 417 • ローカルにデータベースサーバーを立てるため通信が速く、円滑にデータ管理を行うことができる。
- 419 • モジュールの組み立て工程を管理し、組み立て工程に応じた試験項目の表示や解析を行うことで生
420 産者の適切な処理を助ける。
- 421 • モジュールに特化したデータ管理、結果解析を行うことで異常をいち早く検知できる。
- 422 • 試験者情報や試験時間など、品質試験結果以外の必要な情報を正確に管理できる。

423 一方で欠点として、ローカルデータベースと中央データベース間で同期を行う必要があり、それを正確
424 に実行することが技術的に困難であることがあげられる。この問題は以下により解決する。それぞれの詳
425 細については後述する。

- 426 • 同期ツールの開発.
- 427 • データベース操作の流れの確立.
- 428 • 操作方法の普及に向けたチュートリアル、ドキュメント作成.

429 4.2.2 先行研究と開発課題

430 先行研究で開発された項目と、本研究で取り組んだ開発課題を以下に示す。

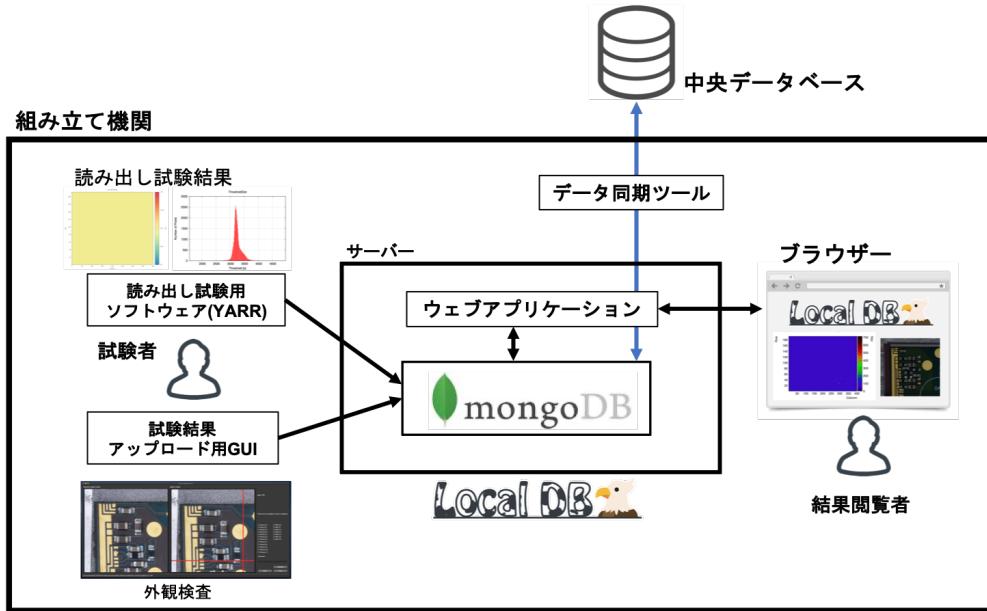


図 4.1 ローカルデータベースシステムの概要。各組み立て機関で MongoDB とローカルデータベース用ウェブアプリケーションの立ち上げを行い、独自にデータ管理をするシステムとなっている。ウェブアプリケーションには Python パッケージである Flask を使用している。品質試験者は図のようにいくつかのソフトウェアを用いて試験結果を MongoDB に保存する。保存された結果はウェブアプリケーションによって閲覧することができる。また結果は中央データベースに集める必要があるため、同期ツールを用いて試験結果の共有を行う。

431 先行研究

432 先行研究 [24]においてデータベースシステムの設計と開発がなされ、図 4.1におけるいくつかの項目が
433 開発された。以下に実装項目を記す。

- 434 • 読み出し試験結果保存のための MongoDB 内部構造の設計.
- 435 • 1 の構造を用いて YARR から MongoDB への読み出し試験結果アップロード機能.
- 436 • 試験結果のソート及び閲覧が可能な、Flask を用いたウェブアプリケーションの開発と実装.

437 モジュールの生産、品質試験に向けたデータ管理、そしてローカルデータベースの利点を活かすには以
438 下のような開発課題が残されていた。

- 439 1. 中央データベースの内部構造の実装 (節 4.3.1).
- 440 2. 中央データベースとローカルデータベース間の同期ツールの開発 (節 4.3.2).
- 441 3. ローカルデータベースにおける品質試験に特化したデータ管理と機能提供.
- 442 4. 量産時におけるデータベース操作の流れの確立 (節 4.3.7).

443 本研究ではこれらの開発課題に対して取り組みを行なった。特に開発項目 3に関しては、以下の機能の
444 開発、実装を行なった。

- 445 (i) 品質試験者及びシステムユーザ管理機能及びコメント付加等の各種ユーザ機能 (節 4.3.3).
- 446 (ii) 品質試験結果の登録と組み立て工程の自動更新 (節 4.3.4).
- 447 (iii) 読み出し試験におけるピクセル解析ツールの開発 (節 4.3.5).

448 (iv) 読み出し試験結果の検索機能（節4.3.6）。

449 なお先行研究で開発したデータ構造やアプリケーションといった項目は既に複数の機関で試験運用され
450 ていたため、本研究ではその構造やソフトウェアを大きく変更せずに、拡張する形で開発を行なった。

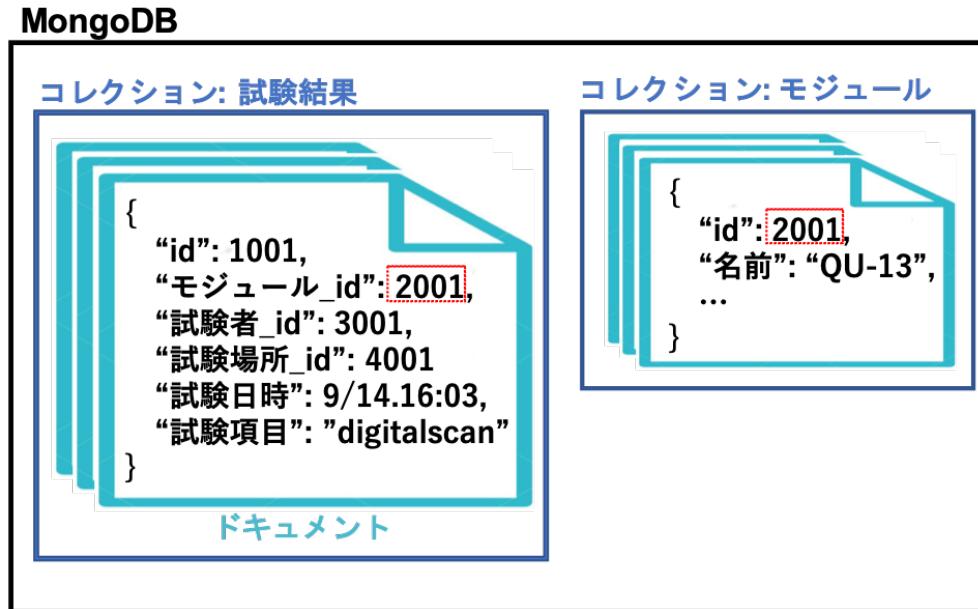


図 4.2 MongoDB の構造の例。図のように MongoDB では JSON 形式でデータを格納する。1 枚の JSON インスタンスをドキュメントと呼び、複数のドキュメントが格納されている枠組みをコレクションと呼ぶ。ドキュメントの構造及びコレクション間の関係等を決めることでデータベースの構造を定義する。ドキュメント間の紐付けは、各ドキュメント内部に ID を持つことで可能となる。図の例では試験結果のドキュメントが {“モジュール_id”:2001} の情報を持っており、これがモジュールに保存されているドキュメントの ID に対応する。

4.2.3 MongoDB と内部構造 [25]

MongoDB とは NoSQL に分類されるデータベースである。MongoDB の構造について簡単に表したもの

を図 4.2 に示す。一般的な SQLDB のようにテーブル形式ではなく、JSON 形式で情報を格納する。

情報を保持している一枚の JSON インスタンスを「ドキュメント」と呼び、「コレクション」と呼ばれる

枠に複数のドキュメントが格納されている。各ドキュメントは「ID」と呼ばれるハッシュ値を持ってい

て、異なるコレクションにおけるドキュメント間の紐付けはこの ID を用いて行う。

ローカルデータベースシステムにおいて、MongoDB を使用する主な利点を以下に示す。

- 各コレクションに格納するドキュメントの構造が動的であるため、開発を柔軟に行うことができる。
- JSON 形式でデータを保持するため情報取得の際の整形処理が容易であり、ウェブアプリケーションとの親和性が高い。
- データのキャッシュをメモリ上に置き処理を実行するため、高速な読み書きが可能 [26]。

モジュール及び品質試験に用いる主なコレクションを表 4.1 に示す。また先行研究で設計された読み出し試験結果に関する内部構造を図 4.3 に示す。

表4.1 品質試験に用いる主なコレクション。ローカルデータベースシステムにおいて、MongoDB 内に 2 つのデータベースを設置し、使用する。

データベース名	コレクション名	情報
localdb	component	モジュール情報、FE チップ情報
	childParentRelation	FE チップとモジュールの関係性
	QC.module.status	各モジュールに対する組み立て工程及び選択された試験結果
	QC.result	品質試験結果
	testRun	読み出し試験結果
	user	読み出し試験実施者
	institute	読み出し試験実施場所
	componentTestRun	component と testRun の関係性
	comments	コメント情報
localdbtools	QC.status	組み立て工程及び試験項目
	viewer.user	登録ユーザの情報
	viewer.query	読み出し結果キーワード、検索機能実行時に使用
	viewer.tag.docs	モジュールや試験結果に付けるタグの情報

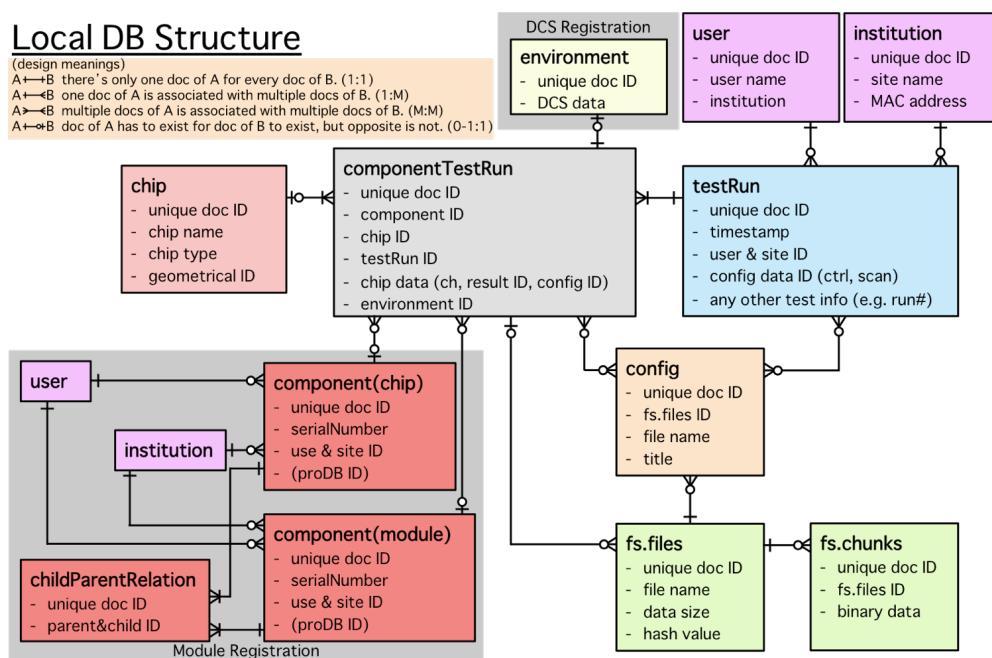


図 4.3 先行研究により設計された読み出し試験結果の MongoDB 内部構造 [24]。それぞれの四角はコレクション、直線は ID によるドキュメント間のリンクを示している。直線が十字になっている場合は対応するドキュメントが 1 つであることを示し、分岐しているものは複数であることを示す。また直線状の白丸は対応するドキュメントが存在しない場合があることを示している。

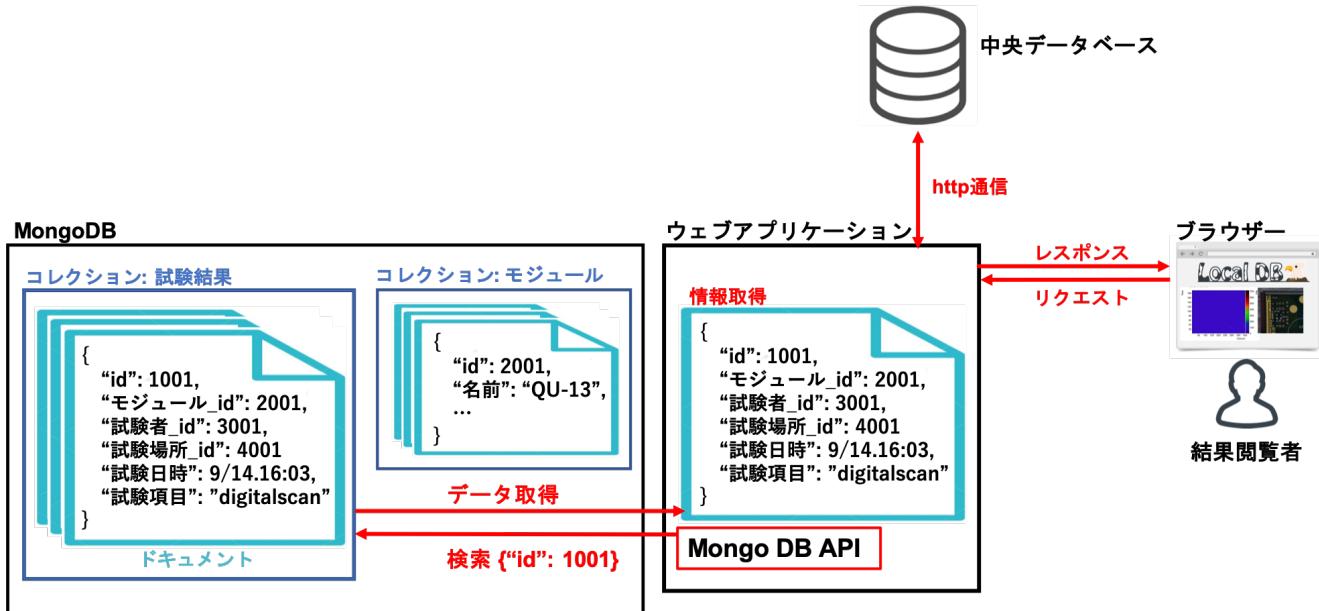


図 4.4 ウェブアプリケーション処理のイメージ。ウェブアプリケーションでは MongoDB と通信を行う API(PyMongo) を用いて、データベースのコレクションに検索をかけることで情報を取得する。取得した情報は整形されたのちブラウザに送信、中央データベースとの同期等の処理に用いられる。

4.2.4 ウェブアプリケーション

各組み立て機関において、試験者が品質試験結果を閲覧、管理するツールとして、ウェブアプリケーションを提供している。アプリケーション開発には、Python のウェブフレームワークである Flask を使っている。またアプリケーションにおいて MongoDB との通信に用いる API として、Python ライブリで PyMongo[27] を用いている。アプリケーション処理に特化したイメージを図 4.4 に示す。このようにアプリケーションはデータベースとブラウザ、データベース間のインターフェースとなっている。

試験結果を迅速に分かりやすく見るシステムを作り、円滑な生産補助や異常結果の早期発見を目的としている。またデータベースの情報管理のみならず、同期ツールや、後述する試験結果解析ツールなどの外部スクリプトの実行、結果取得等、生産時における多くのデータベース操作はこのアプリケーションを用いて行う。

ウェブアプリケーションでは、現在以下の機能を使用することができる。ある品質試験の結果ページを図 4.5 に示す。

- 登録モジュール情報及び品質試験結果の閲覧、解析.
- ローカルデータベースにおけるユーザ管理機能.
- データベース同期実行機能.

Result: 1348

Information

Component

Result

Scan

Key	Data
runNumber	1348
testType	std_digitalscan
stage	MODULEWIREBONDING
component	20UPGR00000001 20UPGFC9999999
startTime	2020/11/13 19:49:40
finishTime	2020/11/13 19:49:45
user	hokuyama
site	lazulite
targetCharge	-1
targetTot	-1
exec	-r configs/controller/specCfg.json -c db-data/connectivity.json -s configs/scans/rd53a/std_digitalscan.Json -W
stopwatch	analysis: 626 config: 37 processing: 1 scan: 2821
QC	False
environment	True
plots	L1Dist EnMask OccupancyMap
passed	True
qcTest	False
qaTest	False
summary	False

Output Data

Type	Format	Chip	Display	Download
ctrlCfg	json			
dbCfg	json			
siteCfg	json			
userCfg	json			
scanCfg	json			
beforeCfg	json	20UPGFC9999999		
afterCfg	json	20UPGFC9999999		
EnMask	json	20UPGFC9999999		
OccupancyMap	json	20UPGFC9999999		
L1Dist	json	20UPGFC9999999		

PLOT JSROOT

L1Dist plotly

20UPGFC9999999

EnMask plotly

20UPGFC9999999

OccupancyMap plotly

20UPGFC9999999

DCS plot

© 2019 ATLAS Japan ITk and [Tokyo Tech](#) with the great help of LBNL.

Released under the GNU license, please read [LICENSE.TXT](#)

Please refer to the LocalDB official document. [Link](#)

If found any problem, please contact to [Hide Oide](#), [Eunchong Kim](#), [Arisa Kubota](#), [Hiroki Okuyama](#), [Satoshi Kinoshita](#)

図 4.5 品質試験結果ページの例。図は品質試験項目であるデジタル回路読み出しの結果を表している。図の上部に試験情報や設定値、下部に結果のグラフが表示されているのを確認できる。

4.3 本研究における開発項目の詳細

481 以下は本研究で開発した項目である。

4.3.1 開発項目 1: 中央データベースの内部データ構造の実装

484 モジュール及びその品質試験に関する情報を中央データベースに保存するために、情報構造の定義、実
485 装を行う必要がある。以下の項目について、中央データベースが提供している API を用いて構造の定義
486 を行った。

- 487 1. モジュールの種類とその構成部品 (図 4.6).
- 488 2. モジュール組み立て工程と付随する品質試験 (表 4.2).

489 また項目 1 に関して、Quad モジュールに関する例を図 4.7 に示す。

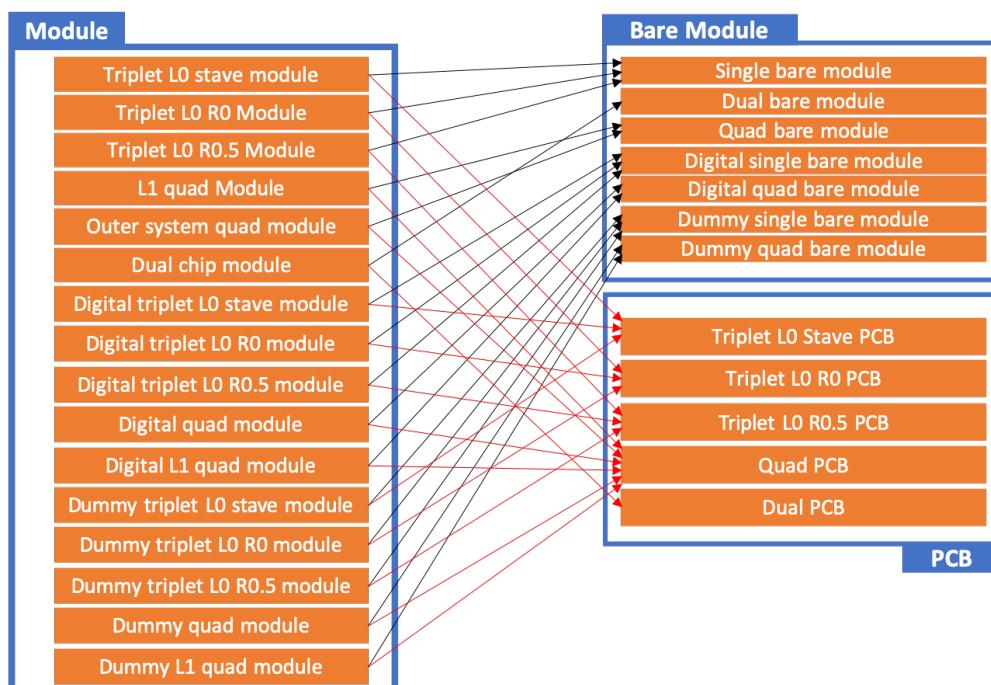


図 4.6 中央データベースにおけるモジュールの種類と構造。中央データベースにモジュールを登録するときの情報として、モジュールの種類、構成部品を図のように実装した。図の左側に実装したモジュールの種類を示しており、Triplet、Quad というように、モジュールの種類ごとに登録できるシステムとなっている。また矢印は構成部品を指しており、各モジュールは対応する Bare Module(ベアモジュール)と PCB(フレキシブル基板)を持つ。DB の中でモジュールと構成部品の紐付けも同時にを行うことができる。

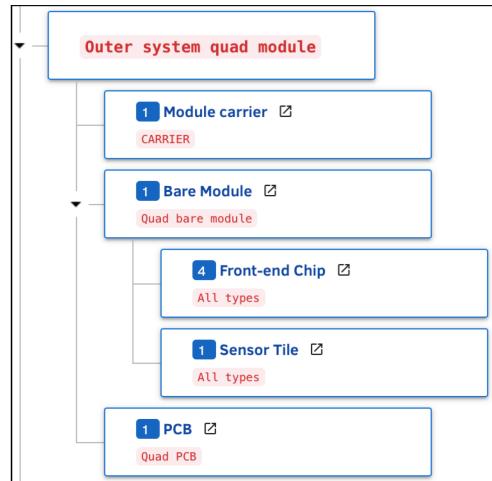


図 4.7 中央データベース内におけるモジュール構造の一例 (Quad モジュール)。例として Outer system quad module の中央データベース内の構造を示している。この種類では構成要素としてそれぞれ対応する種類の Module carrier、Bare Module、PCB を持つことがわかる。さらに Bare Module は FE chip を 4、Sensor を 1 持つことが分かり、Quad モジュールの構造が正しく実装されていることが分かる。

表 4.2 中央データベースにおける組み立て工程と付随するテスト項目。モジュールの組み立て工程及び品質試験をアップロードするため、表のような構造を実装した。データベース内でこの表に沿った組み立て工程の登録、更新、試験結果のアップロードができるようになった。

組み立て項目	付随する組み立て情報及び品質試験項目
1. Bare to PCB assembly	Visual Inspection Metrology Mass measurement Glue information
2. Wirebonding	Visual Inspection Wirebond information (Wirebond pull test) First power up Sensor IV SLDO VI Chip configuration Pixel failure test
3. Wirebond Protection	Visual Inspection Potting information Sensor IV Register test Readout for basic electrical
4. Parylene Coating	Visual Inspection Parylene information Mass measurement Sensor IV Register test Readout for basic electrical Bump bond quality
5. Thermal Cycling	Visual Inspection Thermal cycling info Sensor IV Register test Readout for basic electrical Bump bond quality
6. Burn-in	Visual Inspection Metrology Mass Measurement First power up Sensor IV SLDO VI Chip configuration Pixel failure test
7. Reception	

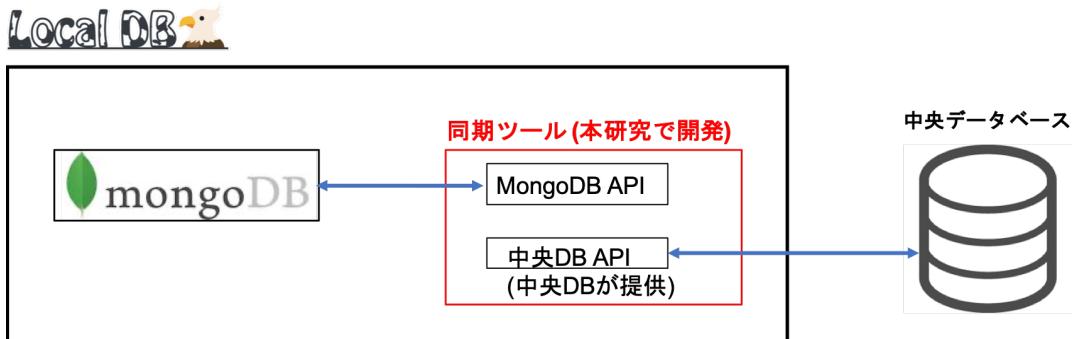


図 4.8 同期ツール処理のイメージ。本研究で開発を行っているのは図の赤線の領域に対応する同期ツールである。このツールは Python を用いて開発しており、処理の中でローカルの MongoDB と通信する API と、中央データベースが開発、提供をしている API を用いることで、2 つのデータベース間の同期を行っている。このとき、中央データベースとの通信は http 通信で行われる。

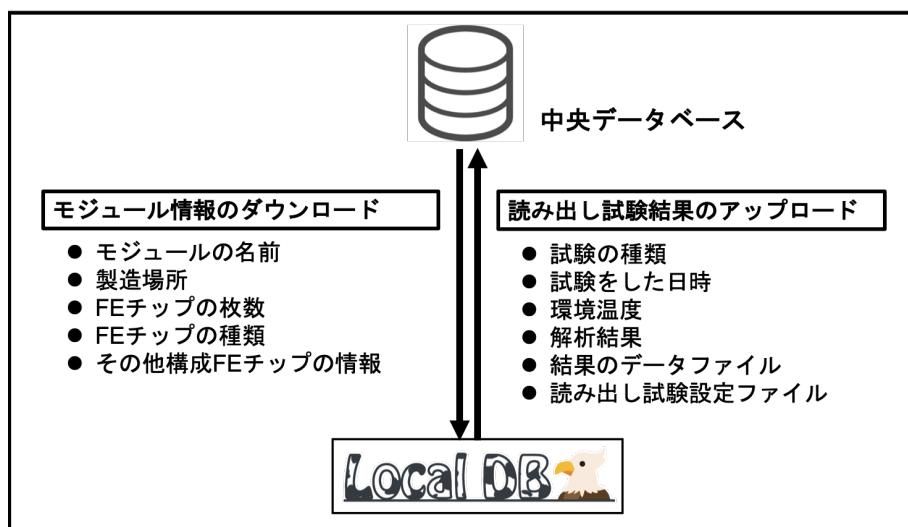


図 4.9 同期機能の概要。本研究ではモジュール情報のダウンロードと読み出し試験結果のアップロード機能を実装した。図に示している情報を主に同期する機能となっている。

4.3.2 開発項目 2: データベース同期ツールの開発

モジュール情報や品質試験結果の共有のために、中央データベースとローカルデータベースの間で同期が行われる必要がある。これを行うツールを設計、開発を行った。

ツールの中では中央データベースが開発、提供している中央データベース通信用 API と、節 4.2.4 で述べたローカルの MongoDB と通信する API の 2 つを用いることで同期を行っている。

同期ツール処理のイメージを図 4.8 に示す。

特に本研究ではツールの枠組み設計に加えて、以下の機能を実装した。

- モジュール及び構成する FET 情報のダウンロード機能.
- 読み出し試験結果のアップロード機能.

これらの機能のイメージを図 4.9 に示す。実装の詳細及び処理時間測定について 8 章で述べる。

500 4.3.3 開発項目 3-(i): ユーザ管理機能及び各種機能

501 異常があった際に確認することを目的として、誰が試験を行ったかを記録することが必要である。ま
502 た、モジュールの登録や中央データベースとの同期など、データベースの機能使用を制限することも必要
503 である。これらを目的として、試験者及びデータベース使用者情報の管理機能を開発、実装した。この詳
504 細を以下で述べる。

505 **機能概要**

506 データベース権限の段階として、管理者、権限付きユーザ、一般ユーザの3段階を設けた。各ユーザが
507 使うことのできる機能を表4.3に示す。

508 権限付きユーザの機能としてモジュール及び試験結果にコメント、タグを付ける機能を実装した。使用
509 したときの様子を図4.10、4.11に示す。

510 **ユーザ登録操作**

511 表4.3において管理者と権限付ユーザの登録について説明する。

512 データベースシステム導入時に管理者のアカウントを作成する。コマンドプロンプト上で開発したスク
513 リプトを用いて実行することで管理者登録が行われる。この際ユーザ名とパスワードを入力する。

514 権限付ユーザについて、全ての品質試験者及びデータベースユーザ機能使用者は管理者によってユーザ
515 登録される必要がある。登録はウェブアプリケーションを用いて行い、以下の情報を入力する。

- 516 ● ユーザ名
517 ● 氏名
518 ● 所属機関
519 ● メールアドレス

520 管理者が登録を完了すると、登録されたメールアドレスに登録完了メールと仮パスワードが届く。この
521 メールに従い、ウェブアプリケーション上でユーザがパスワード登録を完了する。

522 このようにメール機能を用いることでパスワード漏洩の防止、管理者操作の削減を目的としている。

表 4.3 ローカルデータベースユーザ権限及び使用機能一覧。ローカルデータシステムにおけるユーザとして、管理者、権限付きユーザ、一般ユーザの3つを設けた。全てのユーザがウェブアプリケーションの閲覧をすることができる。管理者、権限付きユーザにはデータベース読み書き権限とウェブアプリケーションログイン権限が与えられ、試験結果のアップロード、アプリケーション上のユーザ機能の実行ができる。また管理者は権限付きユーザを登録することができる。

ユーザ	付加される権限	使用できる機能
管理者	ユーザ管理権限 データベース読み書き権限 ウェブアプリケーションログイン権限	権限付きユーザ登録機能
権限付ユーザ	データベース読み書き権限 ウェブアプリケーションログイン権限	試験結果のアップロード 中央データベースとの同期機能 その他ウェブアプリケーションの機能（コメント、タグ）
一般ユーザ		モジュール情報及び試験結果の閲覧

Comment	componentType	Name	Institution	Date
Good results!!	front-end_chip	okuyama	TokyoTech	2020-11-29 09:24:03.498000

図 4.10 ウェブアプリケーションにおけるコメント機能。権限付きユーザ及び管理者はモジュールや試験結果に対してコメントをすることができる。図のようにページの右側にコメント欄があり、コメントをテキスト形式で記述することができる。

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
	JohnDoe_0	std_digitalscan	okuyama	default_host	2020/10/27 15:40:34	result page	anomaly
	JohnDoe_0	std_digitalscan	okuyama	default_host	2020/10/27 15:38:14	result page	good
	JohnDoe_0	std_digitalscan	okuyama	default_host	2020/10/27 15:37:41	result page	anomaly

図 4.11 ウェブアプリケーションにおけるタグ機能。権限付きユーザ及び管理者はモジュールや試験結果に対してタグをつけることができる。図は試験結果の一覧ページであり、図の表において一番右の列がつけられたタグを示しており、図では anomaly や good といったタグが付けられていることが分かる。

機能の仕組み

ユーザ登録の際には内部で以下の2つの処理が行われるように実装した。

1. MongoDBアカウントの作成、読み書き権限の付与.
2. ウェブアプリケーションで用いるユーザ情報ドキュメントの作成.

1の処理を行う理由は、登録ユーザが試験結果をMongoDBにアップロードできるようにするためにある。2の情報は、ウェブアプリケーション内でのログイン判断、ユーザの情報保持に使う。この情報は表4.1のviewer.userに保存される。2つの処理について、実際に保存されるドキュメントの例をコード4.1、4.2以下に示す。

ソースコード4.1 MongoDBアカウント情報を持つドキュメントの例。リスト中の”roles”より、localdbとlocaldbtoolsの読み書き権限が付加されていることが分かる。

```

531 1 {
532 2     "_id" : "localdb.hokuyama",
533 3     "userId" : UUID("fee321eb-83b8-434a-a4a0-fff638b5db36"),
534 4     "user" : "hokuyama",
535 5     "db" : "localdb",
536 6     "credentials" : {
537 7         ...
538 8     },
539 9     "roles" : [
540 10        {
541 11            "role" : "readWrite",
542 12            "db" : "localdb"
543 13        },
544 14        {
545 15            "role" : "readWrite",
546 16            "db" : "localdbtools"
547 17        }
548 18    ]
549 19 }
550 }
```

ソースコード4.2 ウェブアプリケーションで扱うユーザ情報を持つドキュメントの例。リスト4.1で示したものとは別に、ウェブアプリケーション内でユーザ情報を扱うためにこのドキュメントを保持する必要がある。ウェブにおいてログインはこのドキュメントの存在確認をもってなされる。パスワードはhash化して保存している。

```

552 1 {
553 2     "_id" : ObjectId("5f0bbe84ef87af2628865de7"),
554 3     "sys" : {
555 4         "rev" : 0,
556 5         "cts" : ISODate("2020-07-13T10:53:07.943Z"),
557 6         "mts" : ISODate("2020-07-13T10:53:07.943Z")
558 7     },
559 8     "username" : "hokuyama",
560 9     "name" : "Hiroki\Okuyama",
561 10    "auth" : "readWrite",
562 11    "institution" : "Tokyo\Institute\of\Technology",
563 12    "Email" : "okuyama@hep.phys.titech.ac.jp",
564 13    "password" : "5f4dcc3b5aa765d61d8327deb882cf99"
565 14 }
```

568 4.3.4 開発項目 3-(ii): 品質試験結果の登録と組み立て工程の自動更新

569 ローカルデータベースへアップロードした品質試験結果の中から、本結果として中央データベースへ同
570 期する結果を選択する必要ある。これは不必要的結果を同期せず、中央データベースを簡潔に保つ目的が
571 ある。この結果選択機能を開発した。品質試験は各モジュール、各組み立て工程に対して行うものである
572 ため、結果選択も同様に工程毎に行うことを想定している。結果選択後、データベースにおける組み立て
573 工程は次のものへ自動的に更新する機能となっている。

574 概要

575 あるモジュール、組み立て工程に対して結果を選択する様子を図 4.12 に示す。組み立て工程も自動更
576 新されていることがわかる。

577 仕組み

578 コード 4.3、4.4 のようなドキュメントを作成、保存する。コード 4.3 は全てのモジュールに対して共通
579 のドキュメントであり、組み立て工程と各工程における品質試験項目を記録する。これらの情報は中央
580 データベースから取得する。この情報を参照することでローカルデータベース内部で組み立て工程の管理
581 が可能となる。

582 コード 4.4 は各モジュールに対して 1 つ存在し、以下のような情報を保持する。

- 583 ● モジュールの現組み立て工程.
584 ● 各工程における選択された品質試験結果の ID.

ソースコード 4.3 組み立て工程及び品質試験一覧情報のドキュメント。このようなドキュメントを作
成、保持しておくことで組み立て工程及び品質試験の情報を扱う。ローカルデータベース内に 1 つこ
のドキュメントを保持し、品質試験結果選択、組み立て工程の更新時にこのドキュメントを参照する。
このドキュメントは中央データベースより取得して作成する。

```
585 1 {
586 2     "_id" : ObjectId("5fc89aa232d56b29091fd64d"),
587 3     "sys" : {
588 4         "mts" : ISODate("2020-12-03T07:58:26.310Z"),
589 5         "cts" : ISODate("2020-12-03T07:58:26.310Z"),
590 6         "rev" : 0
591 7     },
592 8     "dbVersion" : 1.01,
593 9     "proddbVersion" : 1.01,
594 10    "stage_flow" : [
595 11        "MODULETOPCB",
596 12        "MODULEWIREBONDING",
597 13        "MODULEWIREBONDPROTECTION",
598 14        "MODULEPARYLENECOATING",
599 15        "MODULETHERMALCYCLING",
600 16        "MODULEBURNIN",
601 17        "MODULERECEPTION"
602 18    ],
603 19    "stage_test" : {
604 20        "MODULETOPCB" : [
605 21            "OPTICAL",
606 22            "GLUE_MODULE_FLEX_ATTACH",
607 23            "MASS",
608 24            "METROLOGY"
609 25        ],
610 26        "MODULEWIREBONDING" : [
611 27            "WIREBONDING",
612 28            "OPTICAL",
613 29            "SENSOR_IV",
614 30            "PIXEL_FAILURE_TEST",
615 31        ]
616    }
617 }
```

```

616    31      "SLDO_VI",
617    32      "WIREBOND",
618    33      "CHIP_CONFIGURATION"
619    34  ],
620    35      "MODULEWIREBONDPROTECTION" : [
621    36          "OPTICAL",
622    37          "POTTING",
623    38          "MASS",
624    39          "READOUT_IN_BASIC_ELECTRICAL_TEST",
625    40          "SENSOR_IV",
626    41          "REGISTER_TEST"
627    42      ],
628    43      ...
629    44  },
630    45  ...
631  }

```

ソースコード 4.4 モジュールの組み立て工程及び品質試験結果管理のためのドキュメント例。各モジュールにおいて現在の組み立て工程及び選択された品質試験結果がこのドキュメントに保存される。ドキュメント内の”currentStage”に現工程を保持する。また選択した試験結果の ID を”QC_results”に、組み立て工程ごとに保持する。

```

633 1  {
634 2      "_id" : ObjectId("5fc4be4c12a45922a91b0e75"),
635 3      "sys" : {
636 4          "mts" : ISODate("2020-11-30T09:41:32.411Z"),
637 5          "cts" : ISODate("2020-11-30T09:41:32.411Z"),
638 6          "rev" : 0
639 7      },
640 8      "dbVersion" : 1.01,
641 9      "proddbVersion" : 1.01,
642 10     "component" : "5fa79114e615fa000a1a5976",
643 11     "currentStage" : "MODULEWIREBONDPROTECTION",
644 12     "latestSyncedStage" : "MODULEWIREBONDING",
645 13     "status" : "created",
646 14     "rework_stage" : [],
647 15     "QC_results" : {
648 16         "MODULETOPCB" : {
649 17             "OPTICAL" : "5fc4c2cfb6c93d451e2c9ac1",
650 18             "GLUE_MODULE_FLEX_ATTACH" : "-1",
651 19             "MASS" : "5fc4c2da27766dc6e89c024f",
652 20             "METROLOGY" : "5fc4c2eaf1f19d9cb5859f00"
653 21         },
654 22         "MODULEWIREBONDING" : {
655 23             "WIREBONDING" : "-1",
656 24             "OPTICAL" : "5fc4c4c8b7d0c86912b4958f",
657 25             "SENSOR_IV" : "5fc4c59e9e283a57ccaa1088",
658 26             "PIXEL_FAILURE_TEST" : "5fca342f6e9f1f5eafedfb92",
659 27             "SLDO_VI" : "-1",
660 28             "WIREBOND" : "-1",
661 29             "CHIP_CONFIGURATION" : "-1"
662 30         },
663 31         "MODULEWIREBONDPROTECTION" : {
664 32             "OPTICAL" : "-1",
665 33             "POTTING" : "-1",
666 34             "MASS" : "-1",
667 35             "READOUT_IN_BASIC_ELECTRICAL_TEST" : "-1",
668 36             "SENSOR_IV" : "-1",
669 37             "REGISTER_TEST" : "-1"
670 38         },
671 39     ...
672 40   ...
673 41 }

```

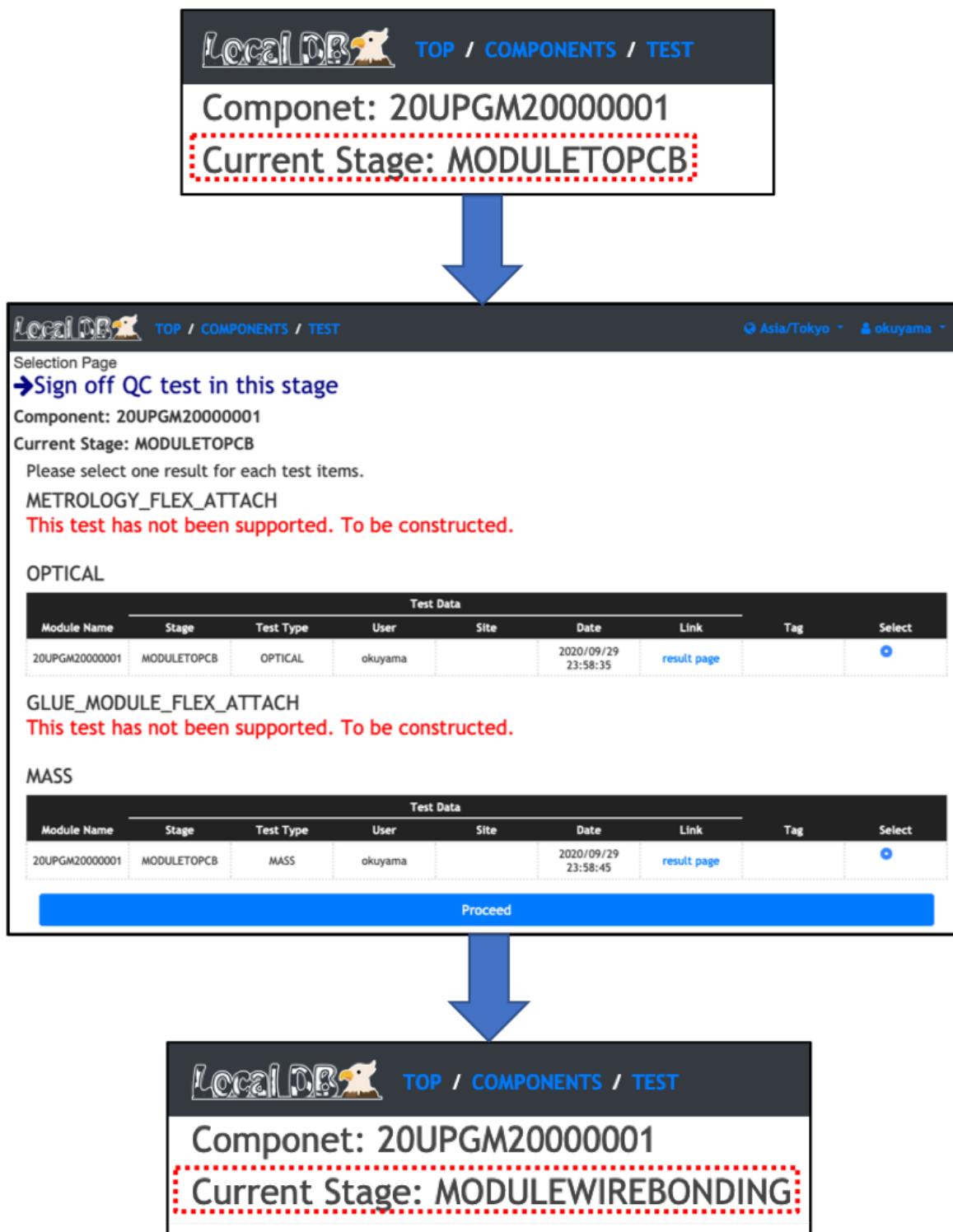


図 4.12 結果選択画面及び組み立て工程表示の例。図の上部で組み立て工程が”MODULETOPCB”である。図の中部において品質試験結果選択処理を行なっており、この図では”OPTICAL”と”MASS”の結果を選択している。この時、ローカルデータベース内部では選択された結果にタグ付けがなされる。これらの結果は中央データベースと同期される。また結果選択後は組み立て工程が自動的に更新される。図の下部では”MODULEWIREBONDING”になっていることが分かる。

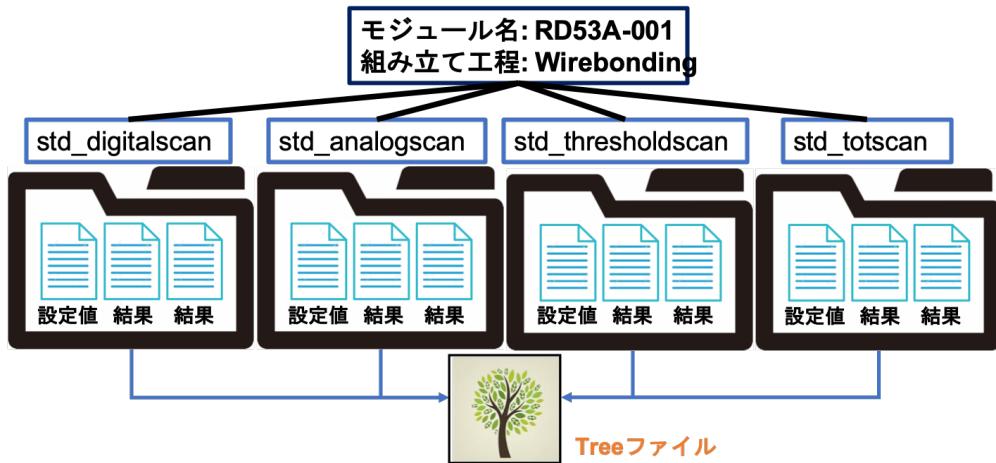


図 4.13 ピクセル解析ツールにおけるファイル統合処理のイメージ。YARR の出力ファイル及びディレクトリは std_digitalscan や std_thresholdscan というように読み出し項目ごとである。ピクセル解析ツールでは、図のようにあるモジュールに関する結果ファイルを統合し、ピクセルごとに行う解析処理を簡易化する狙いがある。

676 4.3.5 開発項目 3-(iii): 読み出し試験結果におけるピクセル解析ツール

677 節 3.2.6 で述べたように、読み出し試験ではピクセル解析を行う。これを円滑に行うために、ピクセル
678 解析ツールを開発した。また開発した解析ツールをローカルデータベースシステムに組み込んだ。この
679 ツールについての詳細を以下に示す。

680 概要

681 YARR で読み出し試験を行った場合、結果ファイル及びディレクトリは各試験項目ごとにわかれて生
682 成される。また各結果ファイルにはモジュール上の全ピクセル結果が JSON の形で保存されている。

683 一方、ピクセル解析において、いくつかの試験結果を統一的に扱い、各ピクセルごとに解析を行う必要
684 がある。そこで、開発した解析ツールでは複数の結果ファイルを 1 つに統合し、ピクセルごとの解析処理
685 を単純化する役割を担っている。開発には Python と C++ を用いた。また CERN が提供している解析
686 フレームワークである ROOT[28] を使用し、いくつかの試験データの統一ファイルとして、ROOT 内部
687 機能である Tree を使用した。このファイル統合処理のイメージを図 4.13 に示す。

688 実際に作った Tree ファイルと、データ構造のイメージを図 4.14 に示す。

689 ツールの内部構造と処理の流れ

690 開発したツールは、主に以下で説明する 3 つの実行ファイルで構成される。それぞれの役割について説
691 明する。

692 getData.py (Python)

693 データベースから対象となるデータファイルを取得、キャッシュファイルとしてサーバー上の一時
694 ディレクトリに保存。

695 makeTree (C++)

696 getData.py を用いて生成されたキャッシュファイルを読み込み、Tree ファイルを作成。



図 4.14 Tree ファイルとそのデータ保持のイメージ。実際にこのツールを用いて作った Tree ファイルの内部構造の様子(左)とそのデータ保持のイメージ(右図)を示す。Tree ファイルでは、右図のように 1 つの表に試験結果をまとめている。各行が std_digitalscan といった各読み出し項目に対応し、各列が 1 ピクセルに対応する。モジュール上の行列 (Row, Col) の番号を表の上部に持っておくことで、モジュール上におけるピクセルの位置情報を保持する。

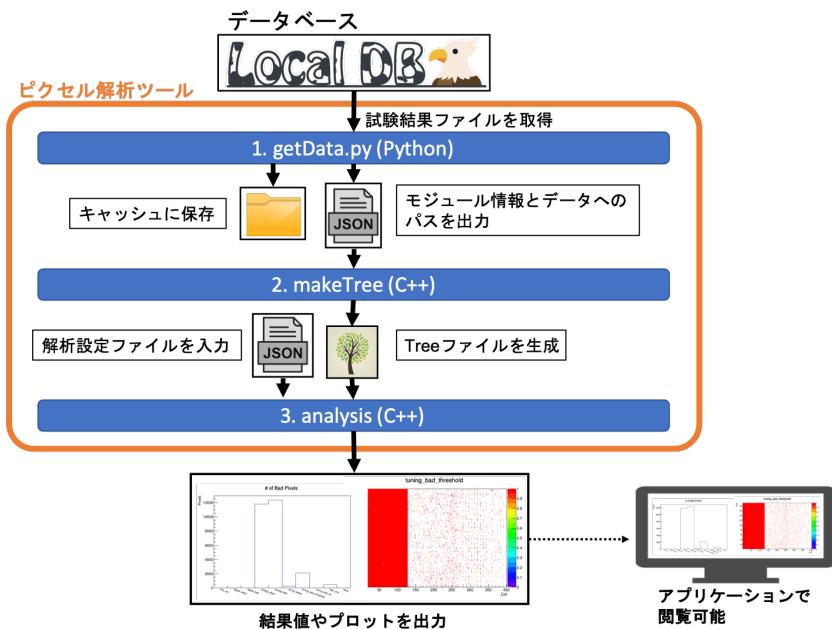


図 4.15 ピクセル解析ツールの処理の流れ。ピクセル解析ツールは、図のように 3 つの実行ファイルにより構成される。getData.py(Python) にてデータベースから結果の取得を行い、makeTree(C++) にて Tree ファイルの作成、analysis(C++) にてピクセル解析及び結果のプロットが出力される。

697 analysis (C++)

698 作成した Tree ファイルを読み込みピクセル解析を実行、結果値やプロットを出力。

700 701 702 処理の流れのイメージを図 4.15 に示す。データベースとの通信に関しては MongoDB や現システムとの親和性を考慮し、Python を使用した。Tree ファイル作成やその後の解析処理のスクリプトは、ROOT を使用する観点から C++ を使用した。またピクセル解析以外の解析に対しても適応可能とするため、Tree 作成部と解析処理部のファイルは分割した。

The screenshot shows a web-based database interface titled "LocalDB" with a "TEST" component selected. At the top, there is a search bar containing the text "QU-13" and several search mode options: "Partial match" (selected), "Perfect match", and "Search". Below the search bar is a navigation menu with links from "TOP / COMPONENTS / TEST". The main area is titled "Scan List" and contains a table of search results. The table has columns: Module Name, Chip Name, Test Type, User, Site, Date, Link, and Tag. The "Module Name" column shows repeated entries for "QU-13". The "Test Type" column includes "thresholdscan", "digtalscan", "selftrigger", and "totscan". The "User" column consistently lists "yuta_miyazaki". The "Site" column lists "kyushu_universit y". The "Date" column shows various dates in June 2019. The "Link" column contains blue hyperlinks labeled "result page". The "Tag" column is empty. Red arrows point from the text "入力欄" to the search input field and from the text "検索結果一覧表" to the table.

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	thresholdscan	yuta_miyazaki	kyushu_universit y	2019/06/11 19:23:18	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	thresholdscan	yuta_miyazaki	kyushu_universit y	2019/06/11 19:17:33	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	thresholdscan	yuta_miyazaki	kyushu_universit y	2019/06/11 19:12:03	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	digtalscan	yuta_miyazaki	kyushu_universit y	2019/06/11 19:05:12	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	selftrigger	yuta_miyazaki	kyushu_universit y	2019/06/11 18:33:26	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	digtalscan	yuta_miyazaki	kyushu_universit y	2019/06/11 17:57:48	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	selftrigger	yuta_miyazaki	kyushu_universit y	2019/06/11 17:55:46	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	digtalscan	yuta_miyazaki	kyushu_universit y	2019/06/11 17:19:38	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	selftrigger	yuta_miyazaki	kyushu_universit y	2019/06/11 16:09:06	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	digtalscan	yuta_miyazaki	kyushu_universit y	2019/06/11 15:33:23	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	selftrigger	yuta_miyazaki	kyushu_universit y	2019/06/11 15:32:52	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	digtalscan	yuta_miyazaki	kyushu_universit y	2019/06/11 14:57:39	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	selftrigger	yuta_miyazaki	kyushu_universit y	2019/06/11 14:57:32	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	thresholdscan	yuta_miyazaki	kyushu_universit y	2019/06/11 14:45:32	result page	
QU-13	QU-13_chipd1 QU-13_chipd2 QU-13_chipd3 QU-13_chipd4	totscan	yuta_miyazaki	kyushu_universit y	2019/06/11 14:44:16	result page	

図 4.16 ウェブアプリケーションにおける検索機能の様子。図は検索結果一覧表示のページである。図の上部に入力欄があり（赤破線）、ここにキーワードを入力し検索を実行する。図の例で“QU-13”と入力しており、検索結果にはモジュール名「QU-13」の試験結果が一覧表示されていることが分かる。

4.3.6 開発項目 3-(iv): 読み出し試験結果の検索機能

登録モジュールや品質試験結果の一覧ページに検索機能を実装した。確認したいモジュール情報や試験結果を迅速に取得し、閲覧することを目的としている。検索機能を使用している様子を図 4.16 に示す。キーワードを入力し、検索することができる仕組みとなっていて、一般的なウェブページの検索エンジンのように扱うことができる。生産に向けて、検索にかかる処理時間測定を行った。検索機能実装方法の詳細と処理時間についての詳細は、6 章で述べる。現在は单一キーワード検索の他に、以下の機能を実装している。

- 完全一致、部分一致検索。
- AND、OR 検索。

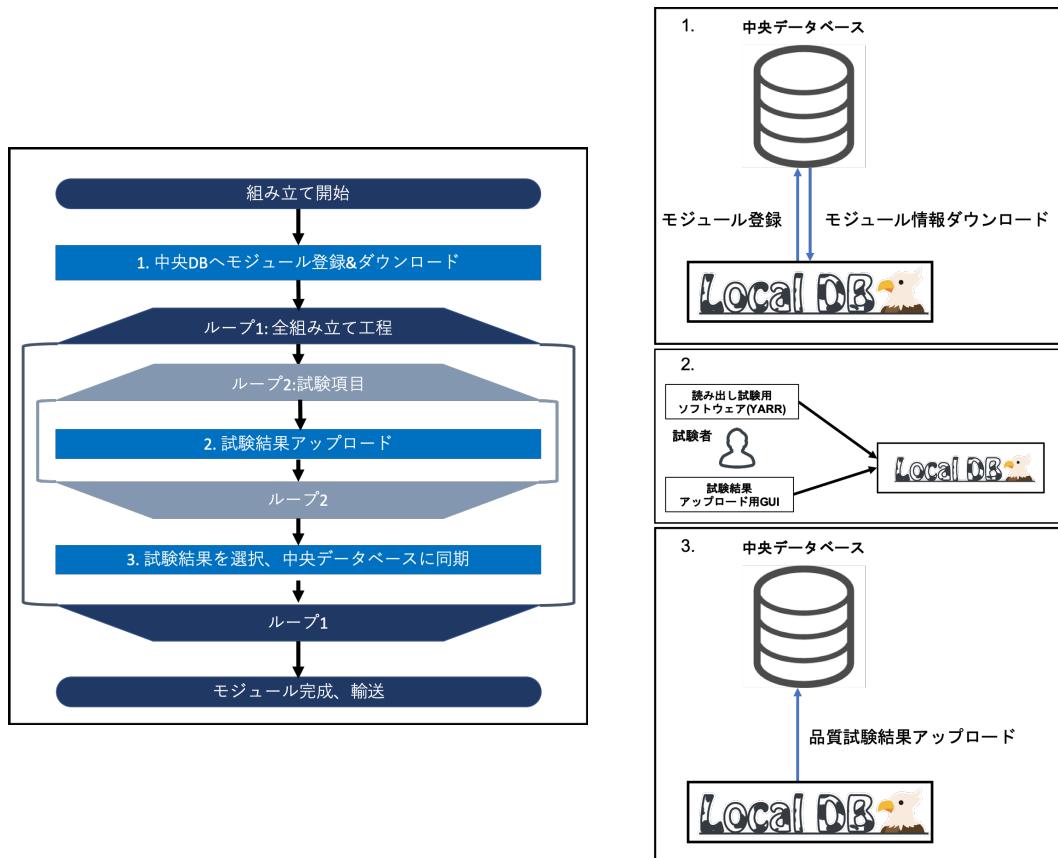


図 4.17 各モジュールにおけるデータベース操作の流れ。モジュール組み立てにおけるデータベース操作の初めに、中央データベースにモジュール登録及びローカルデータベースへモジュール情報のダウンロードを行う（処理 1）。その後、ダウンロードしたモジュールに対して組み立て工程に応じた試験結果を生成、ローカルデータベースに保存する（処理 2）。各組み立て工程の終わりに試験結果の選択を行い、中央データベースに試験結果を同期する（処理 3）。

712 4.3.7 開発項目 4: 量産時におけるデータベース操作の流れの確立

713 量産時におけるデータベース操作の流れを確立した。以下に従い、モジュール組み立て時におけるデータ
714 管理がなされる。

- 715 1. 中央データベースへモジュール登録及び登録情報のダウンロード。
716 2. 1で登録したモジュールに対して、品質試験結果をローカルデータベースへアップロード。
717 3. 組み立て工程毎に品質試験結果の選択と中央データベースへアップロード。

718 流れのイメージを図 4.17 に示す。品質試験結果のアップロードは各組み立て工程毎に行う。ローカル
719 データベースで品質試験結果を組み立て工程毎にまとめて扱い、各モジュールの現組み立て工程を正確に
720 管理する目的がある。

721 全組み立て工程が終了すると、モジュールの情報及び品質試験結果が全て中央データベースへ同期され
722 ている状態となる。

723 この品質試験におけるデータベース操作の流れの確立に向けて、2月に CERN に行なったシステム
724 チュートリアルでこの一連の流れを扱った。ユーザに対する操作方法の周知と機能確認を行うことができ
725 た。チュートリアルの概要は付録 C に記す。またシステムのドキュメント [29] を作成し、その中に具体

726 的なデータベース操作の流れを記述している。量産時に各機関が適切な流れでデータ管理ができるように
727 した。

728 データベース操作の流れの試験として、実際に品質試験のデモンストレーションを行い、各機能が正常
729 に動作するのかの確認を行なった。詳細を5章で述べる。

730 第5章

731 品質試験のデモンストレーション

732 品質試験のデモンストレーションを行った。今回のデモンストレーションでは、品質試験項目として読
 733 み出し試験を行い、データベース機能や試験の流れを確認した。この章の前半では試験で使用するソフト
 734 ウェア、ハードウェアについて説明し、後にデモンストレーションの内容、機能確認について述べる。

735 5.1 用いたソフトウェア

736 試験で用いたソフトウェアをいかに示す。また、これらソフトウェアの概要を図 5.1 に示す。

- 737 • YARR(commit:6b3ffe92)
- 738 • MongoDB(version: v4.2.6)
- 739 • ローカルデータベース用ウェブアプリケーション (tag: ldbtoolv1.4)
- 740 • 中央データベースとの同期ツール (tag: ldbtoolv1.4)
- 741 • ピクセル解析ツール (tag: v1.0.2)
- 742 • 時系列データ用データベース (InfluxDB[30](version: 1.8.0))
 - 743 – 時系列情報に特化したデータベース。このシステムにおいては温度、電圧など DCS 情報を時
 744 間情報と共に保存、管理するために用いる。
- 745 • InfluxDB 解析ソフト (Grafana[31](version: 5.1.0))
 - 746 – InfluxDB に保存された情報の解析、閲覧に用いる。ウェブブラウザー上で DCS データを閲
 747 覧することができる。
- 748 • 電源操作用ソフト
 - 749 – 電源を遠隔で操作し、モジュールに電圧を供給する。また電圧、電流値を取得し、InfluxDB
 750 にアップロードする。CERN で開発されている PySerial[32](commit:0d14fcdb) を改良し作
 751 成した。
- 752 • 温度読み出し用ソフト
 - 753 – GPIO 通信により取得できる ADC 値を取得、温度に変換し InfluxDB へアップロードする処
 754 理を作成したもの。RaspberryPi 上に保存し、処理を実行。

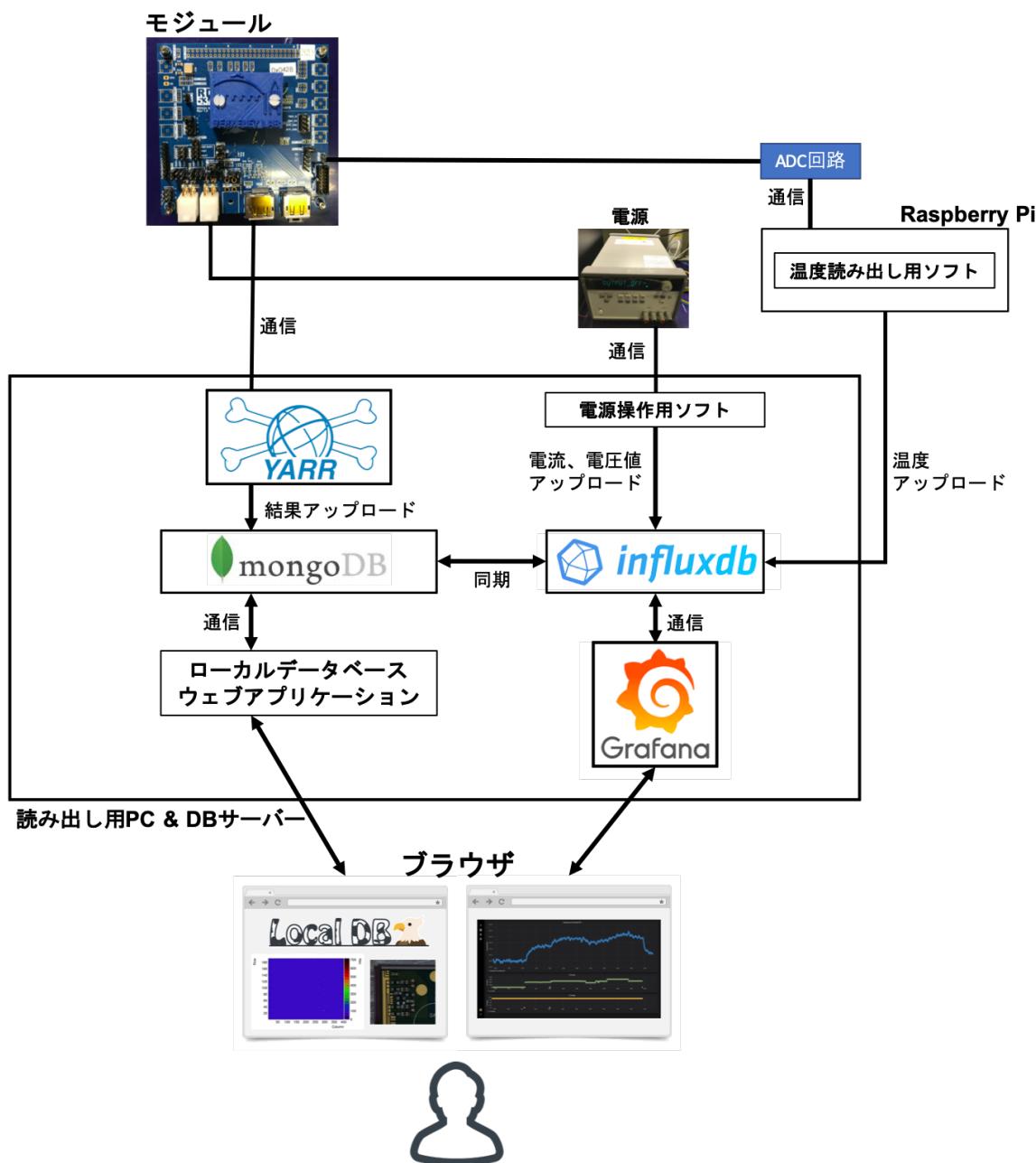


図 5.1 読み出し試験に用いるソフトウェアの概要。FE チップの読み出しとそのデータ通信は YARR を用いて行われる。試験結果は MongoDB にアップロードされ、試験者はウェブアプリケーションを通じて結果を確認することができる。電源操作用ソフトを用いて電源のスイッチ、電圧、電流値の取得がなされ、取得値は InfluxDB に保存される。モジュール付属のサーミスタ読み出しシステムを用いて FE チップ付近の温度を読み出し、InfluxDB に保存する。InfluxDB に保存された DCS データは Grafana を用いてブラウザ上で確認することができる。また読み出し試験に関わる DCS 情報は MongoDB に同期されるため、ローカルデータベースアプリケーションを通して確認ができる。

755 5.2 用いたハードウェア

756 読み出し試験に用いたハードウェアについて、以下に詳細を記す。

757 5.2.1 各装置

758 RD53A シングルモジュール (RD53A Single Chip Card, SCC)[33]

759 今回読み出しに使うモジュールとして、研究室で所有している RD53A シングルモジュール (**RD53A**
760 **Single Chip Card, SCC**) を使用した。SCC は試験用に作られた FE チップを一枚搭載するモジュー
761 ルである。また今回使用したものはシリコンセンサーを持たない。SCC は FE チップ電源端子、データ
762 転送端子をもち、読み出しを行う際はそれぞれ配線をする。FE チップ付近には NTC サーミスタを搭載
763 していて、ボード上の端子からその抵抗値を取得することで温度を測定することができる。図 5.2 に写真
764 を示す。

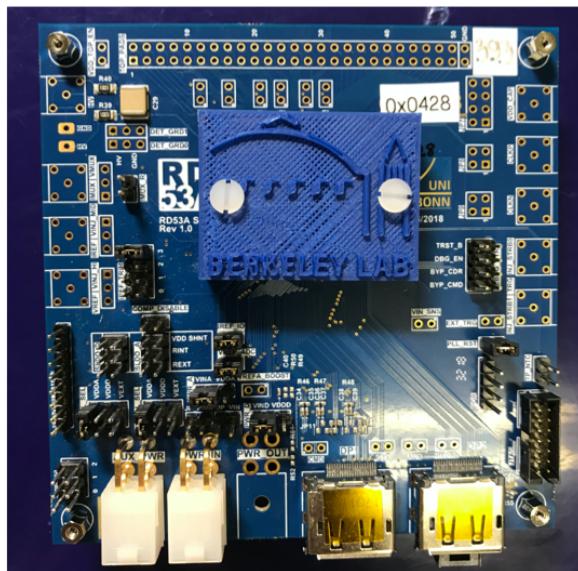


図 5.2 RD53A シングルモジュール (SCC)[33]。RD53A の FE チップを一枚搭載したモジュールである。図の中心部の青いカバーで囲われた中に FE チップが設置されている。このボード上にはデータ転送端子、電源端子、サーミスタ抵抗取得端子が設置されており、これらの端子に対応する配線をすることで読み出し試験のセットアップを組む。

765 モジュールサーミスタ温度読み出しシステム

766 モジュール付属サーミスタの抵抗値を取得し温度を測定するために、ADC、Raspberry Pi を用いた温
767 度読み出しシステムを作成した。このシステムの中で扱った装置を表 5.1 に、回路図、実際に配線した様
768 子を図 5.3 に示す。サーミスタの抵抗値に対応する ADC 値を温度に変換することで読み出しを行ってい
769 る。ADC 値は GPIO 通信 [34] により取得している。

770 電源

771 モジュールの FE チップに対する電圧供給に KEYSIGHT の E3646A 60W デュアル出力電源 [37](図
772 5.4) を用いた。

表 5.1 溫度読み出しシステムに使用した装置一覧。

装置	機種
10kΩ 抵抗	-
ADC	MCP3002[35]
RaspberryPi	Raspberry Pi 3 Model B Plus Rev 1.3[36]

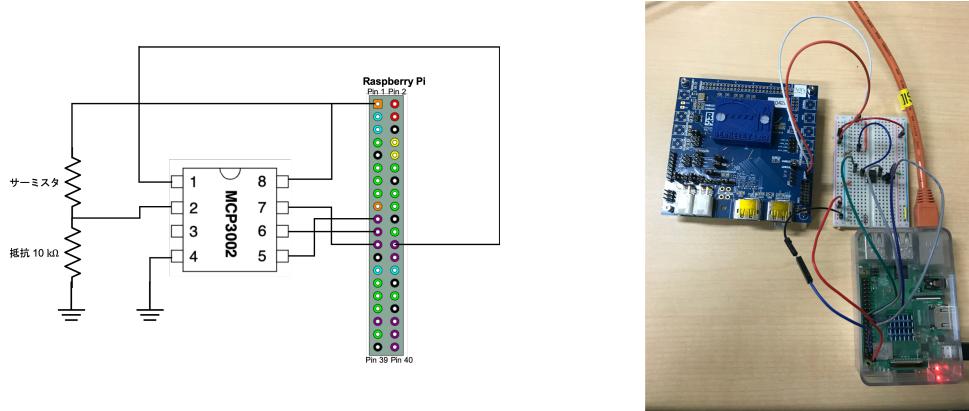


図 5.3 モジュール付属サーミスタを用いた温度読み出し回路。図は回路図(左図)と実際に配線し、読み出しを行っている様子(右図)を示している。抵抗、MCP3002、Raspberry Pi を用いて読み出し回路を作成した。抵抗と MCP3002 は右図のようにブレッドボード上に設置した。ADC と RaspberryPi は GPIO 通信 [34] を行うことで、ADC 値を取得している。



図 5.4 用いた電源。FE チップ電圧供給のための電源として KEYSIGHT の E3646A 60W デュアル出力電源 [37] を用いた。

773 FPGA ボード

774 FPGA ボードに XpressK7[38] を用いた。YARR ファームウェアを FEGA 上にプログラムし通信を行なった。XpressK7 を図 5.5 に示す。

776 FMC-DisplayPort 変換カード

777 FPGA ボードには FMC 端子がついていて、これをモジュールを接続するためには FMC-DisplayPort
778 変換カードが必要となる。使用した変換カード(オハイオカード)を図 5.6 に示す。



図 5.5 使用した FPGA ボード (XpressK7[38])。中央に FMC 端子、右側に FPGA チップ、下側に PCI Express が配置されている。FPGA チップの上にはファンをついているため、この図では確認できない。

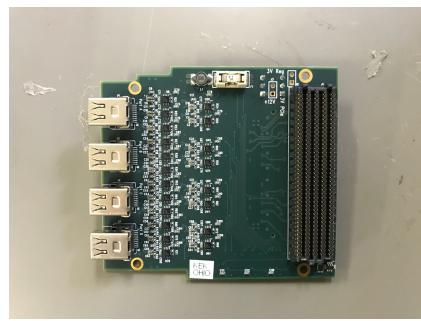


図 5.6 使用した FMC-DisplayPort 変換カード (オハイオカード)。図の右側に FMC 端子、左側に miniDisplayPort が 4 つ配置されており、FMC 端子を 4 つの miniDisplayPort に変換する。今回読み出す FE チップは 1 枚であるため、1 つの端子だけを用いる。

表 5.2 読み出しに使用した PC の性能。研究室で所有する PC を使用した。OS は centOS7 である。

CPU	Core	Thread	Clock speed[GHz]	Memory [GB]	Disk [GB]
Type					
Intel(R) Core(TM) i7-8700K CPU	6	12	3.7	16.18	214

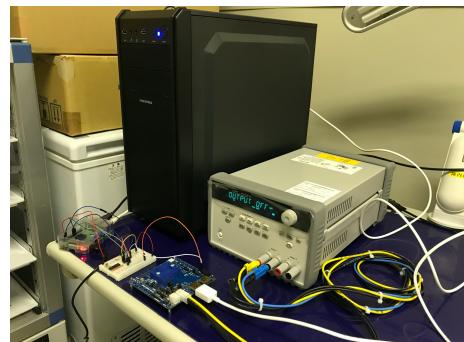
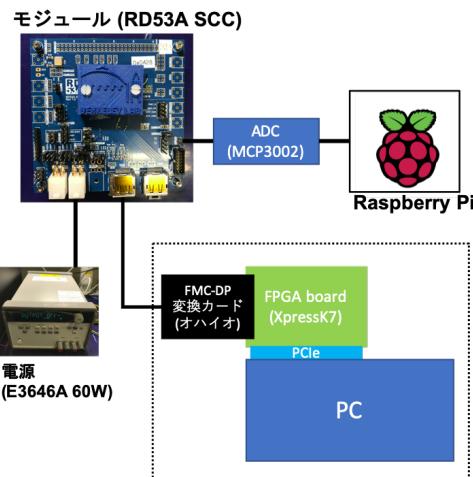


図 5.7 ハードウェアセットアップ。図はセットアップの概要 (左図) と実際に設置した様子 (右図) を示している。FE チップ読み出し装置、電源、サーミスタ読み出し装置をそれぞれ設置、配線し、読み出し操作、データ取得を行った。

779 PC

780 YARR ソフトウェアをインストールし、読み出しを行なった PC の性能を表 5.2 に示す。

781 5.2.2 セットアップ

782 読み出し試験に用いるハードウェアのセットアップを概要を図 5.7 に示す。

783 5.3 デモンストレーションの流れ

784 デモンストレーションで用いるため、中央データベースにおける登録 ID の定義方法 [39] に従い以下の
785 モジュール、FE チップを登録した。

786 モジュール ID 20UPGR00000001

787 FE チップ ID 20UPGFC9999999

788 今回のデモンストレーションではこれらの ID を用いて試験結果の紐付け等のデータベース操作を行う。
789 確認した機能を行った流れの順に以下に示す。

- 790 1. 中央データベースからモジュール情報のダウンロード.
- 791 2. 読み出し試験実施、結果をローカルデータベースに保存.
- 792 3. DCS 情報の取得、監視.
- 793 4. 試験結果検索.
- 794 5. 試験結果閲覧.
- 795 6. 結果選択とピクセル解析機能.
- 796 7. 中央データベースへ試験結果のアップロード

Latest Result								
Module Name	Chip Name	Current Stage	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC999999	MODULEWIREBONDING	PIXEL_FAILURE_TEST	hokuyama		2020/12/05 07:05:51	result page	Tag List

図 5.8 ダウンロードしたモジュール ID 確認画面。図の表において、今回登録した 20UPGR00000001 の ID を持つモジュールがローカルデータベースのアプリケーション上で確認できていることが分かる。また対応する FE チップの ID も確認することができる。

5.4 機能確認

読み出し試験を通して、各ソフトウェア機能が正しく動くことを確認した。詳細を以下に記す。

5.4.1 中央データベースからモジュール情報のダウンロード

中央データベースから登録したモジュール情報をダウンロードし、ウェブアプリケーションで確認した。確認した画面を図 5.8 に示す。今回行った試験結果はこのモジュールに紐つける形でローカルデータベースに保存される。

5.4.2 読み出し試験実施

以下の流れに沿って読み出しを行ない、結果をローカルデータベースに保存した。

1. デジタル回路読み出し (`std_digitalscan`).
2. アナログ回路読み出し (`std_analogscan`).
3. 調整前 Threshold 測定.
4. Threshold 調整.
5. ToT 調整.
6. Threshold 再調整.
7. 調整後 Threshold 測定 (`std_thresholdscan`).
8. ToT 測定 (`std_totscan`).
9. ノイズ測定 (`std_noisescan`).

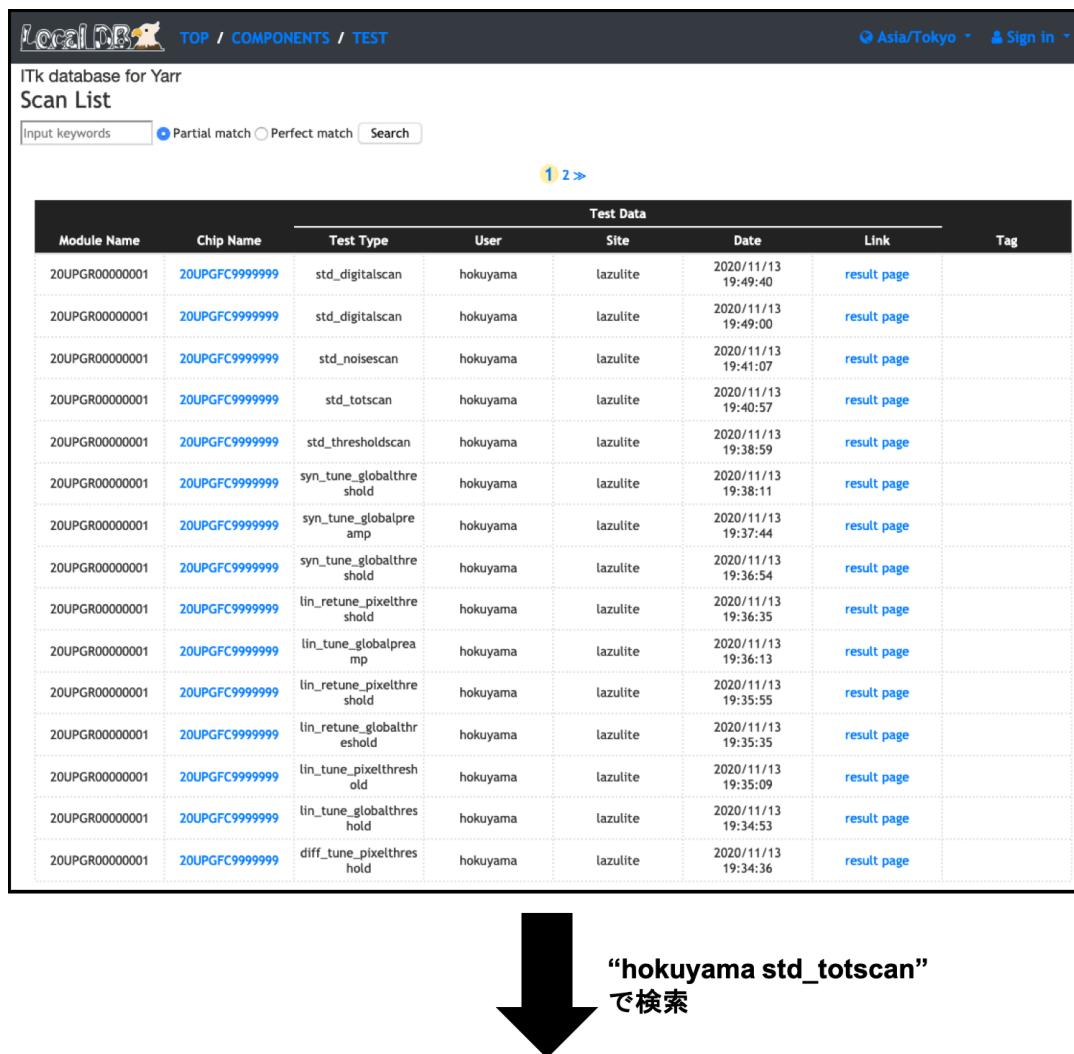
また読み出し試験を通して DCS 情報は InfluxDB を用いて監視し、試験結果と同様にローカルデータベースに保存した。



図 5.9 DCS 情報のモニタリングの様子。図において横軸は時間であり、縦軸は上から温度(青)、電流(緑)、電圧(黄)を示す。それぞれデータの監視が行えていることが分かる。温度に関して、電源オン、オフの付近で変化が大きいことが分かる。電流に関して、読み出し(チューニング)を行っている途中にも微小変化していることが分かる。

5.4.3 DCS 情報の監視

読み出し試験は、DCS 情報を監視しながら行った。それぞれの値は対応するソフトウェアを用いて InfluxDB にアップロードした。その値を Grafana を使って監視をした。その様子を図 5.9 に示す。

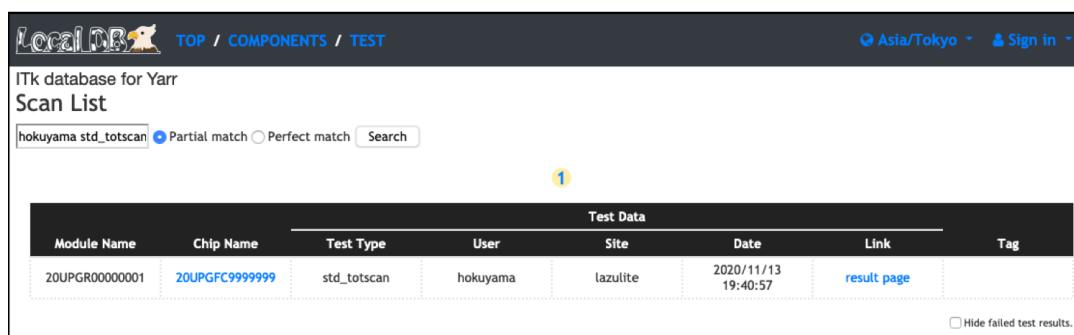


The screenshot shows the LocalDB Scan List page. At the top, there are navigation links: TOP / COMPONENTS / TEST, a location indicator (Asia/Tokyo), and a sign-in button. Below this is a search bar with the placeholder "Input keywords" and options for "Partial match" or "Perfect match". A "Search" button is also present. The main area displays a table titled "Test Data" with columns: Module Name, Chip Name, Test Type, User, Site, Date, Link, and Tag. The table contains 16 rows of data, each representing a different test run. A large black arrow points downwards from the top table to the bottom table.

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC999999	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:40	result page	
20UPGR00000001	20UPGFC999999	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:00	result page	
20UPGR00000001	20UPGFC999999	std_noisescan	hokuyama	lazulite	2020/11/13 19:41:07	result page	
20UPGR00000001	20UPGFC999999	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page	
20UPGR00000001	20UPGFC999999	std_thresholdscan	hokuyama	lazulite	2020/11/13 19:38:59	result page	
20UPGR00000001	20UPGFC999999	syn_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:38:11	result page	
20UPGR00000001	20UPGFC999999	syn_tune_globalpreamp	hokuyama	lazulite	2020/11/13 19:37:44	result page	
20UPGR00000001	20UPGFC999999	syn_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:36:54	result page	
20UPGR00000001	20UPGFC999999	lin_retune_pixelthreshold	hokuyama	lazulite	2020/11/13 19:36:35	result page	
20UPGR00000001	20UPGFC999999	lin_tune_globalpreamp	hokuyama	lazulite	2020/11/13 19:36:13	result page	
20UPGR00000001	20UPGFC999999	lin_retune_pixelthreshold	hokuyama	lazulite	2020/11/13 19:35:55	result page	
20UPGR00000001	20UPGFC999999	lin_retune_globalthreshold	hokuyama	lazulite	2020/11/13 19:35:35	result page	
20UPGR00000001	20UPGFC999999	lin_tune_pixelthreshold	hokuyama	lazulite	2020/11/13 19:35:09	result page	
20UPGR00000001	20UPGFC999999	lin_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:34:53	result page	
20UPGR00000001	20UPGFC999999	diff_tune_pixelthreshold	hokuyama	lazulite	2020/11/13 19:34:36	result page	

↓

**“hokuyama std_totscan”
で検索**



The screenshot shows the LocalDB Scan List page after a search. The search bar now contains the query "hokuyama std_totscan". The search results table shows a single row of data, which corresponds to the 5th row in the table above. The "Link" column for this row contains a link labeled "result page".

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC999999	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page	

図 5.10 検索機能確認の様子。図は検索実行前の試験結果一覧（上図）と実行後（下図）を示す。図の例で“hokuyama std_totscan”的2つのキーワードで検索を行っており、実行後は対応する試験結果1つが表示されていることが分かる。この試験結果について試験実施者は“hokuyama”、試験項目は“std_totscan”であるため、検索機能が正常に動いていることが分かる。

5.4.4 検索機能

819 検索機能の確認を行い、正常に使用できることを確認した。検索機能実行の様子を図 5.10 に示す。

820 検索機能の確認を行い、正常に使用できることを確認した。検索機能実行の様子を図 5.10 に示す。

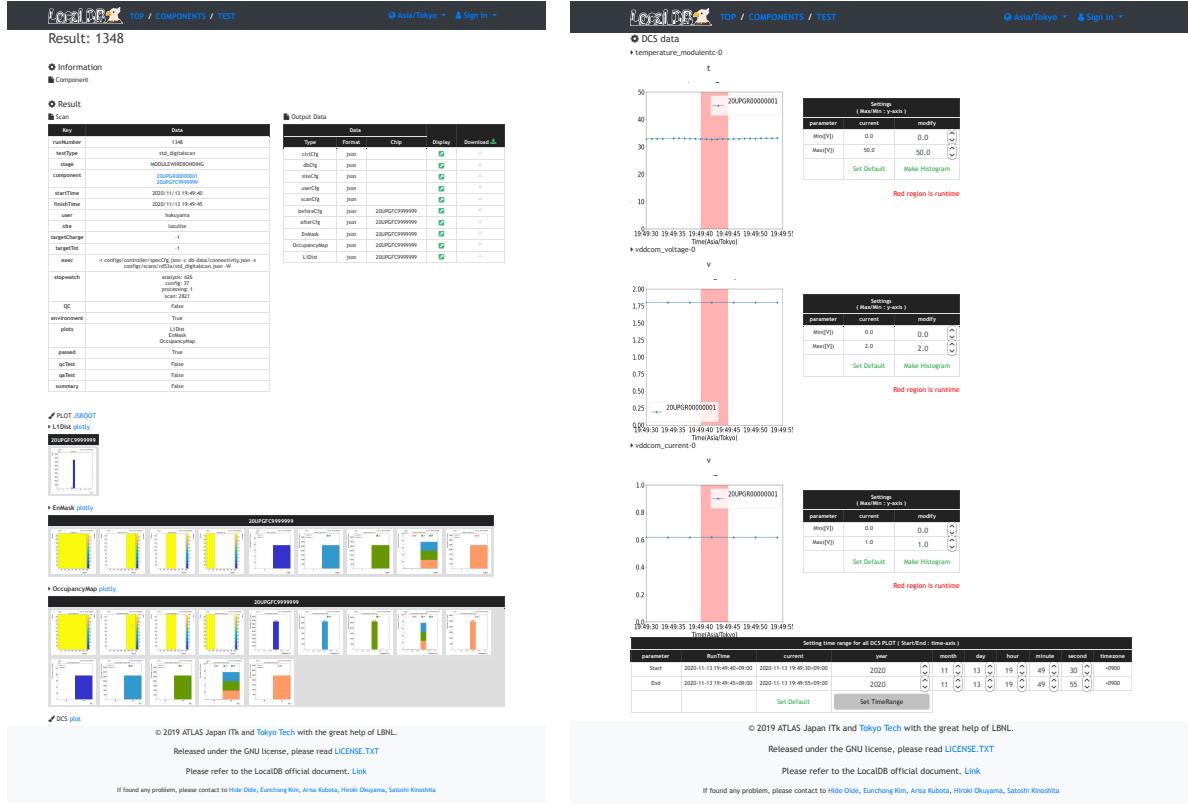


図 5.11 試験結果の閲覧。図は試験結果（左図）と試験におけるDCSデータのグラフ（右図）を示しており、上から温度、電圧、電流となっている。図はstd_digitalscanの結果であり、試験情報及び結果のグラフが確認できる。また右図よりDCSデータも正常にローカルデータベース上に保存され、表示されていることが分かる。

5.4.5 試験結果閲覧

ウェブアプリケーションを用いて、試験結果を閲覧した。その様子を図 5.11 に示す。

823 5.4.6 結果選択とピクセル解析

824 読み出し試験結果を選択し、ピクセル解析を行なった。結果選択画面を図 5.12 に示す。このデモンス
825 トレーションにおける不良評価基準は 3 章に述べた表 3.1 の中から、現時点でシステムに実装している以
826 下の項目を抜粋した。

- 827 1. Digital Dead.
- 828 2. Digital Bad.
- 829 3. Analog Dead.
- 830 4. Analog Bad.
- 831 5. Tuning Failed.
- 832 6. Tuning Bad for Threshold.
- 833 7. Tuning Bad for ToT.
- 834 8. High ENC.
- 835 9. Noisy.

836 解析結果を図 5.13、それぞれの評価基準における不良ピクセルの分布を図 5.14 に示す。

837 5.4.7 試験結果アップロード

838 読み出し試験の結果を中央データベースにアップロードし、情報が正しくアップロードされていること
839 を確認した。図 5.15 に中央データベースのウェブページを示す。結果ファイルや解析結果が正しくアッ
840 プロードされていることが分かる。

Selection Page

→Select scan results for QC result in this stage.

Component: 20UPGR00000001

Current Stage: MODULEWIREBONDING

Please select one result for each test items.

►std_digitalscan

Test Data										
run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1392	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	tokyo_institut e_of_technology	2020/12/08 22:35:47	result page		<input type="radio"/>
1391	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	tokyo_institut e_of_technology	2020/12/08 22:29:39	result page		<input type="radio"/>
1351	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	tokyo_institut e_of_technology	2020/12/03 14:54:43	result page		<input type="radio"/>
1350	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	tokyo_institut e_of_technology	2020/11/30 19:57:19	result page		<input type="radio"/>
1348	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:40	result page		<input type="radio"/>
1347	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:00	result page		<input type="radio"/>
1328	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:31:02	result page		<input checked="" type="radio"/>
1327	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:29:24	result page		<input type="radio"/>
1326	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:28:38	result page		<input type="radio"/>

►std_analogscan

Test Data										
run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1329	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_analogscan	hokuyama	lazulite	2020/11/13 19:31:13	result page		<input checked="" type="radio"/>

►std_thresholdscan

Test Data										
run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1344	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_threshold scan	hokuyama	lazulite	2020/11/13 19:38:59	result page		<input checked="" type="radio"/>
1330	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_threshold scan	hokuyama	lazulite	2020/11/13 19:31:24	result page		<input type="radio"/>

►std_totscan

Test Data										
run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1345	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page		<input checked="" type="radio"/>

►std_noisescan

Test Data										
run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1346	20UPGR00000001	20UPGFC999999	MODULEWIRE BONDING	std_noisescan	hokuyama	lazulite	2020/11/13 19:41:07	result page		<input checked="" type="radio"/>

Proceed

[Back to the component page](#)

© 2019 ATLAS Japan ITk and [Tokyo Tech](#) with the great help of LBNL.

Released under the GNU license, please read [LICENSE.TXT](#)

Please refer to the LocalDB official document. [Link](#)

If found any problem, please contact to [Hide Oide](#), [Eunchong Kim](#), [Arisa Kubota](#), [Hiroki Okuyama](#), [Satoshi Kinoshita](#)

図 5.12 読み出し試験結果の選択。図は読み出し試験結果選択画面を表す。読み出し試験実施後、ピクセル解析と中央データベースとの同期を実行するために試験結果を選択する必要がある。図では std_digitalscan、std_analogscan、std_thresholdscan、std_totscan、std_noisescan の 5 項目を選択している。

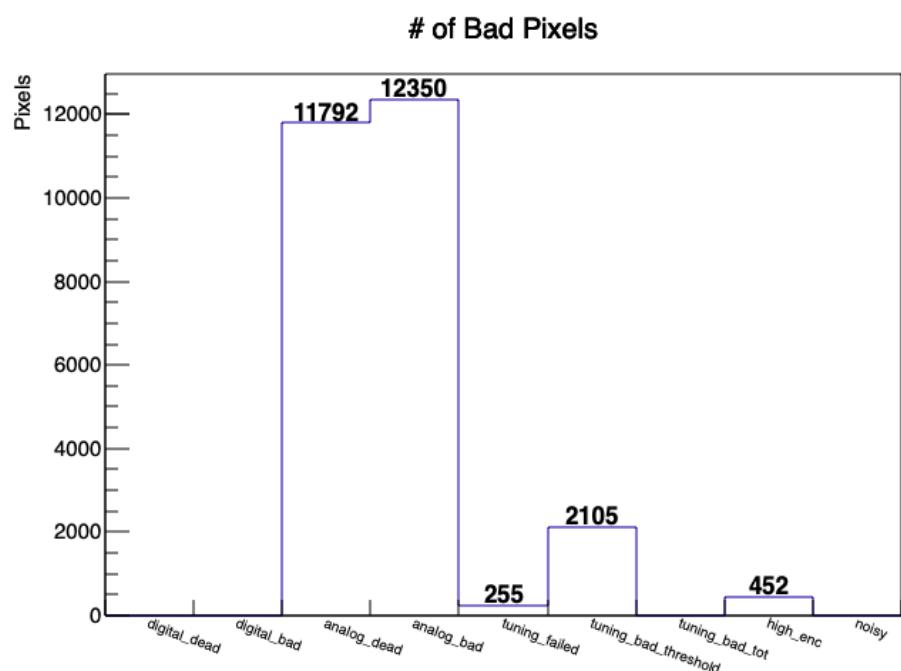


図 5.13 ピクセル解析結果。図において横軸は評価基準、縦軸は該当するピクセル数を表す。
analog_dead、analog_bad の割合が多いいため、アナログ回路読み出しに失敗しているピクセルが多いことが読み取れる。

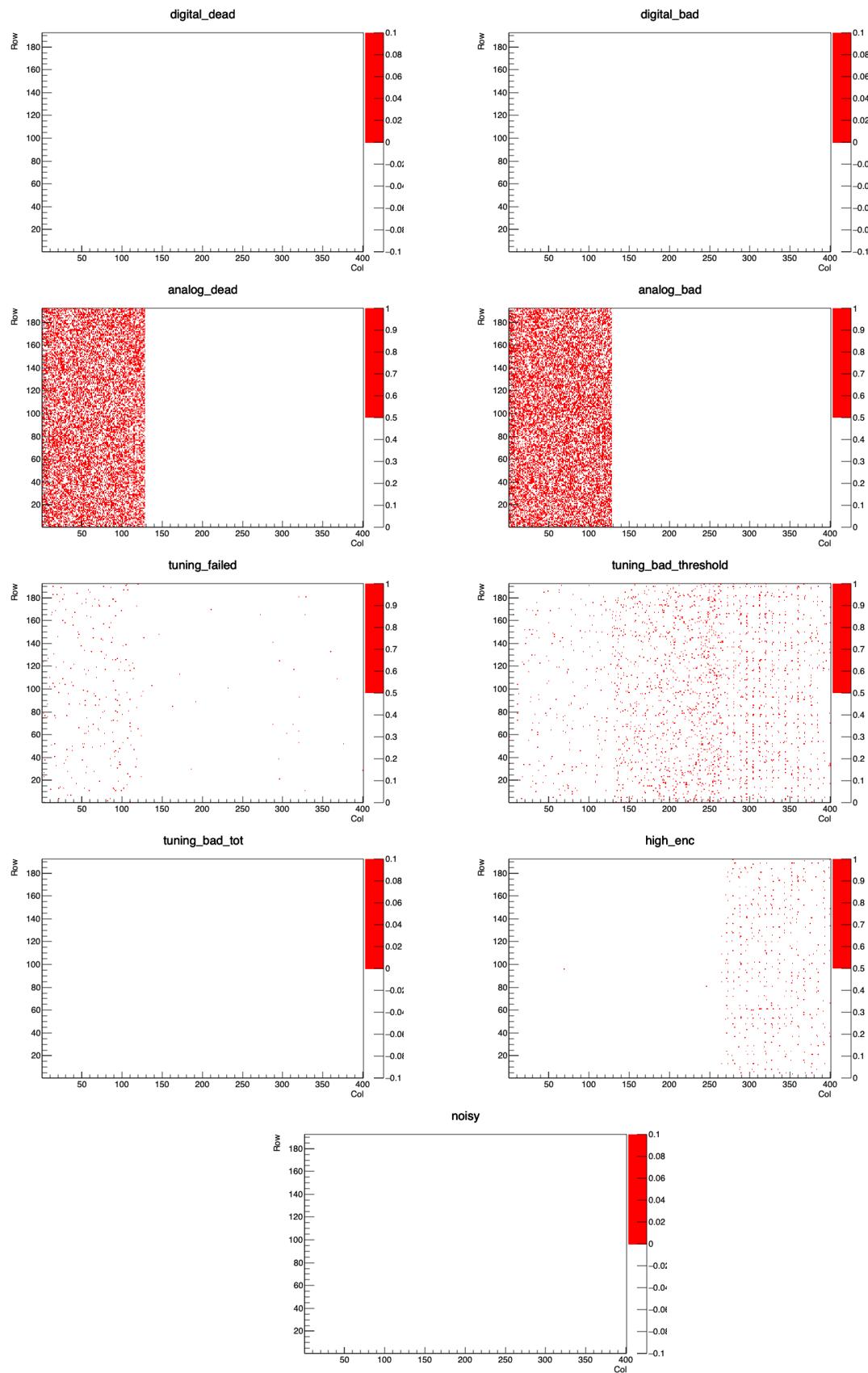


図 5.14 各評価基準における不良ピクセルの分布。各図は二次元ヒストグラムであり、横軸が FE チップにおける各ピクセルの列番号、縦軸が行番号を示しており、赤い領域が不良ピクセルを表している。図 5.13 においてアナログ回路読み出しに失敗しているピクセルが多いことがわかったが、その不良ピクセルは synchronous フロントエンド上に存在していることが分かる。また Threshold 調整に失敗しているピクセル (tuning_bad_threshold) は全体的に分布していることが分かる。

The screenshot shows the 'Test Run Details' page for a test run named 'ModuleQC::PixelFailureTest'. The page has a header with the database name and a user profile for Hiroki Okuyama. The main content area is titled 'ModuleQC::PixelFailureTest' and contains sections for 'Basic Properties', 'Associated Components', 'Properties', 'Results', 'Defects', 'Attachments', and 'Comments'. The 'Results' section lists various failure types with their counts: Noisy (0), High ENC (452), Tuning Bad ToT (0), Tuning Bad Threshold (2105), Tuning Failed (255), Analog Bad (12350), Analog Dead (11792), Digital Bad (0), and Digital Dead (0). The 'Attachments' section shows several zip files uploaded for different scan types. The bottom of the page includes a copyright notice.

図 5.15 中央データベースにおける読み出し試験及びピクセル解析結果画面。読み出し試験及びピクセル解析結果を中央データベースにアップロードした。図は中央データベースのウェブページを示しており、アップロードされた試験結果画面である。図の中央部に各評価基準における不良ピクセルの数、下部に各試験項目ごとの結果ファイルをまとめた Zip ファイルが表示されていることが分かる。試験結果のアップロードが正常に完了し、中央データベース上で結果を確認できることが分かる。

841 第6章

842 ローカルデータベースにおける検索機能 843 とその処理時間

844 モジュール組み立てにおいて、読み出し試験は複数回行われ、一回の試験で行う項目も複数存在する。
 845 そのため、生産時には試験結果が数多くデータベースにアップロードされることになる。4章で述べたよ
 846 うに、任意のタイミングで必要な結果を取得するために検索機能を実装した。詳細について以下に示す。

847 6.1 実装方法

848 一般的にウェブで用いられているフリーワードの検索エンジンのような機能を実装しようと考えた。
 849 ユーザの操作を最小限にし、直感的かつ柔軟な検索ができるようにするためである。
 850 読み出し試験において、対象とする検索キーワードを以下の項目に絞った。システムにおいて、アップ
 851 ロードされた試験結果に関わるデータベース内の情報は後から編集する機能はサポートしない方針を取っ
 852 ている。品質試験の結果が後から上書きされることは、結果の信頼性を失うと考えているからである。そのた
 853 め、ユーザが対象としたい検索キーワードは以下の項目に限られ、これらをサポートすれば十分であると
 854 考えた。

- 855 • モジュール及びFEチップのID.
- 856 • 読み出し試験項目(例:std_digitalscan)
- 857 • 読み出し試験者.
- 858 • 読み出し試験場所.
- 859 • 試験日時.
- 860 • タグ機能を用いてつけられたタグ.

861 そこで実装方法として、以下の2つを考えた。

- 862 1. 各試験に関する情報をプログラム上で配列(Pythonリスト)に保持し、検索キーワードが含まれる
 863 かを確認する方法.
 - 864 2. 各試験に関する情報を持つドキュメント、コレクションを予め作成、それを参照し検索を行う方法.
- 865 これらについて以下で詳細を説明する。

866 6.1.1 方法 1: Python リストを用いた一致確認

867 Python リストを使う実装の場合、以下のような流れで検索処理を行う。

- 868 1. ユーザが検索キーワードを入力し、処理を実行.
- 869 2. アップロードされた全ての読み出し試験結果に関する情報を取得.
- 870 3. 各試験結果に関する情報を Python リストに保持、検索キーワードとの一致を確認、試験を選別.
- 871 4. ブラウザに送信.

872 アルゴリズムのイメージを図 6.1 に示す。この方法では、データベース内の試験結果とアプリケーションの関数だけで全ての処理を行うことが可能なため、シンプルな実装方法であると言え、直感的に始めに思いつく方法であった。

875 しかしこの方法を試験実装したところ、ドキュメント数の増加に対して検索処理時間を大きく要して
876 しまう問題が発生した。図 6.2 のようにデータベース内の構造は複数のコレクションを跨いで情報を保
877 持しているため、試験結果全てに対してリアルタイムでこの処理を行うと、検索時間が大きくかかる
878 しまう。このシステムのデータ構造においては、表 4.1 において各試験結果の情報を保存する testRun、
879 component と testRun の関係を保存する componentTestRun の構造による遅延であると考えられる。
880 試験結果数を n とすると、testRun が n 、componentTestRun がそれぞれ $O(n)$ のドキュメント数を持
881 つことになる。これらのドキュメントを全て検索し、一致確認を行うと全体で $O(n^2)$ の時間がかかる。
882 イメージを図 6.3 に示す。この実装方法について、ドキュメント数と処理時間の関係を測定したところ、
883 図 6.4 のようになり、確かにドキュメント数に対して 2 次的に増加していた。測定の詳細については後述
884 する。

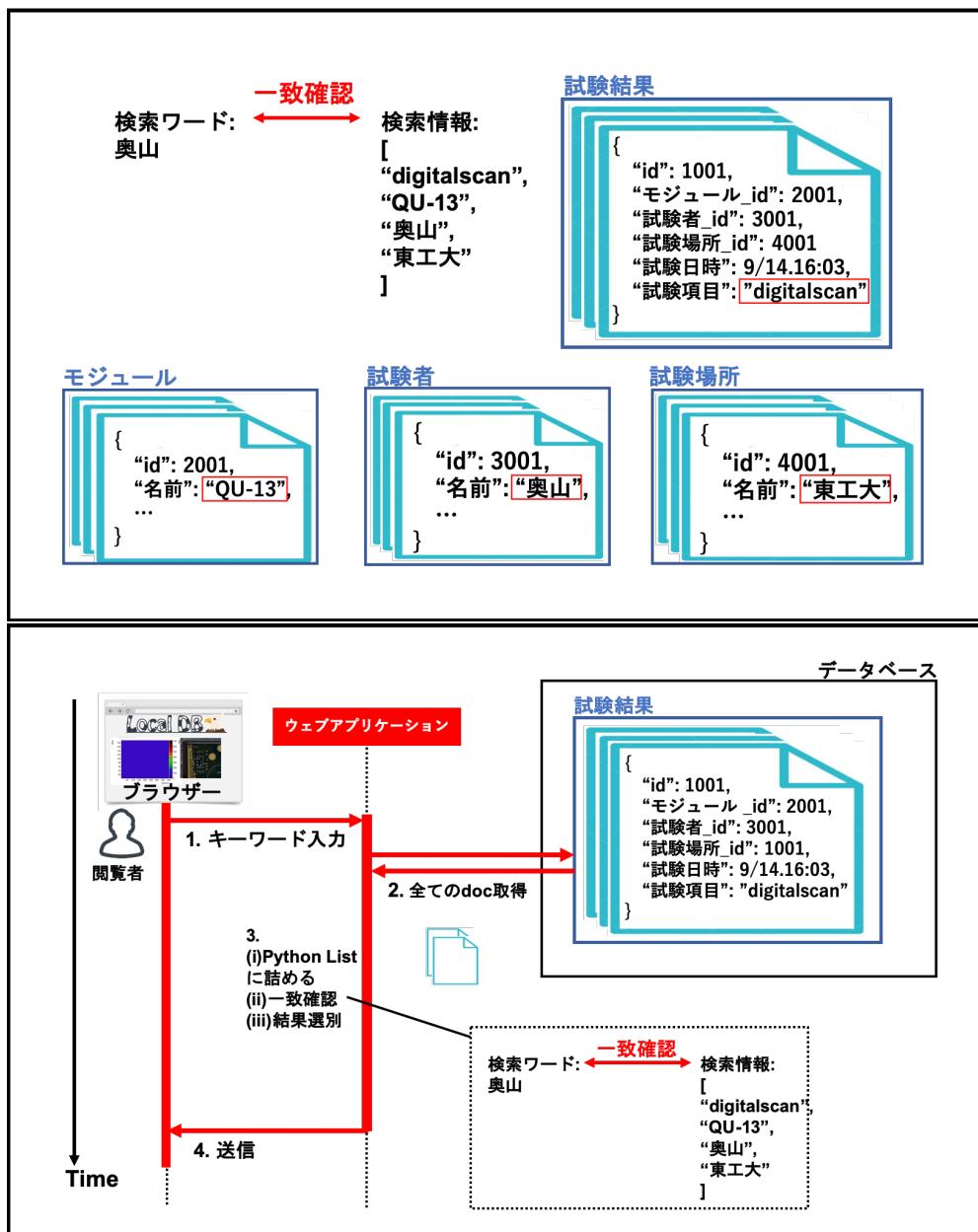


図 6.1 検索機能実装方法 1:Python リストを用いた場合の検索処理のイメージ図。上図は各コレクションと保存されている情報の例を示しており、下図は実際に検索を行った時の処理の流れを表している。上図中の赤枠で囲われた情報のように検索対象となる名前の情報は複数のコレクションにまたがって保存されている。各試験結果に対してこれらの情報を集めリスト内に保持し、各要素とキーワードとの一致を確認することで検索処理を行う。

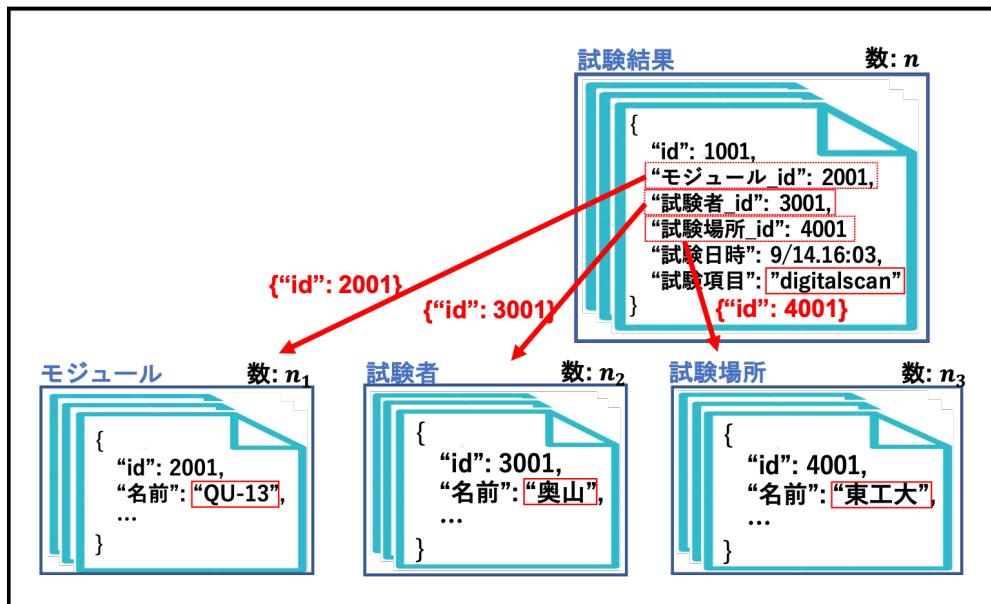


図 6.2 検索機能実装方法 1 の問題点のイメージ図。図 6.1 でも述べたように、検索対象となる情報はこの図のように複数のコレクションにまたがって保存される。そのため、コレクションを超えて検索を行うような場合は試験結果の数以上に、処理に時間がかかるてしまう。この図の例の場合、あるコレクション検索にかかる時間がドキュメント数に対して線形とすると、 $O(n * (n_1 + n_2 + n_3))$ の処理時間がかかると考えられる。

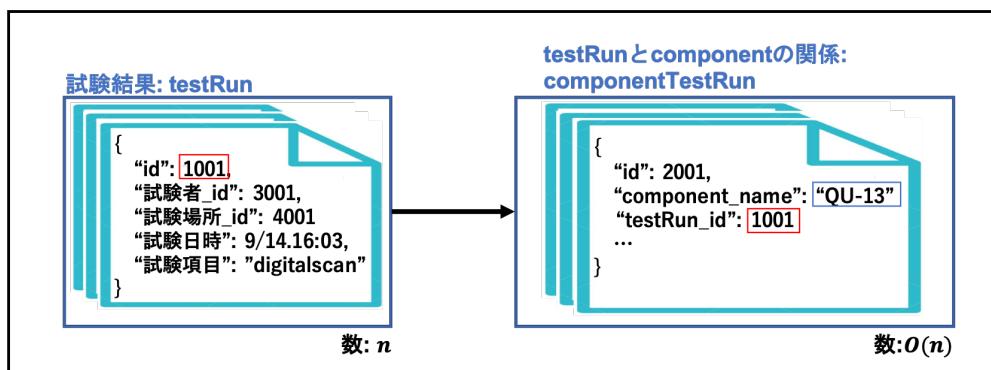


図 6.3 検索処理時間のボトルネックとなっているデータ構造の図。読み出し試験におけるデータ構造(図 4.3)の 1 つに、各試験結果情報を保存する testRun、component と testRun の関係性を保持する componentTestRun という構造が存在する。試験結果数を n とすると、testRun のドキュメント数は n 、componentTestRun の数は試験数と component の数の積となるため $O(n)$ となる。結果的に検索処理に $O(n^2)$ の時間がかかるてしまう。

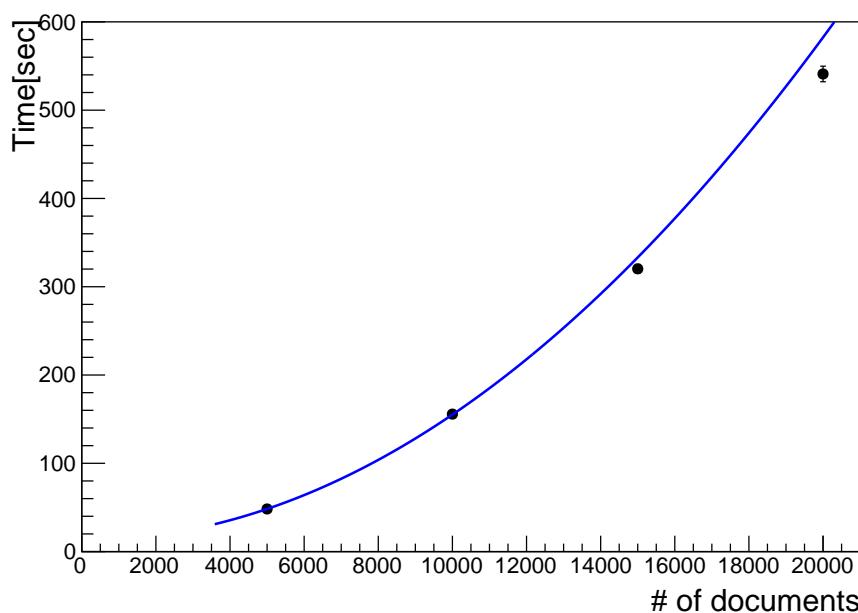


図 6.4 方法 1 における検索処理速度測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。図 6.3 で述べたように、方法 1 の検索処理では試験結果数 n に対して $O(n^2)$ の検索時間がかかるため、試験結果のドキュメント数に対して処理時間は二次関数になっていることがわかる。近似関数や生産時における処理時間の見積もりに関しては後述する。

885 6.1.2 方法2: 検索情報を持つドキュメントを作成、使用

886 検索機能を改善するため方法2を考案し、実装を行った。ここで読み出し試験のデータ構造及び使用し
887 ているPythonフレームワークの変更はせずに処理時間を改善することを試みた。

888 改善策として、検索キーワードを別のドキュメントに予め保持しておく、処理実行時にはそれを参照す
889 ることで検索を行う方法を考案し、試験を行った。アルゴリズムのイメージを図6.5に示す。検索情報コ
890 レクションに入るドキュメント数は、試験結果数と同じである。そのため試験結果数に対する処理時間は
891 $O(n)$ と考えられ、方法1に比べて検索コストを削減できると考えた。なお、このコレクションはウェブ
892 アプリケーションの立ち上げ時に生成するシステムとした。

893 検索情報ドキュメントの例をコード6.1に示す。

ソースコード6.1 検索情報コレクションに入るドキュメントの例。このように試験結果のID、試験
日時、検索対象となる名前情報が保存される。

```
894 1 {  
895 2     "_id" : ObjectId("5fd489f60e2ca70557e44a8b"),  
896 3     "runId" : "5fc4d027b1ef7c6297c91040",  
897 4     "timeStamp" : ISODate("2020-11-30T10:57:19Z"),  
898 5     "data" : [  
899 6         "20UPGR00000001",  
900 7         "20UPGFC99999999",  
901 8         "std_digitalscan",  
902 9         "hokuyama",  
903 10        "tokyo_institute_of_technology",  
904 11        "2020/12/09"  
905 12    ]  
906 13 }
```

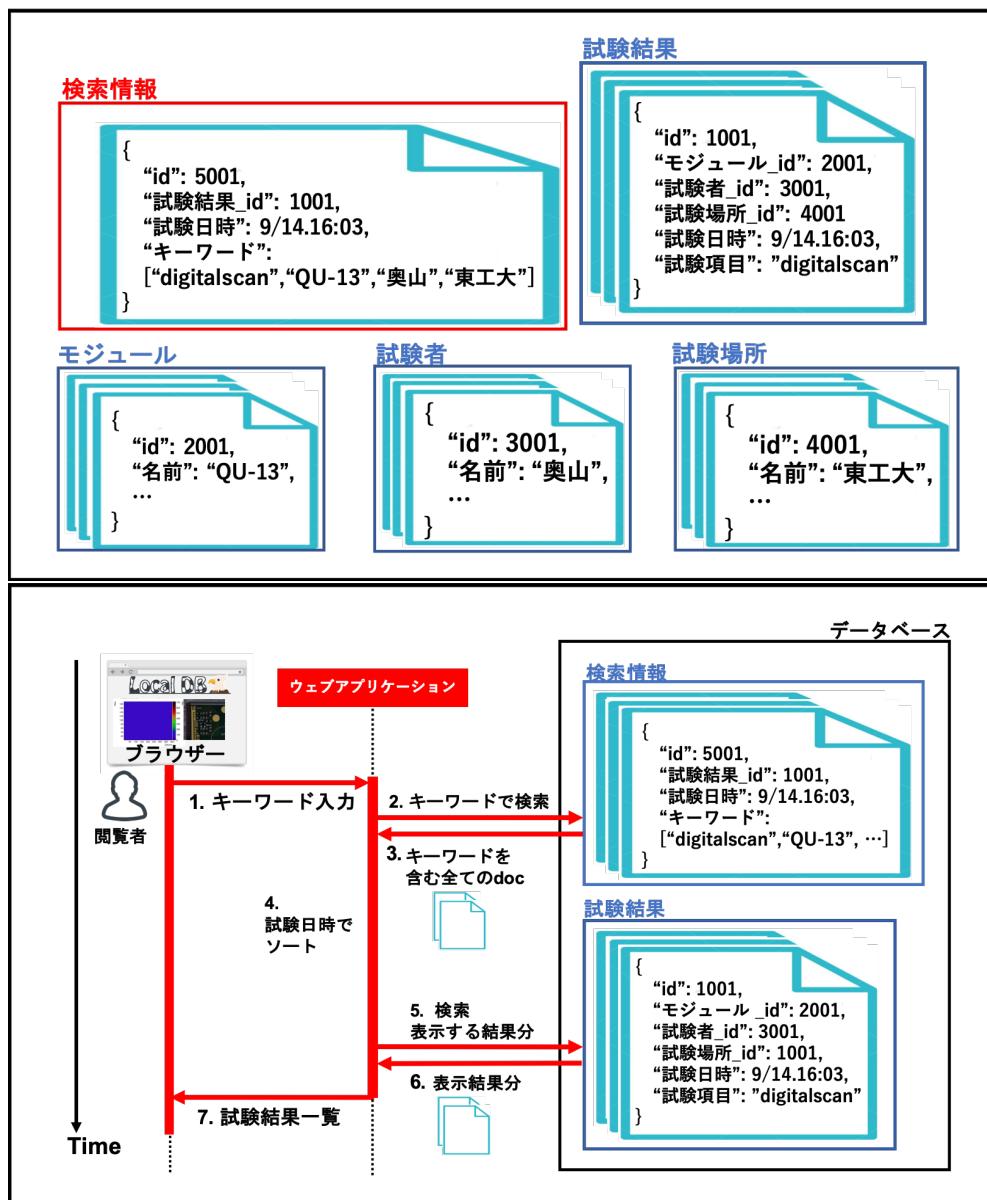


図 6.5 検索機能実装方法 2:検索キーワード専用コレクションを用いた場合のイメージ図。上図は各コレクションと保存されている情報の例を示しており、下図は実際に検索を行った時の処理の流れを表している。方法 2 では上図の赤枠で囲っている検索情報のコレクションを新たに設け、これを参照することで検索処理を行う。このコレクション内には各試験結果に対応したドキュメントが 1 つ保存され、全検索情報と試験結果の ID を持つものとなっている。処理の流れとしては下図のように、検索情報のコレクションよりキーワードに対応する試験結果を抽出する処理と、表示の際に必要な情報を試験結果のコレクションから取得する処理の 2 つに分かれた構造となっている。このような処理をすることにより、各検索処理にかかる時間は試験結果数に対して $O(n)$ になると考えられる。

表 6.1 測定に使用したノート PC(MacBookAir(13-inch,2017)) の性能。検索処理時間の測定に個人的に使用しているノート PC を使用した。

種類	CPU	Type	Core	Thread	Clock speed[GHz]	Memory [GB]	Disk [GB]
MacBookAir(13-inch,2017)	Intel Core i5	2	4	1.8		8	256

表 6.2 検索機能処理時間測定の詳細。測定を行った試験結果数、回数、キーワード、検索モード、検索情報の詳細を示している。

試験結果数	5000, 10000, 15000, 20000
測定回数	各測定点に対して 20 回
検索キーワード	okuyama
検索モード	部分一致
各試験結果が持つ検索情報	全試験結果に対して同じ、以下に詳細
検索情報一覧	モジュール名: 20UPGR10000005 FE チップ名: 20UPGTU9000000 試験項目: std_digitalscan 試験者: okuyama 試験場所: default_host 試験日時: 2020/12/06

6.2 処理時間測定

考案した方法を実装し、検索処理時間の測定を行った。詳細を以下に示す。また方法 1 において行った処理時間測定も同様の条件で行った。

使用した装置

測定には個人的に使用しているノート PC(MacBookAir(13-inch,2017)) を用いた。性能を表 6.1 に示す。

測定内容

コマンドプロンプトから以下のコマンドを実行し、ある試験結果ページのリクエストに対するアプリケーションのレスポンス時間を測定した。ここでは検索キーワードは “okuyama” とし、検索モードは部分一致としている。実際にアプリケーションを使用する際には、ブラウザに一覧表示をする時間が今回の測定時間に加算されることになる。

```
1 curl "http://127.0.0.1:5000/localdb/scan?keywords=okuyama&match=partial" -o /dev/null -w "%{time\_total}\n" 2> /dev/null -s
```

その他測定に関する詳細を表 6.2 に示す。

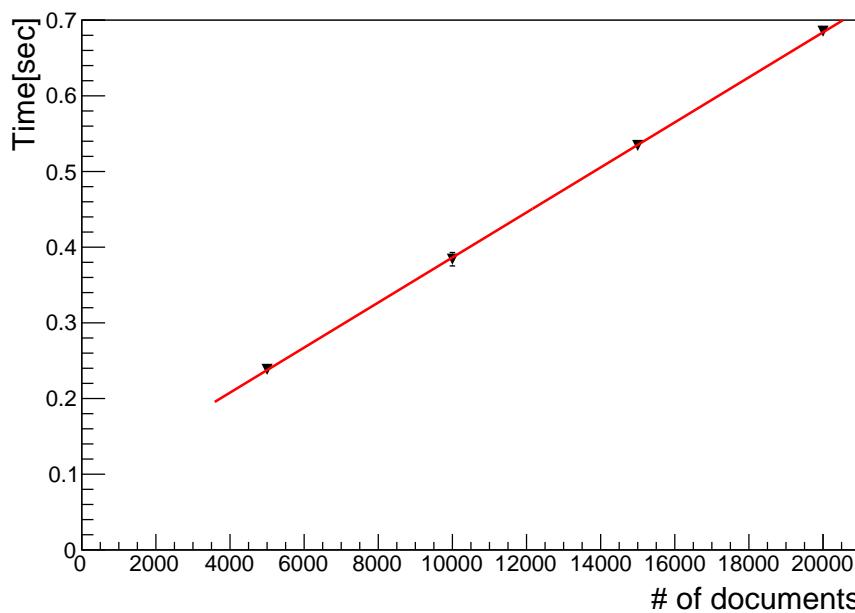


図 6.6 方法 2 における検索処理時間測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。線形性を示していることが確認でき、検索実行時における処理時間は方法 1 に比べて優れている。

925 測定結果

926 方法 2 を用いて得られた結果を図 6.6 に示す。また方法 1 の結果に関しては、既に図 6.4 で示した。方
927 法 1、方法 2 で得られた近似関数を式 6.1、6.2 に示す。方法 1 に対して、方法 2 はアルゴリズムの改善が
928 見られる。

$$y = ax^2 + b \quad (6.1)$$

$$a = (1.4 \pm 0.0) \times 10^{-6}$$

$$b = 13 \pm 0$$

$$y = ax + b \quad (6.2)$$

$$a = (3.0 \pm 0.1) \times 10^{-5}$$

$$b = (8.9 \pm 0.8) \times 10^{-2}$$

929 現在は方法 2 で検索機能を実装し、サービスの 1 つとして提供している。
930 方法 2 では、アプリケーションの起動時に検索情報コレクションを生成するアルゴリズムとしている。
931 ここで検索情報コレクションの生成にかかる処理時間として、ドキュメント数に対するアプリケーション
932 の起動時間を測定した。結果を図 6.7 に示す。基本的には方法 1 と同じ操作を行うことで検索情報を収集
933 し、ドキュメントを作成する。そのため、処理時間は $O(n^2)$ となる。この処理を前もって行なうことで、
934 検索実行時の処理性能を上げている。

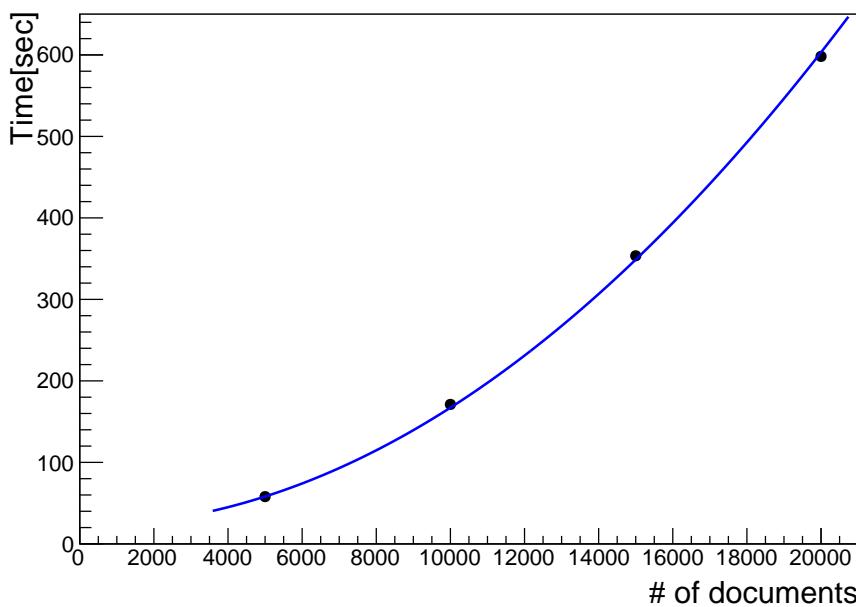


図 6.7 方法 2 における検索情報コレクションの生成時間測定結果。検索情報コレクションの生成にかかる処理時間として、ドキュメント数に対するアプリケーションの起動時間を測定した。横軸が試験結果のドキュメント数、縦軸が時間を示している。生成には、方法 1 と同様の処理を行なっているため $O(n^2)$ となっているのが分かる。

935 生産時における検索時間の見積もり

936 各方法について、生産時における処理時間の見積もりを行う。簡単のため今回使用したデバイスと生産
 937 時に使うサーバーの性能差は無視する。ここでデータベースで管理するモジュール数は日本が最多とし、
 938 その数を予定している 2,000 とする。保存する読み出し試験数は 3 章で述べた項目の合計とし、1 つのモ
 939 ジュールあたり 42 とする。全ての生産が終了した際の検索処理時間を見積もる。検索処理実行時間は、
 940 上で得られた関係式を用いて方法 1、2 に対して式 6.3、6.4 のように見積もることができる。

$$\{(1.4 \pm 0.0) \times 10^{-6}\} \times (2000 \times 42)^2 + (13 \pm 0) = (9.8 \pm 0) \times 10^3 [\text{sec}] \quad (6.3)$$

941

$$\{(3.0 \pm 0.1) \times 10^{-5}\} \times (2000 \times 42) + \{(8.9 \pm 0.8) \times 10^{-2}\} = 2.6 \pm 0.1 [\text{sec}] \quad (6.4)$$

942 方法 1 では 1 回の検索に対して約 2.7 時間と見積もられ、生産時には検索機能として運用不可能なシス
 943 テムであることがわかる。方法 2 では終了時点においても数秒で処理を終えることができるため、生産を
 944 通して使用できると考えられる。

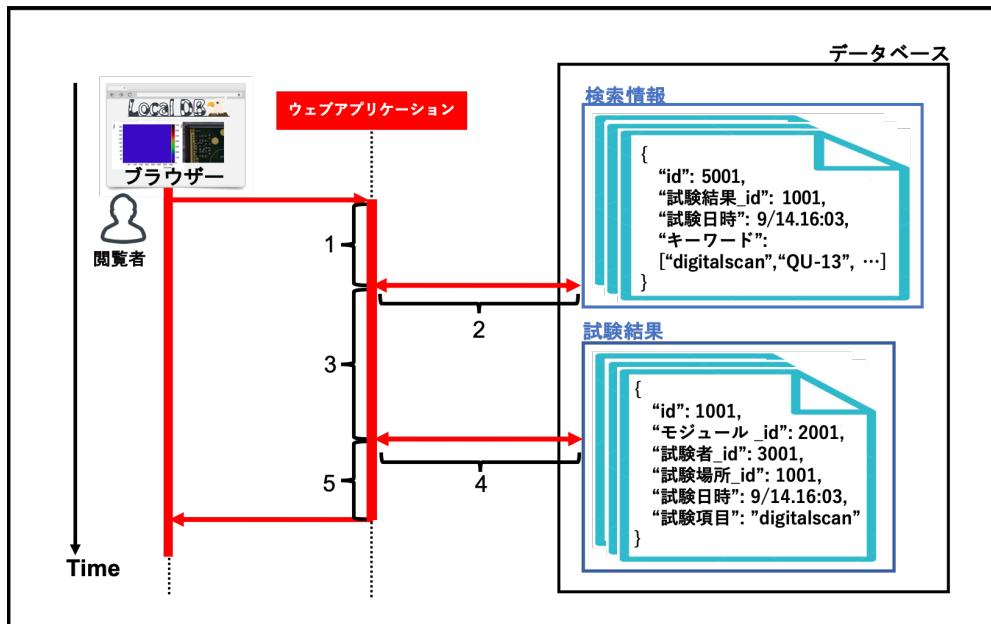


図 6.8 方法 2 の検索における詳細処理。図のように全体の流れにおけるアプリケーション内部での各処理、データベースから情報取得の各処理に 1 から 5 の番号をつけそれぞれにかかる処理時間を測定した。

945 6.3 改善方法と処理時間測定

946 より処理時間を短くすることを目的として、新たな検索処理アルゴリズムの考案と測定を行った。詳細
947 について以下に示す。

948 6.3.1 方法 2 における検索処理時間の詳細調査

949 先述したように、方法 2 では処理時間が改善した。この方法 2 について、処理時間の詳細を知るために
950 追加で測定を行った。アプリケーション層での各処理について、以下のように番号をつける。

- 951 1. キーワードを受け取り、検索情報コレクションに検索をかけるまでの処理。
- 952 2. 検索情報コレクションに検索をかけ、情報を受け取る処理。
- 953 3. ドキュメントを受け取り該当する試験結果 ID をまとめ、試験結果に対して検索をかけるまでの
954 処理。
- 955 4. 試験結果コレクションに検索をかけ、情報を受け取る処理。
- 956 5. ドキュメントを受け取りデータを整形、ブラウザにレスポンスを返すまでの処理。

957 イメージを図 6.8 に示す。

958 ポトルネックとなっている処理を測定するために、上述した各処理にかかる時間の測定を行った。測定
959 は試験結果数が 10,000 の場合に行った。

960 結果を図 6.9 に示す。

961 図より処理 3、5 の割合が大きいことがわかる。これらの処理について、特に以下の処理の割合が大き

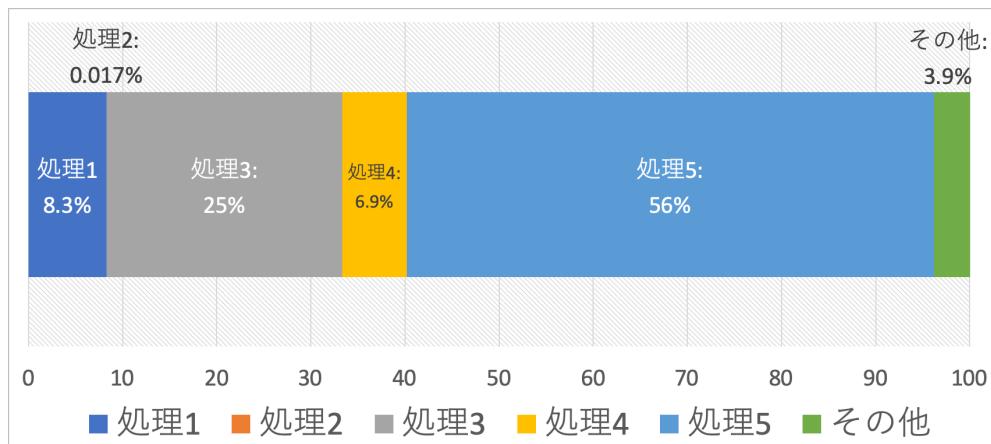


図 6.9 方法 2 における詳細処理時間の測定結果。図 6.8 における 5 つの詳細処理にかかる時間の割合を表している。処理 3、5 にあたるコレクション検索実行後のアプリケーション内での処理に多く時間がかかっていることが分かる。

表 6.3 処理 3,5 における型変換処理 6.5 の割合。処理 3、5 について、表の割合より、型変換処理 6.5 が支配的であることが分かる。

処理	全体 [sec]	処理 6.5[sec]	割合 [%]
3	0.091 ± 0.011	0.089 ± 0.005	97 ± 0
5	0.21 ± 0.00	0.18 ± 0.00	86 ± 1

962 いことがわかった。

取得した複数ドキュメントを Python リストへ型変換する処理. (6.5)

963 図 6.8 における処理 3、5 において、型変換処理 6.5 の割合を表 6.3 まとめた
 964 この変換処理について、あるコレクションにおける全ドキュメント数に対する Python リスト変換処理
 965 時間の関係を測定した。結果を図 6.10 に示す。全ドキュメント数に対して線形性を示していることがわ
 966 かる。方法 2 の検索処理については型変換処理 6.5 が支配的であることが分かった。

967 6.3.2 改善点

968 測定を踏まえ、改善方法として以下の項目を検討した。ここでは、上述したように使用しているデータ
 969 構造やフレームワークの変更はせずに処理時間を改善することを前提としている。

- 970 ● コレクション検索処理の回数を減らす.
- 971 ● 検索対象コレクションのドキュメント数を減らす.

972 上述した 2 つを目的として、以下の 2 つの方法を新しく考え処理時間測定を行った。

- 973 3. 検索情報のコレクションに一覧表示に必要な情報を保持、参照.
- 974 4. 方法 3 に付け加えて、検索情報のドキュメントを複数コレクションに分散、マルチスレッドを用い
 975 た検索処理の並列化.

976 方法 3 については一覧表示に必要な情報を検索情報のドキュメントが持つことで、データベースに対す

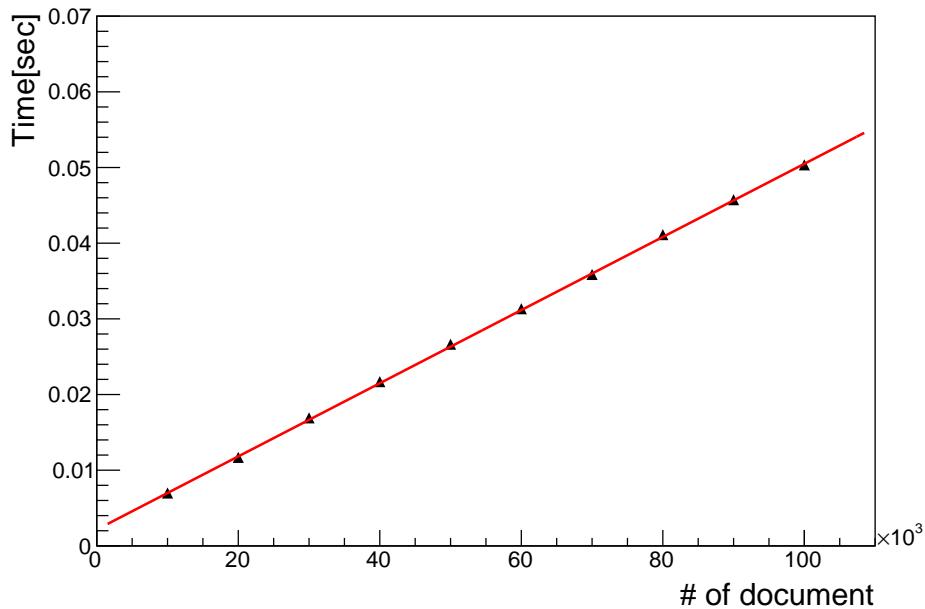


図 6.10 ドキュメント数に対する型変換処理時間の関係。コレクション検索後の型変換に要する処理時間は、図のようにドキュメント数に対して線形となっていることが分かる。

る検索の回数を減らすことを目的としている。方法 4 については方法 3 の検索処理の数を減らすことに加えて、ドキュメントの数を分散し並列処理をすることで処理時間の改善を図っている。イメージをそれぞれ図 6.11、6.12 に示す。

方法 3、4 について、章 6.2 と同じ内容の測定を行った。方法 2 のものと合わせた結果を 6.13 に示す。方法 4 について、分散するコレクション数は 10 個、スレッド数には 2 とした。方法 2 に比べて、方法 3、4 共に処理時間が改善していることがわかる。

方法 3、4 を比べると傾きに差が見られる。そのため、方法 4 はドキュメント数が多くなった時に有効であると考えられる。方法 4 に関しては今回はコレクション数を 10、スレッド数を 2 としたが、それぞれ最適な数を検討することで更なる改善ができる可能性がある。

得られた方法 3,4 に関する関係を式 6.6、6.7 に示す。

$$\begin{aligned}
 y &= ax + b \\
 a &= (2.9 \pm 0.1) \times 10^{-5} \\
 b &= (5.0 \pm 1.1) \times 10^{-3}
 \end{aligned} \tag{6.6}$$

$$\begin{aligned}
 y &= ax + b \\
 a &= (2.7 \pm 0.1) \times 10^{-5} \\
 b &= (2.2 \pm 1.0) \times 10^{-2}
 \end{aligned} \tag{6.7}$$

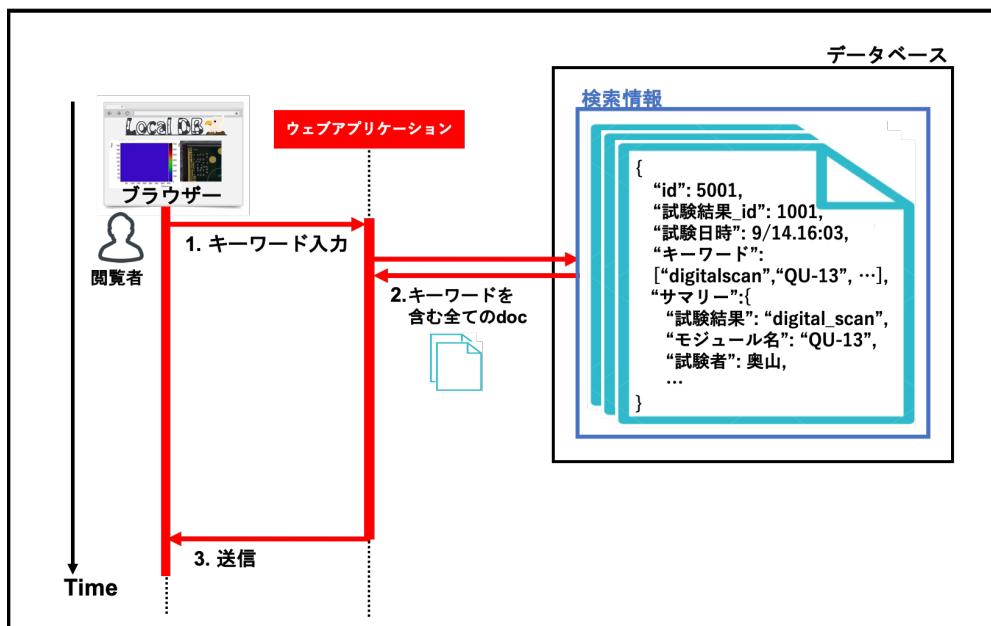


図 6.11 検索機能実装方法 3:検索情報と共に一覧表示に必要な情報を保持、参照。方法 2 では検索情報コレクションより、条件に一致する試験結果 ID を取得し、実際の試験結果に対して再度検索をかける流れとなっていたが、この方法では一覧表示に必要な情報も全て検索情報のコレクション内で保持する。こうすることで検索機能において必要な情報の全てが 1 つのコレクションにまとまり、コレクション検索の処理が 1 回で済むため、処理時間が改善すると考えられる。

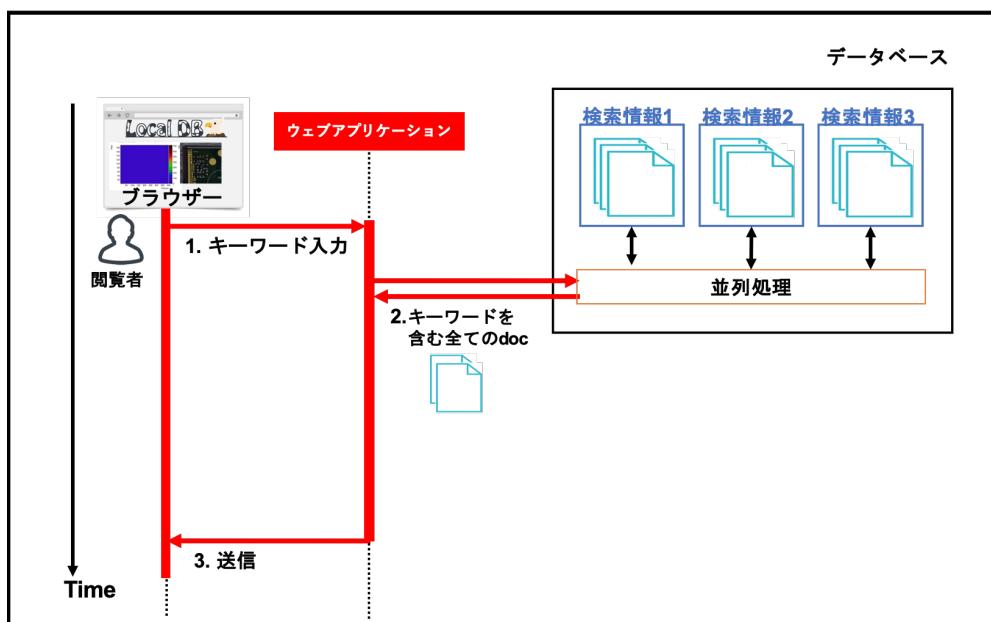


図 6.12 検索機能実装方法 4:検索情報コレクションを分散、マルチスレッドを使用。方法 3 に加えて、検索情報コレクションを分散し、並列処理を行うことで、1 つのコレクションあたりに含まれるドキュメント数を減らし、コレクション検索にかかる時間を削減できると考えられる。

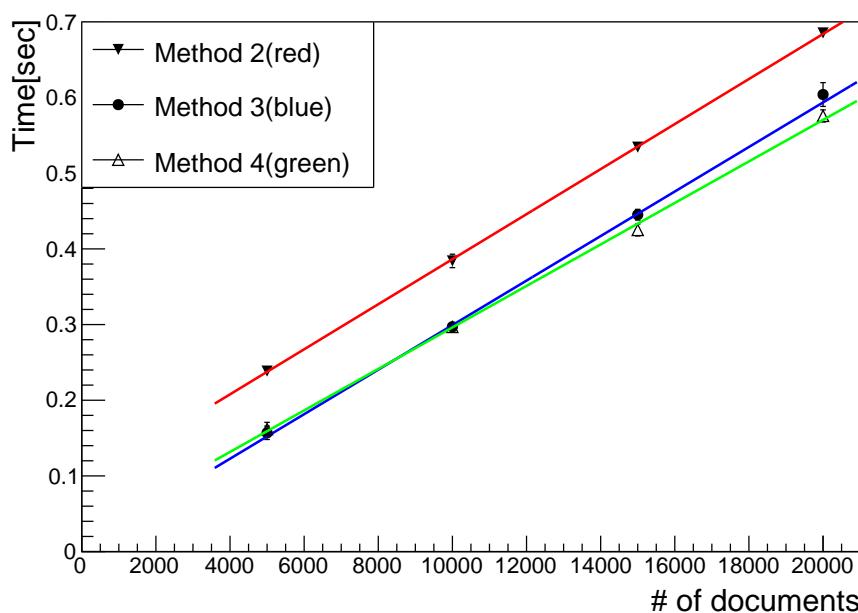


図 6.13 方法 3、4 に対する処理時間測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。赤、青、緑がそれぞれ方法 2、3、4 を用いたものである。方法 2 に比べて 3、4 共に改善していることが分かる。方法 3、4 を比べると傾きに差があることが分かる。これよりドキュメント数が大きい場合に方法 4 は有効である。

987 第7章

988 中央データベースとローカルデータベー 989 スの同期

990 中央データベースとローカルデータベースの同期機能についての調査と機能開発を行った。詳細につい
991 て以下で説明する。

992 7.1 サーバーの設置場所による処理時間の違い

993 4章で述べたように、中央データベースはチェコに設置されている。そのため通信に要する処理時間
994 は、組み立て機関の場所に依存すると考えられる。世界的に同期ツールが不自由なく動くことを向かた開
995 発、改善に役立てることを目的として、データベース間の通信にかかる処理時間を測定した。組み立て機
996 関及びローカルデータベースの設置場所はヨーロッパ、アメリカ、日本の3つの地域に分布している(付
997 錄C)。それぞれにおける代表機関として以下の3つに設置されたサーバーを用いて調査を行った。

- 998 • 日本、高エネルギー加速器研究所 (KEK)
- 999 • アメリカ、バークレー研究所 (LBL)
- 1000 • スイス、欧州原子核研究機構 (CERN)

1001 各サーバーの性能を表7.1に示す。また各サーバーが置かれている場所の位置関係を図7.1に示す。

1002 これらのサーバーは実際に生産の際に使用するものと同程度の性能を持ち、サーバーが置かれている
1003 ネットワーク環境も生産時と同じであると仮定している。

表7.1 各ローカルデータベースサーバーの性能一覧。今回の調査に利用したサーバーの性能を示す。
KEK(日本)、LBL(アメリカ)、CERN(スイス)に設置されたサーバーを用いた。

設置機関	CPU	Type	Core	Thread	Clock speed[GHz]	Memory [GB]	Disk [GB]
KEK(日本)	Intel(R) Core(TM) i7-9700K		8	16	3.6	32.66	1800 + 1800
LBL(アメリカ)	Intel(R) Core(TM) i7-8700		6	12	3.7	32.63	233
CERN(スイス)	Intel(R) Core(TM) i7-4790		4	8	3.6	32.69	238.5 + 3700 + 3700

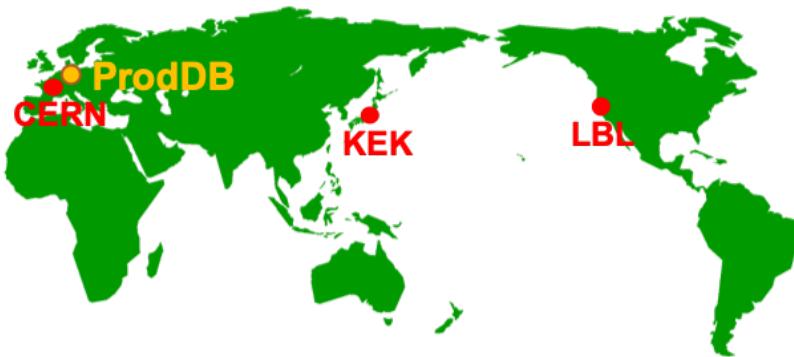


図 7.1 各サーバーの設置場所。赤点で示しているのがそれぞれローカルデータベースサーバーの設置位置であり、オレンジで示しているのが中央データベースである。距離としては CERN が一番近く、LBL、KEK の順番となっている。

表 7.2 同期ツールの中で使用する中央データベースの主な API 一覧。同期ツールにおいて、中央データベースの情報取得には提供されているいくつかの API を用いており代表的なものをいかに示す。この API を Python を用いて実行することで、情報取得や試験結果のアップロードをすることができる。

関数名	処理の内容	本ツールでの使用用途
getComponent	登録した部品情報の取得	主にダウンロード時におけるモジュールや FE チップの情報取得に用いる。
listComponents	登録した部品情報一覧の取得	主にダウンロード時におけるモジュール情報一覧取得に用いる。
uploadTestRunResults	テスト結果生成	読み出し試験結果生成の際に用いる。
createTestRunAttachment	あるテスト結果に対するバイナリファイルの添付	読み出し試験結果ファイルを添付する際に用いる。

1004 7.1.1 同期ツールに使用する API

1005 中央データベースとの同期には、中央データベースで開発された API を使用している。ローカルデータベースとの同期ツールの中で使用している主な API を表 7.2 に示す。

1007 7.1.2 API 使用にかかる時間

1008 上述した API 使用時の処理時間を各サーバーで測定した。以下の 3 つの測定を行なった。

- 1009 • getComponent を用いた、登録モジュール情報 1 つの取得時間測定。
- 1010 • createTestRunAttachment を用いて、ある試験結果ページに 1Byte のデータファイルを添付する時間測定。
- 1011 • createTestRunAttachment を用いて、ある試験結果ページに容量の異なるデータファイルを添付、容量に対する時間依存性を測定。

1014 最初の 2 項目に関して、まとめたものを表 7.3 に示す。ファイル容量と処理時間の関係を図 7.2 に示す。ここで、どの場合においても KEK における処理時間が最も長いことがわかる。そのため本研究では、開発した同期ツールの処理確認及び処理時間測定を KEK サーバーで行い、その有用性を評価することとした。

表 7.3 中央データベース API 実行時の処理時間測定結果。左の結果は表 7.2 における”getComponent”を用いてモジュール 1 つの情報を取得するのにかかった時間、右は”createTestRunAttachment”を用いて 1Byte のファイル送信にかかった時間である。どのサーバーにおいても 0.28 秒以上の処理時間がかかっていることがわかり、データベースへの接続、情報取得にかかる時間が読み取れる。3 つのサーバーを比べると、どちらの場合も KEK サーバーでの処理時間が一番大きいことが分かる。

サーバー	処理時間 [秒]	サーバー	処理時間 [秒]
KEK	0.47 ± 0.01	KEK	0.83 ± 0.01
LBL	0.37 ± 0.02	LBL	0.34 ± 0.03
CERN	0.28 ± 0.01	CERN	0.48 ± 0.03

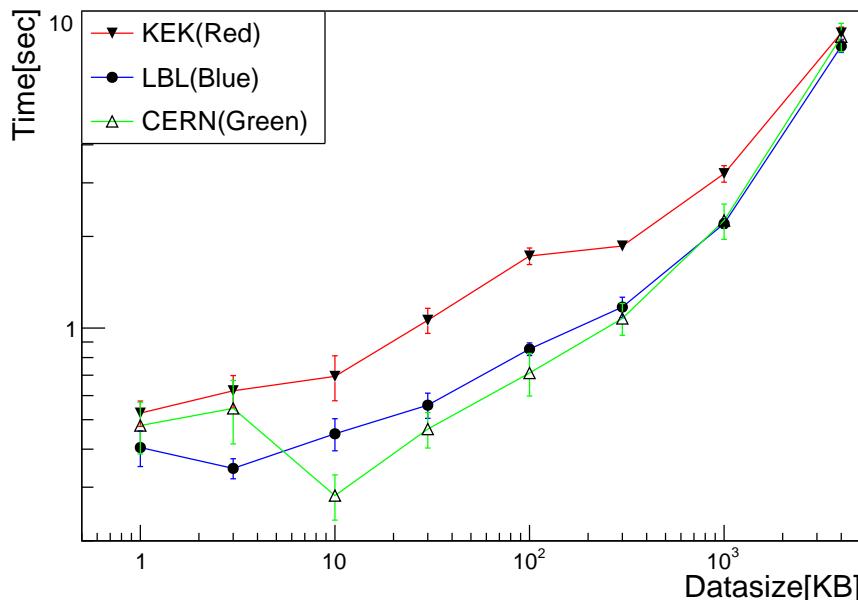


図 7.2 ”createTestRunAttachment”を用いた添付処理におけるファイル容量と処理時間の関係。それぞれのサーバーで 1、3、10、30、100、300KB、1、4MB のファイル送信にかかる時間を測定した。どの点においても KEK サーバーが最も処理時間を要していることが分かる。またこのグラフについての考察を付録 E に記す。

ここで、KEK、LBL のサーバーにおいてネットワークの混雑状況等の理由による処理時間遅延の影響を調べるために、中央データベースウェブページへの通信時間を 1 週間通して測定した。測定は 1 月 10 日午後 8 時から 1 月 18 日午後 16 時の、偶数時間 (0,2,4,6,8,10,12,14,16,18,20,22 時) に行なった。CERN のサーバーにおいては、サーバーが一時的にダウンしたためデータを取得できなかった。測定結果を図 7.3 に示す。KEK、LBL 共に処理時間の変化は小さく、どの時間帯においても KEK の方が遅いことが分かる。

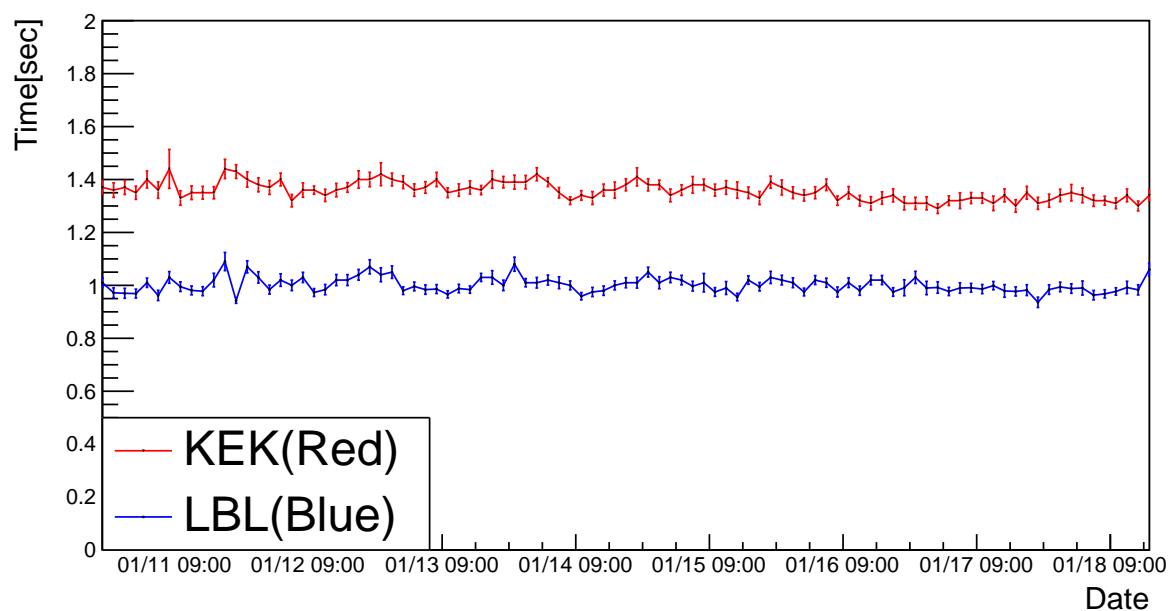


図 7.3 中央データベースウェブページへの通信時間の測定結果。横軸は測定日時、縦軸は中央データベースウェブページへの通信時間を表す。1月10日午後8時から1月18日午後16時の期間で測定を行い、各日の偶数時間にデータ取得を行なった。赤線、青線はそれぞれKEK、LBLのサーバーを用いた測定結果である。どちらのサーバーにおいても時間帯による通信時間の大きな変位は見られず、全ての時間帯においてKEKの通信時間の方が遅いことが分かる。

表 7.4 ダウンロード機能を用いて保存する情報一覧。ダウンロード機能を用いて中央データベースからローカルデータベースに保存する情報の一覧を示している。保存の際にはモジュール、FE チップにそれぞれ分かれたドキュメントに保存される。

部品	情報
モジュール	シリアルナンバー
	搭載 FE チップの種類
	登録機関
	搭載 FE チップの枚数
FE チップ	シリアルナンバー
	FE チップ ID(モジュール上の位置を表す情報)
	登録機関

7.2 モジュール情報のダウンロード

7.2.1 ダウンロードする情報と構造

中央データベースから、モジュール及び FE チップの情報をダウンロードする機能を開発、実装した。

ダウンロードする情報の詳細について表 7.4 に示す。

ダウンロードされたモジュール、FE チップのドキュメントの例をコード 7.1、7.2 に示す。

ソースコード 7.1 ダウンロードしたモジュール情報のドキュメントの例。ドキュメントが表 7.4 の情報を持つことが分かる。

```

1 {
2     "_id" : ObjectId("5fa79114e615fa000a1a5976"),
3     "name" : "20UPGR00000001",
4     "chipType" : "RD53A",
5     "serialNumber" : "20UPGR00000001",
6     "chipId" : -1,
7     "componentType" : "module",
8     "address" : "5fd597fdf7339bbf26b87fb2",
9     "children" : 1,
10    "sys" : {
11        "mts" : ISODate("2020-12-13T04:26:37.989Z"),
12        "cts" : ISODate("2020-12-13T04:26:37.989Z"),
13        "rev" : 0
14    },
15    "dbVersion" : 1.01,
16    "user_id" : -1,
17    "proDB" : true
18 }
```

ソースコード 7.2 ダウンロードした FE チップ情報のドキュメントの例。ドキュメントが表 7.4 の情報を持つことが分かる。

```

1 {
2     "_id" : ObjectId("5fa79560e615fa000a1a5a16"),
3     "name" : "20UPGFC9999999",
4     "chipType" : "RD53A",
5     "serialNumber" : "20UPGFC9999999",
6     "chipId" : 0,
7     "componentType" : "front-end_chip",
8     "address" : "5fd597fdf7339bbf26b87fb2",
9     "children" : -1,
10    "sys" : {
11        "mts" : ISODate("2020-12-13T04:26:37.984Z"),
12        "cts" : ISODate("2020-12-13T04:26:37.984Z"),
13        "rev" : 0
14    },
15    "dbVersion" : 1.01,
```

```
1065   16      "user_id" : -1,  
1066   17      "proDB" : true  
1067  18 }
```

1069 7.2.2 処理の流れ

1070 ダウンロード機能における処理の流れのイメージを図 7.4 に示す。

1071 7.2.3 機能確認

1072 KEK で組み立てられた 6 台の Quad モジュールを中央データベースに登録し、ダウンロードを行つ
1073 た。登録したモジュールを表 7.5、ダウンロードをしてアプリケーションで確認した様子を図 7.6 に示す。

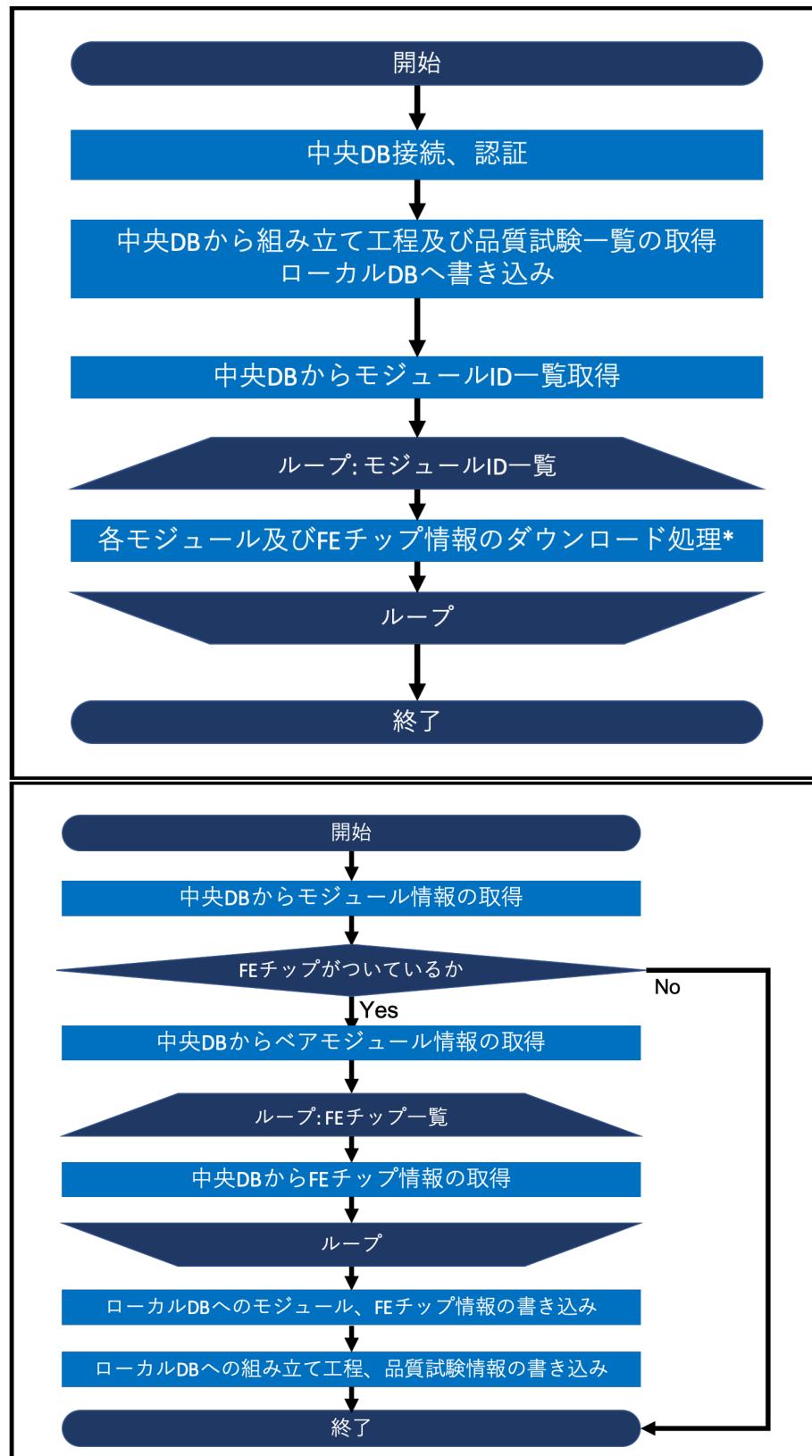


図 7.4 ダウンロード処理における流れのイメージ図。上図が処理全体の流れを表すものであり、下図は各モジュール情報のダウンロードにおける処理の流れを示している。上図中のループ構造の 1 処理が下図に対応している。流れの中で複数回中央データベースに接続し、モジュールや FE チップの情報取得をしていることが分かる。

Module	Bare Module	Sensor	PCB	Carrier	FE chip
20UPGM20030004	20UPGB40500019	20UPGS83300002	20UPGPQ0030004	20UPGMC0210000	20UPGFC0014659 20UPGFC0014658 20UPGFC0014675 20UPGFC0014691 20UPGFC0014644 20UPGFC0014628 20UPGFC0014660 20UPGFC0014708 20UPGFC0014677 20UPGFC0014629 20UPGFC0014693 20UPGFC0014646 20UPGFC0016282 20UPGFC0016281 20UPGFC0016279 20UPGFC0016280 20UPGFC0016278 20UPGFC0016267 20UPGFC0016235 20UPGFC0016242 20UPGFC0016276 20UPGFC0016266 20UPGFC0016220 20UPGFC0016227
20UPGM20030001	20UPGB40500020	20UPGS83300003	20UPGPQ0030001	20UPGMC0210001	
20UPGM20030003	20UPGB40500021	20UPGS83300004	20UPGPQ0030003	20UPGMC0210002	
20UPGM20030006	20UPGB40500022	20UPGS83300001	20UPGPQ0030006	20UPGMC0210003	
20UPGM20030022	20UPGB40500023	20UPGS83300005	20UPGPQ0030022	20UPGMC0210004	
20UPGM20030024	20UPGB40500024	20UPGS83300006	20UPGPQ0030024	20UPGMC0210005	

図 7.5 登録した Quad モジュールとその構成部品のシリアルナンバー一覧。左から、登録したモジュール、搭載されているベアモジュール、シリコンセンサー、PCB、モジュールキャリア、FE チップの中核データベース内でのシリアルナンバーを示している。Quad モジュールであるため、FE チップをそれぞれ 4 つ搭載している。

ProdDB web

IN PROGRESS Module - Outer system quad module 20UPGM20030001

IN PROGRESS Module - Outer system quad module 20UPGM20030024

IN PROGRESS Module - Outer system quad module 20UPGM20030022

IN PROGRESS Module - Outer system quad module 20UPGM20030006

IN PROGRESS Module - Outer system quad module 20UPGM20030003

IN PROGRESS Module - Outer system quad module 20UPGM20030004

Download

LocalDB TOP / COMPONENTS / TEST

ITk database for Yarr Component List

Input keywords Partial match Perfect match Search

RD53A (11 modules)

Module Name	Chip Name	Latest Result						Tag
		Current Stage	Test Type	User	Site	Date	Link	
20UPGR90020026	20UPGFC0028965 20UPGFC0028950 20UPGFC0028951 20UPGFC0028952	None	None	None	None	None		
20UPGR90000004	20UPGFC0008036 20UPGFC0007994 20UPGFC0007972 20UPGFC0008035	None	None	None	None	None		
20UPGR30000001	20UPGR33000000 20UPGR33000001 20UPGR33000002 20UPGR33000003	None	None	None	None	None		
20UPGR10099999	20UPGFC999995 20UPGFC999996 20UPGFC999997 20UPGFC999998	None	None	None	None	None		
20UPGR00000001	20UPGFC999999	None	None	None	None	None		
20UPGM20030024	20UPGFC0016276 20UPGFC0016266 20UPGFC0016220 20UPGFC0016227	None	None	None	None	None		
20UPGM20030022	20UPGFC0016278 20UPGFC0016267 20UPGFC0016235 20UPGFC0016242	None	None	None	None	None		
20UPGM20030006	20UPGFC0016282 20UPGFC0016281 20UPGFC0016279 20UPGFC0016280	None	None	None	None	None		
20UPGM20030004	20UPGFC0014659 20UPGFC0014658 20UPGFC0014675 20UPGFC0014691	None	None	None	None	None		
20UPGM20030003	20UPGFC0014677 20UPGFC0014629 20UPGFC0014693 20UPGFC0014646	None	None	None	None	None		
20UPGM20030001	20UPGFC0014644 20UPGFC0014628 20UPGFC0014660 20UPGFC0014708	None	None	None	None	None		

図 7.6 登録した Quad モジュールのダウンロードの様子。上図が中央データベースのウェブページを表しており、下図がローカルデータベースのものである。上図で登録したモジュール一覧を確認でき、赤枠で囲っているところでシリアルナンバーを見ることができる。ダウンロード実行後は下図のようにローカルデータベースで対応するモジュールを確認することができる。ローカルデータベースではモジュール情報に加えて FE チップの情報も取得するため、下図の表ではこれらのシリアルナンバーも確認できる。

表 7.5 登録したモジュールのダウンロード処理時間測定結果。登録したそれぞれのモジュールについてダウンロードにかかる時間を測定した。表より 1 つあたり平均 4 秒の時間がかかっていることが分かる。

モジュール	処理時間
20UPGM20030004	3.8
20UPGM20030001	3.7
20UPGM20030003	5.9
20UPGM20030006	3.6
20UPGM20030022	3.8
20UPGM20030024	3.3
平均	4.0 ± 0.4

1074 7.2.4 処理時間測定

1075 ダウンロードの処理時間を測定した。これについてまとめたものを表 7.5 に示す。Quad モジュール 1
1076 つに対して平均して 4.0 ± 0.4 秒の処理時間がかかっている。

1077 7.2.5 処理時間詳細

1078 ダウンロード処理時間の改善に向けて、処理時間の詳細について以下の測定した。

- 1079 1. 中央データベースからモジュール情報の取得.
- 1080 2. データベースでの FE チップ確認処理.
- 1081 3. 中央データベースからベアモジュール情報の取得.
- 1082 4. 中央データベースから FE チップ情報の取得 (4 枚分).
- 1083 5. ローカルデータベースへの情報の書き込み (モジュール、FE チップ、品質試験情報).

1084 情報取得のイメージを表 7.7 に示す。このように Quad モジュールの場合、ダウンロードの流れの中で
1085 合計して 7 回、データベース API を用いて情報取得を行う。

1086 結果を表 7.6 に示す。

1087 この結果より、各構成部品情報の取得 (モジュール、ベアモジュール、FE チップ) の取得にそれぞれ均
1088 等に処理時間がかかっていることがわかった。

1089 7.2.6 生産時における見積もり

1090 現在ダウンロード機能のオプションとして、以下の 2 つを実装している。

- 1091 1. モジュール 1 つをダウンロードする機能.
- 1092 2. 登録されている全てのモジュールの一括ダウンロード機能.

1093 オプション 1 の見積もり値は、表 7.5 の平均値として 4.0 ± 0.4 [sec] となる。オプション 2 の見積もり
1094 値は、生産時には最大で 10,000 台のモジュールが中央データベースに登録されることから、以下のように

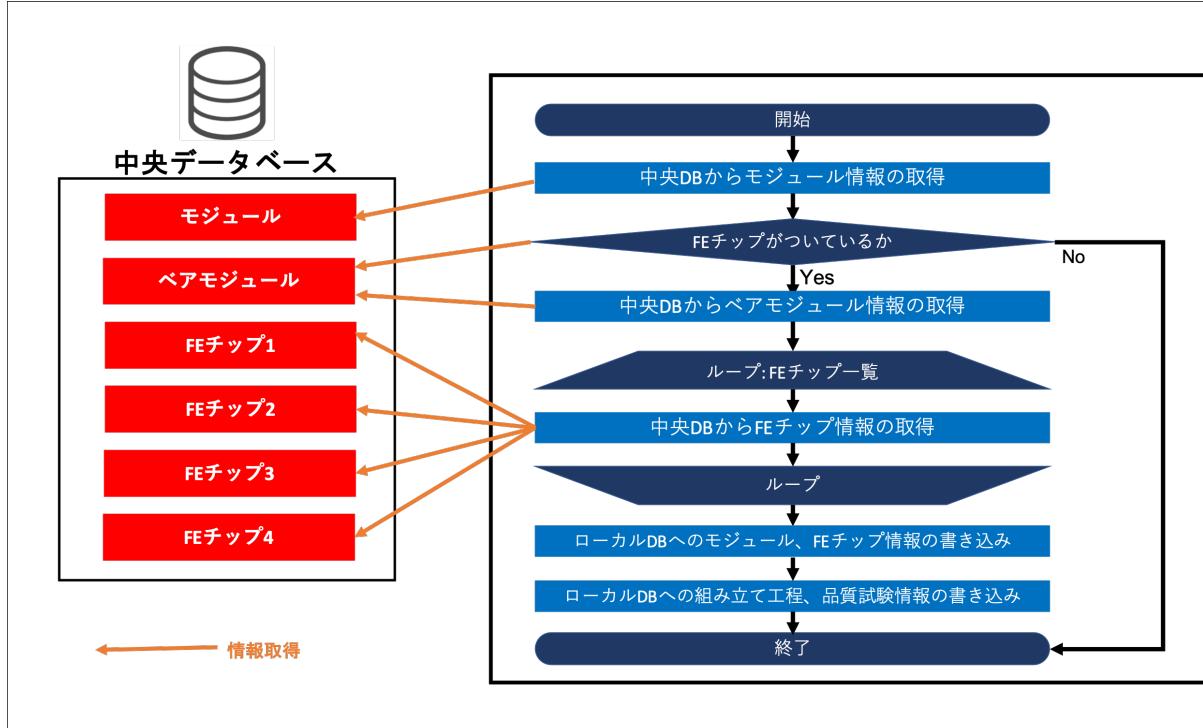


図 7.7 モジュール及び構成部品情報取得のイメージ。図のように処理の流れの中で合計 7 回中央データベースに接続し、モジュール、ベアモジュール、FE チップの情報取得を行っている。

表 7.6 ダウンロード機能における詳細処理にかかる時間測定。図 7.7 より、中央データベースに接続、情報取得を合計して 7 回行っており、それが処理 1 から 4 に対応する。どの処理においても 0.5 秒程度の時間がかかっていることが分かる。処理 5 はローカルデータベースへの書き込み処理であるが、他の処理に比べて十分に小さいことが分かる。

処理	時間
1	0.60 ± 0.07
2	0.55 ± 0.07
3	0.61 ± 0.04
4	0.46 ± 0.04 0.71 ± 0.19 0.51 ± 0.04 0.57 ± 0.12
5	0.0025 ± 0.0011
合計	4.0 ± 0.1

1. 中央データベースからモジュール情報の取得.
2. データベースでの FE チップ確認処理.
3. 中央データベースからベアモジュール情報の取得.
4. 中央データベースから FE チップ情報の取得 (4 枚分).
5. ローカルデータベースへの情報の書き込み (モジュール、FE チップ、品質試験情報).

1095 になる。

$$(4.0 \pm 0.4)[\text{sec}] \times 10,000 = 11 \pm 1[\text{hour}] \quad (7.1)$$

1096 データベース操作の流れにおいて(図4.17)、各機関で各モジュールの組み立てを始める際に中央データベースへのモジュール登録及びローカルデータベースへのダウンロードを行うことを想定している。これは1つずつ行うため、このような場合はオプション1を用いる。

1099 オプション2の使用例として、モジュールの組み立て工程の途中で、複数のモジュール($O(100)$ 程度)を機関1から機関2に輸送する場合をあげる。機関2では機関1で登録された全てのモジュール情報をダウンロードしてくる必要があるため、このオプションを使用する。輸送に要する時間として1日と仮定すると、この時間を利用してダウンロード処理を行う必要がある。

1103 7.2.7 改善点の考案と見積もり

1104 オプション2について、11時間の処理時間を要する見積もりとなっている。より円滑な同期処理に向けて、改善方法について考える。

1106 一括ダウンロード機能については以下の改善点が考えられる。

- 1107 1. モジュールの現在位置に対応したものののみのダウンロード.
- 1108 2. FEチップの登録機関を取得しない.
- 1109 3. モジュールのプロパティとして、ダウンロードに必要な情報を全て保存.
- 1110 4. データベースAPIを改良し、モジュール一覧取得の際に構成要素の情報を取得できるようにする.

1111 これらについて詳細と処理時間の見積もりを以下で行う。

1112 改善案1: モジュールの現在位置に対応したものののみのダウンロード

1113 全てのモジュール情報をダウンロードする必要はなく、中央データベースではモジュールの現在位置情報保持しているため、機能実行者と位置が同じもののみをダウンロードするアルゴリズムにすれば処理時間は改善できる。見積もりとしては、ダウンロード対象となるモジュール数を n とすると、以下のようになる。

$$(11 \pm 1) \times \frac{n}{10000}[\text{hour}] \quad (7.2)$$

1117 改善案2: FEチップの登録機関を取得しない

1118 表7.7よりダウンロードの際に、FEチップの情報取得を行っている。これはFEチップ登録機関の情報を取得しローカルデータベースに保存するためであるが、登録機関の情報は組み立て現場で扱う作業としては、必要な情報ではない。そのため、現段階ではFEチップのデータ取得処理は割愛することができる。これにかかる処理時間は表7.6より、合計して $2.3 \pm 0.2[\text{sec}]$ となるため、その場合オプション2の処理時間の見積もりは、以下のようになる。

$$\{(4.0 \pm 0.4) - (2.3 \pm 0.2)\}[\text{sec}] \times 10,000 = 4.9 \pm 0.8[\text{hour}] \quad (7.3)$$

1123 この改善策のデメリットとしては、FEチップの情報取得処理を省くとローカルデータベースで扱いたい情報が将来的にできた場合に保存できないことである。例えば各FEチップの最適動作電圧のようにモジュール読み出しに対して有益な情報は保存し、迅速に確認したいという方針になることがあげられる。

1126 改善案3: モジュールのプロパティとして、ダウンロードに必要な情報を全て保存

1127 モジュールのプロパティとして、FEチップの名前等のダウンロードに必要な情報を書いておくと、表
1128 7.2におけるlistComponentsによるモジュール一覧取得の際にその情報を参照することができる。こう
1129 することで、表7.7において、ペアモジュールやFEチップの情報取得を省くことができる。これらの処
1130 理時間は、合計して $2.9 \pm 0.2[\text{sec}]$ となるため、その場合オプション2の処理時間の見積もりは、

$$\{(4.0 \pm 0.4) - (2.9 \pm 0.2)\}[\text{sec}] \times 10,000 = 3.1 \pm 0.8[\text{hour}] \quad (7.4)$$

1131 このデメリットは、データベースの中でデータが冗長になってしまうことである。FEチップの名前
1132 情報がモジュールのプロパティにも保存されていると、データベース内部で冗長性を持ってしまい、編集
1133 が加えられた場合などこれを管理するのが難しくなる。

1134 改善案4:データベースAPIを改良し、モジュール一覧取得の際に構成要素の情報を取得できるようにする

1135 現在、表7.2のlistComponentsを用いた時にはモジュール一覧の情報は取得できるが、各モジュール
1136 に対して構成要素の情報は取得できない。そのため表7.7のようにモジュールごとに中央データベースに
1137 接続し、部品情報を取得している。ダウンロードに必要な情報をlistComponentsで一括で取得できるよ
1138 うな仕様にAPIの変更を行えば、中央データベースへの接続は一回ですみ、処理時間を削減できると考え
1139 られる。この場合、中央データベースの内部構造を知り、一括で取得しデータ送信をする場合にどれだ
1140 けの時間を要するかを見積もり、今の場合と比較する必要がある。

1141

1142 現段階では組み立ての試験段階であり、現場で必要な情報、世界各地での組み立て工程の流れ等を検討
1143 している段階である。ここで述べたような改善策を組み合わせ、変更を加えていく必要がある。また中央
1144 データベースとの同期におけるネットワーク速度は地理的な距離により異なるため、この点も考慮に入れ
1145 る必要がある。今回の測定は日本での測定であるため、他の国においては処理速度は改善すると考えられる。

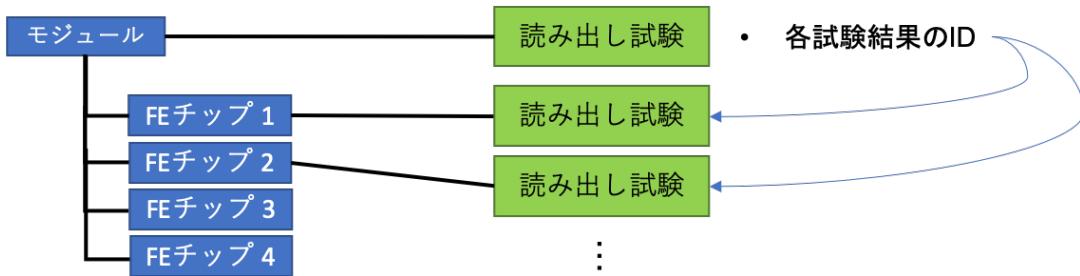


図 7.8 中央データベースにおける読み出し試験結果の構造。YARR の出力ファイル及びローカルデータベースのデータ構造において、読み出し試験結果は全て FE チップに紐つけられている。そのため、図のように中央データベースにおいてもこのデータ構造を保持する形でアップロードを行う。モジュールの結果は各 FE チップの試験結果に対する ID を持つことで紐付けを行っている。

表 7.7 中央データベースにおける読み出し試験結果に関する情報一覧。モジュール及び FE チップが中央データベース内で持つ試験結果の情報を示している。

部品	試験情報、結果	添付ファイル
モジュール	モジュール環境温度 FE チップにつく読み出し試験結果の ID	
FE チップ	ピクセル解析結果	試験結果データファイル 読み出し設定ファイル その他設定ファイル

1146 7.3 読み出し試験結果のアップロード

1147 4 章で述べたように、読み出し試験結果について中央データベースへアップロードするツールを開発し
1148 た。以下で詳細を述べる。

1149 7.3.1 アップロードする情報とその構造

1150 読み出し試験結果について、中央データベースにアップロードする情報を以下に記す。

- 1151 • 試験日時.
- 1152 • モジュール周りの環境温度.
- 1153 • ピクセル解析結果.
- 1154 • 各試験結果データファイル.
- 1155 • 読み出し設定ファイル.
- 1156 • その他設定ファイル (DB、ユーザ、組み立て機関等).

1157 中央データベースにおける読み出し試験の構造に関して、YARR を用いて行った読み出し結果は全て
1158 FE チップ毎に取得、保存される。そのため、データベースの内部でも FE チップに読み出し試験結果を
1159 紐づける構造を設け、モジュールの結果では各 FE チップの結果ページの ID を持つ構造とした。イメー
1160 ジを図 7.8 に示す。

1161 中央データベースにおいてモジュール、FE チップの試験結果が持つ情報を表 7.7 にまとめた。

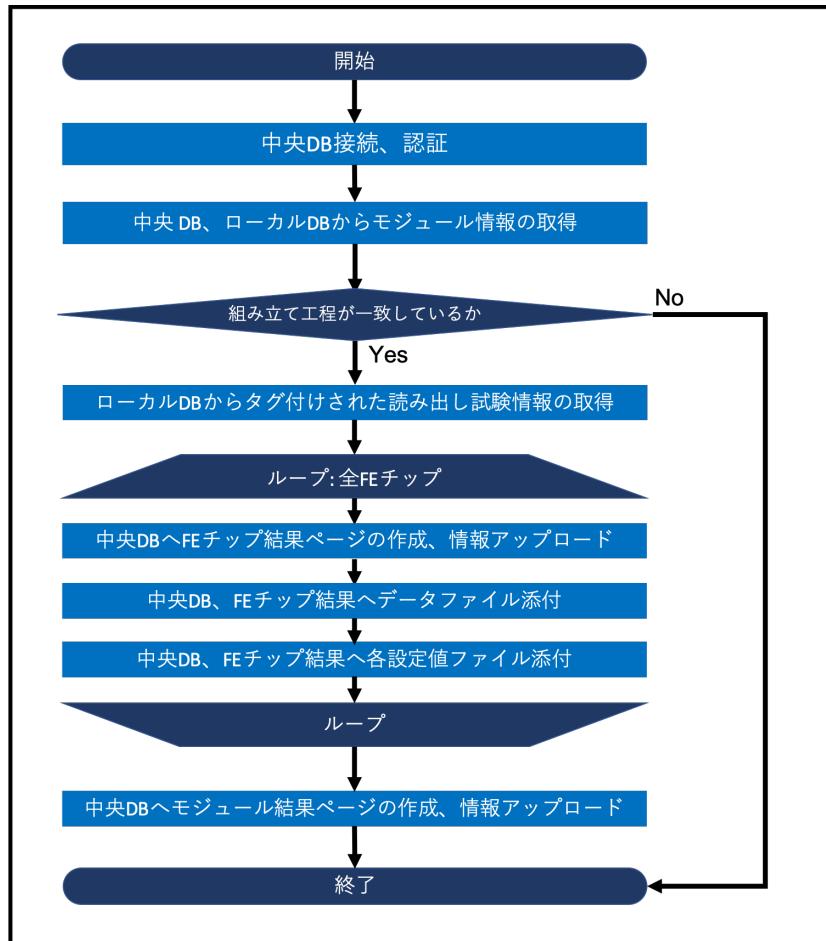


図 7.9 アップロード処理に関する流れのイメージ図。流れの中には FE チップに関するループ構造があり、ここで FE チップの結果生成、結果ファイルのアップロードを行う。最後にモジュールに対する結果生成とアップロードを行う。

1162 7.3.2 処理の流れ

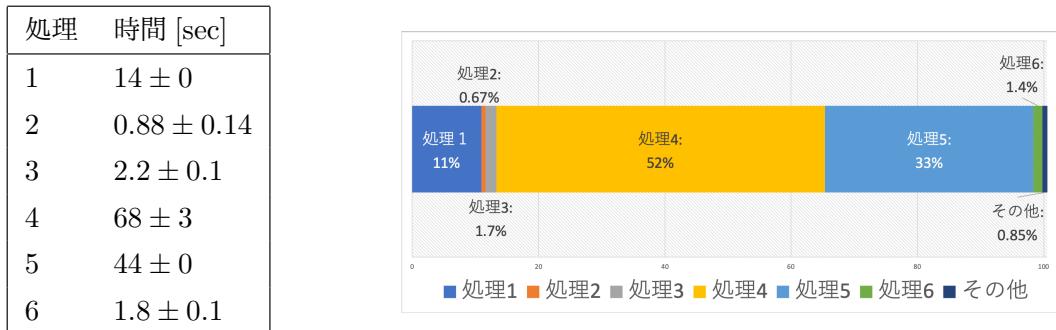
1163 アップロード機能における処理の流れのイメージを図 7.9 に示す。流れの中には共通して行われる処理
1164 と、各 FE チップに対して行われる処理がある。

1165 7.3.3 処理時間測定

1166 上述した処理のツールを開発し、処理時間測定を行った。ここで行ったアップロードする試験項目は 5
1167 章で行ったデモンストレーションのものと同じで以下の項目とする。

- 1168 • デジタル回路読み出し.
- 1169 • アナログ回路読み出し.
- 1170 • Threshold 測定.
- 1171 • ToT 測定.
- 1172 • ノイズ占有率測定.

1173 これらの項目についてアップロードを行い、その処理時間を測定した。KEK のサーバーを用いてアッ



1. 中央データベース接続、認証。
2. 中央データベース、ローカルデータベースからモジュール情報の取得。
3. 中央データベースに FE チップ結果ページの作成。結果情報をアップロード。
4. 2で作成した結果に対して、各データファイルを添付。
5. 2で作成した結果に対して、各設定値ファイルを添付。
6. 中央データベースにモジュールの結果ページの作成、結果情報をアップロード。

図 7.10 アップロード機能における詳細処理測定結果。アップロード機能において、各詳細処理時間測定した結果である。左図は測定値であり、右図はそれぞれの割合を示したものである。右図より、処理 4、5 の結果ファイルの添付、設定値ファイルの添付に多く時間がかかっていることが分かる。

₁₁₇₄ プロード処理を 20 回行い、5 項目全体でかかる時間を測定した。以下のようにになった。

$$(1.3 \pm 0.0) \times 10^2 [\text{sec}] \quad (7.5)$$

₁₁₇₅ ここで処理流れの表 7.9 より特に以下の詳細処理を抜粋し、それぞれにかかる時間を測定した。

1. 中央データベース接続、認証。
2. 中央データベース、ローカルデータベースからモジュール情報の取得。
3. 中央データベースに FE チップ結果ページの作成。結果情報をアップロード。
4. 2で作成した結果に対して、各データファイルを添付。
5. 2で作成した結果に対して、各設定値ファイルを添付。
6. 中央データベースにモジュールの結果ページの作成、結果情報をアップロード。

₁₁₈₂ 結果を表 7.10 に示す。結果データや各設定値のファイル添付に大きく時間がかかっていることが分かった。

₁₁₈₄ 生産時における見積もり

₁₁₈₅ Quad モジュールにおける読み出し試験結果アップロード処理合計時間の見積もりを行った。上述した
₁₁₈₆ 測定は SCC であるため FE チップに対する処理は 1 回であるため、Quad モジュールの場合は表 7.10 を
₁₁₈₇ 用いて以下のように計算できる。

$$\begin{aligned} \text{FE チップ処理} &: \left\{ (2.2 + 68 + 44) \pm \sqrt{(0.1)^2 + 3^2 + 0^2} \right\} \times 4 \\ &= (4.6 \pm 0.1) \times 10^2 [\text{sec}] \end{aligned} \quad (7.6)$$

$$\begin{aligned} \text{合計} &: (4.6 \pm 0.1) \times 10^2 [\text{sec}] + (14 \pm 0) + (0.88 \pm 0.14) + (1.8 \pm 0.1) \\ &= 7.9 \pm 0.1 [\text{min}] \end{aligned} \quad (7.7)$$

モジュール読み出し試験 1 回に対して、約 8 分程度かかる見積もりとなった。円滑なモジュール組み立て、データ管理を行うために同期は速やかに行われることが要求されること、外観検査や平坦性測定のように他の品質試験も同期する必要があることを考慮し、処理時間の改善を試みた。詳細を以下で示す。

7.3.4 改善策

図 7.10 より処理 4、5 のファイル添付に大きく要していることが分かる。この現状を踏まえ、各ファイルにおける添付処理時間の測定を行った。測定は上述したものと同様に KEK サーバーを用いて合計 20 回行った。添付する結果ファイル、設定ファイルの種類とデータ容量、添付処理実行結果、処理時間を表 7.8 に示す。ここで 4MB を超える容量のファイル添付は失敗していることがわかり、アップロード機能の問題点を発見した。

表 7.8: アップロード処理における結果、設定値ファイル添付実行結果と処理時間。図 7.10 より、読み出し試験に対して出力される各ファイルのアップロード実行結果、データ容量と処理時間をまとめた。ここで扱うファイルサイズの合計は 94MB である。図よりファイルのデータ容量が大きいほど処理時間が長いことが分かる。`std_thresholdscan` のようにファイル数が多い項目の場合、合計して大きい処理時間を要することが分かる。全項目において読み出しの設定ファイルにあたる `beforeCfg_chipCfg.json`, `afterCfg_chipCfg.json` のアップロードは、中央データベースの容量制限により失敗していることが分かる。

読み出し項目	ファイル名	実行結果	容量 [KB]	処理時間 [sec]	全体 [sec]
std_digitalscan	EnMask.json	Ok	1,300	3.3 ± 0.1	17±0
	OccupancyMap.json	Ok	1,500	2.9 ± 0.2	
	L1Dist.json	Ok	0.53	0.75 ± 0.12	
	ctrlCfg_ctrlCfg.json	Ok	0.46	0.61 ± 0.08	
	dbCfg_dbCfg.json	Ok	0.60	0.69 ± 0.16	
	siteCfg_siteCfg.json	Ok	0.033	0.61 ± 0.06	
	userCfg_userCfg.json	Ok	0.14	0.66 ± 0.09	
	scanCfg_std_digitalscan.json	Ok	2.2	0.55 ± 0.06	
	beforeCfg_chipCfg.json	Error	7,200	3.0 ± 0.2	
	afterCfg_chipCfg.json	Error	7,200	4.0 ± 0.2	
std_analogscan	EnMask.json	Ok	1,300	3.9 ± 0.1	17±0
	OccupancyMap.json	Ok	1,400	2.6 ± 0.1	
	L1Dist.json	Ok	0.60	0.69 ± 0.16	
	ctrlCfg_ctrlCfg.json	Ok	0.46	0.54 ± 0.05	
	dbCfg_dbCfg.json	Ok	0.60	0.49 ± 0.04	
	siteCfg_siteCfg.json	Ok	0.033	0.48 ± 0.04	
	userCfg_userCfg.json	Ok	0.14	0.58 ± 0.08	
	scanCfg_std_analogscan.json	Ok	2.1	0.45 ± 0.03	
	beforeCfg_chipCfg.json	Error	7,200	2.9 ± 0.2	
	afterCfg_chipCfg.json	Error	7,200	3.9 ± 0.3	
std_thresholdscan	Scurve-30-96.json	Ok	0.98	1.3 ± 0.1	49±1
	Scurve-110-96.json	Ok	0.98	0.45 ± 0.03	
	Scurve-70-96.json	Ok	0.98	0.47 ± 0.04	
	Scurve-150-96.json	Ok	1.0	0.46 ± 0.04	
	Scurve-190-96.json	Ok	1.0	0.64 ± 0.12	
	Scurve-230-96.json	Ok	1.0	0.49 ± 0.03	
	Scurve-270-96.json	Ok	1.0	0.47 ± 0.04	
	Scurve-310-96.json	Ok	1.0	0.47 ± 0.04	
	Scurve-350-96.json	Ok	1.0	0.49 ± 0.03	
	Scurve-390-96.json	Ok	1.0	0.52 ± 0.06	
	Scurve-40-96.json	Ok	1.0	0.46 ± 0.03	
	Scurve-80-96.json	Ok	0.99	0.68 ± 0.13	
	Scurve-120-96.json	Ok	1.0	0.54 ± 0.07	
	Scurve-160-96.json	Ok	1.0	0.51 ± 0.05	
	Scurve-200-96.json	Ok	1.0	0.49 ± 0.04	
	Scurve-240-96.json	Ok	1.0	0.50 ± 0.05	
	Scurve-280-96.json	Ok	1.0	0.48 ± 0.04	
	Scurve-320-96.json	Ok	1.0	0.49 ± 0.05	
	Scurve-360-96.json	Ok	1.0	0.49 ± 0.06	
	Scurve-400-96.json	Ok	1.0	0.45 ± 0.05	
	Scurve-10-96.json	Ok	1.0	0.42 ± 0.03	
	Scurve-50-96.json	Ok	0.99	0.49 ± 0.05	
	Scurve-90-96.json	Ok	0.99	0.46 ± 0.05	
	Scurve-130-96.json	Ok	1.0	0.47 ± 0.05	
	Scurve-170-96.json	Ok	1.0	0.52 ± 0.04	
	Scurve-210-96.json	Ok	1.0	0.51 ± 0.04	
	Scurve-250-96.json	Ok	1.0	0.58 ± 0.10	
	Scurve-290-96.json	Ok	1.0	0.64 ± 0.13	
	Scurve-330-96.json	Ok	1.0	0.64 ± 0.09	
	Scurve-370-96.json	Ok	1.0	0.49 ± 0.06	

表 7.8: アップロード処理における結果、設定値ファイル添付実行結果と処理時間。図 7.10 より、読み出し試験に対して出力される各ファイルのアップロード実行結果、データ容量と処理時間をまとめた。ここで扱うファイルサイズの合計は 94MB である。図よりファイルのデータ容量が大きいほど処理時間が長いことが分かる。`std.thresholdscan` のようにファイル数が多い項目の場合、合計して大きい処理時間を要することが分かる。全項目において読み出しの設定ファイルにあたる `beforeCfg_chipCfg.json`, `afterCfg_chipCfg.json` のアップロードは、中央データベースの容量制限により失敗していることが分かる。

	Scurve-60-96.json	Ok	0.99	0.51 ± 0.06
	Scurve-100-96.json	Ok	1.0	0.48 ± 0.05
	Scurve-140-96.json	Ok	1.0	0.48 ± 0.06
	Scurve-180-96.json	Ok	1.0	0.52 ± 0.06
	Scurve-220-96.json	Ok	1.0	0.54 ± 0.05
	Scurve-260-96.json	Ok	1.0	0.51 ± 0.05
	Scurve-300-96.json	Ok	1.0	0.66 ± 0.09
	Scurve-340-96.json	Ok	1.0	0.51 ± 0.06
	Scurve-380-96.json	Ok	1.0	0.55 ± 0.05
	sCurve-0.json	Ok	49	1.0 ± 0.1
	ThresholdDist-0.json	Ok	4.6	0.56 ± 0.06
	ThresholdMap-0.json	Ok	2,200	4.3 ± 0.1
	NoiseDist-0.json	Ok	2.3	0.42 ± 0.04
	Chi2Map-0.json	Ok	2,300	4.5 ± 0.1
	StatusMap-0.json	Ok	1,300	2.8 ± 0.1
	StatusDist-0.json	Ok	0.49	0.48 ± 0.04
	NoiseMap-0.json	Ok	2,200	4.0 ± 0.2
	Chi2Dist-0.json	Ok	1.1	0.50 ± 0.04
	TimePerFitDist-0.json	Ok	3.1	0.56 ± 0.12
	ctrlCfg_ctrlCfg.json	Ok	0.46	0.49 ± 0.05
	dbCfg_dbCfg.json	Ok	0.60	0.52 ± 0.06
	siteCfg_siteCfg.json	Ok	0.033	0.54 ± 0.07
	userCfg_userCfg.json	Ok	0.14	0.60 ± 0.07
	scanCfg_std_thresholdscan.json	Ok	2.2	0.46 ± 0.03
	beforeCfg_chipCfg.json	Error	7,200	3.2 ± 0.2
	afterCfg_chipCfg.json	Error	7,200	3.5 ± 0.1
std_totscan	MeanTotMap-0.json	Ok	1,900	4.8 ± 0.1
	SigmaTotMap-0.json	Ok	2,200	4.1 ± 0.2
	MeanTotDist-0.json	Ok	0.59	0.54 ± 0.07
	SigmaTotDist-0.json	Ok	1.8	0.47 ± 0.03
	L1Dist.json	Ok	0.59	0.60 ± 0.08
	ctrlCfg_ctrlCfg.json	Ok	0.46	0.47 ± 0.03
	dbCfg_dbCfg.json	Ok	0.60	0.67 ± 0.24
	siteCfg_siteCfg.json	Ok	0.033	0.59 ± 0.10
	userCfg_userCfg.json	Ok	0.14	0.56 ± 0.05
	scanCfg_std_totscan.json	Ok	2.0	0.59 ± 0.10
std_noisescan	beforeCfg_chipCfg.json	Error	7,200	3.0 ± 0.1
	afterCfg_chipCfg.json	Error	7,200	3.9 ± 0.3
	Occupancy.json	Ok	1,300	4.0 ± 0.1
	NoiseOccupancy.json	Ok	1,300	2.5 ± 0.1
	NoiseMask.json	Ok	1,300	2.4 ± 0.1

1197 表 7.8 より、データ容量の大きいものにアップロード時間がかかっていることがわかる。また添付処理
 1198 を行うオフセットがあることから、`std_thresholdscan` のように各容量が大きくなくてもファイル数が
 1199 多いものにはアップロード時間が合計して多くかかってしまうことがわかる。

1200 これらのことと 4MB 以上の容量を持つファイル添付処理の失敗をなくすために、次のような改善策を
 1201 考えた。

- 1202 • 各試験項目に対する結果データ、設定ファイルをそれぞれ Zip ファイルに統合し、圧縮後にアップ
 1203 ロードを行う。

1204 こうすることで、アップロードするファイルの容量、数共に削減することができる。圧縮率によっては
 1205 アップロード処理の失敗もなくすことができると考えた。

1206 これを踏まえアップロードツールを改良し、再び各ファイルの添付処理にかかる時間を測定した。合計
 1207 処理時間は以下のようにになり、全てのファイルのアップロードに成功した。

$$36 \pm 1[\text{sec}] \quad (7.8)$$

表 7.9: アップロード処理改善後における結果、設定値ファイル添付実行結果と処理時間。各試験結果毎に結果ファイル、設定値ファイルを Zip ファイルにまとめアップロードする処理とした。これにより、ファイル数、容量の削減に成功し、アップロード時間が改善した。全てのファイルのアップロードに成功していることが分かる。

読み出し項目	ファイル名	実行結果	容量 [KB]	処理時間 [sec]	全体 [sec]
<code>std_digitalscan</code>	<code>std_digitalscan_datafiles.zip</code>	Ok	10	0.77 ± 0.18	1.9 ± 0.2
	<code>std_digitalscan_configfiles.zip</code>	Ok	56	1.1 ± 0.1	
<code>std_analogscan</code>	<code>std_analogscan_datafiles.zip</code>	Ok	46	1.0 ± 0.2	2.2 ± 0.3
	<code>std_analogscan_configfiles.zip</code>	Ok	58	1.2 ± 0.2	
<code>std_thresholdscan</code>	<code>std_thresholdscan_datafiles.zip</code>	Ok	1,500	2.6 ± 0.1	3.5 ± 0.1
	<code>std_thresholdscan_configfiles.zip</code>	Ok	190	0.86 ± 0.08	
<code>std_totscan</code>	<code>std_totscan_datafiles.zip</code>	Ok	730	1.7 ± 0.2	2.5 ± 0.2
	<code>std_totscan_configfiles.zip</code>	Ok	190	0.83 ± 0.15	
<code>std_noisescan</code>	<code>std_noisescan_datafiles.zip</code>	Ok	19	0.56 ± 0.07	1.7 ± 0.1
	<code>std_noisescan_configfiles.zip</code>	Ok	190	1.1 ± 0.1	

1208 生産時における見積もり

1209 上記の見積もりと同様に、改善後のツールにおけるアップロード時間の見積もりを行った。表 7.10 に
 1210 おいて、添付処理に対応する処理 4、5 以外は同じとする。改良後の処理 4、5 の処理時間は、

$$\begin{aligned} \text{FE チップ処理} : & \left\{ (2.2 + 6.7 + 5.1) \pm \sqrt{(0.1)^2 + (1.1)^2 + (0.3)^2} \right\} \times 4 \\ & = 56 \pm 5[\text{sec}] \end{aligned} \quad (7.9)$$

$$\begin{aligned} \text{合計} : & (56 \pm 5)[\text{sec}] + (14 \pm 0) + (0.88 \pm 0.14) + (1.8 \pm 0.1) \\ & = 1.2 \pm 0.1[\text{min}] \end{aligned} \quad (7.10)$$

1211 約 1 分でアップロードを完了できる見積もりとなった。改善前に比べて 15% の処理時間となった。現
 1212 在は改善後の方針を用いたツールを提供している。

1213 ファイル添付以外の処理

1214 ここではファイル添付処理に関する処理時間検討を行なったが、以下にあげたそれ以外の処理について言及する。

- 1216 1. ローカルデータベース内部処理.
- 1217 2. 中央データベースへの接続、認証処理.
- 1218 3. ファイル添付以外の処理に関する、中央データベース API 使用.
- 1219 4. ネットワーク速度の改善.

1220 項目1に関して、読み出し試験におけるローカルデータベースの内部構造は世界的に使われているものである。内部構造の変更をすることなく改善を図りたいという開発方針から、この項目に関する処理時間削減の検討を行わなかった。また図7.10よりローカルデータベース内部処理は十分に小さいものであつたため、検討の必要がないと考えた。

1224 項目2に関して、中央データベースへの接続、認証処理は中央データベースのAPI及び中央データベースの内部構造によるものである。この部分の処理を改善するには、これらの変更を検討する必要がある。

1226 項目3に関して、アップロード機能ではファイル添付以外に、以下の処理の際に中央データベースのAPIを使用している。

- 1228 • モジュール情報の取得.
- 1229 • 結果ページの作成(モジュールとFEチップで5回分).

1230 これらの処理は必要であるため、APIの使用回数の減らし処理時間を削減することは難しい。

1231 項目4に関して、中央データベースと通信する全ての処理はネットワーク速度に依存していると考えられる(付録E)。サーバーが置かれている位置やネットワーク環境の改善により、本ツールの処理速度も改善すると考えられる。

1234 第8章

1235 まとめ

1236 8.1 本論文のまとめ

1237 HL-LHCに向けてATLAS内部飛跡検出器の総入れ替えを予定しており、これに向けてピクセルモ
 1238 デュールを世界で10,000台生産する予定である。各モジュールに対して品質試験を行い、全てのモジュール
 1239 及び品質試験の結果は中央データベースに保存する。

1240 本研究では、この生産及び品質試験に向けてデータベースシステムの構築を行った。各組み立て機関に
 1241 てデータ管理をするローカルデータベースを確立し、品質試験結果検索や中央データベースとの同期機能
 1242 など、生産時に必要となる諸ツールの開発を行った。

1243 開発した諸ツールを含め、生産において必要な機能の確認を行った。本番を想定したソフトウェア、
 1244 ハードウェアのセットアップと各ツールの処理を実行し、機能が使用可能であることを確認した。

1245 主な開発項目の1つ目として品質試験検索機能を述べた。開発当初はデータベース内部構造により、
 1246 試験結果数 n に対して処理時間が $O(n^2)$ かかってしまう問題が発生した。MongoDB 内に新しいコレク
 1247 ションを設け検索に必要な情報を予め1つの場所に保持しておくことにより、処理時間の改善に成功し
 1248 た。実際に処理時間の測定を行い、データ数の増加に対しても検索機能が不都合なく使えることを確認し
 1249 た。本番を想定した見積もりを行い、84,000件のデータ数に対して $2.6 \pm 0.1[\text{sec}]$ で処理が実行できる見
 1250 込みであり、生産時において十分に有用な機能であることを確認した。

1251 2つ目に中央データベースとローカルデータベースの同期ツールを開発した。世界的に使われるツール
 1252 であり、全ての組み立て機関でこのツールをサポートするために中央データベースへの通信処理時間調査
 1253 を KEK、LBL、CERN のサーバーを用いて行った。KEK のサーバーを用いた場合に最も時間がかかる
 1254 ことを確認し、このサーバーにおいて十分に使うことができる機能開発を達成すれば世界的に問題がない
 1255 と考えた。開発した中央データベース同期ツールについて KEK サーバーを用いて処理速度測定を行っ
 1256 た。モジュール情報のダウンロード機能に関して、モジュール1つあたり $4.0 \pm 0.4[\text{sec}]$ の処理時間がか
 1257 かることを確認した。処理の詳細を調査すると、モジュールや構成部品の情報取得するために行っている
 1258 中央データベース API の使用に時間がかかっていた。処理時間の改善策をいくつか考案し、それぞれに
 1259 ついて見積もりを行った。読み出し試験結果のアップロード機能に関して、開発当初はモジュール1つに
 1260 対して結果アップロード処理時間の見積もり値が $7.9 \pm 0.1[\text{min}]$ であった。処理時間改善に向けて処理
 1261 の詳細を調査したところ、結果ファイルの添付処理に大きく時間が要していることが分かった。ファイル
 1262 添付についての詳細を調べると、添付処理時間がファイル数とファイル容量に依存していた。このこと
 1263 から結果ファイルを ZIP ファイルにまとめ、圧縮しアップロードを行うことで処理時間の改善を図った。
 1264 ここで圧縮前後のファイル容量は、それぞれ 94、3.9[MB/1FE チップ] となった。結果として処理時間の

1265 改善に成功し、その見積もり値が $1.2 \pm 0.1[\text{min}]$ となった。またファイル添付以外の処理に関して、時間
1266 削減の余地がないかを検討した。

1267 8.2 現状と今後の課題

1268 8.2.1 ソフトウェアリリースとユーザサポート

1269 本論文で述べたツールの他に、読み出し試験コマンド統括ソフト、品質試験結果アップロード用ソフト
1270 などの開発もチームとして行っている。全てのソフトウェアを含めて、品質試験のデータ管理を達成する
1271 ようなアプリケーションスイートを目指している。2020年12月9日にファーストバージョンのリリース
1272 を行い、いくつかの機関で全体のシステム及びソフトウェアが使われている現状である。

1273 またCERNで行ったチュートリアルを経て、世界的に機能普及が進んでいる。そのためユーザサポート
1274 としてソフトウェア使用のためのドキュメント[40]の作成、整備も行っている。開発者の連絡先やロー
1275 カルデータベース専用掲示板へのリンクもドキュメントに記している。何か問題が生じた時などに簡単に
1276 問い合せができる仕組みを整えている。

1277 8.2.2 開発課題

1278 本研究では検索機能や同期機能など、読み出し試験を対象とした機能を重点的に開発した。ローカル
1279 データベース開発は、読み出し試験の結果を管理したいという要求から始まり、現在はそれ以外の品質試
1280 験も含め、全ての結果や組み立て工程の管理も目標としている。今後の開発課題として以下の機能をあ
1281 げる。

- 1282 • 読み出し試験以外(外観検査、平坦性測定等)の結果同期機能.
- 1283 • 中央データベースからローカルデータベースへ品質試験結果の同期.
- 1284 • 品質試験結果解析とモジュール選別機能.
- 1285 • 組み立て工程管理を世界的にサポート.

1286 最後の項目に関して、モジュールの組み立て工程は各機関ごとに異なるため、全ての現場における工程
1287 を調査しそれをサポートするシステムを実装する必要がある。例えば、日本では「ベアモジュール・フレ
1288 キシブル基板貼り付け」工程の後に、モジュール冷却のための構造である「セル搭載」を行う予定であり、
1289 これは他の組み立て機関とは異なる。各地域における柔軟な生産を許しているため、組み立て工程は世界
1290 的に細かく統一されていない。データベースシステムでは多様な組み立て工程に対応できるシステムとす
1291 る必要がある。

1292 このシステム実装において、中央データベースには選択可能な組み立て工程を定義する機能が存在し、
1293 これを使用することを考えている。そのイメージを図8.1に示す。

1294 以下の開発項目をあげる。

- 1295 • 中央データベースにおいて選択可能な工程定義機能を使い、全ての組み立て機関における工程をサ
1296 ポートする構造を設計、実装.
- 1297 • 本研究で開発した同期ツールを拡張、ローカルデータベース上にも中央データベースと同様の組み
1298 立て工程構造を保持.

1299 これらを達成することにより、全ての機関における組み立て工程情報の管理ができるようになると考え



図 8.1 中央データベースにおける選択可能な組み立て工程のイメージ。図に示すように中央データベースでは選択可能な組み立て工程を定義することができる。図ではセル搭載が選択可能となっていて、ペア・基板貼り付け工程の後に、どちらの工程に進むのかを選択できる。この機能を用いて世界的に多様な組み立て工程をサポートすることを考えている。

1300 ている。

1301 付録 A

1302 シリコン検出器の原理

1303 A.1 半導体 [12]

1304 固体は、絶縁体、半導体、導体の 3 つに大別できる。物質の電気伝導度に関して、絶縁体は非常に低い
 1305 値 ($10^{-18} \sim 10^{-8}$ S/cm)、導体は高い値 ($10^4 \sim 10^6$ S/cm) を持つ。半導体の電気伝導度はこれらの中間
 1306 であり、温度、光、磁界および微量の不純物に対し非常に敏感である。この特徴のために半導体はエレク
 1307 トロニクスにおける最も重要な材料の 1 つになっている。半導体は元素半導体と化合物半導体に分けら
 1308 れ、多くの物質がその候補となる。元素半導体の中で代表的なものとして Si があげられ、ATLAS ピク
 1309 セル検出器に使われる半導体は Si がベースとなっている。不純物が入っていない、全ての原子が Si の半
 1310 導体を真性半導体と呼ぶ。真性半導体中の Si は 4 つの Si と共有結合を構成し、結晶を作る。(図 A.1)
 1311 真性半導体に対し、As などの最外殻電子を 5 つもつ原子を不純物としてドープしたものを n 型半導体、
 1312 B などの 3 つのものをドープしたものを p 型半導体と呼ぶ。それぞれキャリアとして電子、ホールを持
 1313 つことになり、キャリア移動の特性を組み合わせて様々なデバイスに応用することができる。

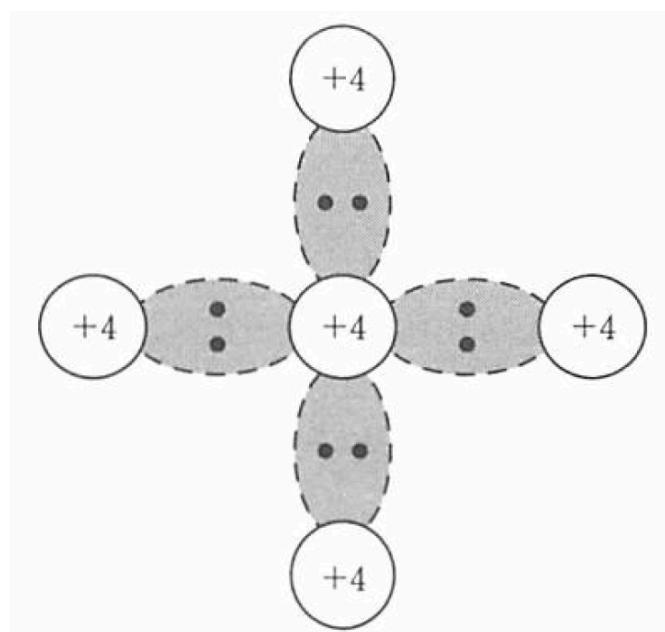


図 A.1 真性半導体中のシリコン [12]

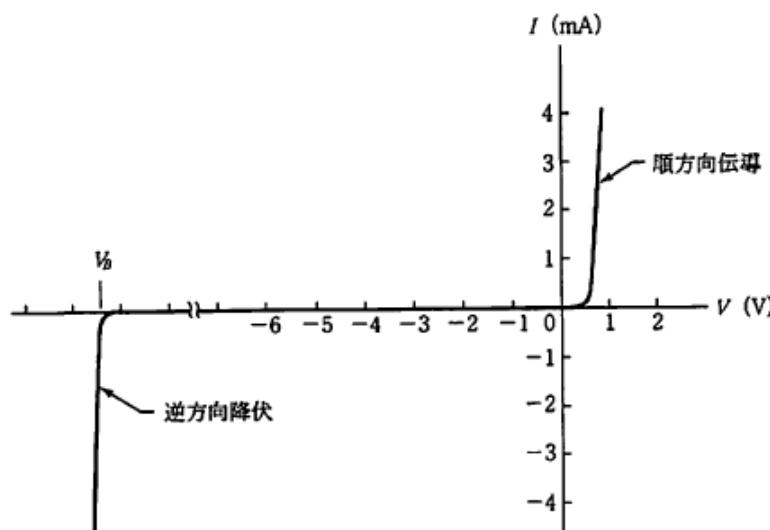


図 A.2 pn 接合の電流 - 電圧特性 [12]



図 A.3 空乏層 [12]

1314 A.1.1 pn 接合

1315 n 型半導体と p 型半導体を接合し、その接合部を pn 接合と呼ぶ。この接合は各種半導体素子で様々な
 1316 形で応用されており、ピクセル検出器にも用いられている。

1317 pn 接合の最も重要な特徴は特定の方向にだけ電流が流れやすい整流性である。図 A.2 に示すように正
 1318 電圧をかけると電流は急速に増加する。逆方向にかけた場合、始めのうちは電流はほとんど流れない。あ
 1319 る臨界電圧に達すると電流は急激に増大する。

1320 逆方向電圧をかけた場合、図 A.3 に示すように pn 接合付近はキャリアが存在しない空乏層領域が形成
 1321 される。この時、それぞれの半導体のエネルギー準位に差が生じている状態となっている。印加電圧 V
 1322 と空乏層幅 W は以下のような関係がある。

$$W \propto \sqrt{V} \quad (\text{A.1})$$

1323 A.2 検出原理

1324 荷電粒子が物質中を通過するとき、以下の Bethe-Bloch の公式によってエネルギーを損失する [13]。

$$-\left\langle \frac{dE}{dx} \right\rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left(\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{\max}}{I^2} - \beta^2 + \dots \right) \quad (\text{A.2})$$

$\frac{dE}{dx}$: 荷電粒子のエネルギー損失量 [eV · g⁻¹ · cm²]
 K : $4\pi N_A r_e^2 m_e c^2 = 0.307075$ [MeVcm²]
 z : 荷電粒子の電荷量
 Z : 物質の原子番号 (Si 14)
 A : 物質の原子量 (Si 28)
 $m_e c^2$: 電子の静止エネルギー (0.511 MeV)
 β : 光速を 1 とした入射粒子の速度
 γ : ローレンツ因子 $1/\sqrt{1 - \beta^2}$
 I : 励起エネルギーの期待値 (シリコン 137eV)

(A.3)

1325 また T_{\max} は質量 M の入射粒子による 1 つの電子への最大運動エネルギー移行であり、以下の式で書
1326 ける。

$$T_{\max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e / M + (m_e / M)^2} \quad (\text{A.4})$$

1327 荷電粒子が半導体を通過したとき、そのエネルギー損失量に応じて電子・ホール対が生成し、その量を
1328 測定することができる。

1329 例として、大きい速度 ($\beta\gamma \gg 1$) を持つ荷電粒子が、300 μm の厚さを持つシリコン半導体を通過し
1330 た時の生成キャリア数及び信号強度を計算する。

1331 荷電粒子の速度が十分に早い場合、式 A.2 は物質の密度 ρ を用いて以下のように近似できる [41]。

$$-\left\langle \frac{dE}{dx} \right\rangle = 1 \sim 2\rho [\text{MeV}/\text{cm}] \quad (\text{A.5})$$

1332 シリコンの密度を 2.33 g/cm³、シリコンにおける 1 キャリア対生成に必要な平均電離エネルギー
1333 3.62 eV より、300 μm の厚さを持つシリコン半導体を通過した時の生成キャリア数は以下のように計算
1334 できる。

$$1 \sim 2 \times 2.33 \times (3 \times 10^4) \times 1/3.62 \sim 20,000 \sim 39,000 \quad (\text{A.6})$$

1335 素電荷 $e = 1.6 \times 10^{-19}$ C より、信号強度は以下のようになる。

$$20,000 \sim 39,000 \times 1.6 \times 10^{-19} \sim (3.2 \sim 5.1) \times 10^{-15} [\text{C}] \quad (\text{A.7})$$

1336 付録 B

1337 RD53A の回路図とフレキシブル基板

1338 RD53A の各 Front-end が持つアナログ回路を図 B.1 に示す。

1339 B.1 アナログ回路

1340 B.2 試験用電荷入射のイメージ

1341 RD53A の各ピクセルが持つ試験用電荷入射回路の簡略図を図 B.3 に示す。

1342 B.3 RD53A のデータフォーマット

1343 RD53A のデータフォーマットの模式図を図 B.4 に示す。

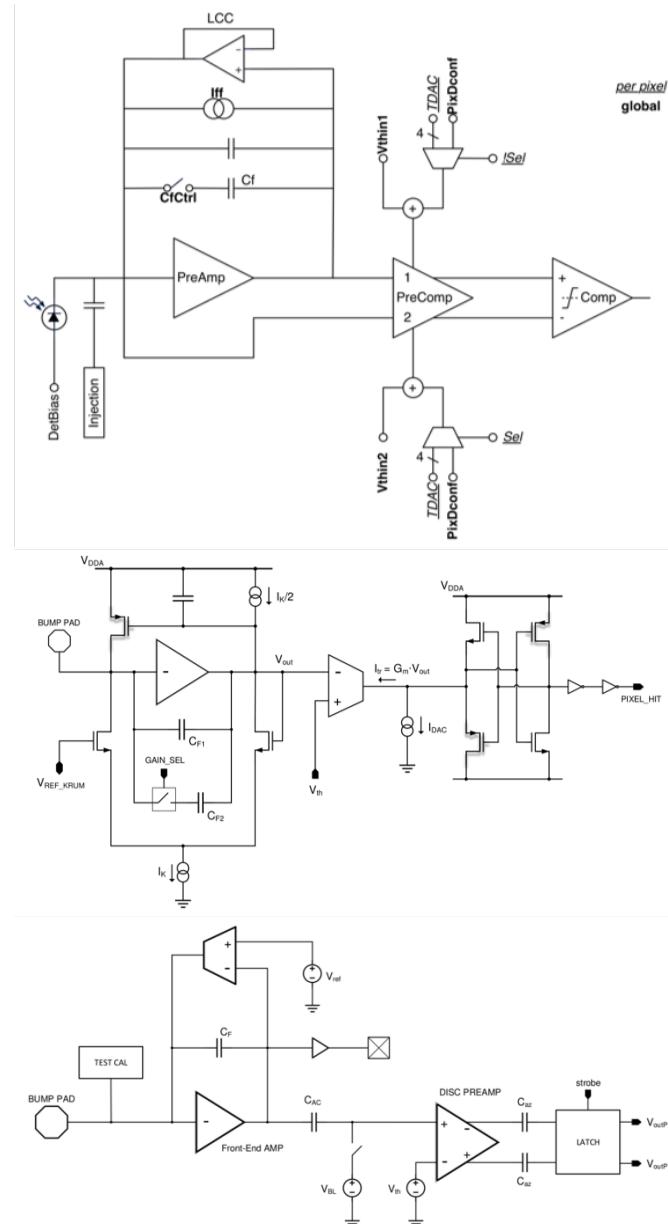


図 B.1 アナログフロントエンド [12]

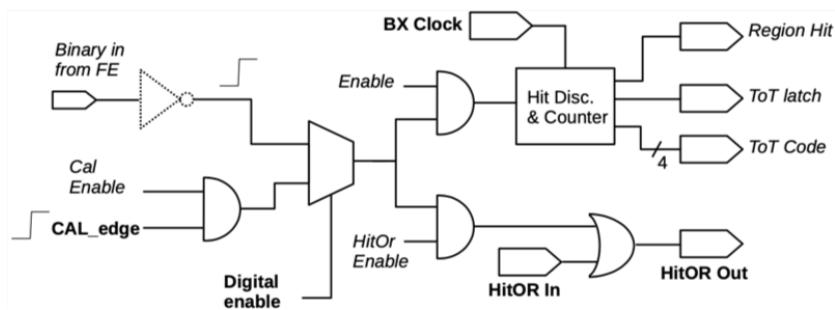


図 B.2 デジタルフロントエンド [12]

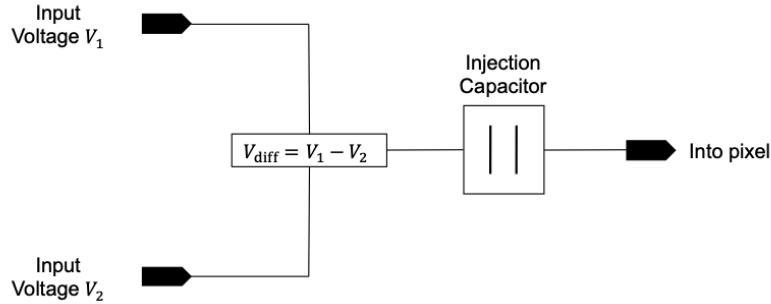


図 B.3 RD53A の各ピクセルが持つ試験用電荷入射回路の簡略図 [43]。図のように 2 つの電位を入力し、その差分の電圧を回路内のコンデンサにかける。これを開放することで、電荷をピクセル回路内に入力する。

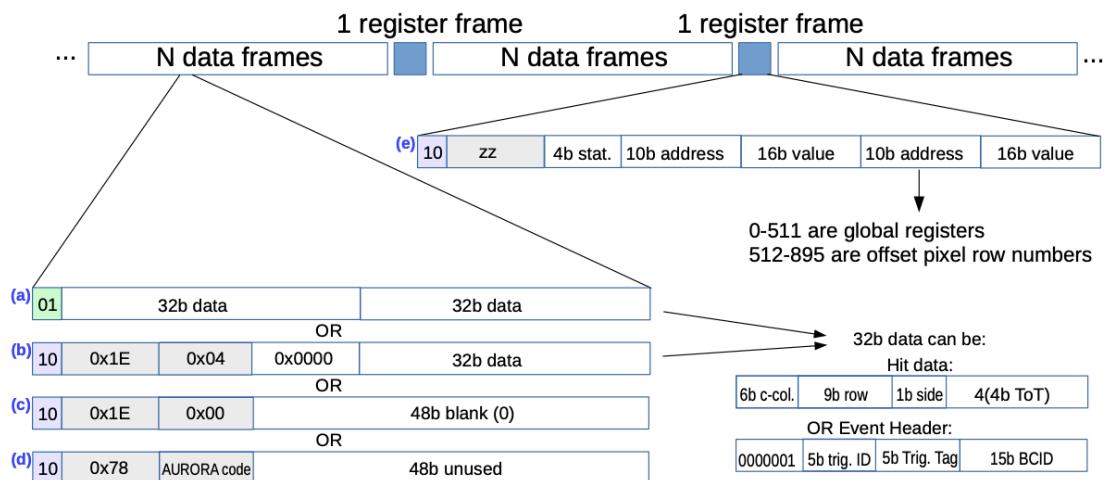


図 B.4 RD53A におけるデータフォーマットのイメージ [12]。RD53A のデータフォーマットは図のように、N 個の data frame と 1 個の register frame から成る。特にデータフレームに関しては header 部と hit data 部で構成され、header 部に trigger の情報が入っている。クロック信号や外部トリガーで送った情報はここに格納され、そのトリガーに応じてデータを取得する。

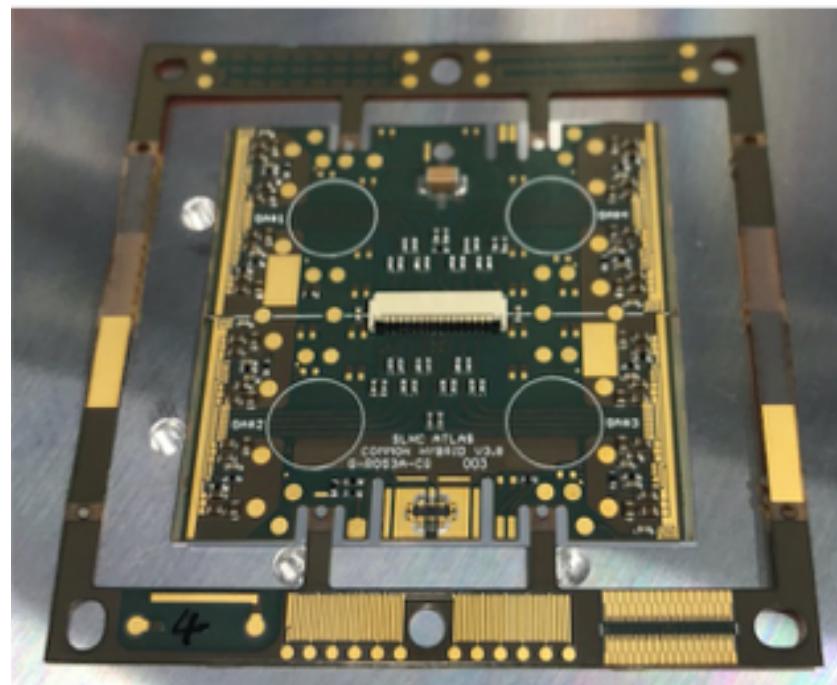


図 B.5 フレキシブル基板

1344 B.4 フレキシブル基板

付録 C

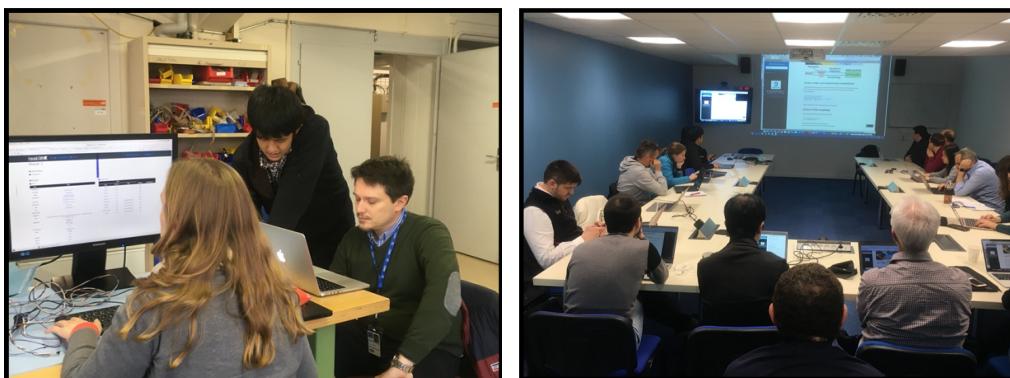
ローカルデータベースのチュートリアル と普及状況

ローカルデータベースの機能の普及を目的として、2020年2月にCERN研究所にてシステムのチュートリアルを行った。このチュートリアルは以下のような2つのセッションに分けて行った。

- 参加者が実際にサーバーの設定、各ソフトウェアのインストールを行いながら機能を実践するセッション（2月3日から6日まで）
- 私が参加者の前で実際に機能を実践し、システムや使い方に対して議論を行うセッション（2月7日）

それぞれのセッションの様子を図C.1に示す。数多くの議論を行い、有益なフィードバックを得ることができた。また品質試験の流れにおいて、一連の機能確認をすることができた。

これを経て現在ローカルデータベースは世界18箇所にて導入され、試験運用が開始している。将来的には全組み立て機関で使うことが決定しており、それに向けたシステム開発、サポートが必要となっている状況である。ローカルデータベースについて、導入及び試験運用を行っている機関を以下に示す。また世界地図をC.2に示す。



図C.1 ローカルデータベースシステムチュートリアルの様子。2020年2月にCERNでローカルデータベースシステムのチュートリアルを行った。参加者が実際にシステムの設置、機能実行を行うハンズオンセッション（左図）と、参加者の前で実際に機能の動作をみせ、議論を行うハンズオフセッション（右図）に分けて行った。システムに有益な情報を獲得したと共に、システムの機能普及に成功した。



図 C.2 ローカルデータベースシステム導入及び試運転場所。赤文字は設置している地域、括弧内の数字はその地域におけるシステム導入場所の数を示している。2020 年 11 月現在、ローカルデータベースシステムは世界 18 の機関で試験運転がなされている。日本を除いてその多くはヨーロッパとアメリカに位置していることが分かる。

- 1360 ● 高エネルギー加速器研究機構 (KEK), 日本
- 1361 ● 欧州原子核研究機構 (CERN), スイス
- 1362 ● University of Liverpool, イギリス
- 1363 ● University of Oxford, イギリス
- 1364 ● University of Glasgow, イギリス
- 1365 ● Paris-Saclay University, フランス
- 1366 ● パリ第 6 大学, フランス
- 1367 ● フランス国立科学研究中心, フランス
- 1368 ● University of Grenoble, フランス
- 1369 ● University of Gottingen, ドイツ
- 1370 ● University of Siegen, ドイツ
- 1371 ● University of Genoa, イタリア
- 1372 ● University of Salento, イタリア
- 1373 ● University of Milan, イタリア
- 1374 ● University of Udine, イタリア
- 1375 ● University of Trento, イタリア
- 1376 ● University of Oklahoma, アメリカ
- 1377 ● Argonne National Laboratory, アメリカ
- 1378 ● Lawrence Berkeley National Laboratory(LBL), アメリカ

付録 D

モジュール生産状況の解析

データベースシステムを使って、モジュール生産状況の解析を行うことができる。モジュールの組み立て工程は各組み立て機関のローカルデータベース上に記録され、組み立て工程ごとに中央データベースへ同期される。そのため全てのモジュールの現在工程を中央データベース上で取得できることができ、この情報を用いて世界的な生産状況の解析を行うことができる。現在は生産は行われていないが、想定している解析結果のイメージを図 D.1 に示す。

生産数や生産レートのモニタリングを行うことで、今後の生産計画や問題解決に役立てることが可能である。

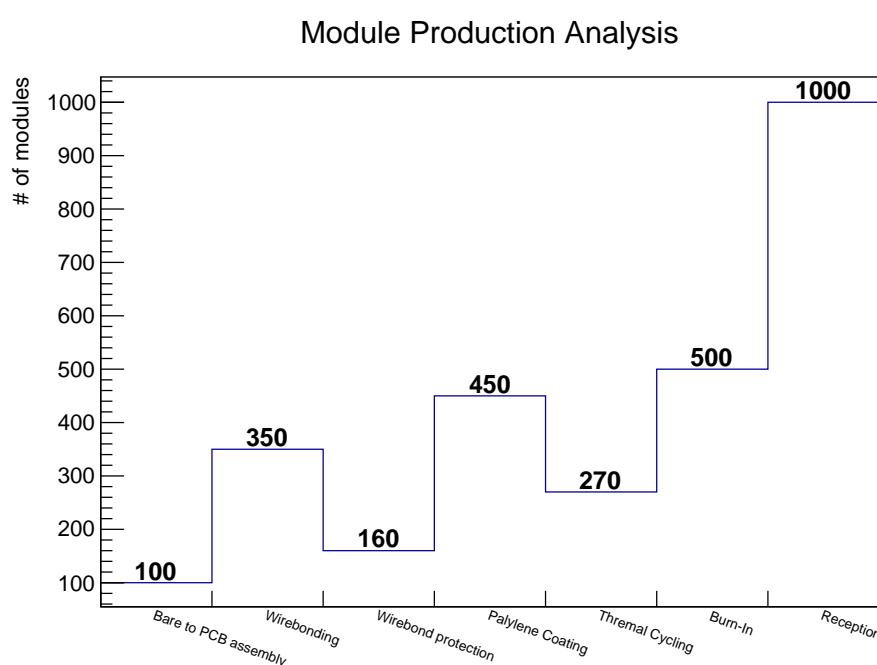


図 D.1 生産時のモジュール組み立て状況解析の例。ローカルデータベースにて組み立て工程は管理され、工程毎に中央データベースに同期されるため、生産時には全てのモジュールの現工程を中央データベースで取得できる。図の例のように各段階におけるモジュール数を見ることで生産状況の確認ができる。さらにある期間ごとに工程を取得することで生産レートなども計算することができ、今後の生産計画やモジュール選別の参考とすることができます。

付録 E

ファイル送信時におけるデータ容量と処理時間の関係について

1391 始めに、処理速度遅延の原因として以下をあげる。

- 1392 ● サーバーの読み書き速度.
- 1393 ● サーバーのファイル転送アルゴリズム.
- 1394 ● サーバーのファイル転送時におけるパケットサイズ.
- 1395 ● サーバー間のネットワーク上の距離.
- 1396 ● サーバー間ネットワークの処理性能.

1397 KEK と LBL に設置されているサーバーにおいて、中央データベースへのファイル送信処理時間に差
1398 が出る理由について考察する。

1399 ここで、ファイル送信時におけるデータ容量と処理時間の関係は、線形性を示さない。図 E.1 は KEK
1400 から LBL のサーバーに scp コマンドを用いてファイル送信を行い、データ容量と処理時間の関係を取得
1401 したものである。赤線が線形フィットであるが、測定点は優位にずれている。これは TCP 通信において
1402 パケットの送信に輻輳制御 [44] と呼ばれる技術が使われており、データ送信量を変化させながら情報通信
1403 を行っているためである。

1404 scp によるファイル送信を以下の 2 つの場合に対して行い、ファイル容量と処理時間の関係を取得した。

- 1405 1. KEK から LBL
- 1406 2. LBL から KEK

1407 結果を図 E.2 に示す。ここで処理時間に差が生まれる原因について調査したものを以下に示す。

- 1408 ● 読み書き速度を測定したところ同程度であった。
- 1409 ● 輻輳制御アルゴリズムは Cubic をどちらも使用していた。
- 1410 ● パケットサイズは変わらなかった。
- 1411 ● ping による応答時間確認は同程度であった (111 msec)。

1412 よって各サーバが置かれているローカルネットワークに差異があると考えた。一般的には上りより下り
1413 の方が太いと考えると、KEK ローカルの上りネットワークの性能が、LBL ローカルの上りネットワーク
1414 と比べて悪いと考えられる。これにより、ファイル送信時間に差が生まれている。

1415 次に、下の 2 つの場合に対して行い、ファイル容量と処理時間の関係を取得した。

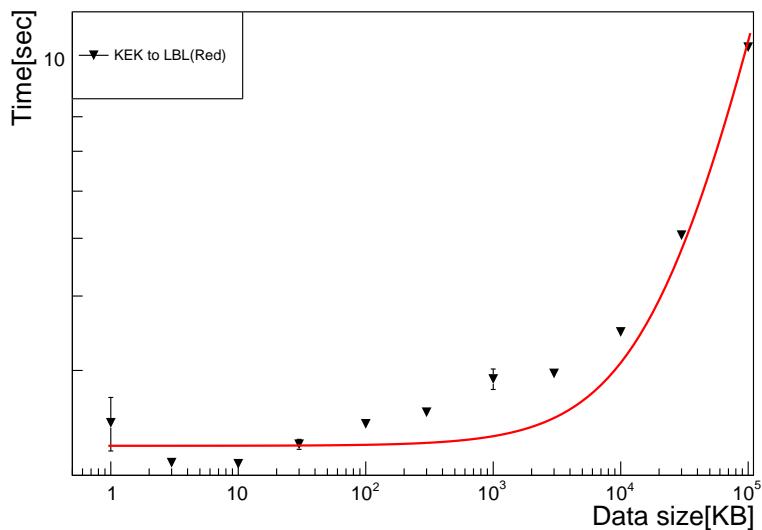


図 E.1 添付するファイルサイズと処理時間の関係

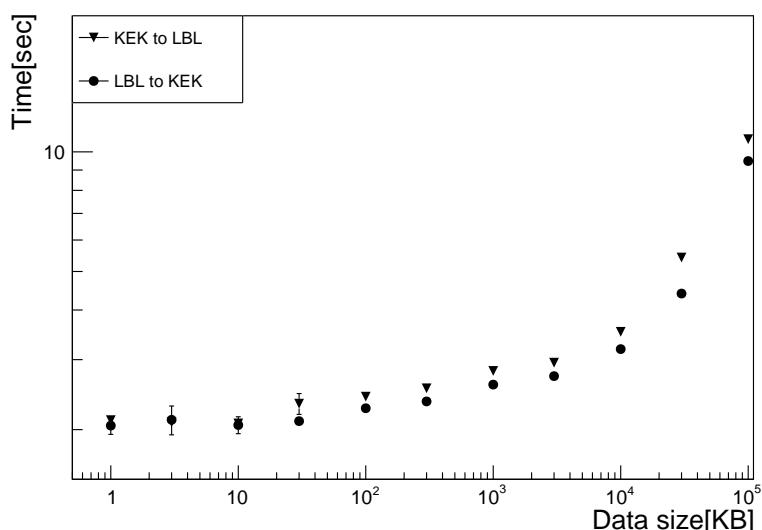


図 E.2 KEK、LBL 間のファイル送信

- 1416 1. KEK から CERN(Lxplus)
 1417 2. LBL から CERN(Lxplus)

1418 KEK ローカルの上りネットワークは細いため、処理時間に差が生まれる。加えてこの場合はサーバ間
 1419 の距離による遅延も含まれていると考えられる。ping による応答時間が測定 1 では 170 msec 程度なの
 1420 に対し、測定 2 は 150 msec 程度であった。そのため、これも処理速度遅延に影響していると考えられる。
 1421 CERN と中央データベースが地理的に近い距離であることを考慮し、上述したことをまとめると KEK
 1422 と LBL の間で処理時間の差が生まれる要因は以下であると考えた。

- 1423 ● ローカルネットワークの性能。
 1424 ● ネットワーク上の距離差。

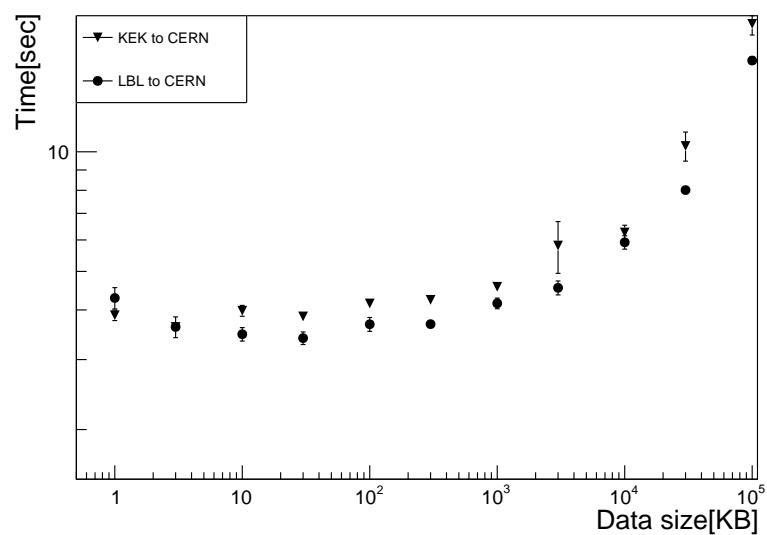


図 E.3 KEK、LBL と CERN 間のファイル送信

参考文献

- [1] “現代素粒子物理実験的観点からみる標準理論”
 末包文彦・久世正弘・白井淳平・湯田春雄, 森北出版株式会社, 2016 年 12 月第 1 版第 1 刷発行.
- [2] ATLAS Collaboration. “Combined measurements of Higgs boson production and decay using up to 80 fb^{-1} of proton–proton collision data at $\sqrt{s}=13 \text{ TeV}$ collected with the ATLAS experiment”. CERN Document Server. 2019-9 更新.
<http://cdsweb.cern.ch/record/2629412/>, (2020-1)
- [3] Stephen P. Martin. “A Supersymmetry Primer”. arXiv.org. 2016-1 更新.
<https://arxiv.org/abs/hep-ph/9709356>, (2020-1)
- [4] Damerau,H et al. “LHC Injectors Upgrade Technical Design Report”. CERN Document server. 2016-05
<https://cds.cern.ch/record/2153863>, (2020-12)
- [5] Georges Aad et al. “The ATLAS Experiment at the CERN Large Hadron Collider”. Semantic Scholar. 2008-8
<https://www.semanticscholar.org/paper/The-ATLAS-Experiment-at-the-CERN-Large-Hadron-Aad-Gr7d771b20731969fe10c267465582ee60e9383db3>, (2020-12)
- [6] ATLAS Collaboration. “The upgraded Pixel Detector of the ATLAS Experiment for Run 2 at the Large Hadron Collider”. ScienceDirect. 2016-9
<https://doi.org/10.1016/j.nima.2016.05.018>, (2020-12)
- [7] Apollinari, G;Bjar Alonso, I; Brning, O; Lamont, M; Rossi, L. “High-Luminosity Large Hadron Collider (HL-LHC) : Preliminary Design Report”. CERN Document Server. 2015-12
<https://cds.cern.ch/record/2116337>, (2020-12)
- [8] The HL-LHC project. “The HL-LHC project”. CERN Accelerating science. 2020-8
<https://hilumilhc.web.cern.ch/content/hl-lhc-project>, (2020-12)
- [9] ATLAS Collaboration. “Technical Design Report for the ATLAS Inner Tracker Pixel Detector”. CERN Document Server. 2018-8
<https://cds.cern.ch/record/2285585>, (2020-12)
- [10] ATLAS Collaboration. “Projections for measurements of Higgs boson signal strengths and coupling parameters with the ATLAS detector at a HL-LHC”. CERN Document Server. 2020-5 更新.
<https://cds.cern.ch/record/1956710>, (2020-1)
- [11] ATLAS Collaboration. “Observation and measurement of Higgs boson decays to WW with the ATLAS detector”. CERN Document Server. 2020-6 更新.

- 1458 <https://cds.cern.ch/record/1975394>, (2020-1)
- 1459 [12] “Semiconductor Devices Physics and Technology”.
- 1460 S.M. Sze 著, 南日康夫・川辺光央・長谷川文夫訳, 産業図書, 2015 年 3 月第 2 版第 11 刷発行.
- 1461 [13] “Pixel Detectors”.
- 1462 Rossi, L., Fischer, P., Rohe, T., Wermes, N. Springer, 2006-7-8
- 1463 [14] “トータルドーズ効果”. 天文学辞典. 2018-3
- 1464 <https://astro-dic.jp/total-dose-effect/>, (2020-12)
- 1465 [15] Danilo Giugni. “General approach for thermal-mechanics QA and QC”. CERN Indico. 2019-12
- 1466 <https://indico.cern.ch/event/860761/contributions/3661710/>, (2020-12)
- 1467 [16] Johannes Weller. “Characterisation of Pixel Sensors for the ATLAS ITk Upgrade”. CERN
- 1468 Indico. 2020-7
- 1469 [https://indico.cern.ch/event/935007/contributions/3928837/attachments/2075810/3485805/BA_pres_\\$jweller.pdf](https://indico.cern.ch/event/935007/contributions/3928837/attachments/2075810/3485805/BA_pres_$jweller.pdf), (2020-1)
- 1470 [17] Lakmin Wickremasinghe. “Pixel modules and Hybridization:First results from RD53A production”. CERN Indico. 2020-9
- 1471 <https://indico.cern.ch/event/950039/contributions/4024890/attachments/2108087/3546372/QuadModuleTestResults-ITkWeek.pdf>, (2020-12)
- 1472 [18] Timon Heim. “YARR - A PCIe based Readout Concept for Current and Future ATLAS Pixel
- 1473 Modules”. IOP Science. 2017
- 1474 <https://iopscience.iop.org/article/10.1088/1742-6596/898/3/032053>, (2020-12)
- 1475 [19] “物理数学 II”.
- 1476 西森秀稔 著, 丸善出版, 平成 27 年 9 月 30 日発行.
- 1477 [20] Meng, Lingxin. “RD53A Module Testing Document”. CERN Document server. 2020-9
- 1478 <https://cds.cern.ch/record/2702738>, (2020-12)
- 1479 [21] “Unicorn University - Unicorn College”. Unicorn University. <https://unicornuniversity.net/en/>, (2020-1)
- 1480 [22] “MongoDB: The most popular database for modern apps”. MongoDB, Inc.
- 1481 <https://www.mongodb.com/2>, (2020-12)
- 1482 [23] “Welcome to Flask”. Flask Documentation.
- 1483 <https://flask.palletsprojects.com/en/1.1.x/>, (2020-12)
- 1484 [24] 窪田ありさ. “HL-LHC ATLAS 実験用新型ピクセル検出器の系統評価と量産時に向けた試験管理シ
- 1485 ステムの開発” CERN Box. 2020-3
- 1486 <https://cernbox.cern.ch/index.php/s/BdhvSTAuuE5xHxt>, (2020-1)
- 1487 [25] “Databases and Collections”. MongoDB Manual.
- 1488 <https://docs.mongodb.com/manual/core/databases-and-collections/>, (2020-12)
- 1489 [26] “WiredTiger Storage Engine”. MongoDB, Inc.
- 1490 <https://docs.mongodb.com/manual/core/wiredtiger/>, (2020-1)
- 1491 [27] “PyMongo 3.11.2 Documentation”. PyMongo 3.11.2 Documentation.
- 1492 <https://pymongo.readthedocs.io/en/stable/>, (2020-12)
- 1493 [28] “ROOT: analyzing petabytes of data, scientifically.”. ROOT Team.
- 1494 <https://root.cern>, (2020-12)

- 1499 [29] “QC Software doc”. ITk Docs.
1500 https://itk.docs.cern.ch/pixels/qc__software/rd53a__demo__flow/, (2020-1)
- 1501 [30] “InfluxDB: Purpose-Built Open Source Time Series Database”. influxdata.
1502 <https://www.influxdata.com>, (2020-12)
- 1503 [31] “Grafana: The open observability platform”. Grafana Labs.
1504 <https://grafana.com>, (2020-12)
- 1505 [32] “PySerialComm”. GitLab.
1506 <https://gitlab.cern.ch/solans/PySerialComm>, (2020-12)
- 1507 [33] “RD53A Single Chip Card Configuration”. CERN twiki. 2018-5-14
1508 https://twiki.cern.ch/twiki/pub/RD53/RD53ATesting/RD53A__SCC__Configuration.pdf, (2020-12)
- 1509 [34] Renesas Electronics Corporation. “GPIO”. RENESAS.
1510 <https://www.renesas.com/jp/ja/support/engineer-school/mcu-programming-peripherals-01-gpio>, (2020-12)
- 1511 [35] Microchip Technology. “2.7V Dual Channel 10-Bit A/D Converter with SPITM Serial Interface”.
1512 秋月電子通商. 2006-08
1513 <https://akizukidenshi.com/download/ds/microchip/mcp3002.pdf>, (2020-12)
- 1514 [36] “Raspberry Pi 3 Model B+”. RASPBERRY PI FOUNDATION.
1515 <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>, (2020-12)
- 1516 [37] “E3640A E3649A Programmable DC Power Supplies - Data Sheet”. Keysight Technologies.
1517 2018-3
1518 <https://www.keysight.com/jp/ja/assets/7018-06827/data-sheets/5968-7355.pdf>, (2020-12)
- 1519 [38] “XpressK7-160-Gen2”. Mouser Electronics.
1520 <https://www.mouser.jp/ProductDetail/ReFLEX-CES/XpressK7-160-Gen2?qs=rrS6PyfT74eSJLUPLu1P5g%3D%3D>, (2020-12-22)
- 1521 [39] Wielers, Monika. “Pixel production database serial numbering scheme”. CERN Document
1522 Server. 2020-12-07
1523 <https://cds.cern.ch/record/2728364>, (2020-12)
- 1524 [40] “LocalDB docs”. LocalDB docs.
1525 <https://localdb-docs.readthedocs.io/en/top/>, (2020-12)
- 1526 [41] “Selected Exercises in Particle and Nuclear Physics”.
1527 Lorenzo Bianchini, Springer, 2017-9 published.
- 1528 [42] Garcia-Sciveres, Maurice. “The RD53A Integrated Circuit”. CERN Document Server. 2017-10
1529 <https://cds.cern.ch/record/2287593>, (2020-12)
- 1530 [43] Patrick McCormack, Maurice Garcia-Sciveres, Timon Heim, Benjamin Nachman, Magne
1531 Lauritzen. “New Method for Silicon Sensor Charge Calibration Using Compton Scattering”.
1532 arXiv.org. 2020-8.
1533 <https://arxiv.org/abs/2008.11860>, (2020-1)
- 1534 [44] “TCP 技術入門-進化を続ける基本プロトコル”.
1535 安永遼真, 中山悠, 丸田一輝著, WEB+DB PRESS plus, 2019 年 7 月 6 日発売.

1540 謝辞

図目次

1541	図目次	
1542	1.1 標準模型の素粒子の質量とヒッグス粒子との結合定数の関係	3
1543	1.2 加速器の全体像	4
1544	1.3 極角 θ と擬ラピディティ η の関係図	5
1545	1.4 ATLAS 検出器の全体像	5
1546	1.5 内部飛跡検出器の全体像	6
1547	1.6 ピクセル検出器の全体像	7
1548	1.7 ピクセルモジュールの全体像。	7
1549	1.8 HL-LHC 運転計画	8
1550	1.9 LHC、HL-LHC における粒子の質量とヒッグス粒子の結合定数の関係と統計誤差の見積 もり	9
1551	1.10 1 バンチ衝突あたりのイベント数増加のイメージ	10
1552	1.11 ITk の全体像	10
1553	1.12 ITk の断面図	11
1554	1.13 VBF イベントの図	12
1555	1.14 VBF イベントにおけるジェットの運動方向の η 分布	12
1556	2.1 ピクセルモジュールの構成	14
1557	2.2 新型ピクセルモジュールに搭載するシリコンセンサーの断面図	15
1558	2.3 ToT の概念図	15
1559	2.4 RD53A	16
1560	2.5 ピクセルモジュールにおける信号伝達の様子	17
1561	2.6 搭載するモジュールのプロトタイプと ITk における配置。	18
1562	3.1 組み立て工程のイメージ図	20
1563	3.2 外観検査の様子	21
1564	3.3 平坦性測定の様子。	21
1565	3.4 センサー電流-電圧特性結果の例	22
1566	3.5 FE チップ電流-電圧特性試験結果の例。	22
1567	3.6 読み出し試験の様子	23
1568	3.7 デジタル回路読み出しにおける <i>Occupancy</i> 分布の例。	26
1569	3.8 入射電荷量と <i>Occupancy</i> の関係	27
1570	3.9 Threshold 値とノイズの分布。	27
1571	3.10 ノイズ占有率測定における <i>NoiseOccupancy</i> 分布の例。	28

1573	3.11	組み立て工程と対応する品質試験一覧	30
1574	4.1	ローカルデータベースシステムの概要	33
1575	4.2	MongoDB の構造の例	35
1576	4.3	先行研究により設計された読み出し試験結果の MongoDB 内部構造	36
1577	4.4	ウェブアプリケーション処理のイメージ	37
1578	4.5	品質試験結果ページの例	38
1579	4.6	中央データベースにおけるモジュールの種類と構造	39
1580	4.7	中央データベース内におけるモジュール構造の一例 (Quad モジュール)	40
1581	4.8	同期ツール処理のイメージ	41
1582	4.9	同期機能の概要	41
1583	4.10	ウェブアプリケーションにおけるコメント機能	43
1584	4.11	ウェブアプリケーションにおけるタグ機能	43
1585	4.12	結果選択画面及び組み立て工程表示の例	47
1586	4.13	ピクセル解析ツールにおけるファイル統合処理のイメージ	48
1587	4.14	Tree ファイルとそのデータ保持のイメージ	49
1588	4.15	ピクセル解析ツールの処理の流れ	49
1589	4.16	ウェブアプリケーションにおける検索機能の様子	50
1590	4.17	各モジュールにおけるデータベース操作の流れ	51
1591	5.1	読み出し試験に用いるソフトウェアの概要	54
1592	5.2	RD53A シングルモジュール (SCC)	55
1593	5.3	モジュール付属サーミスタを用いた温度読み出し回路	56
1594	5.4	用いた電源	56
1595	5.5	使用した FPGA ボード (XpressK7)	57
1596	5.6	使用した FMC-DisplayPort 変換カード (オハイオカード)	57
1597	5.7	ハードウェアセットアップ	57
1598	5.8	ダウンロードしたモジュール ID 確認画面	59
1599	5.9	DCS 情報のモニタリングの様子	60
1600	5.10	検索機能確認の様子	61
1601	5.11	試験結果の閲覧	62
1602	5.12	読み出し試験結果の選択	64
1603	5.13	ピクセル解析結果	65
1604	5.14	各評価基準における不良ピクセルの分布	66
1605	5.15	中央データベースにおける読み出し試験及びピクセル解析結果画面	67
1606	6.1	検索機能実装方法 1:Python リストを用いた場合	70
1607	6.2	検索機能実装方法 1 の問題点	71
1608	6.3	検索処理時間のボトルネックとなっているデータ構造	71
1609	6.4	方法 1 における検索処理速度測定結果	72
1610	6.5	検索機能実装方法 2:検索キーワード専用コレクションを用いた場合	74
1611	6.6	方法 2 における検索処理時間測定結果	76

1612	6.7	方法 2 における検索情報コレクションの生成時間測定結果	77
1613	6.8	方法 2 の検索における詳細処理	78
1614	6.9	方法 2 における詳細処理時間の測定結果	79
1615	6.10	ドキュメント数に対する型変換処理時間の関係	80
1616	6.11	検索機能実装方法 3:検索情報と共に一覧表示に必要な情報を保持、参照	81
1617	6.12	検索機能実装方法 4:検索情報コレクションを分散、マルチスレッドを使用	81
1618	6.13	方法 3、4 に対する処理時間測定結果	82
1619	7.1	各サーバーの設置場所	84
1620	7.2	”createTestRunAttachment”を用いた添付処理におけるファイル容量と処理時間の関係	85
1621	7.3	中央データベースウェブページへの通信時間の測定結果	86
1622	7.4	ダウンロード処理における流れのイメージ図	89
1623	7.5	登録した Quad モジュールと構成部品のシリアルナンバー一覧。	90
1624	7.6	登録した Quad モジュールのダウンロードの様子	91
1625	7.7	モジュール及び構成部品情報取得のイメージ図	93
1626	7.8	中央データベースにおける読み出し試験結果の構造	96
1627	7.9	アップロード処理に関する流れのイメージ図	97
1628	7.10	アップロード機能における詳細処理測定結果	98
1629	8.1	中央データベースにおける選択可能な組み立て工程のイメージ	105
1630	A.1	真性半導体中のシリコン	106
1631	A.2	pn 接合の電流 – 電圧特性	107
1632	A.3	空乏層	107
1633	B.1	アナログフロントエンド	110
1634	B.2	デジタルフロントエンド	110
1635	B.3	RD53A の各ピクセルが持つ試験用電荷入射回路の簡略図	111
1636	B.4	RD53A におけるデータフォーマットのイメージ図	111
1637	B.5	フレキシブル基板	112
1638	C.1	ローカルデータベースシステムチュートリアルの様子	113
1639	C.2	ローカルデータベースシステム導入及び試運転場所	114
1640	D.1	生産時のモジュール組み立て状況解析の例	115
1641	E.1	添付するファイルサイズと処理時間の関係	117
1642	E.2	KEK、LBL 間のファイル送信	117
1643	E.3	KEK、LBL と CERN 間のファイル送信	118

表目次

1644	標準模型のフェルミオン	1
1646	標準模型のボソン	2
1647	現行 LHC と HL-LHC の比較	8
1648	ピクセル検出器設置領域の比較	11
1649	搭載するピクセルモジュール数の比較	11
1650	VBF $H \rightarrow WW$ の測定における系統誤差の見積もり	13
1651	RD53A のスペック	16
1652	ピクセル解析の評価基準一覧	29
1653	品質試験に用いる主なコレクション	36
1654	中央データベースにおける組み立て工程と付随するテスト項目	40
1655	ローカルデータベースユーザ権限及び使用機能一覧	43
1656	温度読み出しシステムに使用した装置一覧	56
1657	読み出しに使用した PC の性能	57
1658	測定に使用したノート PC の性能	75
1659	検索機能処理時間測定の詳細	75
1660	処理 3,5 における型変換処理 6,5 の割合	79
1661	各ローカルデータベースサーバーの性能一覧	83
1662	同期ツールの中で使用する中央データベースの主な API 一覧	84
1663	中央データベース API 実行時の処理時間測定結果。	85
1664	ダウンロード機能を用いて保存する情報一覧。	87
1665	登録したモジュールのダウンロード処理時間測定結果	92
1666	ダウンロード機能における詳細処理にかかる時間測定	93
1667	中央データベースにおける読み出し試験結果に関する情報一覧	96
1668	アップロード処理における結果、設定値ファイル添付実行結果と処理時間	99
1669	アップロード処理における結果、設定値ファイル添付実行結果と処理時間	100
1670	アップロード処理改善後における結果、設定値ファイル添付実行結果と処理時間	101