

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)

2020 年 12 月 30 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 searched, 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 guranularity. 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 detestors and covers much more wider solid angle acceptance than the current Inner Detector.

For the production of the ITk, we are planning to produce $O(10,000)$ modules and conduct a series of tests for quarity 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 Check Republic, to save the performance of the module itself as the reference for the operation of the ITk. For this production, I have developed a "local database" system in order to manage data at local production sites and to synchronize informations. The system is now under test-use among production sites towards full production.

For spreading the DB system, I organized a tutorial for users at CERN in February 2020.

Concering the development of the DB system, 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 amount of data at the production to measure the processing time of the tools.

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、LHC-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、衝突確率に対応する量である瞬間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。瞬間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 実験では最内装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空間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 10,000 台の生産、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。品質試験は種類が多く、モジュール組み立て工程の中で何度も行うものである。またモジュール情報及び品質試験の結果は固体性能の保持や ITk 運転時における参照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。私はモジュール量産に向けて、各現場においてモジュール及び品質試験の情報管理と中央データベースとの同期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の開発と機能普及を行った。

機能普及に向けて、2020 年 2 月に生産時におけるシステム利用者を対象としたチュートリアルを CERN で行った。このチュートリアルを経て、現在ではいくつかの生産機関でシステム試験運転が開始されており、機能普及が進んでいることを確認した。

データベースシステムの開発に関して、先行研究で開発されたシステムに拡張する形で試験結果検索やデータ同期など量産時に必要となるいくつかのツールを実装した。また実装した機能を含めデータベース機能全体が使用可能であることを学内設備を用いて確認した。さらに開発した機能が量産時のデータ数に対して十分に使用可能であるかを、機能の処理時間測定を行い確認した。

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 1 つの衝突点で行われている実験であり、設置している ATLAS 検出器を
₆ 用いて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、
₇ これに向けて 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
₈ レード計画について説明する。

₉ 1.1 LHC について

₁₀ 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。
₁₁ 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
₁₂ である。

₁₃ 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る
₁₄ ことで陽子を生成する。その後最初の線形加速器 (Linear Accelerator: LINAC) で 50 MeV まで加速し、
₁₅ 陽子シンクロトロンブースター (Proton Synchrotron Booster: PSB) で 1.4 GeV、陽子シンクロトロン
₁₆ (Proton Synchrotron: PS) で 25 GeV、スーパー陽子シンクロトロン (Super Proton Synchrotron) で
₁₇ 450 GeV まで加速されたのち LHC に入射する。CERN にある加速器の概要を図 1.1 に示す。LHC に
₁₈ は 4 つの衝突点があり、それぞれ ALICE(A Large Ion Collider Experiment)、LHCb、CMS(Compact
₁₉ Muon Solenoid)、ATLAS(A Troidal LHC Apparatus) 実験が行われている。それぞれの衝突点には崩
₂₀ 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されており、それら検出器で取得したデータを
₂₁ 元に多様な物理解析が行われている。

₂₂ 1.2 ATLAS 実験

₂₃ 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、
₂₄ 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
₂₅ 中心に向かう方向であり、 y 軸は地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
₂₆ z 軸とのなす角である。大抵、極角 θ は以下のようにローレンツ不変量 η で表される。

$$\eta = -\ln \tan \left(\frac{\theta}{2} \right) \quad (1.1)$$

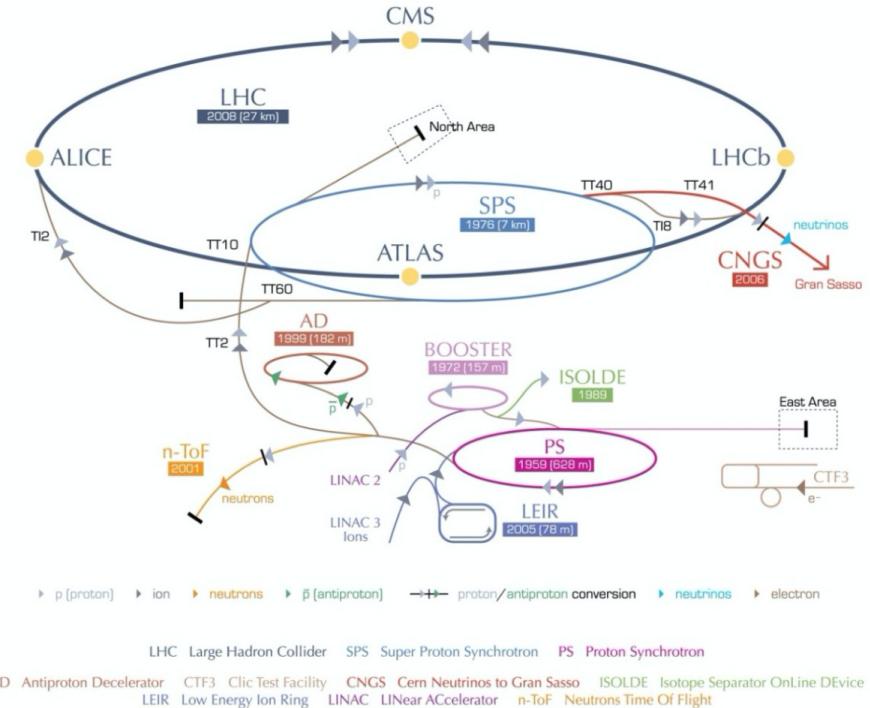


図 1.1 LHC の全体像 [1-1]

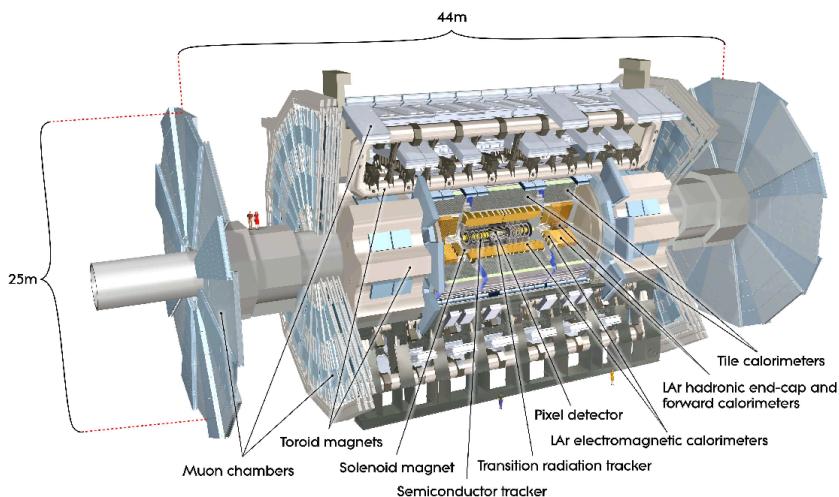


図 1.2 ATLAS 検出器 [1-2]

1.2.1 ATLAS 検出器

ATLAS 検出器の全体図を図 1.2 に示す。最内装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、カロリメータ、トレノイド磁石、ミューオン検出器の順に設置されている。ビームパイプ以外をほとんど検出器で覆うようなデザインとなっている。

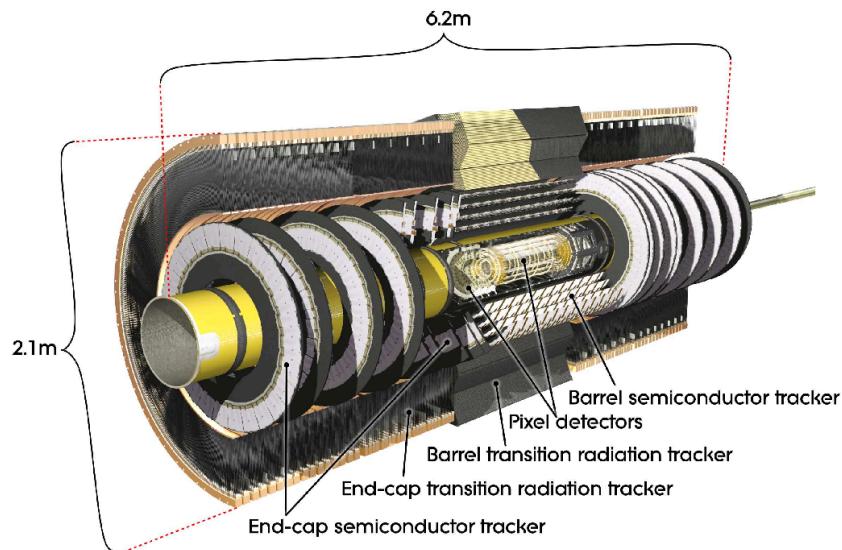


図 1.3 内部飛跡検出器 [1-2]

31 1.2.2 内部飛跡検出器

32 内部飛跡検出器の全体図を図 1.3 に示す。内部飛跡検出器は半径 1.15m、長さ 7m の円柱形であり、
33 $|\eta| \leq 2.5$ の領域を覆っている。超伝導ソレノイド磁石で 2 T の磁場が z 方向にかけられる。この検出器
34 はさらに 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、遷移放射検出器の順に
35 設置されている。

36 検出器は η の範囲によってバレル部とエンドキャップ部に分かれる。図 1.4 にビーム軸方向の端面図を
37 示す。

38 ピクセル検出器

39 ピクセル検出器の全体図を図 1.5 に示す。ピクセル検出器はバレル層が 4 層、エンドキャップ層が
40 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、
41 Layer-2 となっている。

42 ピクセル検出器は、モジュール構造を持つ検出器の集合である。モジュールを図 1.6 に示す。このピク
43 セルモジュールの詳細については 2 章で述べる。

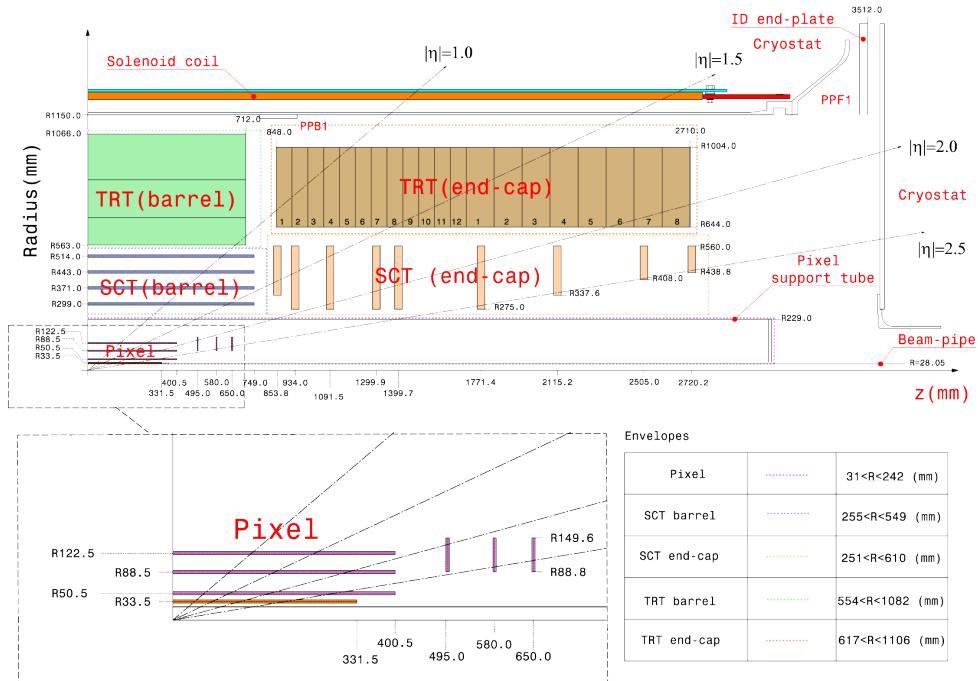


図 1.4 内部飛跡検出器 [1-4]

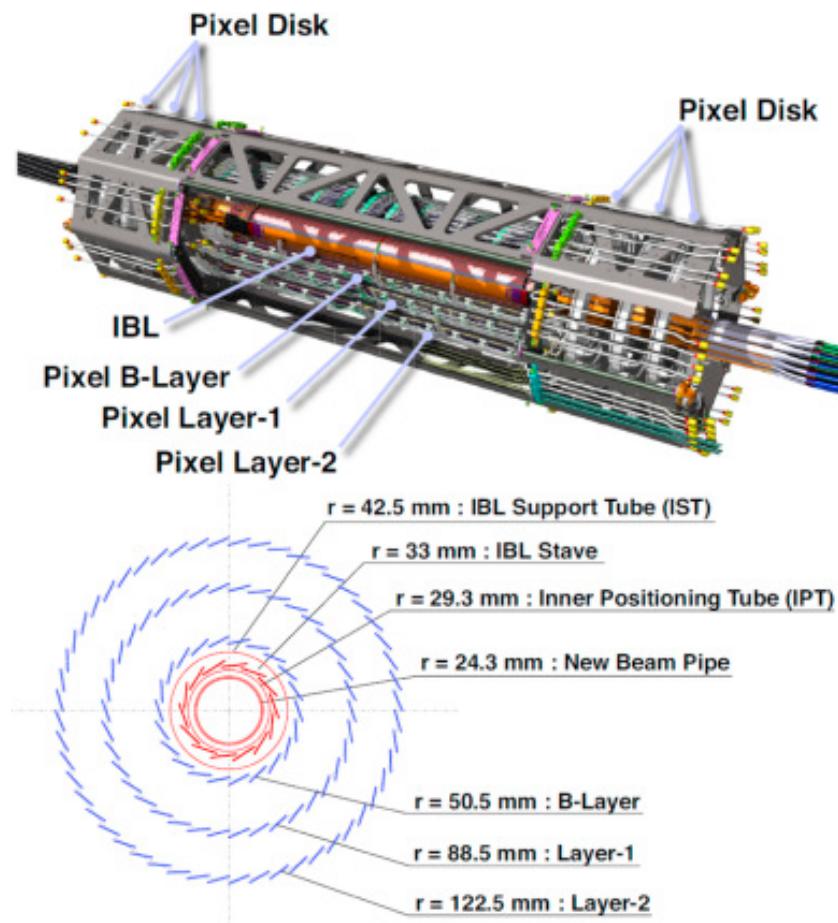


図 1.5 ピクセル検出器全体 [1-5]

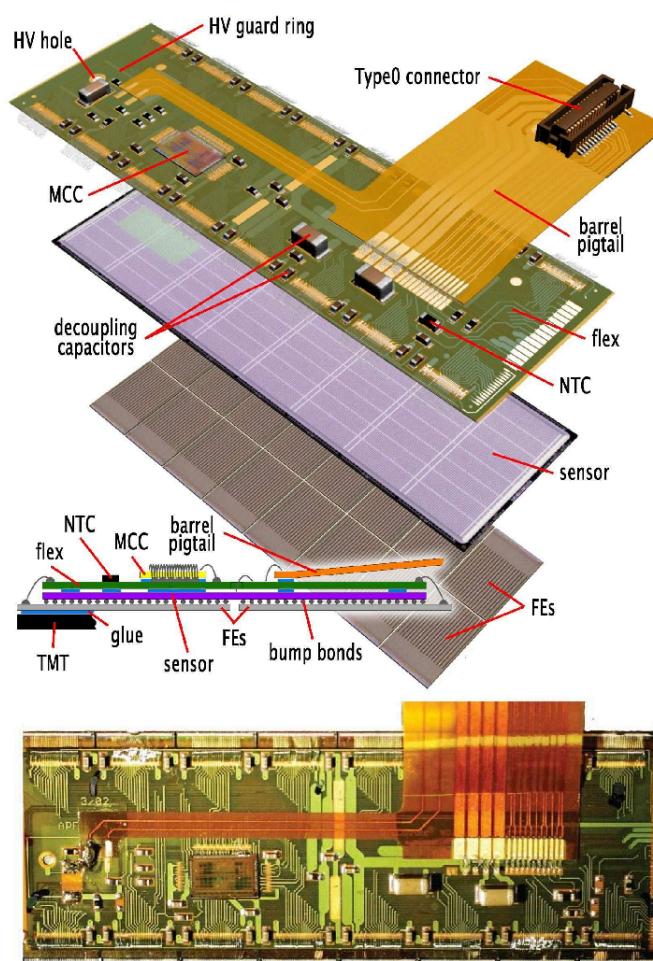


図 1.6 ピクセルモジュール [1-2]

表 1.1 LHC の比較 [1-6]

	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衝突あたりのイベント数	23	138



図 1.7 HL-LHC 運転計画 [1-7]

44 1.3 HL-LHC 実験アップグレード計画

45 上述したように、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
46 計画と呼ぶ。詳細を以下に示す。

47 1.3.1 概要

48 HL-LHC ではルミノシティ呼ばれる陽子バンチ密度を上げることで、衝突確率を大きくし、取得統計
49 数を増やす目的がある。LHC と HL-LHC の比較を表 1.1 に示す。

50 LHC の運転計画を表 1.7 に示す。2020 年 8 月時点の計画では、2025 年の初めより HL-LHC の導入が
51 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。

52 1.3.2 内部飛跡検出器のアップグレード

53 ルミノシティの増加に伴い、検出器には以下の性能が要求される。

- 54 ● 放射線耐性の向上
- 55 ● 高速読み出し
- 56 ● 検出器の細密化

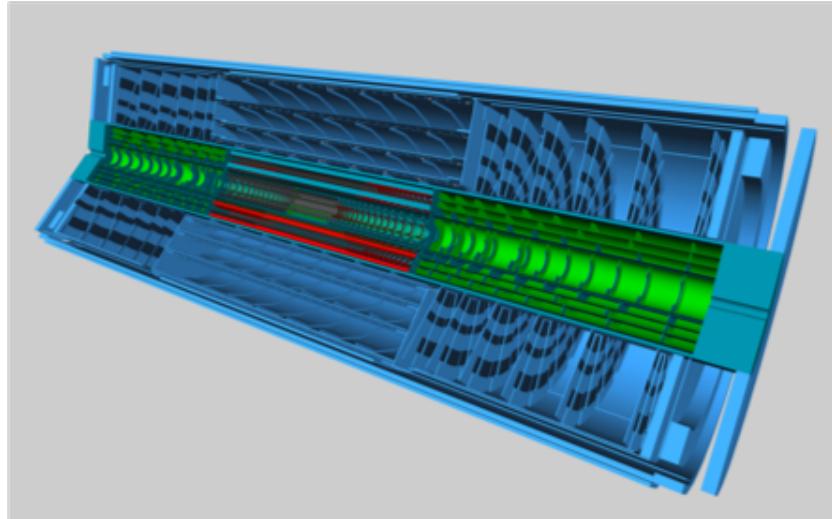


図 1.8 ITk のイメージ [1-3]

表 1.2 ピクセル検出器配置の比較

	現行	ITk
$r[\text{mm}]$	33 129	39 279
$ \eta $	< 2.5	< 4
層の数	4	5

HL-LHCに向けて ATLAS 内部飛跡検出器はアップグレードを予定しており、上記の要求を満たすよう日々開発を進めている。アップグレード後の検出器を ITk(Inner Tracker)と呼ぶ。イメージを図 1.8 に示す。

ITk の構成と現行との比較

図 1.9 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る。ピクセル検出器はバレル、インクラインド、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。

ピクセル検出器の配置に関して、現行と ITk の比較を表 1.2 に示す。またモジュール数の比較を表 1.3 に示す。

1.3.3 期待される物理

ヒッグス粒子の測定

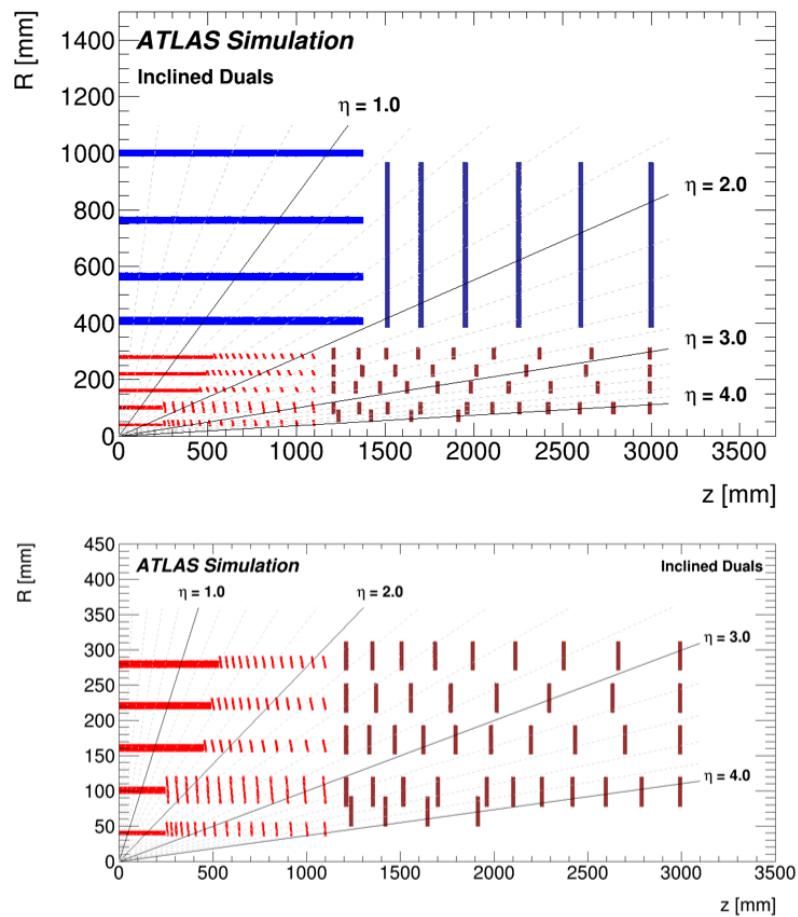


図 1.9 ITk の断面図 [1-3]

表 1.3 ピクセルモジュール数の比較

	バレル部		インクラインド部		エンドキャップ部	
層	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

68 第2章

69 ピクセル検出器

70 2.1 シリコン検出器

71 2.1.1 半導体 [2-1]

72 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
 73 値、導体は高い値を持つ。半導体の電気伝導度はこれらの中間であり、温度、光、磁界および微量の不純
 74 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トロニクスにおける最も重要な材料の1つ
 75 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れ、多くの物質がその候補となる。元素半導
 76 体の中で代表的なものとして Si があげられ、ATLAS ピクセル検出器に使われる半導体は Si がベースと
 77 なっている。不純物が入っていない、全ての原子が Si の半導体を真性半導体と呼ぶ。真性半導体中の Si
 78 は4つの Si と共有結合を構成し、結晶を作る。(図 2.2)

79 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、
 80 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
 81 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。

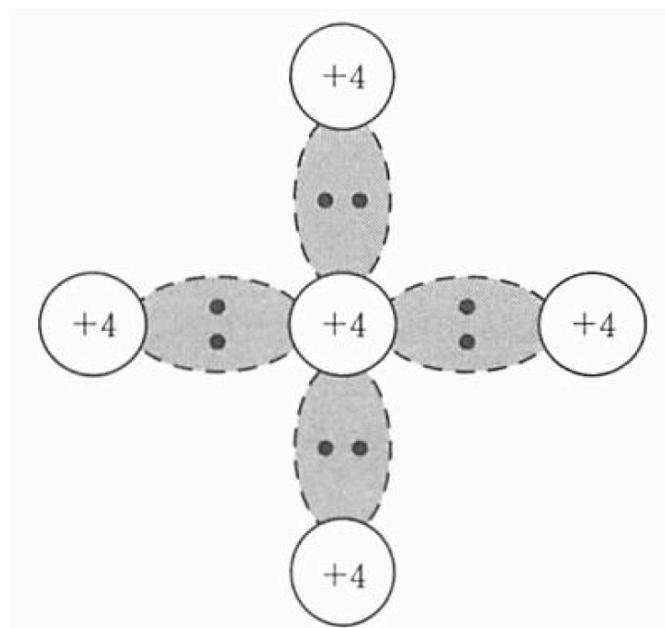


図 2.1 真性半導体中のシリコン [2-1]

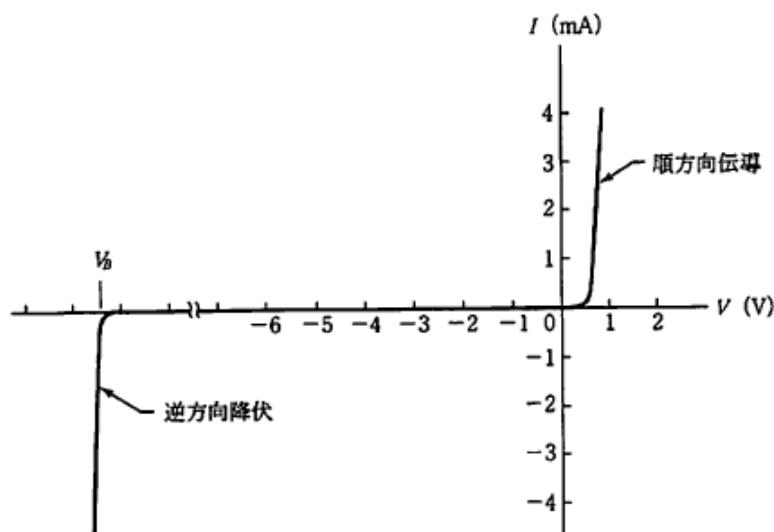


図 2.2 pn 接合の電流 – 電圧特性 [2-1]

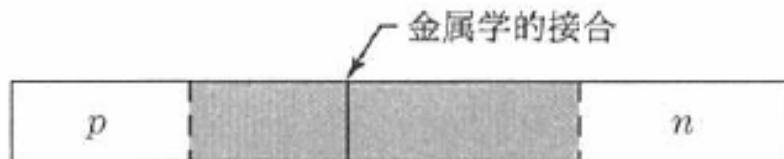


図 2.3 空乏層 [2-1]

82 2.1.2 pn 接合

83 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
84 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。

85 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??に示すように正電
86 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。ある
87 臨界電圧に達すると電流は急激に増大する。

88 逆方向電圧をかけた場合、図 2.3 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
89 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
90 と空乏層幅 W は以下のような関係がある。

$$W \propto \sqrt{V} \quad (2.1)$$



図2.4 ピクセルモジュールの構成

91 2.1.3 検出原理

92 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 [2-3]。

$$-\left\langle \frac{dE}{dx} \right\rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left(\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 + \dots \right) \quad (2.2)$$

$\frac{dE}{dx}$: 荷電粒子のエネルギー損失量 [$\text{eV} \cdot \text{g}^{-1} \cdot \text{cm}^2$] (2.3)

K : $4\pi N_A r_e^2 m_e c^2 = 0.307075 [\text{MeVcm}^2]$

z : 荷電粒子の電荷量

Z : 物質の原子番号 (Si14)

A : 物質の原子量 (Si28)

$m_e c^2$: 電子の静止エネルギー (0.511MeV)

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)

93 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
94 ける。

$$T_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e / M + (m_e / M)^2} \quad (2.4)$$

95 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
96 測定することができる。

97 2.2 新型ピクセルモジュール

98 新型ピクセルモジュールの構成と、各部品についての説明を以下で述べる。

99 2.2.1 モジュールの構成

100 モジュールの構成を図 2.4 に示す。また、モジュールの信号伝達の様子を模式的に表したものも図 2.5
101 に示す。

102 モジュールの種類

103 Quad, triplet など

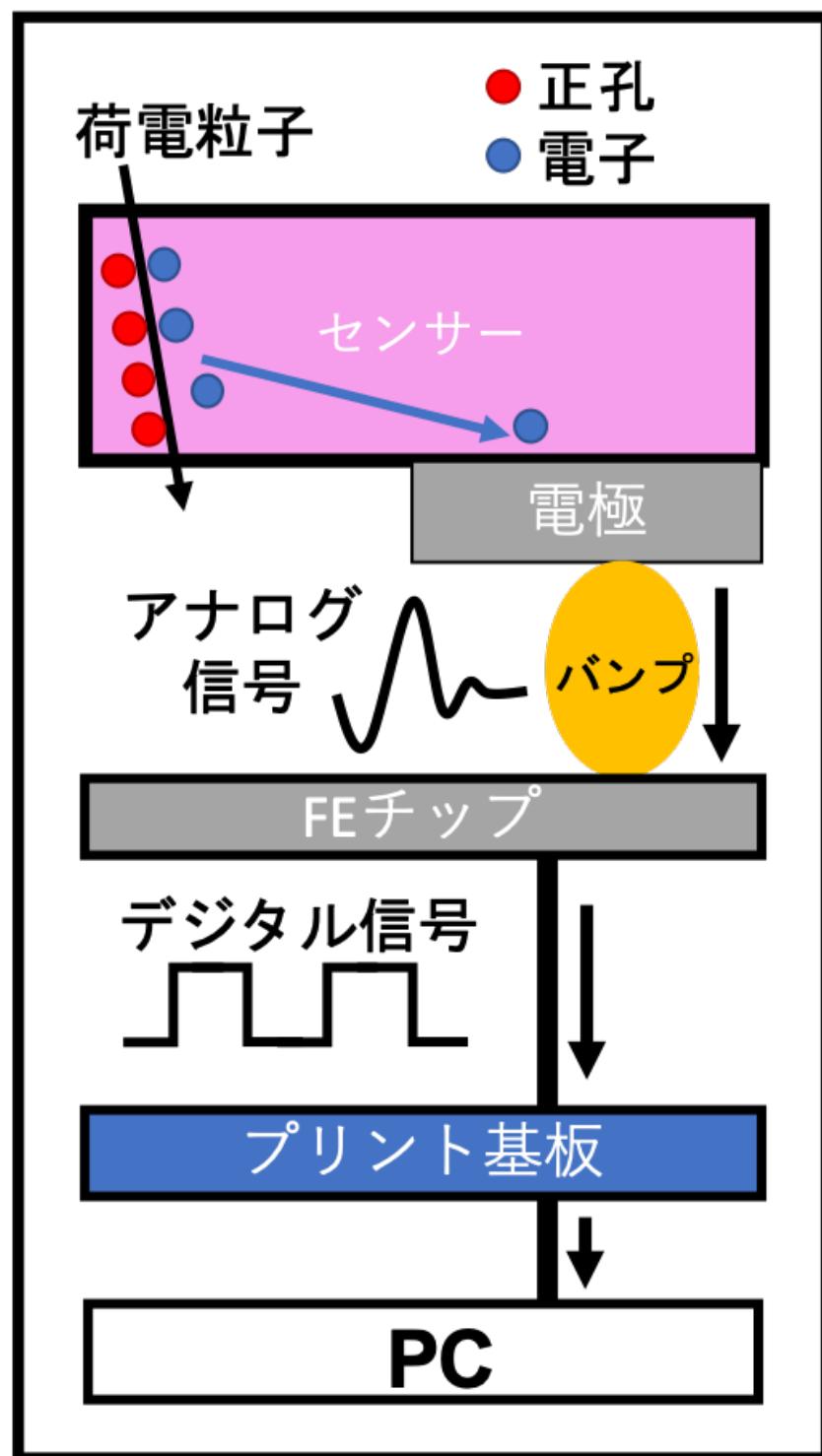


図 2.5 信号伝達

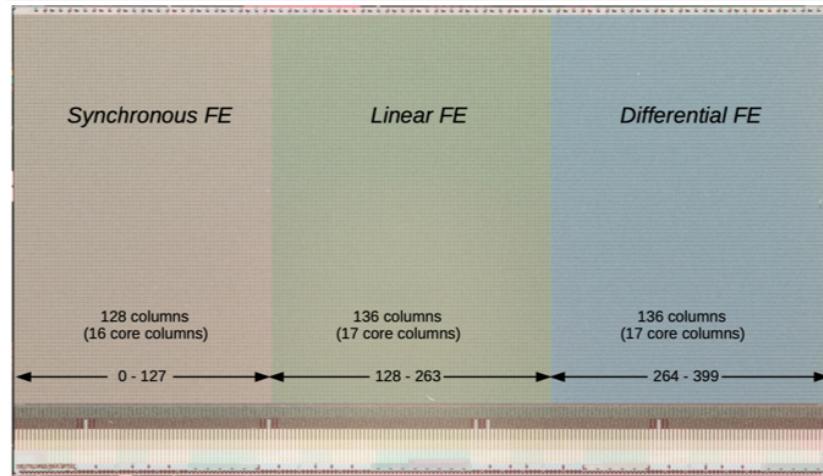


図 2.6 RD53A[2-1]

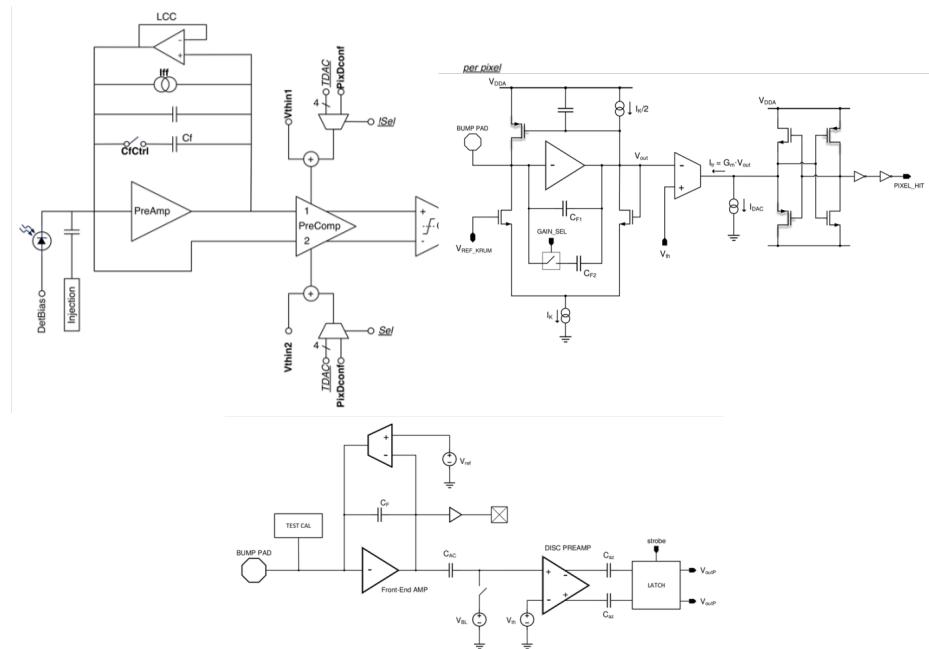


図 2.7 アナログフロントエンド [2-1]

- 104 シリコンセンサー
 105 読み出し FE チップ
 106 プリント基板
 107 モジュールキャリア

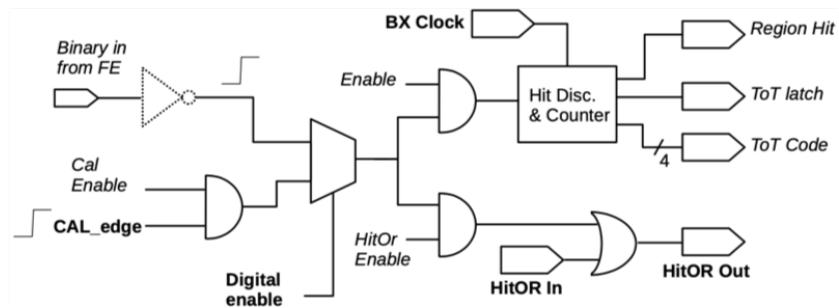


図2.8 デジタルフロントエンド [2-1]

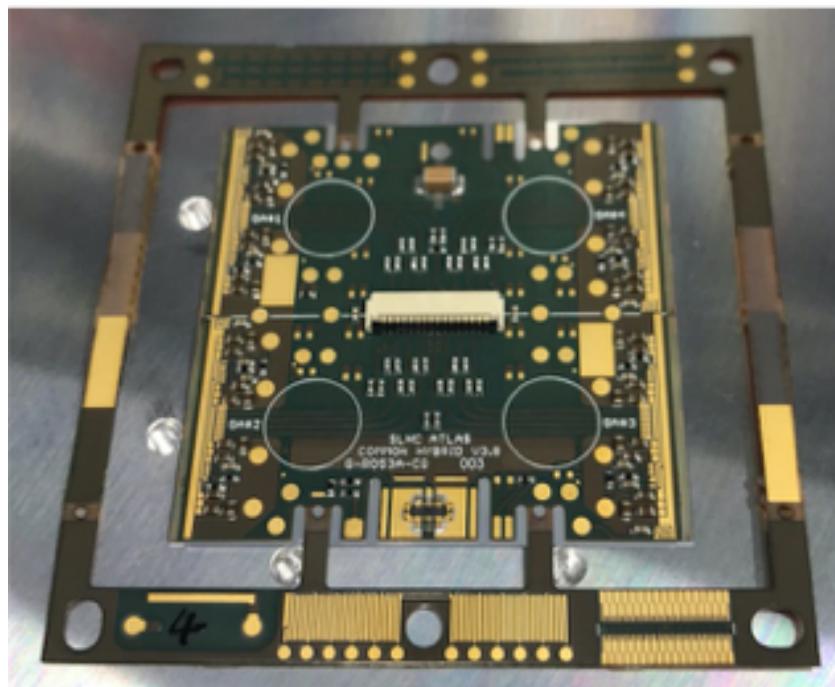


図2.9 プリント基板

108 第3章

109 検出器量産と品質試験

110 3.1 組み立て工程

111 現在モジュールの組み立て工程として以下が設定されている。

112 1. プリント基板・ペアモジュール貼り付け

- 113 • hoge

114 2. ワイヤー配線

- 115 • hoge

116 3. ワイヤー保護

- 117 • hoge

118 4. パリレンコーティング

- 119 • hoge

120 5. 温度サイクル試験

- 121 • hoge

122 6. 低温耐久試験

- 123 • hoge

124 流れと各組み立て工程のイメージを図 3.1 に示す。

125 3.2 品質試験

126 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。

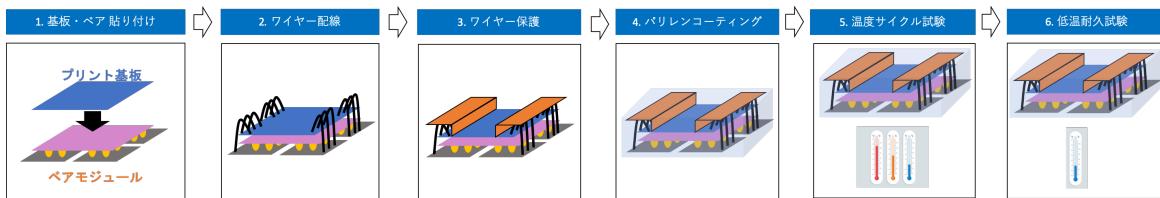


図 3.1 組み立て工程

- 127 3.2.1 外観検査 (Visual Inspection)
- 128 3.2.2 質量測定 (Mass Measurement)
- 129 3.2.3 平坦性測定 (Metrology)
- 130 3.2.4 センサー電流 – 電圧特性確認 (Sensor IV)
- 131 3.2.5 FE チップ電流 – 電圧特性確認 (SLDO VI)
- 132 3.2.6 読み出し試験
- 133 用いる SW と Setup
- 134 レジスタ書き換え (Chip Configuration)
- 135 試験項目詳細
- 136 • レジスタ読み出し (Register test)
 - 137 • デジタル回路読み出し (Digital scan)
 - 138 • アナログ回路読み出し (Analog scan)
 - 139 • Threshold 測定 (Threshold scan)
 - 140 • Threshold グローバルレジスタ調整 (Global threshold tuning)
 - 141 • Threshold ピクセルレジスタ調整、再調整、精密調整 (Pixel threshold tuning)
 - 142 • ToT グローバルレジスタ調整 (ToT tuning)
 - 143 • ノイズ占有率測定 (Noise scan)
 - 144 • スタックピクセル測定 (Stuck pixel scan)
 - 145 • クロストーク測定 (Crosstalk scan)
 - 146 • バンプ接続確認測定 (Disconnected bump scan)
 - 147 • 外部トリガーを用いた測定 (External trigger)
- 148 ここから 1 つ 1 つの詳細について説明する。

- 149 Threshold 調整とピクセル解析
- 150 以下の流れで読み出しを行う。
- 151 • デジタル回路読み出し
 - 152 • アナログ回路読み出し
 - 153 • Threshold 測定
 - 154 • Threshold グローバルレジスタ調整
 - 155 • Threshold ピクセルレジスタ調整
 - 156 • ToT グローバルレジスタ調整
 - 157 • Threshold グローバルレジスタ再調整、精密調整
 - 158 • Threshold 測定
 - 159 • スタックピクセル測定
 - 160 • クロストーク測定

表 3.1 ピクセル解析の評価基準 [3-1]

評価名	読み出し項目	不良評価基準
Digital Dead	Digital scan	Occupancy ≤ 1
Digital Bad	Digital scan	Occupancy ≤ 98 or Occupancy ≥ 102
Merged Bump	Analog scan	Occupancy ≤ 98 or Occupancy ≥ 102
	Crosstalk scan	High Crosstalk
Analog Dead	Analog scan	Occupancy ≤ 1
Analog Bad	Analog scan	Occupancy ≤ 98 or Occupancy ≥ 102
Tuning Failed	Threshold scan	カイ二乗=0
Tuning Bad	Threshold scan	$\pm 5\sigma$
	ToT scan	0 or 15
High ENC	Threshold scan	$\pm 3\sigma$
Noisy	Noise scan	Occupancy $\geq 10^{-6}$
Disconnected Bump	Disconnected bump scan	未決定
	Source scan	Occupancy が平均値の 1%
High Crosstalk	Crosstalk scan	Occupancy ≥ 0 with 25e(sync)
		Occupancy ≥ 0 with 40e(lin and diff)

¹⁶¹ digital scan ・ Explanation ・ Output files

¹⁶² モジュール上のピクセルを解析し、各ピクセルが正常かどうかを判断する。以下の評価基準で解析を
¹⁶³ し、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

¹⁶⁴ 簡易読み出し試験

¹⁶⁵ 簡易読み出し試験では以下の項目を扱う。

- ¹⁶⁶ ● レジスタ読み出し
- ¹⁶⁷ ● デジタル回路読み出し
- ¹⁶⁸ ● アナログ回路読み出し
- ¹⁶⁹ ● Threshold 読み出し
- ¹⁷⁰ ● ToT 読み出し
- ¹⁷¹ ● バンプ接続確認読み出し

¹⁷² バンプ接合確認試験 (Bump bond quality)

¹⁷³ 放射線源を用いてバンプ接合の確認を行う。

¹⁷⁴ 3.2.7 各組み立て工程における品質試験

¹⁷⁵ 各組み立て工程と品質試験項目を図 3.2 に示す。



図3.2 組み立て工程と対応する品質試験

¹⁷⁶ 3.3 検出器量産と品質試験データ管理に対する要求

¹⁷⁷ 読み出し試験については特にデータ管理が大変。

178 第4章

179 モジュール情報及び品質試験結果管理シ 180 ステム

181 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
182 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。システムについては、ITk の全
183 情報を保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
184 である。本章ではこれらのデータベースについて説明する。また、システム開発の中で私が開発を行った
185 機能について詳細に説明する。

186 4.1 中央データベース

187 4.1.1 中央データベースの概要

188 概要

189 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
190 る。ユニコーン大学が開発、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。ITk
191 は、ピクセル検出機とストリップ検出機にから構成される。これらを生産するにあたって、シリコンセン
192 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い、それらを用いたモジュールの組み立て、複
193 数モジュールを搭載した stave や ring の組み立てを経て検出器が完成する。また各組み立て段階におい
194 て、動作確認等を目的とした品質試験を行う。これらの過程における全ての構成部品の情報、及び品質試
195 験結果を中央データベースに保存する。

196 意義

197 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
198 である。品質試験結果を解析、品質のいいモジュールを 10,000 台の中から選別、ITk に搭載する。また、
199 モジュールの配置も品質を考慮して決定する。例として $|\eta|$ が小さく、粒子が多く通過する場所には品質
200 のいいモジュールを搭載するといった位置決定がなされる。

201 なんか物理っぽい議論ができればなあ。

202 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
203 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
204 運転時に参照することができる。また運転前の状態における検出器の性能、運転前後での性能比較を行う

表 4.1 中央データベースにおけるモジュールの種類と構造一覧。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、構成部品を表のように実装した。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おり、PCB などの対応する構成部品の紐付けも同時にを行うことができる。

種類	構成する部品(数)
Triplet L0 stave module	Single bare module(3)
	Triplet stave PCB(1)
Triplet L0 Ring0 module	Single bare module(3)
	Triplet R0 PCB(1)
Triplet L0 Ring0.5 module	Single bare module(3)
	Triplet R0.5 PCB(1)
L1 quad module	Quad bare module(1)
	Quad PCB(1)
Outer system quad moudle	Quad bare module(1)
	Quad PCB(1)
Outer system quad moudle	Dual bare module(1)
	Dual PCB(1)
Digital triplet L0 stave module	Digital single bare module(3)
	Triplet stave PCB(1)
Digital triplet L0 Ring0 module	Digital single bare module(3)
	Triplet R0 PCB(1)
Digital triplet L0 Ring0.5 module	Digital single bare module(3)
	Triplet R0.5 PCB(1)
Digital quad module	Digital quad bare module(1)
	Quad PCB(1)
Digital L1 quad moudle	Digital quad bare module(1)
	Quad PCB(1)
Dummy triplet L0 stave module	Dummy single bare module(3)
	Triplet stave PCB(1)
Dummy triplet L0 Ring0 module	Dummy single bare module(3)
	Triplet R0 PCB(1)
Dummy triplet L0 Ring0.5 module	Dummy single bare module(3)
	Triplet R0.5 PCB(1)
Dummy quad module	Dummy quad bare module(1)
	Quad PCB(1)
Dummy L1 quad moudle	Dummym quad bare module(1)
	Quad PCB(1)

²⁰⁵ ことができる。

²⁰⁶ 4.1.2 モジュール情報構造および構成部品との関係の実装

²⁰⁷ 中央データベースにモジュールを登録するためには、モジュールの種類、その構成部品といった情報構
²⁰⁸ 造を決定し、データベース上に定義しておく必要がある。この情報構造をデータベースに実装し、登録で
²⁰⁹ きる仕組みを整えた。詳細な種類と構造については表 4.1 に示す。また Quad モジュールに関する例を図
²¹⁰ 4.1 に示す。

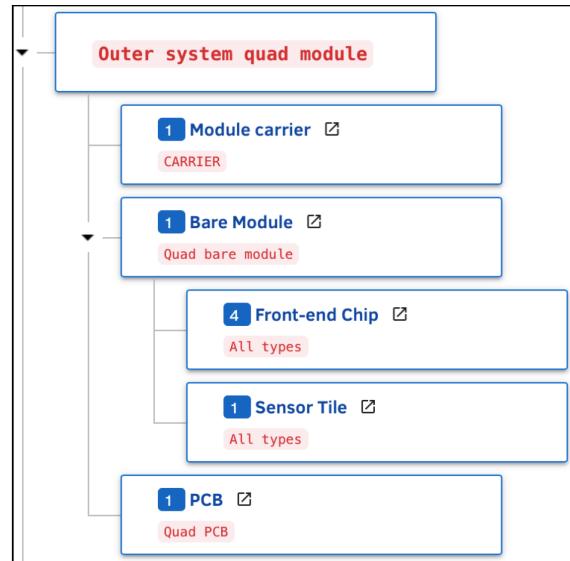


図 4.1 中央データベース内におけるモジュール構造の一例 (Quad モジュール)。例として Outer system quad module の中央データベース内の構造を示している。この種類では構成要素としてそれぞれ対応する種類の Module carrier、Bare Module、PCB を持つことがわかる。さらに Bare Module は FE chip を 4、Sensor を 1 持つことが分かり、Quad モジュールの構造が正しく実装されていることが分かる。

211 4.1.3 組み立て工程および品質試験の情報形式の実装

212 モジュールの情報構造の実装に加えて、品質試験の情報を正確に管理するには、モジュール組み立て工
213 程と付随する品質試験の項目をデータベース上に定義する必要がある。これを実装し、試験結果を適切な
214 組み立て工程へアップロードできる仕組みを整えた。

215 中央データベースにおける組み立て工程と対応する品質試験項目のデータ構造を表 4.2 に示す。

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登録するため、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 Palylene 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	

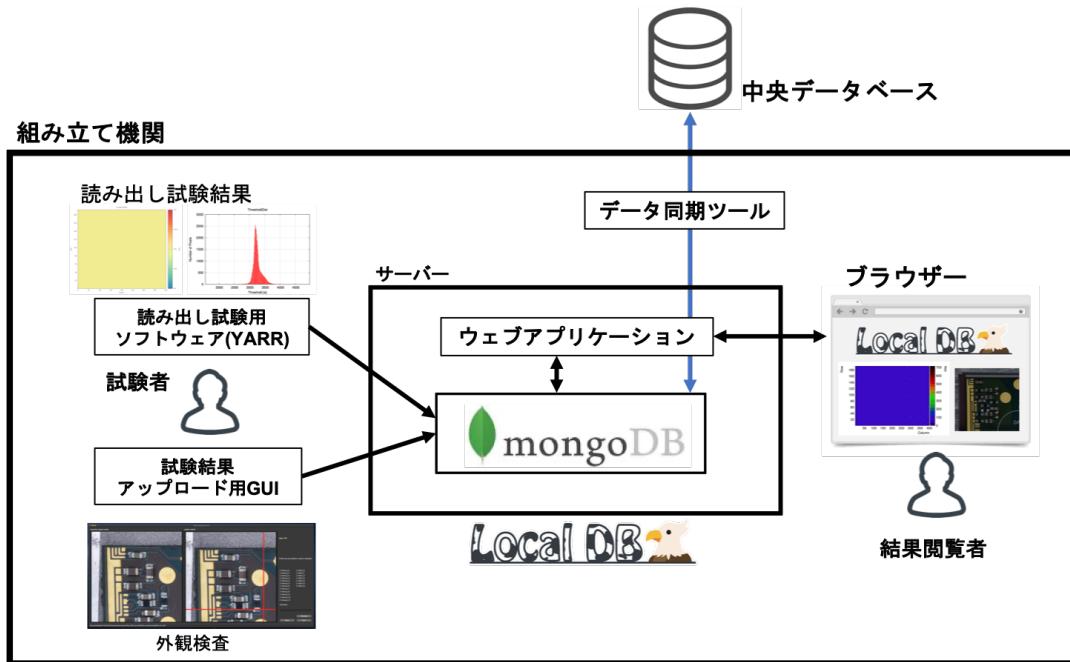


図 4.2 ローカルデータベースシステムの概要。各組み立て機関で MongoDB とローカルデータベース用ウェブアプリケーションの立ち上げを行い、独自にデータ管理をするシステムとなっている。品質試験者は図のようにいくつかのソフトウェアを用いて試験結果を MongoDB に保存する。保存された結果はウェブアプリケーションによって閲覧することができる。また結果は中央データベースに集める必要があるため、同期ツールを用いて試験結果の共有を行う。

216 4.2 ローカルデータベース

217 4.2.1 ローカルデータベースの意義と概要

218 中央データベースでは、前述したようにモジュールの情報のみならず ITk に関わるすべての情報を管
219 理する。データベースの機能としては汎用的に使えるようなものになっている。モジュールの組み立て及
220 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い。1 つの生産
221 現場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、データ管理が簡単にかつ円滑に進むよ
222 うになっているのが好ましい。このような理由から、生産現場での生産性、利便性に特化し、円滑な生産
223 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（ローカルデータベース）を開発している。シス
224 テムの概要図を図 4.2 に示す。オープンソースのサービスである MongoDB[4-1] と、Python のウェブフ
225 レームワークである Flask[4-3] を使用し、開発を行なっているローカルデータベース用ウェブアプリケ
226 ションを併用することで、データ管理や中央データベースとの同期を行うシステムとなっている。

227 具体的にローカルデータベースは以下のようない点を持つ。

- 228 ● ローカルにデータベースサーバーを立てるためアクセス速度が早く、円滑にデータ管理を行うこと
229 ができる。
- 230 ● モジュールの組み立て工程を管理し、生産者の適切な処理を助ける。
- 231 ● モジュールに特化したデータ管理、解析を行うことで異常をいち早く検知できる。
- 232 ● 試験者の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。

233 4.2.2 MongoDB と内部構造 [4-2]

234 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
235 ものを図 4.3 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。
236 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
237 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
238 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。

239 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。

- 240 • 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。
- 241 • 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を容易であり、ウェブアプリケーションとの親和性が高い。
- 242 • 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。

243 モジュール及び品質試験に用いる主なコレクションと内部情報を表 4.3 に示す。

データベース

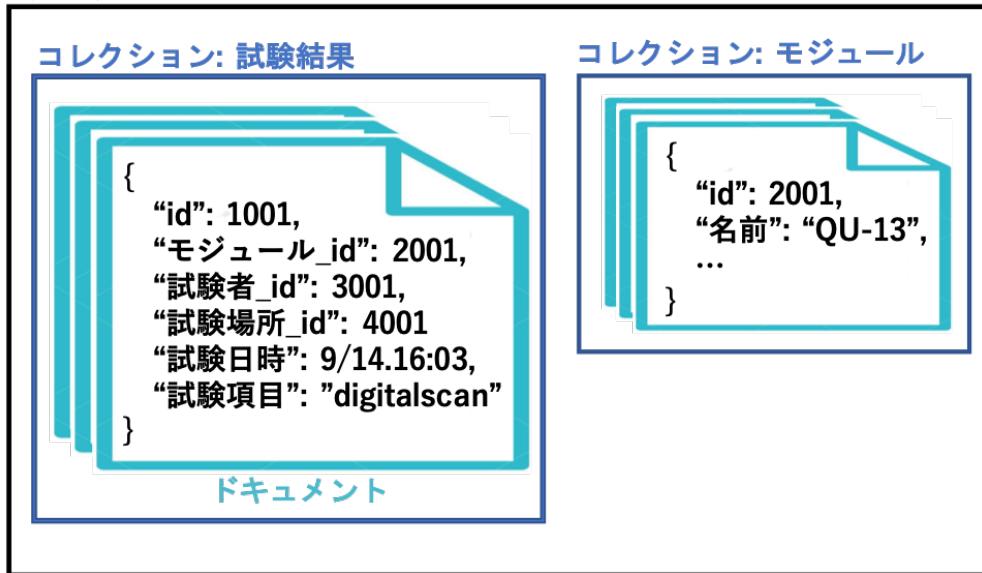


図 4.3 MongoDB の構造の例 [4-2]。図のように MongoDB では JSON 形式でデータを格納する。1 枚の JSON インスタンスをドキュメントと呼び、複数のドキュメントが格納されている枠組みをコレクションと呼ぶ。ドキュメントの構造及びコレクション間の関係等を決めることでデータベースの構造を定義する。

表 4.3 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

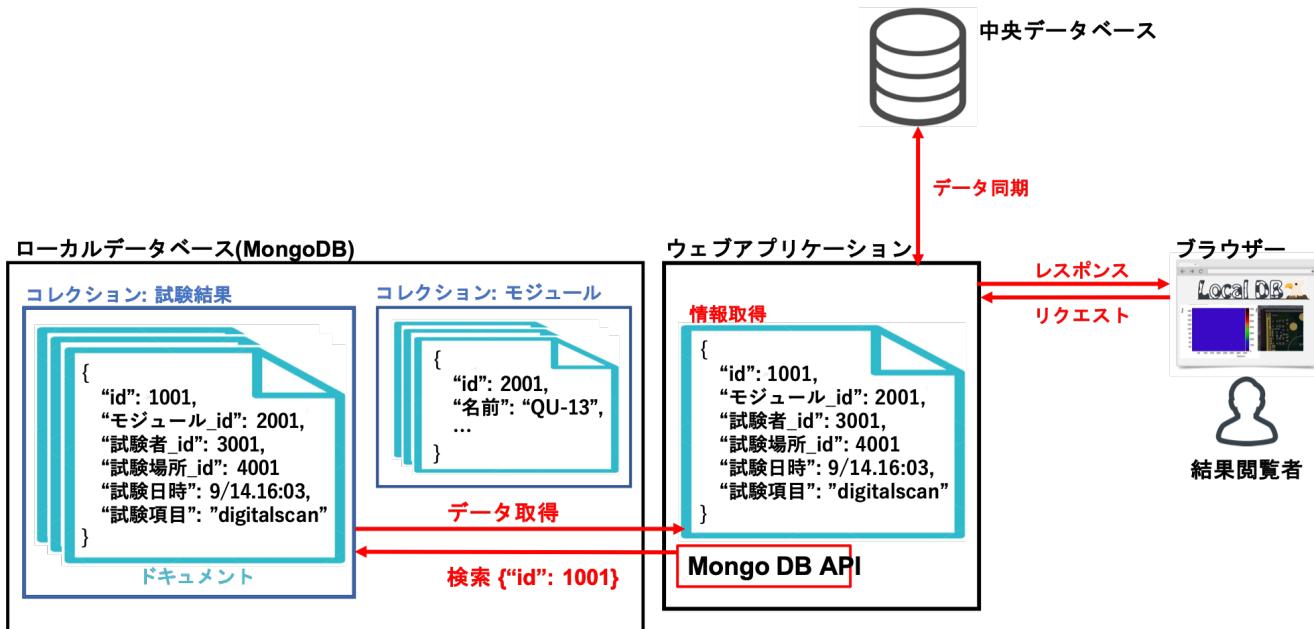


図 4.4 ウェブアプリケーション処理のイメージ。ウェブアプリケーションでは MongoDB 通信 API(PyMongo) を用いて、データベースのコレクションに検索をかけて情報を取り得する。取得した情報は整形されたのちブラウザに送信、中央データベースとの同期等の処理に用いられる。

246 4.2.3 ウェブアプリケーション

247 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
248 用している。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
249 ブラリである PyMongo[4-4] を用いている。ローカルデータベースとアプリケーション間の処理に特化し
250 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
251 のインターフェースとなっている。

252 試験結果を迅速に分かりやすく見るシステムを作り、円滑な生産の補助や異常結果の早期発見を目的と
253 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
254 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
255 用いて行う。

256 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
257 図 4.5 に示す。

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

Result: 1347

Information

Component

Result

Scan

Key	Data
runNumber	1347
testType	std_digitalscan
stage	MODULEWIREBONDING
component	20UPGRD0000001 20UPGFC999999
startTime	2020/11/13 19:49:00
finishTime	2020/11/13 19:49:05
user	hokuyama
site	lazulite
targetCharge	-1
targetTot	-1
exec	-r configs/controller/specConfig.json -c db/data/connectivity.json -s configs/scans/rd53a /std_digitalscan.json -W
stopwatch	analysis: 646 config: 37 processing: 1 scan: 2822
QC	False
environment	True
plots	EnMask L1Dist 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	20UPGFC999999		
afterCfg	json	20UPGFC999999		
EnMask	json	20UPGFC999999		
OccupancyMap	json	20UPGFC999999		
L1Dist	json	20UPGFC999999		

PLOT JSROOT

L1Dist [plotly](#)

20UPGFC999999

EnMask [plotly](#)

20UPGFC999999

OccupancyMap [plotly](#)

20UPGFC999999

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。

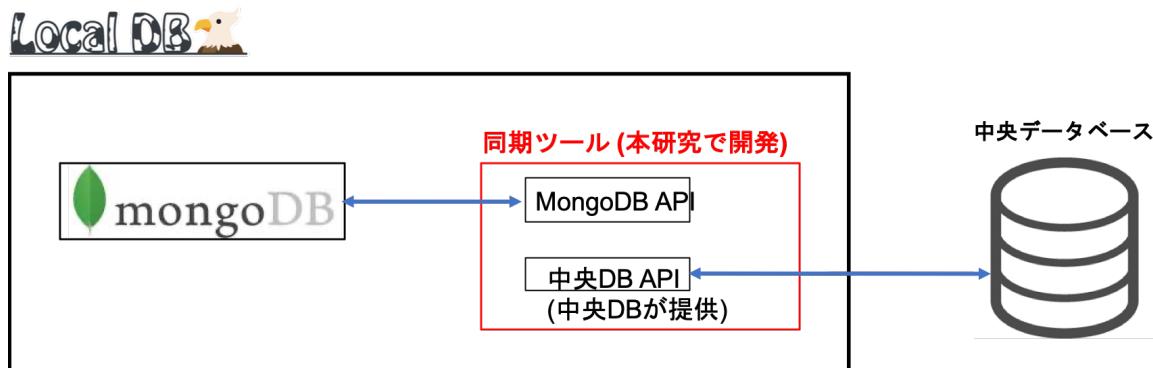


図 4.6 同期ツールのデータ通信のイメージ。本研究で開発を行っているのは図の赤線の領域に対応する同期ツールである。このツールは Python を用いて開発しており、処理の中でローカルの MongoDB と通信する API と、中央データベースが開発、提供をしている API を用いることで、2 つのデータベース間の同期を行っている。このとき、中央データベースとの通信は http 通信で行われる。

262 4.2.4 データベース同期ツール

263 モジュールや品質試験の結果のデータ共有のために、中央データベースとローカルデータベースの間で
264 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。

265 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??で述べ
266 たローカル MongoDB と通信する API の 2 つを用いることで情報共有を行っている。

267 同期ツールのデータ通信のイメージを図 4.6 に示す。

268 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。

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

271 これらの機能のイメージを図 4.7 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。

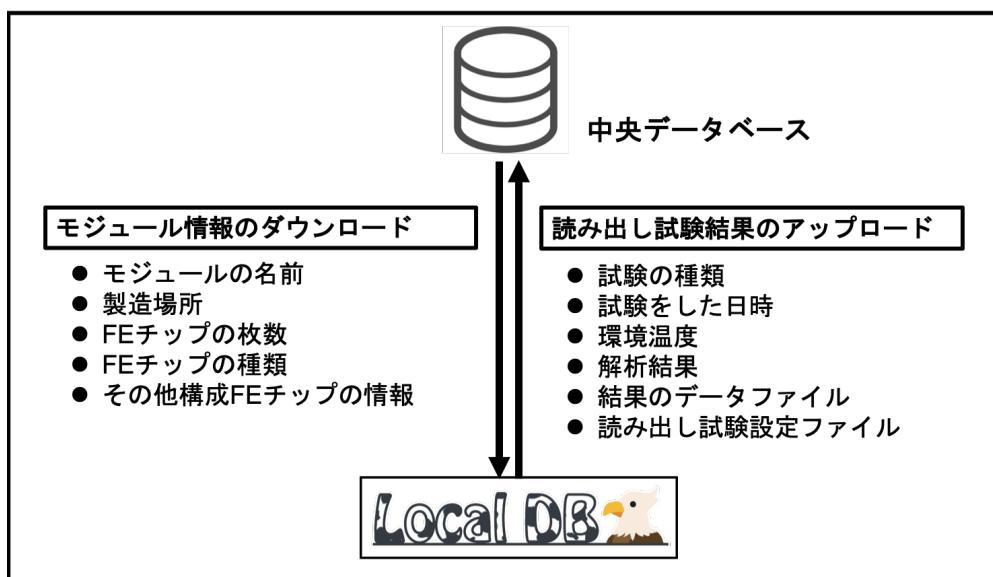


図4.7 同期機能の概要。本研究ではモジュール情報のダウンロードと読み出し試験結果のアップロード機能を実装した。図に示している情報を同期する機能となっている。

4.3 実装した機能

私は、このシステムの中に以下のような機能を実装した。

- ユーザ管理機能
- 品質試験結果の登録と組み立て工程の管理機能
- 読み出し試験におけるピクセル解析ツール
- 読み出し試験結果検索機能
- 節 4.2.4 で述べたデータベース同期ツール

最後のデータベース同期ツール以外の項目について、概要を以下で述べる。

4.3.1 ユーザ管理機能及び各種機能

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システムを開発、実装した。この詳細について以下に述べる。

機能概要

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。各ユーザが使うことのできる機能を表4.4に示す。

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。使用したときの様子を図4.8、4.9に示す。

ユーザ登録操作

表4.4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。

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リプトを用いて実行することで管理者登録がなされ、この際ユーザ名とパスワードを入力する。

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。

- ユーザ名
- 氏名
- 所属機関
- メールアドレス

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。

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 4.4 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データ同期機能 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
<input type="text" value="comment"/>	<input type="text" value="okuyama"/>	<input type="text" value="institution"/>	<input type="button" value="submit"/>	

図 4.8 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 4.9 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。

303 機能の仕組み

304 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。

- 305 1. MongoDBアカウントの作成、読み書き権限の付与
- 306 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

307 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4.3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リスト4.1、4.2以下に示す。

Listing 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。

```
311 {
  "id" : "localdb.hokuyama",
  "userId" : UUID("fee321eb-83b8-434a-a4a0-fff638b5db36"),
  "user" : "hokuyama",
  "db" : "localdb",
  "credentials" : {
    "SCRAM-SHA-1" : {
      "iterationCount" : 10000,
      "salt" : "smkhqvrYnxtsN7rvdadw8g==",
      "storedKey" : "6wC3HyeGTN4px+FV839ou1bylp0==",
      "serverKey" : "/sUu8aX86hnvvMvNq9tC+WZHTDE=="
    },
    "SCRAM-SHA-256" : {
      "iterationCount" : 15000,
      "salt" : "RCC51GuZ4IQK+fINo5HDKP4Vp6LPdersp2gmIA==",
      "storedKey" : "+WwblqTl/SvyyiP7H867kMPPh3Uy2wRIeQ2PgCpdWzs",
      "serverKey" : "651T+2BkP1FE4YEHKbxf+JTnzsR9yWcZQPA/y9RN8e0"
    }
  },
  "roles" : [
    {
      "role" : "readWrite",
      "db" : "localdb"
    },
    {
      "role" : "readWrite",
      "db" : "localdbtools"
    }
  ]
}
```

```
338     }
339   ]
340 }
```

Listing 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。

```
341 {
342   "_id" : ObjectId("5f0bbe84ef87af2628865de7"),
343   "sys" : {
344     "rev" : 0,
345     "cts" : ISODate("2020-07-13T10:53:07.943Z"),
346     "mts" : ISODate("2020-07-13T10:53:07.943Z")
347   },
348   "username" : "hokuyama",
349   "name" : "Hiroki Okuyama",
350   "auth" : "readWrite",
351   "institution" : "Tokyo Institute of Technology",
352   "Email" : "okuyama@hep.phys.titech.ac.jp",
353   "password" : "5f4dcc3b5aa765d61d8327deb882cf99"
354 }
```

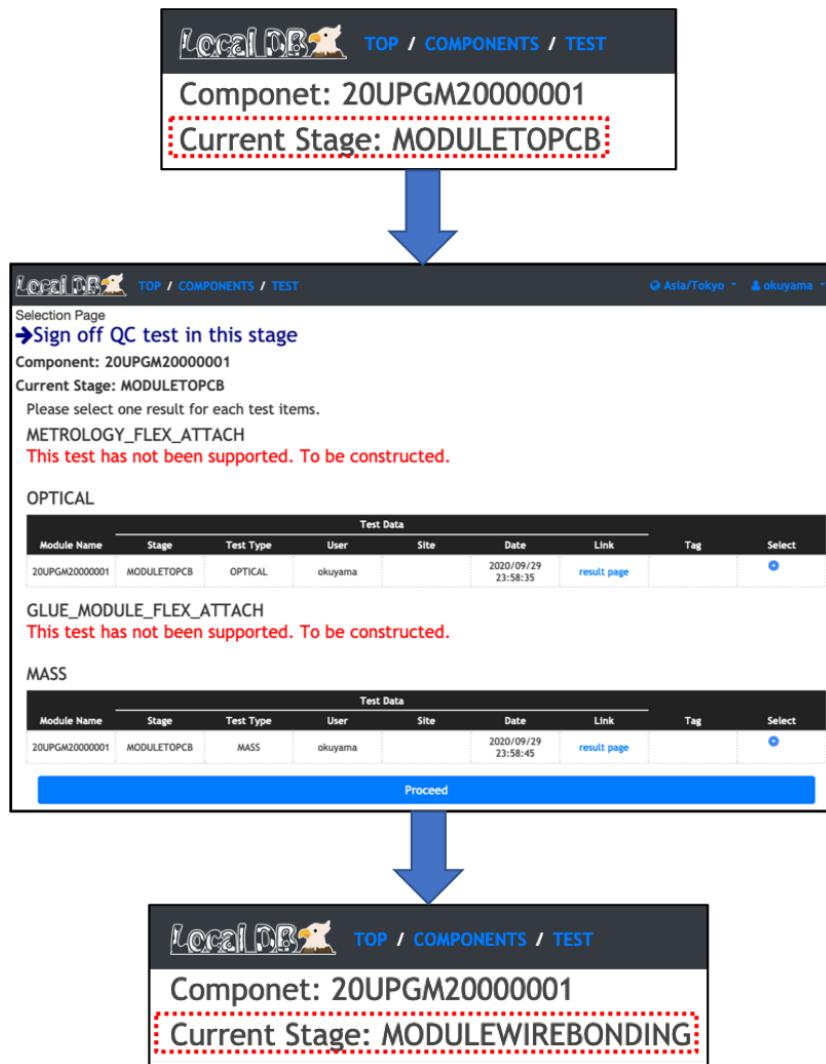


図 4.10 結果選択画面及び組み立て工程表示の例。図の上部で組み立て工程が”MODULETOPCB”である。この段階において結果を選択する処理を行うとローカルデータベース内で選択された結果にタグ付けがなされる。図の中部においてその処理を行なっており、この図では”OPTICAL”と”MASS”の結果を選択している。選択した結果は中央データベースと同期される。また結果選択後は組み立て工程が自動的に更新される。図の下部では”MODULEWIREBONDING”になっていることが分かる。

355 4.3.2 品質試験結果の登録と組み立て工程の自動更新

356 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
 357 アップロードする結果を選択する機能を開発した。品質試験は各モジュール、各組み立て工程に対して行
 358 うものであるため、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
 359 ける組み立て工程の情報は次のものへ自動的に更新する機能となっている。

360 概要

361 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 4.10 に示す。組み立て工程も自動更
 362 新されていることがわかる。

363 仕組み

364 リスト 4.3、4.4 のようなドキュメントを作成、保存する。リスト 4.3 は全てのモジュールに対して共通
365 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
366 データベースより取得される。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の組み立て工程の
367 管理が可能となっている。

368 リスト 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。

- 369 ● 現在工程
370 ● 各工程における品質試験結果の ID

Listing 4.3 組み立て工程及び品質試験一覧情報ドキュメント。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データ取得して作成する。

```
371 {
372     "id" : ObjectId("5fc89aa232d56b29091fd64d"),
373     "sys" : {
374         "mts" : ISODate("2020-12-03T07:58:26.310Z"),
375         "cts" : ISODate("2020-12-03T07:58:26.310Z"),
376         "rev" : 0
377     },
378     "dbVersion" : 1.01,
379     "proddbVersion" : 1.01,
380     "stage_flow" : [
381         "MODULETOPCB",
382         "MODULEWIREBONDING",
383         "MODULEWIREBONDPROTECTION",
384         "MODULEPARYLENECOATING",
385         "MODULETHERMALCYCLING",
386         "MODULEBURNIN",
387         "MODULERECEPTION"
388     ],
389     "stage_test" : {
390         "MODULETOPCB" : [
391             "OPTICAL",
392             "GLUE_MODULE_FLEX_ATTACH",
393             "MASS",
394             "METROLOGY"
395         ],
396     }
397 }
```

```

396     "MODULEWIREBONDING" : [
397         "WIREBONDING",
398         "OPTICAL",
399         "SENSOR_IV",
400         "PIXEL_FAILURE_TEST",
401         "SLDO_VI",
402         "WIREBOND",
403         "CHIP_CONFIGURATION"
404     ],
405     "MODULEWIREBONDPROTECTION" : [
406         "OPTICAL",
407         "POTTING",
408         "MASS",
409         "READOUT_IN_BASIC_ELECTRICAL_TEST",
410         "SENSOR_IV",
411         "REGISTER_TEST"
412     ],
413     ...
414 },
415 ...
416 }

```

Listing 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”に各組み立て工程ごとに持つようになっている。

```

417 {
418     "_id" : ObjectId("5fc4be4c12a45922a91b0e75"),
419     "sys" : {
420         "mts" : ISODate("2020-11-30T09:41:32.411Z"),
421         "cts" : ISODate("2020-11-30T09:41:32.411Z"),
422         "rev" : 0
423     },
424     "dbVersion" : 1.01,
425     "proddbVersion" : 1.01,
426     "component" : "5fa79114e615fa000a1a5976",
427     "currentStage" : "MODULEWIREBONDPROTECTION",
428     "latestSyncedStage" : "MODULEWIREBONDING",
429     "status" : "created",

```

```
430     "rework_stage" : [ ],
431     "QC_results" : {
432         "MODULETOPCB" : {
433             "OPTICAL" : "5fc4c2cfb6c93d451e2c9ac1",
434             "GLUE_MODULE_FLEX_ATTACH" : "-1",
435             "MASS" : "5fc4c2da27766dc6e89c024f",
436             "METROLOGY" : "5fc4c2eaf1f19d9cb5859f00"
437         },
438         "MODULEWIREBONDING" : {
439             "WIREBONDING" : "-1",
440             "OPTICAL" : "5fc4c4c8b7d0c86912b4958f",
441             "SENSOR_IV" : "5fc4c59e9e283a57ccaa1088",
442             "PIXEL_FAILURE_TEST" : "5fca342f6e9f1f5eafedfb92",
443             "SLDO_VI" : "-1",
444             "WIREBOND" : "-1",
445             "CHIP_CONFIGURATION" : "-1"
446         },
447         "MODULEWIREBONDPROTECTION" : {
448             "OPTICAL" : "-1",
449             "POTTING" : "-1",
450             "MASS" : "-1",
451             "READOUT_IN_BASIC_ELECTRICAL_TEST" : "-1",
452             "SENSOR_IV" : "-1",
453             "REGISTER_TEST" : "-1"
454         },
455         ...
456     }
457 }
```

検索欄

検索結果一覧表

Module Name	Chip Name	Test Data					Tag
		Test Type	User	Site	Date	Link	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	thresholdscan	yuta_miyazaki	kyushu_university	2019/06/11 19:23:18	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	thresholdscan	yuta_miyazaki	kyushu_university	2019/06/11 19:17:33	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	thresholdscan	yuta_miyazaki	kyushu_university	2019/06/11 19:12:03	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	digitalscan	yuta_miyazaki	kyushu_university	2019/06/11 19:05:12	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	selftrigger	yuta_miyazaki	kyushu_university	2019/06/11 18:33:26	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	digitalscan	yuta_miyazaki	kyushu_university	2019/06/11 17:57:48	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	selftrigger	yuta_miyazaki	kyushu_university	2019/06/11 17:55:46	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	digitalscan	yuta_miyazaki	kyushu_university	2019/06/11 17:19:38	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	selftrigger	yuta_miyazaki	kyushu_university	2019/06/11 16:09:06	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	digitalscan	yuta_miyazaki	kyushu_university	2019/06/11 15:33:23	result page	
QU-13	QU-13_chip01 QU-13_chip02 QU-13_chip03 QU-13_chip04	selftrigger	yuta_miyazaki	kyushu_university	2019/06/11 15:32:52	result page	

図 4.11 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。図の例では”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。

4.3.3 読み出し試験結果におけるピクセル解析ツール

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。ローカルデータベースにアップロードされた結果を用いて、この解析を行うツールを開発した。開発の詳細とツール確認のためのデモンストレーションについて 5 章で述べる。

4.3.4 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、閲覧でき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 4.11 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。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

また生産に向けて、検索にかかる処理時間測定を行った。検索機能の詳しい実装方法と処理時間についての詳細は、6 章で述べる。

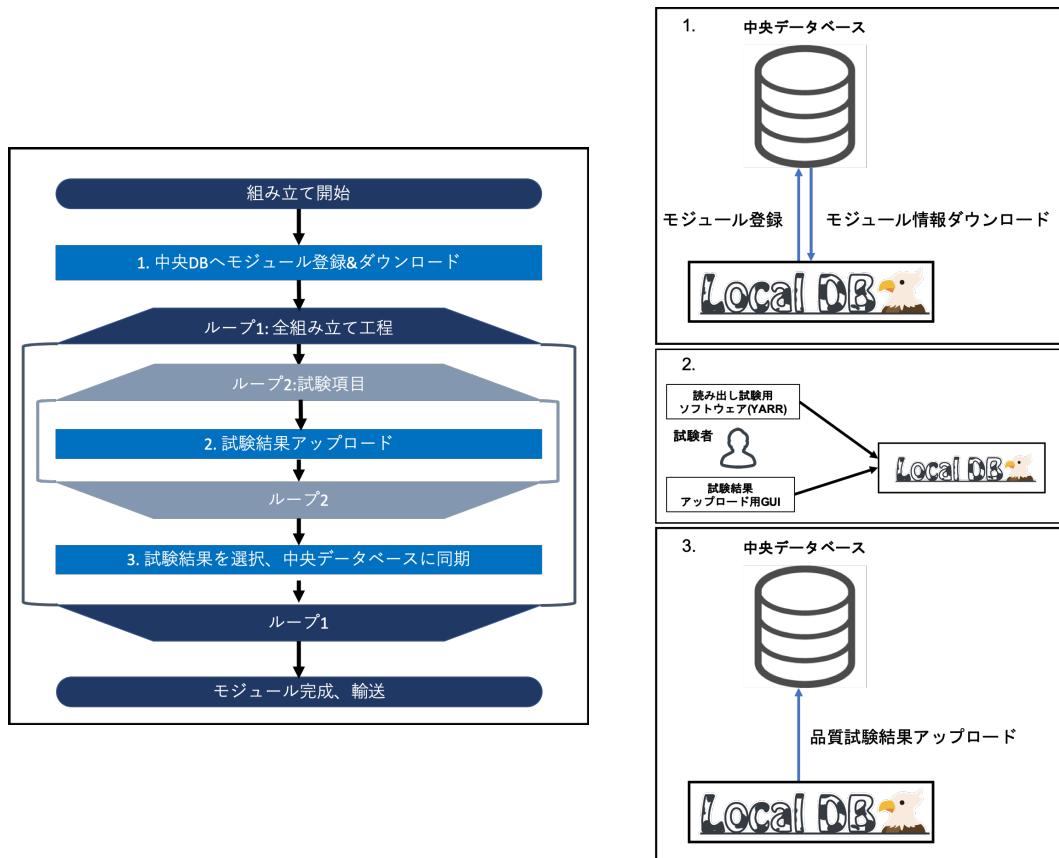


図 4.12 各モジュールにおけるデータベースシステム操作の流れ。モジュール組み立てにおけるデータベース操作の初めに、中央データベースにモジュール登録及びローカルデータベースへモジュール情報のダウンロードを行う(処理 1)。その後、ダウンロードしたモジュールに対して組み立て工程に応じた試験結果を生成、ローカルデータベースに保存する(処理 2)。各組み立て工程の終わりに試験結果の選択を行い、中央データベースに試験結果を同期する(処理 3)。

4.4 量産時の情報登録・同期の流れ

471 量産時におけるデータベース操作は以下の流れで行うことと想定している。

- 472
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
 2. 1で登録したモジュールに対して品質試験結果のローカルデータベースへのアップロード
 3. ステージ毎に品質試験結果の登録と中央データベースへアップロード

473 流れのイメージを図 4.12 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
474 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
475 管理する目的がある。

476 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
477 ている状態となる。データベース操作の流れの一部を学内実験室にて行った。詳細を 5 章で述べる。

Module Production Analysis

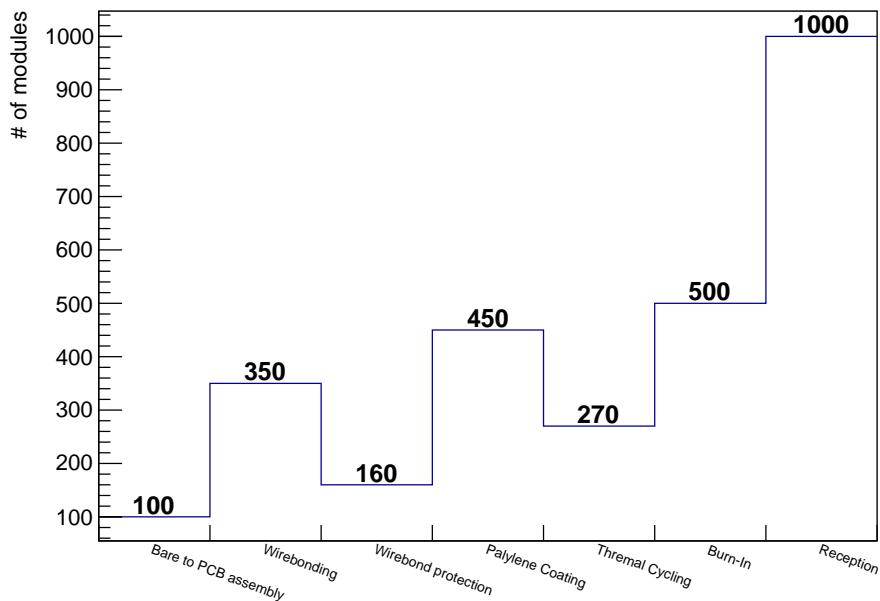


図 4.13 生産時のモジュール組み立て状況解析の例。ローカルデータベースにて組み立て工程は管理され、工程毎に中央データベースに同期されるため、生産時には全てのモジュールの現工程を中央データベースで取得できる。図の例のように各段階におけるモジュール数を見ることで生産状況の確認ができる。さらにある期間ごとに工程を取得することで生産レートなども計算することができ、今後の生産計画やモジュール選別の参考とすることができる。

4.5 モジュール生産状況の解析

上述した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生産場所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。そのため現在組み立てが行われている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 4.13 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。

489 第5章

490 ピクセル解析ツールと読み出し試験のデ 491 モンストレーション

492 品質試験項目の1つである読み出し試験のデモンストレーションを学内実験室にて行った。デモンスト
493 レーションの内容としては、4章で述べた2月にCERNで行ったチュートリアルとほぼ同じ内容である。
494 この章の前半では開発したツールと試験で使用するソフトウェア、ハードウェアについて説明し、後にデ
495 モンストレーションの内容、各ソフトウェアの機能確認について述べる。

496 5.1 ピクセル解析ツールの開発

497 品質試験における読み出し試験では、3章で述べたようにモジュールの性能確認のためにピクセル解析
498 を行う。これを円滑に行うために、ピクセル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
499 データベースシステムに組み込んだ。このツールについての詳細を以下に示す。

500 5.1.1 概要

501 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
502 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。

503 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
504 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
505 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
506 フレームワークであるROOT[5-5]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
507 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5.1に示す。

508 実際に作ったTreeファイルと、データ構造のイメージを図5.2に示す。

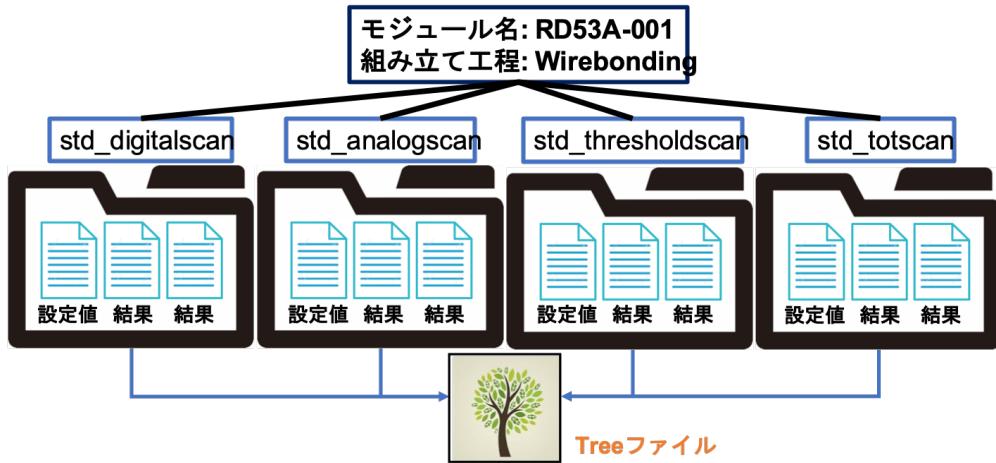


図 5.1 ピクセル解析ツールにおけるファイル統合処理のイメージ。YARR の出力ファイル及びディレクトリは std_digitalscan や std_thresholdscan というように読み出し項目ごとである。ピクセル解析ツールでは、図のようにあるモジュールに関する結果ファイルを統合し、ピクセルごとに解析処理を簡易化する狙いがある。



図 5.2 Tree ファイルとそのデータ保持。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。

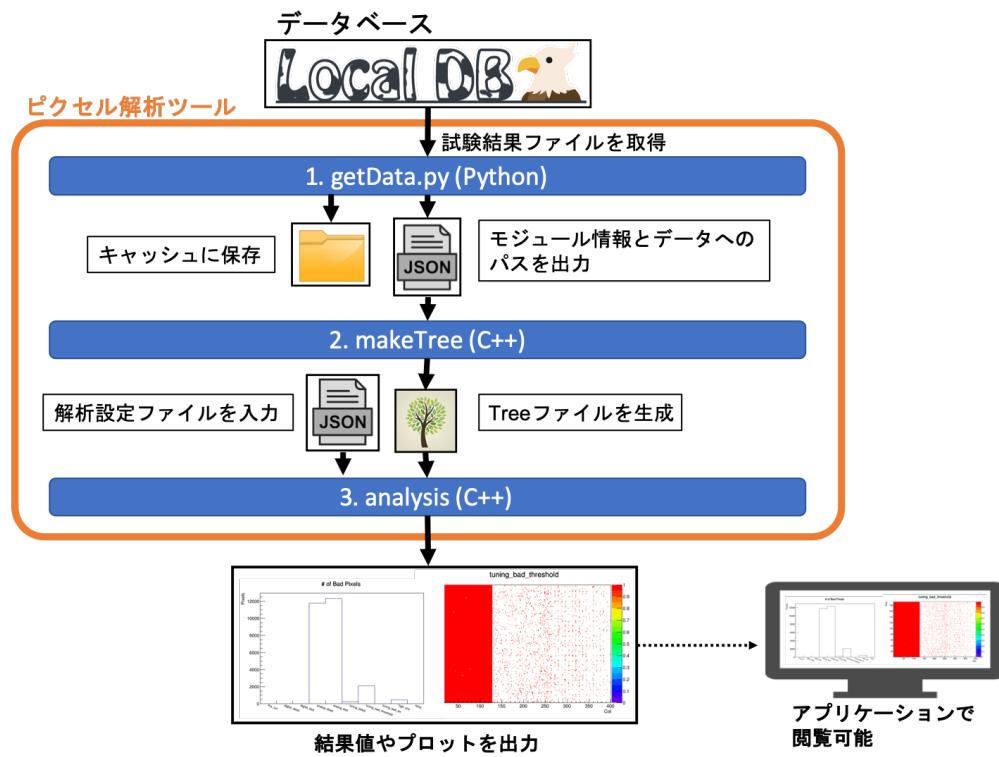


図 5.3 ピクセル解析ツールの処理の流れ。ピクセル解析ツールは、図のように 3 つの実行ファイルにより構成される。getData.py(Python) にてデータベースから結果の取得を行い、makeTree(C++) にて Tree ファイルの作成、analysis(C++) にてピクセル解析及び結果のプロットが outputされる。

5.1.2 ツールの内部構造と処理の流れ

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明する。

5.1.2.1 getData.py (Python)

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ディレクトリに保存。

5.1.2.2 makeTree (C++)

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。

5.1.2.3 analysis (C++)

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。

処理の流れのイメージを図 5.3 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。

5.2 デモンストレーションと機能確認

上述したピクセル解析ツールを含む読み出し試験用ソフトウェアの機能確認を目的として、生産時における流れのデモンストレーションを行なった。その詳細について以下に示す。

5.2.1 用いたソフトウェアの概要

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 5.4 に示す。

- YARR(commit:6b3ffe92)
- MongoDB(version: v4.2.6)
- ローカル DB ウェブアプリケーション (tag: ldbtoolv1.4)
- 中央データベースとの同期ツール (tag: ldbtoolv1.4)
- ピクセル解析ツール (tag: v1.0.2)
- 時系列データ用データベース (InfluxDB[5-6](version: 1.8.0))
 - 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 情報を時間情報と共に保存、管理するために用いる。
- InfluxDB 解析ソフト (Grafana[5-7](version: 5.1.0))
 - 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 データを閲覧することができる。
- 電源操作用ソフト
 - 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 にアップロードする。CERN で開発されている PySerial[5-8](commit:0d14fcdb) を改良し作成した。
- 温度読み出し用ソフト
 - 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理を作成したもの。RaspberryPi 上に保存し、処理を実行。

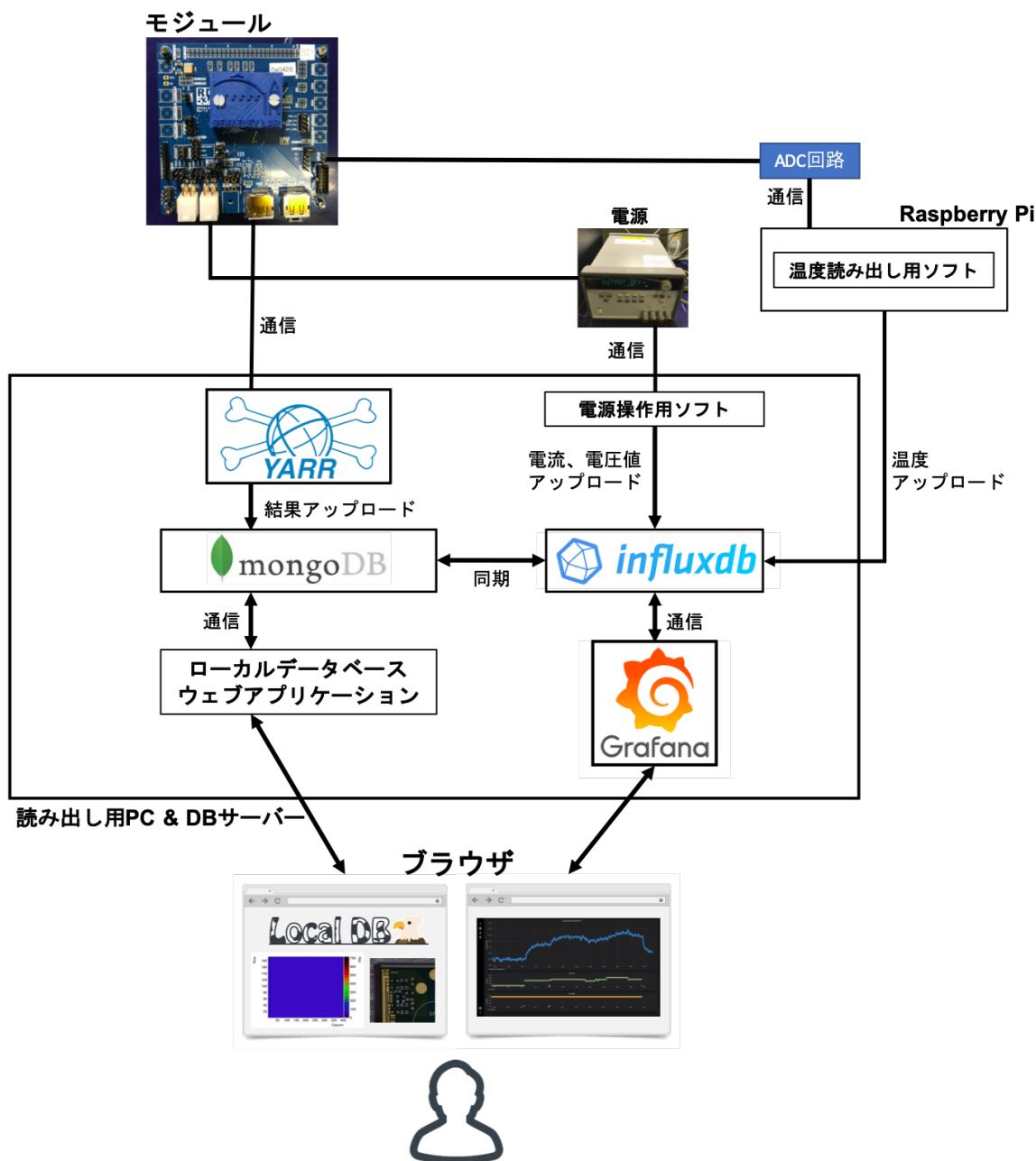


図 5.4 読み出し試験に用いるソフトウェアの概要。FE チップの読み出しとそのデータ通信は YARR を用いて行われる。試験結果は MongoDB にアップロードされ、試験者はウェブアプリケーションを通じて結果を確認することができる。電源操作用ソフトを用いて電源のスイッチ、電圧、電流値の取得がなされ、取得値は InfluxDB に保存される。モジュール付属のサーミスタ読み出しシステムを用いて FE チップ付近の温度を読み出し、InfluxDB に保存する。InfluxDB に保存された DCS データは Grafana を用いてブラウザ上で確認することができる。また MongoDB に同期されるため、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。

546 5.2.2 用いたハードウェア

547 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。

548 RD53A シングルモジュール (RD53A Single Chip Card, SCC)[5-10]

549 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
550 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ルで
551 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、データ転送
552 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 サーミスタを搭載して
553 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 5.5 に写真を
554 示す。

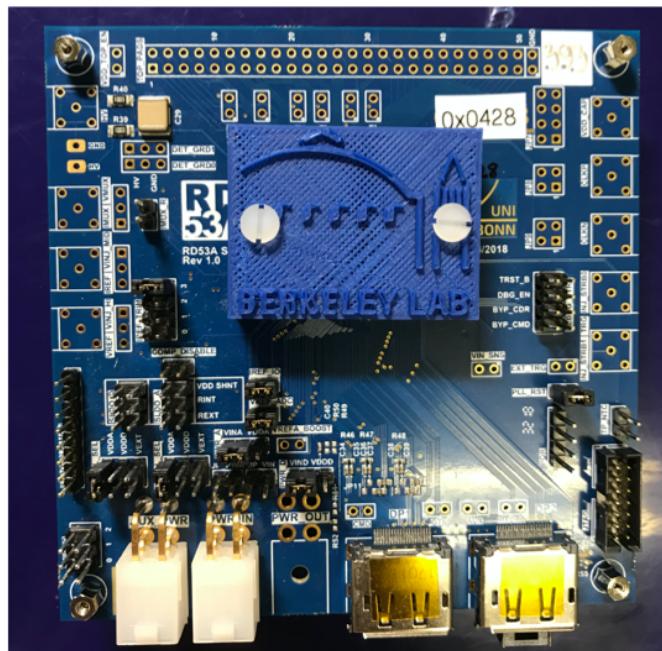


図 5.5 RD53A シングルモジュール (SCC)[5-10]。RD53A の FE チップを一枚搭載したモジュー
ルである。図の中心部の青いカバーで囲われた中に FE チップが設置されている。このボード上には
データ転送端子、電源端子、サーミスタ抵抗取得端子が設置されており、これらの端子に対応する配
線をすることで読み出し試験のセットアップを組む。

555 モジュールサーミスタ温度読み出しシステム

556 モジュール付属のサーミスタ、ADC、Raspberry Pi を用いた温度読み出しシステムを作成した。こ
557 のシステムの中で扱った装置を表 5.1 に、回路図、実際に配線した様子を図 5.6 に示す。スクリプト上
558 で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る。ADC 値は
559 GPIO 通信 [5-9] により取得している。

560 電源

561 モジュールの電圧供給に KEYSIGHT の E3646A 60W デュアル出力電源 [5-1](図 5.7) 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[5-3]
RaspberryPi	Raspberry Pi 3 Model B Plus Rev 1.3[5-4]

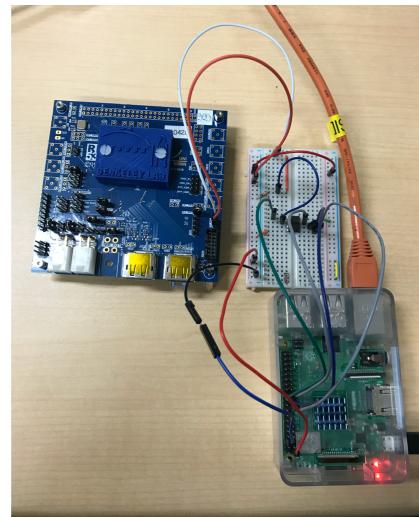
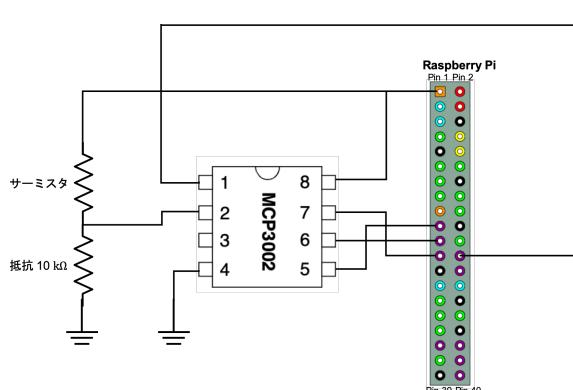


図 5.6 モジュール付属サーミスタを用いた温度読み出し回路。図は回路図（左図）と実際に配線し、読み出しを行っている様子（右図）を示している。抵抗、MCP3002、Raspberry Pi を用いて読み出し回路を作成した。抵抗と MCP3002 は右図のようにブレッドボード上に設置した。ADC と RaspberryPi は GPIO 通信 [5-9] を行うことで、ADC 値を取得している。



図 5.7 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 [5-1] を用いた。

⁵⁶² FPGA ボード

⁵⁶³ FPGA ボードに XpressK7[5-2] を用いた。図 5.8 に示す。

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

564 FMC-DisplayPort 変換カード

565 使用した FMC-DisplayPort 変換カード（オハイオカード）を図 5.9 に示す。



図 5.8 使用した FPGA ボード (XpressK7[5-2])。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。



図 5.9 使用した FMC-DisplayPort 変換カード。FMC 端子を 4 つの mini Display Port 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。

566 PC

567 今回用いた PC の性能を表 5.2 に示す。

568 セットアップ

569 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 5.10 に示す。

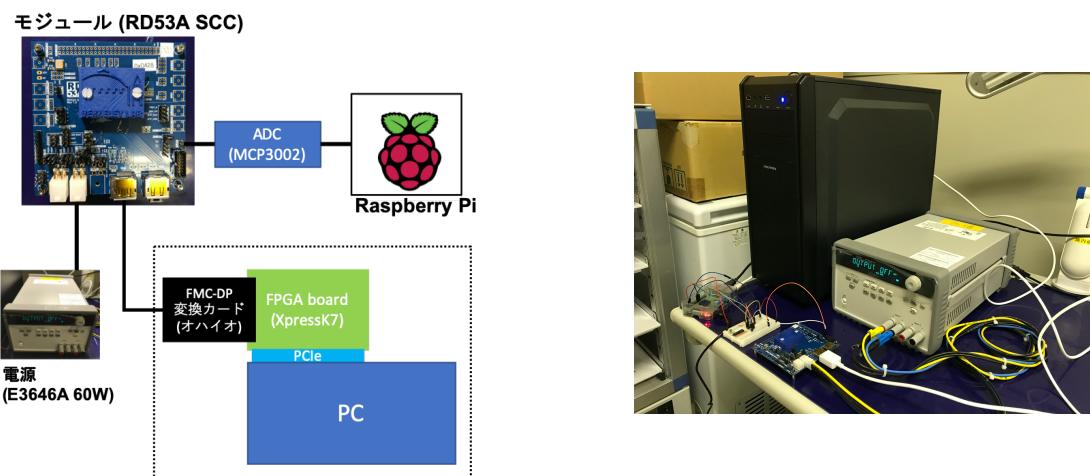


図 5.10 ハードウェアセットアップ。図はセットアップの概要(左図)と実際に設置した様子(右図)を示している。FE チップ読み出し装置、電源、サーミスタ読み出し装置をそれぞれ設置、配線し、読み出し操作、データ取得を行った。

Latest Result								
Module Name	Chip Name	Current Stage	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC9999999	None	None	None	None	None		

図 5.11 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ウェブ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。

5.2.3 デモンストレーションの流れ

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 [5-11] に従い以下のモジュール、FE チップを登録した。

モジュール ID 20UPGR00000001

FE チップ ID 20UPGFC9999999

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。確認した機能を行った流れの順に以下に示す。

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.
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.
3. DCS 情報の取得、監視.
4. 試験結果検索.
5. 試験結果閲覧.
6. 結果選択とピクセル解析機能.
7. 中央データベースへ試験結果のアップロード

5.2.4 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。

中央データベースからモジュール情報のダウンロード

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 5.11 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。



図 5.12 DCS 情報のモニタリングの様子。図において全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。それぞれのデータの監視が行っ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。

589 読み出し試験実施

590 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。

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

600 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
601 ベースに保存した。

602 DCS 情報の監視

603 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
604 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 5.12 に示す。

The screenshot shows two instances of the LocalDB application interface. The top instance displays a 'Scan List' with a table titled 'Test Data'. The table has columns: Module Name, Chip Name, Test Type, User, Site, Date, Link, and Tag. There are 15 rows of data, each corresponding to a different test type (e.g., std_digitalscan, std_noisescan, std_totscan, etc.) and date (e.g., 2020/11/13 19:49:40). The bottom instance shows the same interface after a search for 'hokuyama std_totscan'. The search bar now contains 'hokuyama std_totscan' and the results table shows only one row of data from the previous list, indicating a successful search.

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC9999999	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:40	result page	
20UPGR00000001	20UPGFC9999999	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:00	result page	
20UPGR00000001	20UPGFC9999999	std_noisescan	hokuyama	lazulite	2020/11/13 19:41:07	result page	
20UPGR00000001	20UPGFC9999999	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page	
20UPGR00000001	20UPGFC9999999	std_thresholdscan	hokuyama	lazulite	2020/11/13 19:38:59	result page	
20UPGR00000001	20UPGFC9999999	syn_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:38:11	result page	
20UPGR00000001	20UPGFC9999999	syn_tune_globalpreamp	hokuyama	lazulite	2020/11/13 19:37:44	result page	
20UPGR00000001	20UPGFC9999999	syn_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:36:54	result page	
20UPGR00000001	20UPGFC9999999	lin_retune_globalthreshold	hokuyama	lazulite	2020/11/13 19:36:35	result page	
20UPGR00000001	20UPGFC9999999	lin_tune_globalpreamp	hokuyama	lazulite	2020/11/13 19:36:13	result page	
20UPGR00000001	20UPGFC9999999	lin_retune_globalthreshold	hokuyama	lazulite	2020/11/13 19:35:55	result page	
20UPGR00000001	20UPGFC9999999	lin_retune_globalthreshold	hokuyama	lazulite	2020/11/13 19:35:35	result page	
20UPGR00000001	20UPGFC9999999	lin_tune_globalthreshold	hokuyama	lazulite	2020/11/13 19:35:09	result page	
20UPGR00000001	20UPGFC9999999	lin_tune_globalthresholdes	hokuyama	lazulite	2020/11/13 19:34:53	result page	
20UPGR00000001	20UPGFC9999999	diff_tune_globalthresholdes	hokuyama	lazulite	2020/11/13 19:34:36	result page	

↓

“hokuyama std_totscan”
で検索

Test Data							
Module Name	Chip Name	Test Type	User	Site	Date	Link	Tag
20UPGR00000001	20UPGFC9999999	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page	

図 5.13 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。図に例では”hokuyama std_totscan”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。

605 検索機能

606 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。

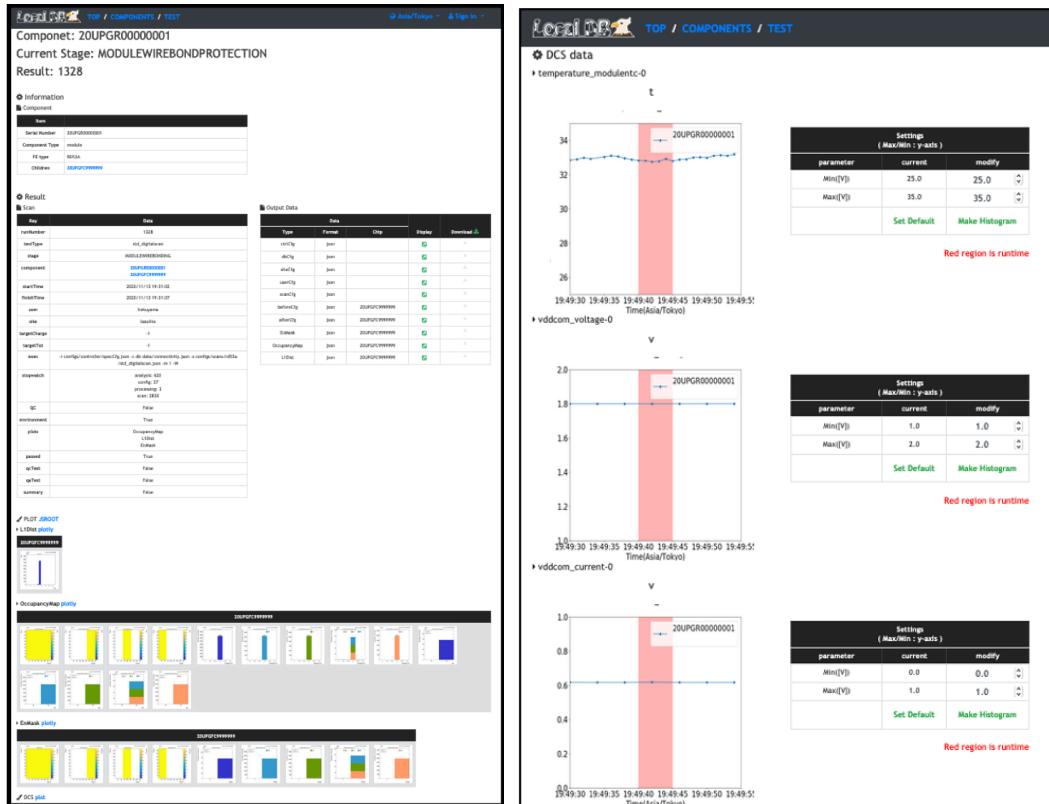


図 5.14 試験結果の閲覧。図は試験結果（左図）と試験における DCS データのグラフ（右図）を示しており、上から温度、電圧、電流となっている。図は std_digitalscan の結果であり、試験情報及び結果のグラフが確認できる。また右図より DCS データも正常にローカルデータベース上に保存され、表示されていることが分かる。

607 試験結果閲覧

608 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 5.14 に示す。

Digital_scan

run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1348	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:40	result page	<input type="radio"/>	<input type="radio"/>
1347	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:49:00	result page	<input type="radio"/>	<input type="radio"/>
1328	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:31:02	result page	<input checked="" type="radio"/>	<input type="radio"/>
1327	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:29:24	result page	<input type="radio"/>	<input type="radio"/>
1326	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_digitalscan	hokuyama	lazulite	2020/11/13 19:28:38	result page	<input type="radio"/>	<input type="radio"/>

Analog_scan

run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1329	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_analogscan	hokuyama	lazulite	2020/11/13 19:31:13	result page	<input checked="" type="radio"/>	<input type="radio"/>

Threshold_scan

run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1344	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_thresholdscan	hokuyama	lazulite	2020/11/13 19:38:59	result page	<input checked="" type="radio"/>	<input type="radio"/>
1330	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_thresholdscan	hokuyama	lazulite	2020/11/13 19:31:24	result page	<input type="radio"/>	<input type="radio"/>

ToT_scan

run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1345	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_totscan	hokuyama	lazulite	2020/11/13 19:40:57	result page	<input checked="" type="radio"/>	<input type="radio"/>

Noise_scan

run Number	Module Name	Chip Name	Stage	Test Type	User	Site	Date	Link	Tag	Select
1346	20UPGR000000001	20UPGFC99999999	MODULEWIREBONDING	std_noisescan	hokuyama	lazulite	2020/11/13 19:41:07	result page	<input checked="" type="radio"/>	<input type="radio"/>

Proceed

図 5.15 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。

609 結果選択とピクセル解析

610 読み出し結果を選択し、ピクセル解析を行なった。結果選択画面を図 5.15 に示す。

611 このデモンス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
612 実装している以下の項目を抜粋した。

- 613 1. Digital Dead
- 614 2. Digital Bad
- 615 3. Analog Dead
- 616 4. Analog Bad
- 617 5. Tuning Failed
- 618 6. Tuning Bad for Threshold
- 619 7. Tuning Bad for ToT
- 620 8. High ENC
- 621 9. Noisy

622 解析結果を図 5.16、それぞれの評価基準における不良ピクセルの分布を図 5.17 に示す。

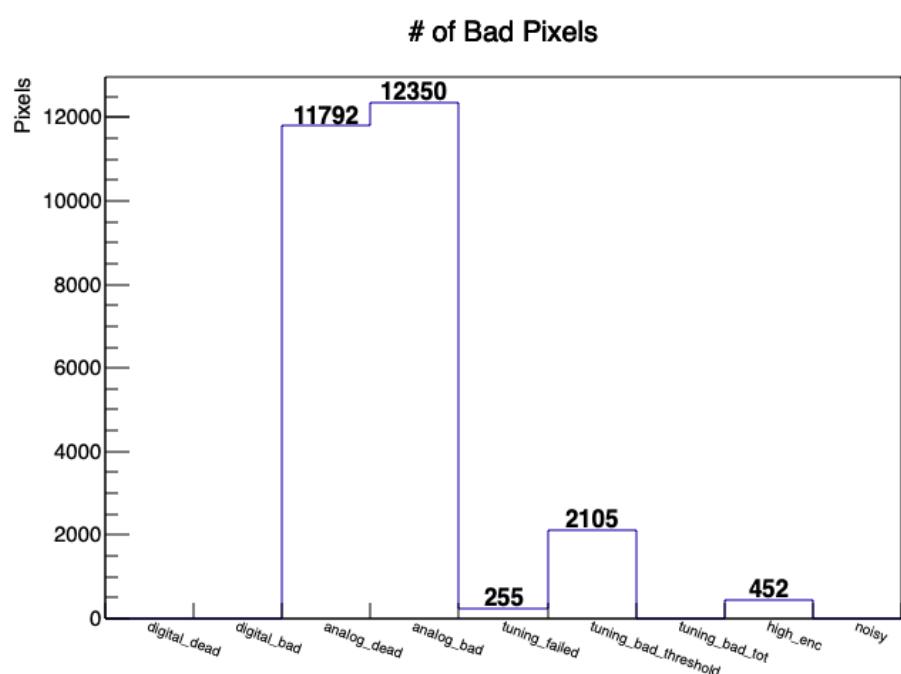


図 5.16 ピクセル解析結果。図において横軸は評価基準、縦軸は該当するピクセル数を表す。
analog_dead, analog_bad の割合が多いいため、アナログ回路読み出しに失敗しているピクセルが多いことが読み取れる。

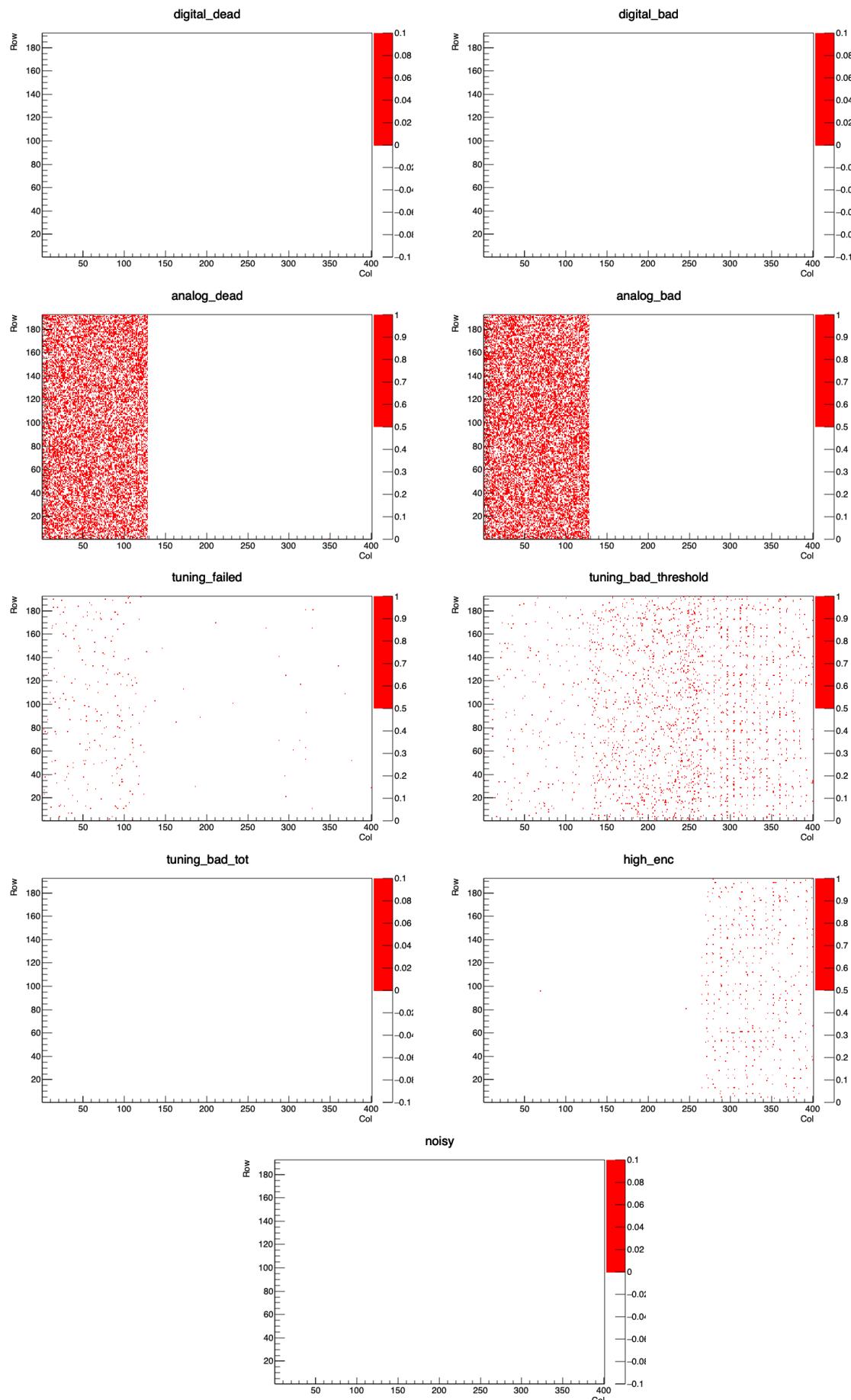


図 5.17 各評価基準における不良ピクセルの分布。各図は二次元ヒストグラムであり、横軸が FE チップにおける各ピクセルの列番号、縦軸が行番号を示しており、赤い領域が不良ピクセルを表している。図 5.16においてアナログ回路読み出しに失敗しているピクセルが多いことがわかったが、その不良ピクセルは synchronous フロントエンド上に存在していることが分かる。また Threshold 調整に失敗しているピクセル (tuning_bad_threshold) は全体的に分布していることが分かる。

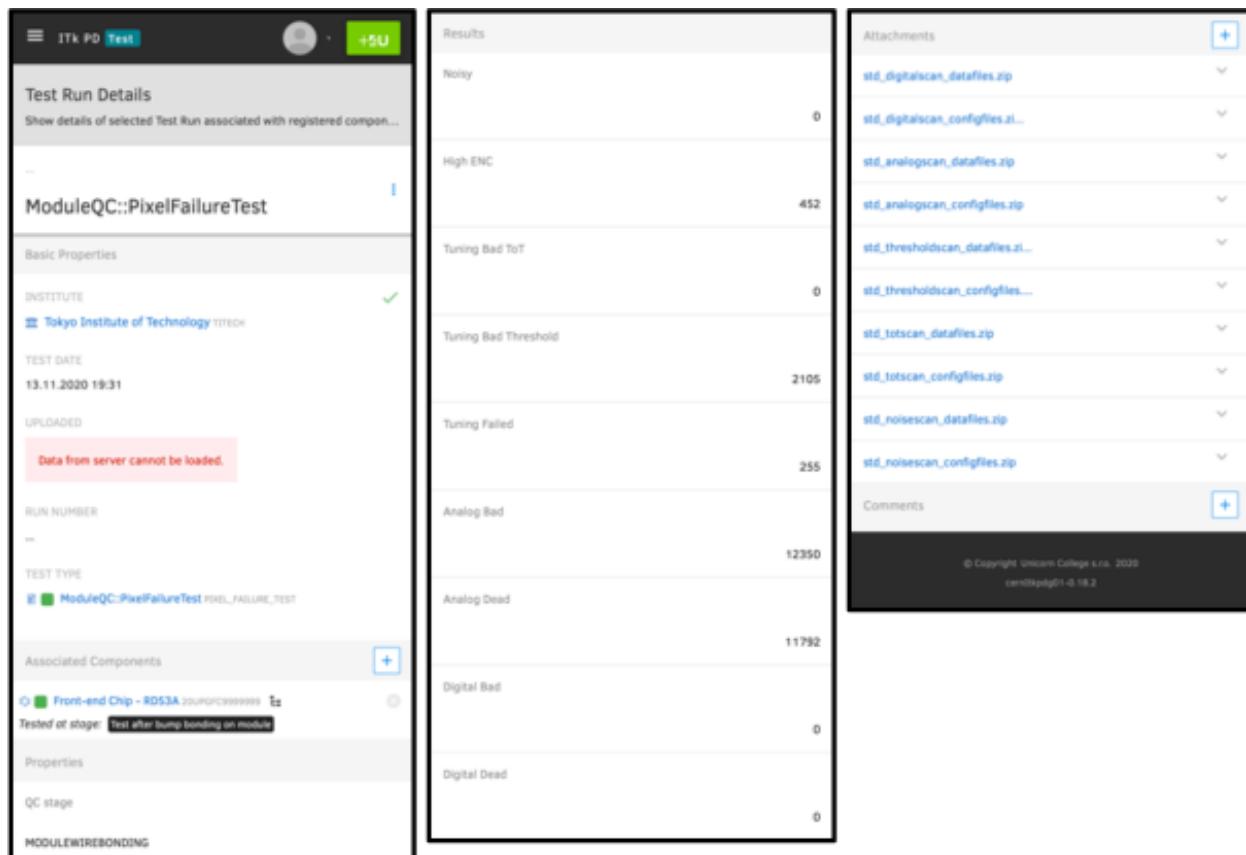


図 5.18 中央データベースにおける読み出し試験及びピクセル解析結果画面。読み出し試験及びピクセル解析結果を中央データベースにアップロードした。図は中央データベースのウェブページを示しており、アップロードされた試験結果画面である。赤線の領域に各評価基準における不良ピクセルの数、青線の領域に各試験項目ごとの結果ファイルをまとめた Zip ファイルが表示されていることが分かる。試験結果のアップロードが正常に完了し、中央データベース上で結果を確認できることが分かる。

623 試験結果アップロード

624 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
625 を確認した。図 5.18 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
626 プロードされていることが分かる。

627 第6章

628 ローカルデータベースにおける検索機能 629 とその処理時間

630 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。
 631 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
 632 うに、任意のタイミングで必要な結果を取得できる検索機能を実装した。詳細について以下に示す。

633 6.1 実装方法

634 今回の実装では、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
 635 ようと考えた。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にある。
 636 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
 637 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
 638 ている。品質試験の結果が後から上書きされると、結果の信頼性を失うと考えているからである。そのた
 639 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、検索キーワードとしてサポートすれば
 640 十分であると考えた。

- 641 • モジュール及びFEチップのID.
- 642 • 読み出し試験項目(例:std_digitalscan)
- 643 • 読み出し試験者.
- 644 • 読み出し試験場所.
- 645 • 試験日時.
- 646 • タグ機能を用いてつけられたタグ.

647 そこで実装方法として、以下の2つを考えた。

- 648 1. 各試験に関する情報をプログラム上で配列に保持し、検索キーワードが含まれるかを確認する方法.
 - 649 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.
- 650 これらについて以下で詳細を説明する。

6.1.1 方法 1: Python リストを用いた一致確認

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。

1. ユーザが検索キーワードを入力し、処理を実行.
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.
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.
4. ブラウザに送信.

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内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。

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てしまう。このシステムのデータ構造においては、表 4.3 において各試験結果の情報を保存する testRun、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。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。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、図 6.4 のようになり、確かにドキュメント数に対して 2 次的に増加していることが分かる。測定の詳細については後述する。

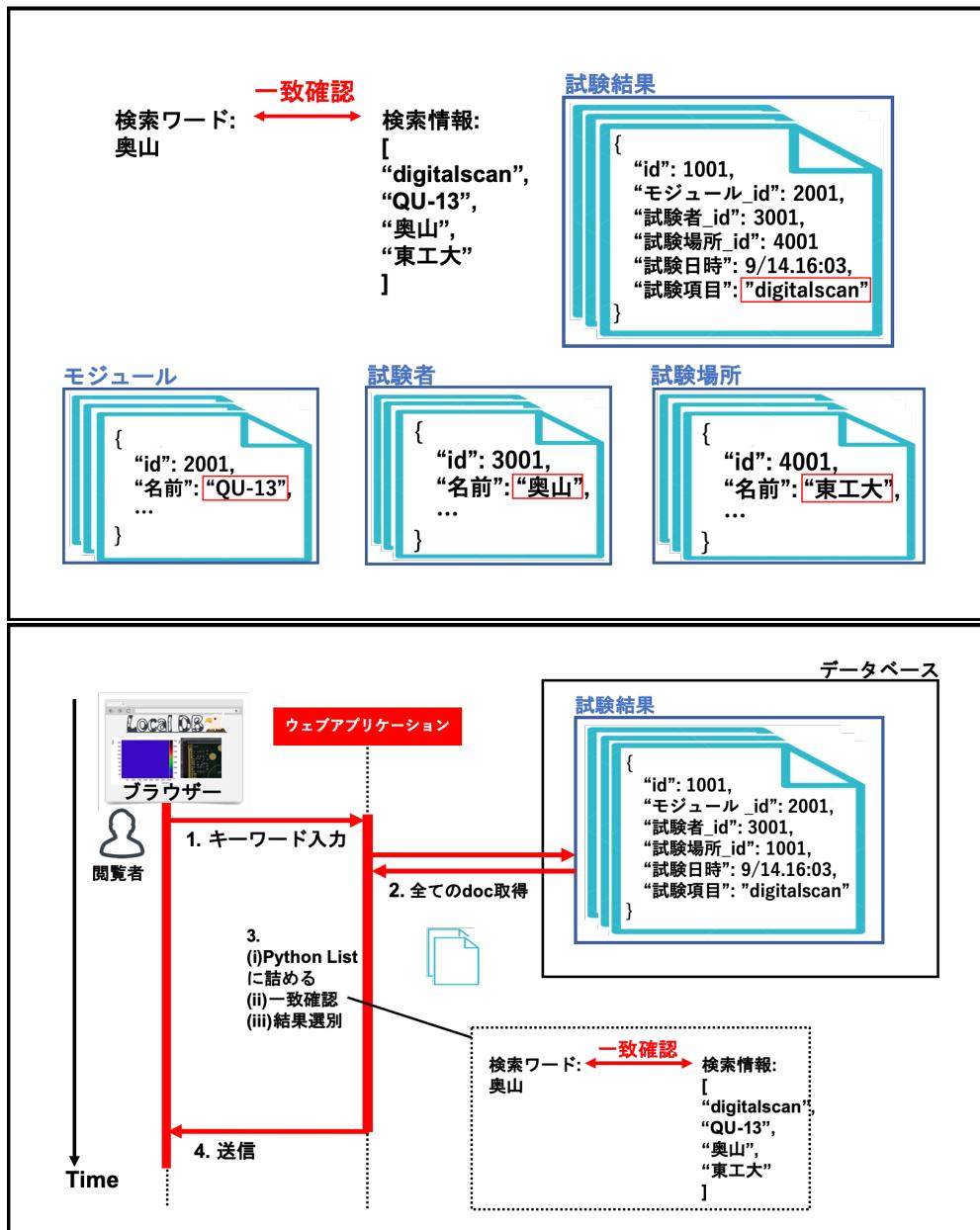


図 6.1 検索機能実装方法 1:Python リストを用いた場合の検索処理のイメージ図。上図は各コレクションと保存されている情報の例を示しており、下図は実際に検索を行った時の処理の流れを表している。上図中の赤枠で囲われた情報のように検索対象となる名前の情報は複数のコレクションにまたがって保存されている。各試験結果に対してこれらの情報を集めリスト内に保持し、各要素とキーワードとの一致を確認することで検索処理を行う。

6.1.2 方法 2: 検索情報を持つドキュメントを作成、使用

検索機能を改善するため方法 2 を考案し、実装を行った。改善の方法として、読み出し試験のアップロードシステム及びウェブアプリケーションでの結果確認システムは既に使われていたため、データ構造及び使用している 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。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き、処理実行時にはそれを参照す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

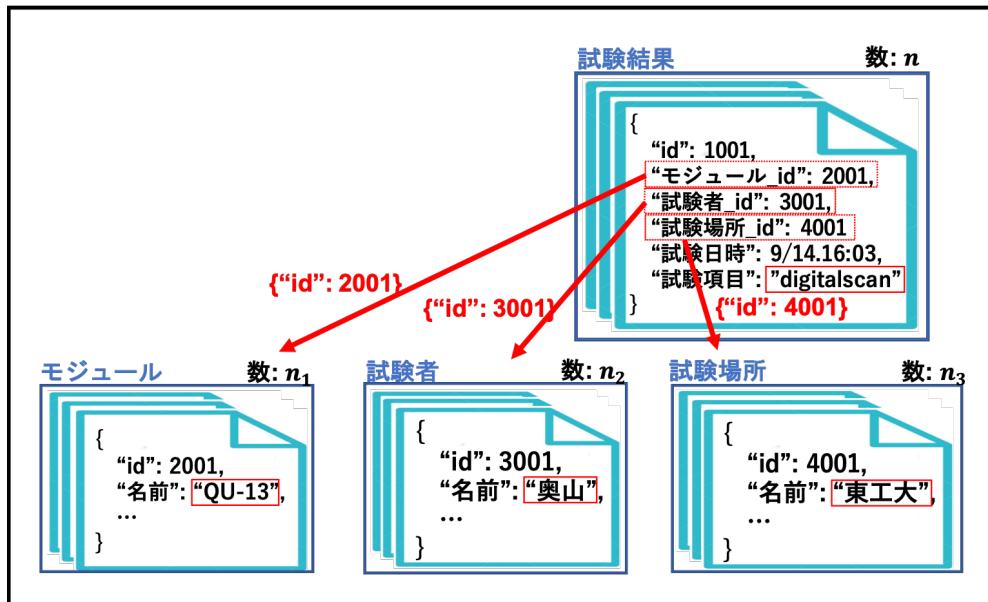


図 6.2 検索機能実装方法 1 の問題点のイメージ図。図 6.1 でも述べたように、検索対象となる情報はこの図のように複数のコレクションにまたがって保存される。そのため、コレクションを超えて検索を行うような場合は試験結果の数以上に、処理に時間がかかるてしまう。この図の例の場合、あるコレクション検索にかかる時間がドキュメント数に対して線形とすると、 $O(n * (n_1 + n_2 + n_3))$ の処理時間がかかると考えられる。

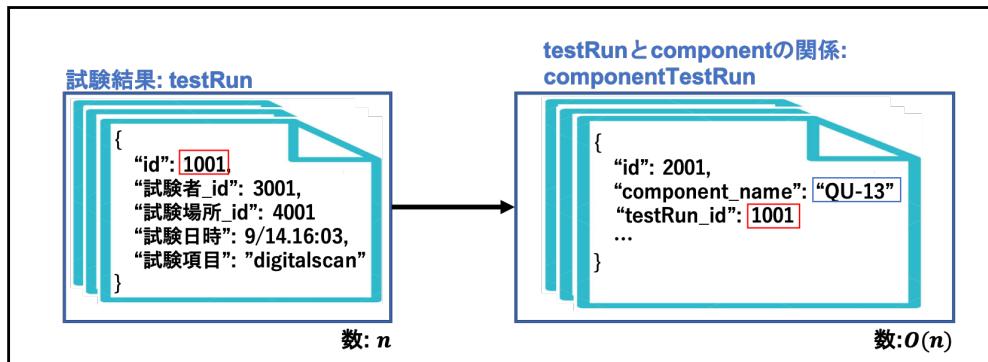


図 6.3 検索処理時間のボトルネックとなっているデータ構造の図。表 4.3 より、各試験結果情報を保存する testRun、component と testRun の関係性を保持する componentTestRun という構造が存在する。試験結果数を n とすると、testRun のドキュメント数は n 、componentTestRun の数は試験数と component の数の積となるため $O(n^2)$ となる。結果的に検索処理に $O(n^2)$ の時間がかかるてしまう。

⁶⁷⁶ 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 $O(n)$ 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。

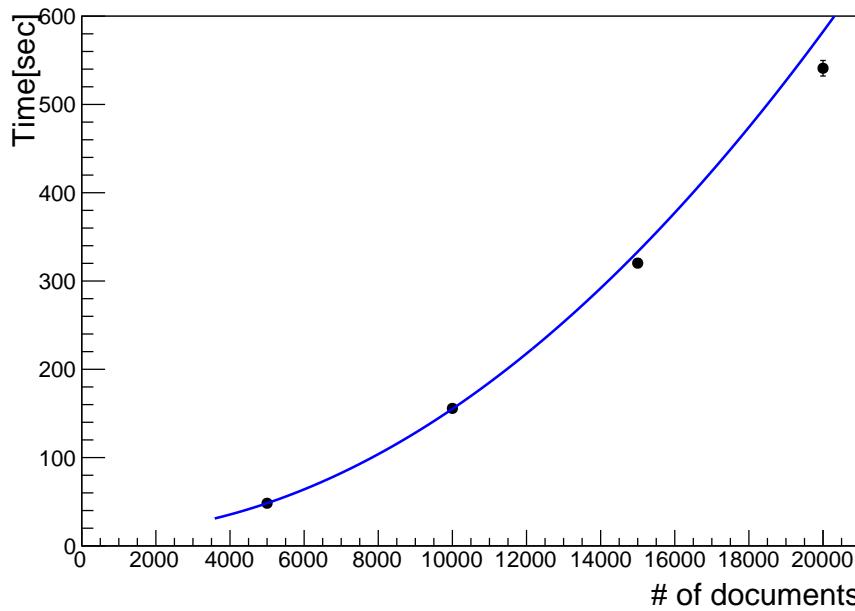


図 6.4 方法 1 における検索処理速度測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。図 6.3 で述べたように、方法 1 の検索処理では試験結果数 n に対して $O(n^2)$ の検索時間がかかるため、試験結果のドキュメント数に対して処理時間は二次関数になっていることがわかる。近似関数や生産時における処理時間の見積もりに関しては後述する。

678 以下に検索情報のドキュメントの例をリスト 6.1 に示す。

Listing 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。

```

679 {
680     " _id" : ObjectId("5fd489f60e2ca70557e44a8b"),
681     " runId" : "5fc4d027b1ef7c6297c91040",
682     " timeStamp" : ISODate("2020-11-30T10:57:19Z"),
683     " data" : [
684         "20UPGR00000001",
685         "20UPGFC9999999",
686         "std_digitalscan",
687         "hokuyama",
688         "tokyo_institute_of_technology",
689         "2020/12/09"
690     ]
691 }
```

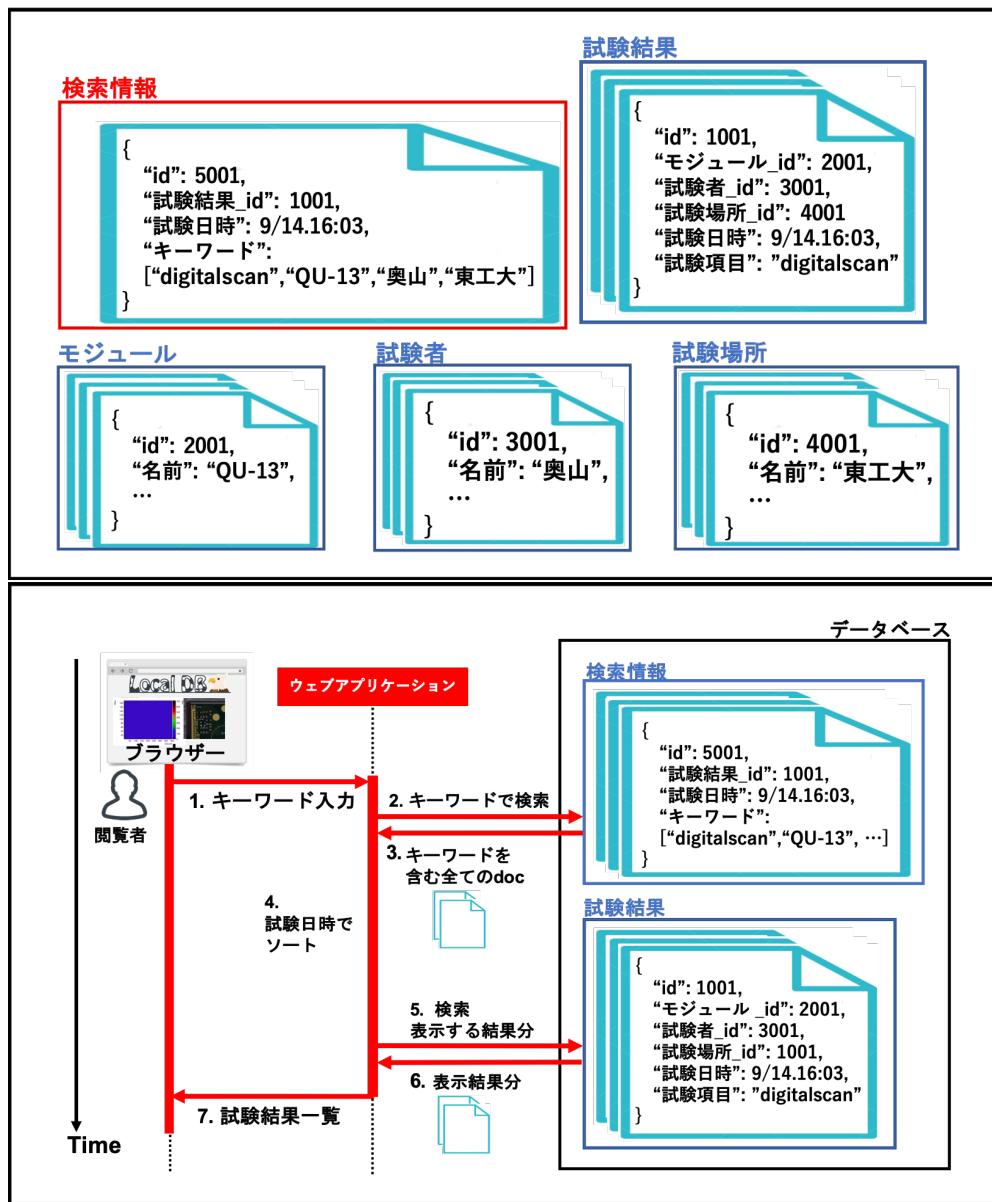


図 6.5 検索機能実装方法 2:検索キーワード専用コレクションを用いた場合のイメージ図。上図は各コレクションと保存されている情報の例を示しており、下図は実際に検索を行った時の処理の流れを表している。方法 2 では上図の赤枠で囲っている検索情報のコレクションを新たに設け、これを参照することで検索処理を行う。このコレクション内には各試験結果に対応したドキュメントが 1 つ保存され、全検索情報と試験結果の ID を持つものとなっている。このコレクションはウェブアプリケーションの立ち上げ時に生成するシステムとした。処理の流れとしては下図のように、検索情報のコレクションよりキーワードに対応する試験結果を抽出する処理と、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”としていて、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。

```
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。

各測定点に対して平均値、標準偏差を算出し、フィッティングを行った。

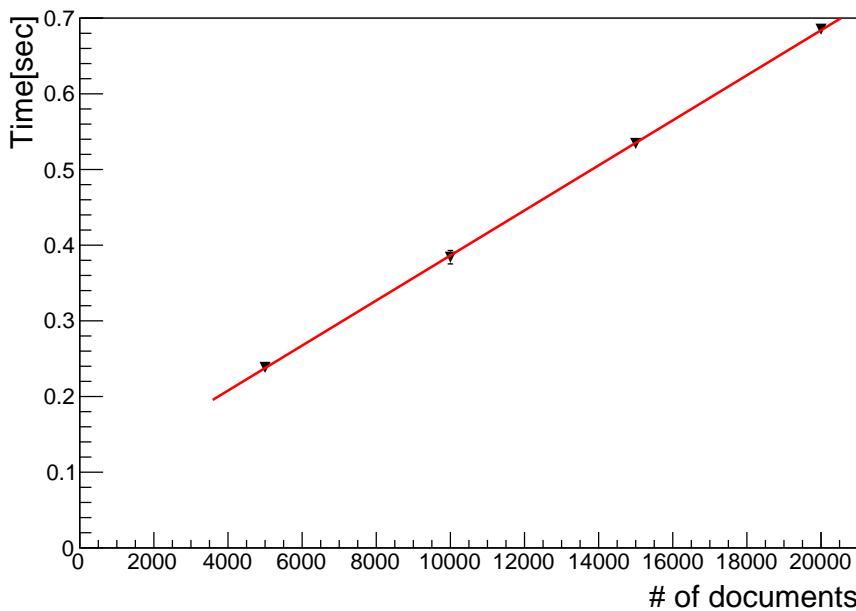


図 6.6 方法 2 における検索処理時間測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。線形性を示していることが確認でき、方法 1 に比べて優れている。

707 測定結果

708 結果を図 6.6 に示す。方法 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
709 2 はアルゴリズムの改善が見られる。

$$y = \{(1.4 \pm 0.0) \times 10^{-6}\}x^2 + (13 \pm 0) \quad (6.1)$$

$$y = \{(3.0 \pm 0.1) \times 10^{-5}\}x + \{(8.9 \pm 0.8) \times 10^{-2}\} \quad (6.2)$$

711 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。

712 生産時における検索時間の見積もり

713 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
714 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、
715 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 章で述べたように、1 つのモジュール
716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。上で得られた関係式を用いて検
717索処理実行時間は方法 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。

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

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

719 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
720 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
721 通して十分に使うことができると考えられる。

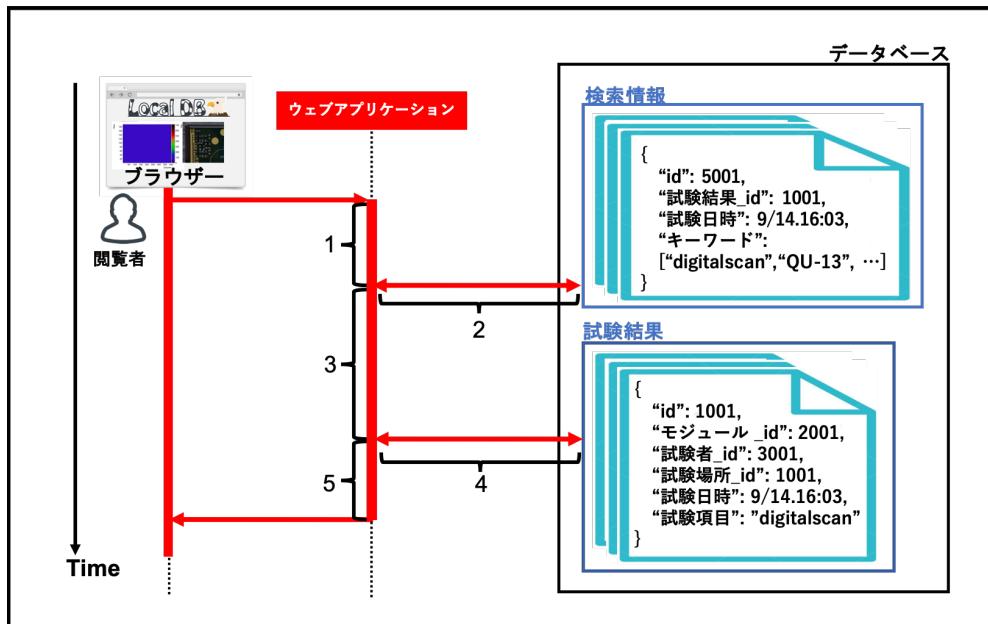


図 6.7 方法 2 の検索における詳細処理。図のように全体の流れにおけるアプリケーション内部での各処理、データベースから情報取得の各処理に 1 から 5 の番号をつけそれぞれにかかる処理時間を測定した。

722 6.3 改善方法の処理時間測定

723 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
724 について以下に示す。

725 6.3.1 方法 2 における検索処理時間の詳細調査

726 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
727 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。

- 728 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。
- 729 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。
- 730 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
731 処理。
- 732 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。
- 733 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。

734 イメージを図 6.7 に示す。

735 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
736 は試験結果数が 10,000 の場合に行なった。また測定は 20 回行った。

737 結果を図 6.8 に示す。

738 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

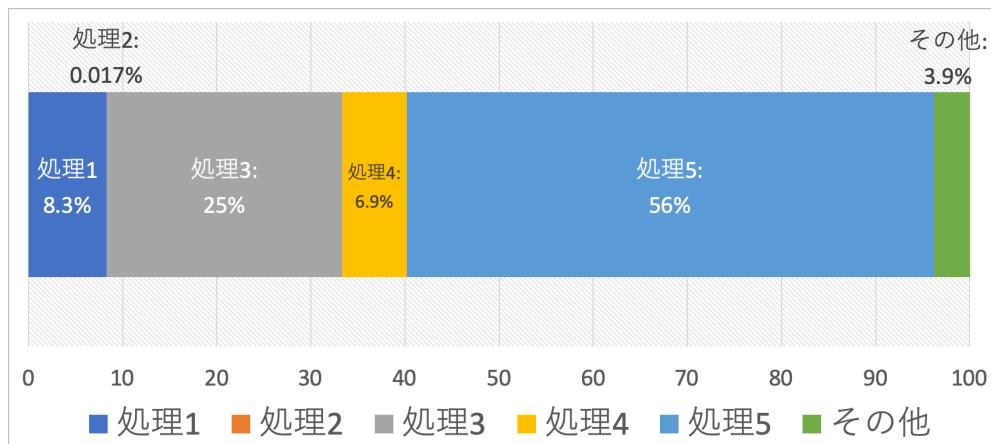


図 6.8 方法 2 における詳細処理時間の測定結果。図 6.7 における 5 つの詳細処理にかかる時間の割合を表している。処理 3、5 にあたるコレクション検索実行後のアプリケーション内での処理に多く時間がかかっていることが分かる。

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

739 いことがわかった。

取得した複数ドキュメントから Python リストへの型変換 (6.5)

740 図 6.7 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
 741 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
 742 時間の関係を測定した。結果を図 6.9 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か
 743 る。方法 2 の検索処理については処理 6.5 が支配的であることが分かった。

744 6.3.2 改善点

745 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
 746 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。

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

749 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。

- 750 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.
- 751 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
 752 た検索処理の並列化.

753 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

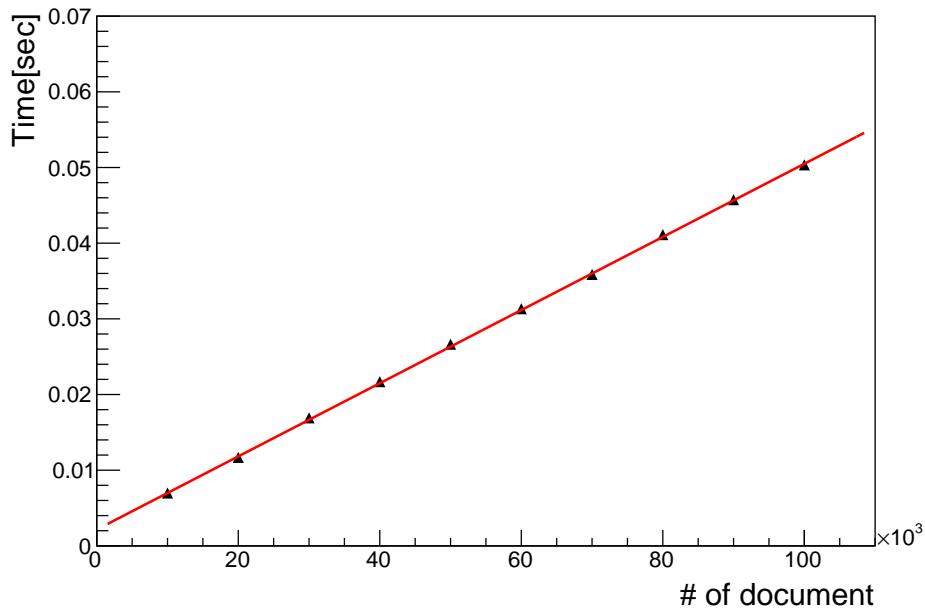


図 6.9 ドキュメント数に対する型変換処理時間の関係。コレクション検索後の型変換に要する処理時間は、図のようにドキュメント数に対して線形となっていることが分かる。

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 6.10、6.11 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 6.12 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。

$$y = \{(2.9 \pm 0.1) \times 10^{-5}\}x + \{(5.0 \pm 11) \times 10^{-3}\} \quad (6.6)$$

$$y = \{(2.7 \pm 0.1) \times 10^{-5}\}x + \{(2.2 \pm 1.0) \times 10^{-2}\} \quad (6.7)$$

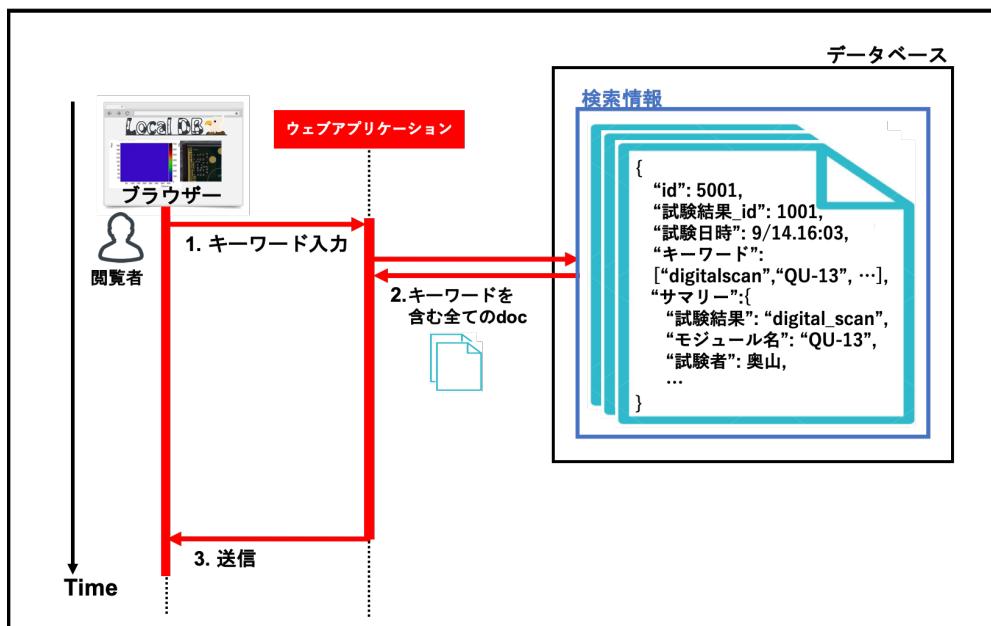


図 6.10 検索機能実装方法 3:検索情報と共に一覧表示に必要な情報を保持、参照。方法 2 では検索情報コレクションより、条件に一致する試験結果 ID を取得し、実際の試験結果に対して再度検索をかける流れとなっていたが、この方法では一覧表示に必要な情報も全て検索情報のコレクション内で保持する。こうすることで検索機能において必要な情報の全てが 1 つのコレクションにまとまり、コレクション検索の処理が 1 回で済むため、処理時間が改善すると考えられる。

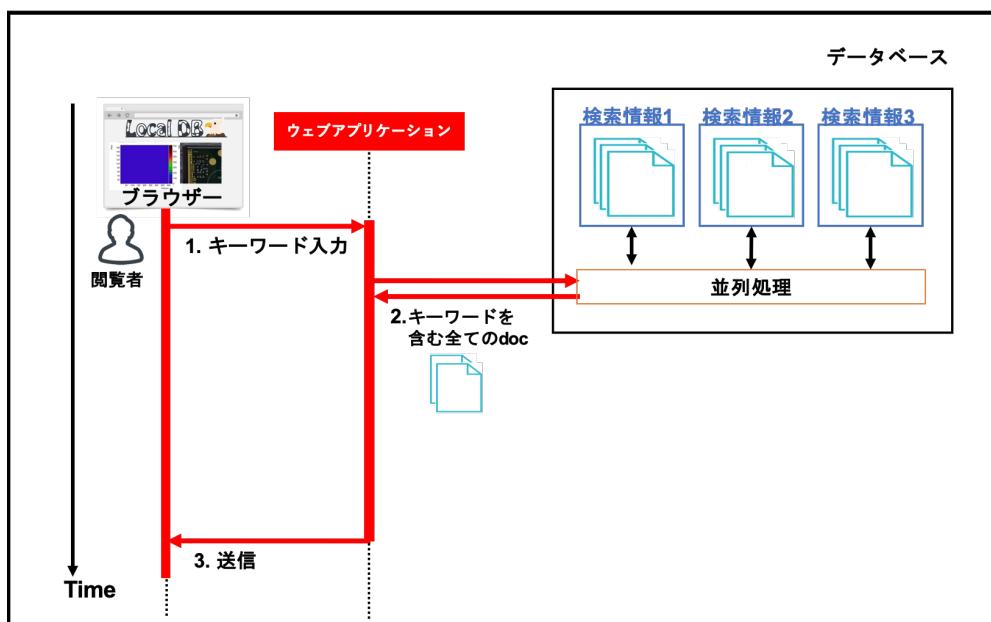


図 6.11 検索機能実装方法 4:検索情報コレクションを分散、マルチスレッドを使用。方法 3 に加えて、検索情報コレクションを分散し、並列処理を行うことで、1 つのコレクションあたりに含まれるドキュメント数を減らし、コレクション検索にかかる時間を削減できると考えられる。

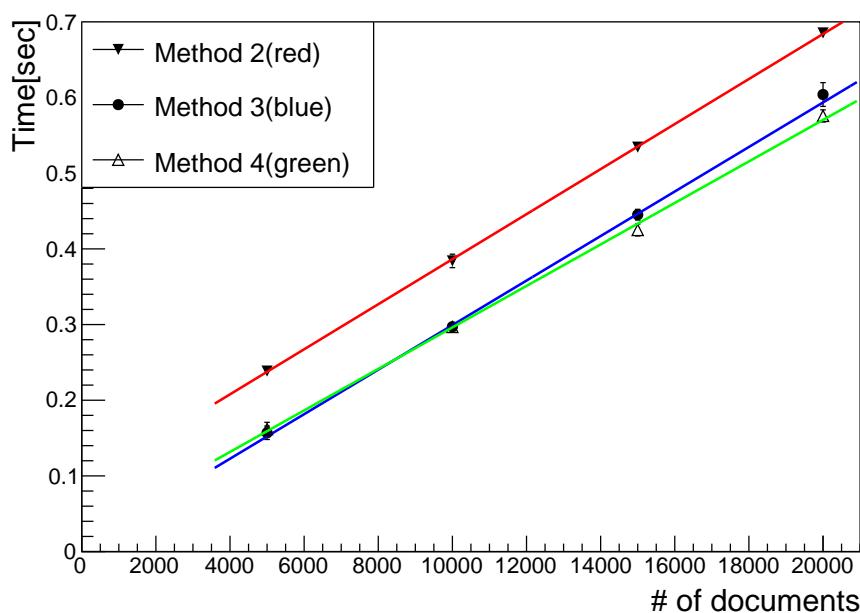


図 6.12 方法 3、4 に対する処理時間測定結果。横軸が試験結果のドキュメント数、縦軸が処理時間を表している。赤、青、緑がそれぞれ方法 2、3、4 を用いたものである。方法 2 に比べて 3、4 共に改善していることが分かる。方法 3、4 を比べると傾きが小さくなっていることが分かる。ドキュメント数が大きい時に対して方法 4 は有効であると言える。

764 第 7 章

765 中央データベースとローカルデータベースの同期

766

767 この章では中央データベースとローカルデータベースのデータ同期機能に関する調査と各ツールにおける改善策の考案、処理時間測定を行った。詳細について以下で説明する。

769 7.1 サーバーの設置場所による処理時間の違い

770 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。そのため試験結果のアップロード
771 に関する、各組み立て機関から接続しデータ送信する処理時間は、機関の場所に大きく依存すると考えら
772 れる。世界的にデータ同期ツールが不自由なく動くことに向か開発、改善に役立てる目的とし
773 て、データベース間の情報通信にかかる処理時間を、以下の 3 つの場所に置かれているサーバーを用いて
774 測定した。組み立て機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
775 域に分布している（付録 A）。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
776 て調査を行った。

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

780 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。
781 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
782 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	Disk
					[GB]	[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 データ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。データ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	読み出し試験結果生成後にファイルを添付する際に用いる。

7.1.1 データ同期ツールに使用する API

中央データベースのデータ取得には、開発された API をいくつか使用している。ローカルデータベースとのデータ同期ツールの中で主に使用している API を表 7.2 に示す。

7.1.2 API 使用にかかる時間

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。

- getComponent を用いた、登録モジュール情報 1 つの取得時間測定。
- 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。
- 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。

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、KEK における処理時間を測定し、ツールの開発、改善について考えることとした。

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 0.3 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]
KEK	0.49 ± 0.02
LBL	0.37 ± 0.02
CERN	0.30 ± 0.04

サーバー	処理時間 [秒]
KEK	0.54 ± 0.04
LBL	0.34 ± 0.03
CERN	0.39 ± 0.02

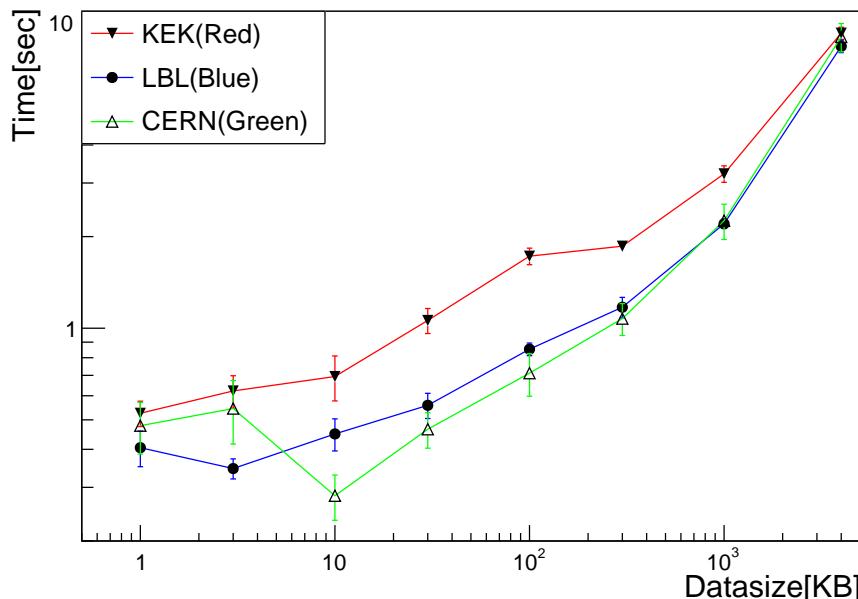


図 7.2 ”createTestRunAttachment”を用いた添付処理におけるファイル容量と処理時間の関係。それぞれのサーバーで 1、3、10、30、100、300KB、1、4MB のファイル送信にかかる時間を測定した。どの点においても KEK サーバーが最も処理時間を要していることが分かる。またこのグラフについての詳しい考察を付録 C に記す。

7.2 モジュール ID のダウンロード機能確認と処理時間測定

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リスト 7.1、7.2 に示す。

Listing 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。

```
801 {
802     "_id" : ObjectId("5fa79114e615fa000a1a5976"),
803 }
```

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

```

803     "name" : "20UPGR00000001",
804     "chipType" : "RD53A",
805     "serialNumber" : "20UPGR00000001",
806     "chipId" : -1,
807     "componentType" : "module",
808     "address" : "5fd597fdf7339bbf26b87fb2",
809     "children" : 1,
810     "sys" : {
811         "mts" : ISODate("2020-12-13T04:26:37.989Z"),
812         "cts" : ISODate("2020-12-13T04:26:37.989Z"),
813         "rev" : 0
814     },
815     "dbVersion" : 1.01,
816     "user_id" : -1,
817     "proDB" : true
818 }
```

Listing 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。

```

819 {
820     "_id" : ObjectId("5fa79560e615fa000a1a5a16"),
821     "name" : "20UPGFC9999999",
822     "chipType" : "RD53A",
823     "serialNumber" : "20UPGFC9999999",
824     "chipId" : 0,
825     "componentType" : "front-end-chip",
826     "address" : "5fd597fdf7339bbf26b87fb2",
```

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

```

827     "children" : -1,
828     "sys" : {
829         "mts" : ISODate("2020-12-13T04:26:37.984Z"),
830         "cts" : ISODate("2020-12-13T04:26:37.984Z"),
831         "rev" : 0
832     },
833     "dbVersion" : 1.01,
834     "user_id" : -1,
835     "proDB" : true
836 }

```

837 7.2.2 処理の流れ

838 ダウンロード機能における処理の流れのイメージを図 7.3 に示す。

839 7.2.3 機能確認

840 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、ダウンロードを行った。登録したモジュールを表 7.4、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 7.5 に示す。

842 7.2.4 処理時間測定

843 ダウンロードした際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。

844 7.2.5 生産時における見積もり

845 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。

846 1. モジュール 1 つのシリアルナンバーを打ち込み、そのモジュールをダウンロードする機能

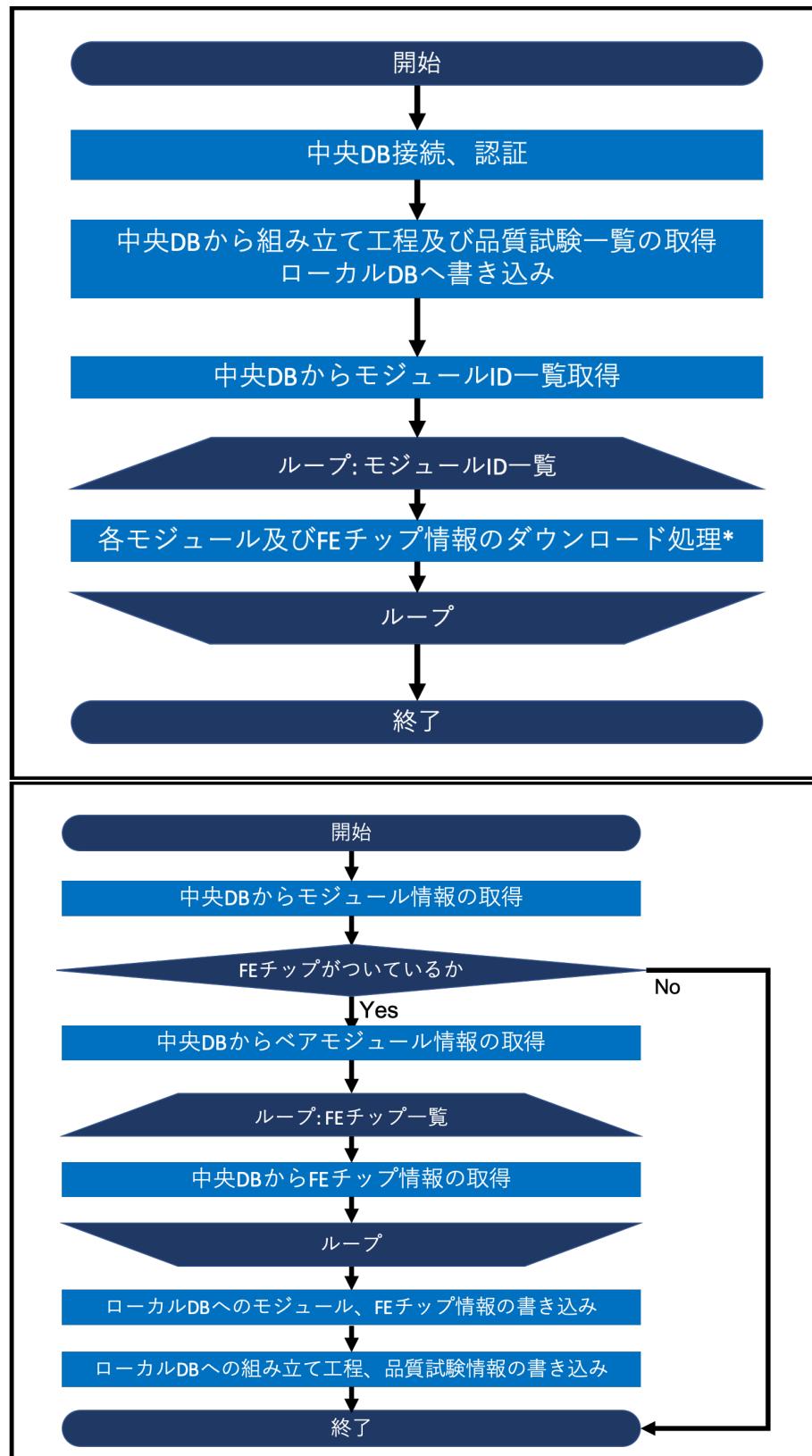


図 7.3 ダウンロード処理における流れのイメージ図。上図が処理全体の流れを表すものであり、下図は各モジュール情報のダウンロードにおける処理の流れを示している。上図中のループ構造の 1 処理が下図に対応している。流れの中で複数回中央データベースに接続し、モジュールや FE チップの情報取得をしていることが分かる。

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 7.4 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、搭載されているペアモジュール、シリコンセンサー、PCB、モジュールキャリア、FE チップの中央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。

847 2. 登録されている全てのモジュールを一括でダウンロードする機能

848 オプション 1 の見積もり値は、表 7.5 の平均値として $4.0 \pm 0.4[\text{sec}]$ となる。オプション 2 の見積もり
 849 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
 850 になる。

$$(4.0 \pm 0.4) \times 10,000 = (4.0 \pm 0.4) \times 10^4[\text{sec}] \quad (7.1)$$

$$= 11 \pm 1[\text{hour}] \quad (7.2)$$

851 このダウンロード機能についてはモジュール登録機能の直後に使用する設計となっている。各機関で各
 852 モジュールの組み立てを始める際に、モジュール登録を行うことを想定している。これは 1 つずつ行われ
 853 るため、現在はオプション 1 のみを提供している。

854 しかし、将来的には世界中で様々なモジュール組み立ての流れが想定される。例えば機関 1 で何台かの
 855 モジュールを登録した後に、機関 2 に輸送するような場合、機関 2 では機関 1 で登録されたモジュールを
 856 ダウンロードしてくる必要がある。このような場合にはオプション 2 を使用すると考えられる。11 時間
 857 の処理時間を要するような今のアルゴリズムでは運用は難しいと考えられるため、改善が必要である。

858 7.2.6 処理時間詳細

859 ダウンロード処理の詳細について以下の測定した。

- 860 1. 中央データベースからモジュール情報の取得.
- 861 2. データベースでの FE チップ確認処理.
- 862 3. 中央データベースからペアモジュール情報の取得.
- 863 4. 中央データベースから FE チップ情報の取得 (4 枚分).

ProdDB web

[IN PROGRESS]	Module - Outer system quad module	20UPGM20000024
[IN PROGRESS]	Module - Outer system quad module	20UPGM20000022
[IN PROGRESS]	Module - Outer system quad module	20UPGM20000006
[IN PROGRESS]	Module - Outer system quad module	20UPGM20000003
[IN PROGRESS]	Module - Outer system quad module	20UPGM20000001
[IN PROGRESS]	Module - Outer system quad module	20UPGM20000004

↓ Download

Local DB TOP / COMPONENTS / TEST Asia/Tokyo Sign In

ITk database for Yarr Component List

Input keywords Partial match Perfect match

RD53A (11 modules)

Latest Result								
Module Name	Chip Name	Current Stage	Test Type	User	Site	Date	Link	Tag
20UPGR90020026	20UPGFC0020950 20UPGFC0020951 20UPGFC0020952	None	None	None	None	None		
20UPGR90000000	20UPGFC0000836 20UPGFC0007994 20UPGFC0007972 20UPGFC0008035	None	None	None	None	None		
20UPGR30000001	20UPGR30000001 20UPGR30000002 20UPGR30000003	None	None	None	None	None		
20UPGR10099999	20UPGFC9999995 20UPGFC9999996 20UPGFC9999997 20UPGFC9999998	None	None	None	None	None		
20UPGR00000001	20UPGFC9999999	None	None	None	None	None		
20UPGM20030024	20UPGFC001626 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 20UPGFC0014679 20UPGFC0014693 20UPGFC0014646	None	None	None	None	None		
20UPGM20030001	20UPGFC0014644 20UPGFC0014628 20UPGFC0014640 20UPGFC0014708	None	None	None	None	None		

図 7.5 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。

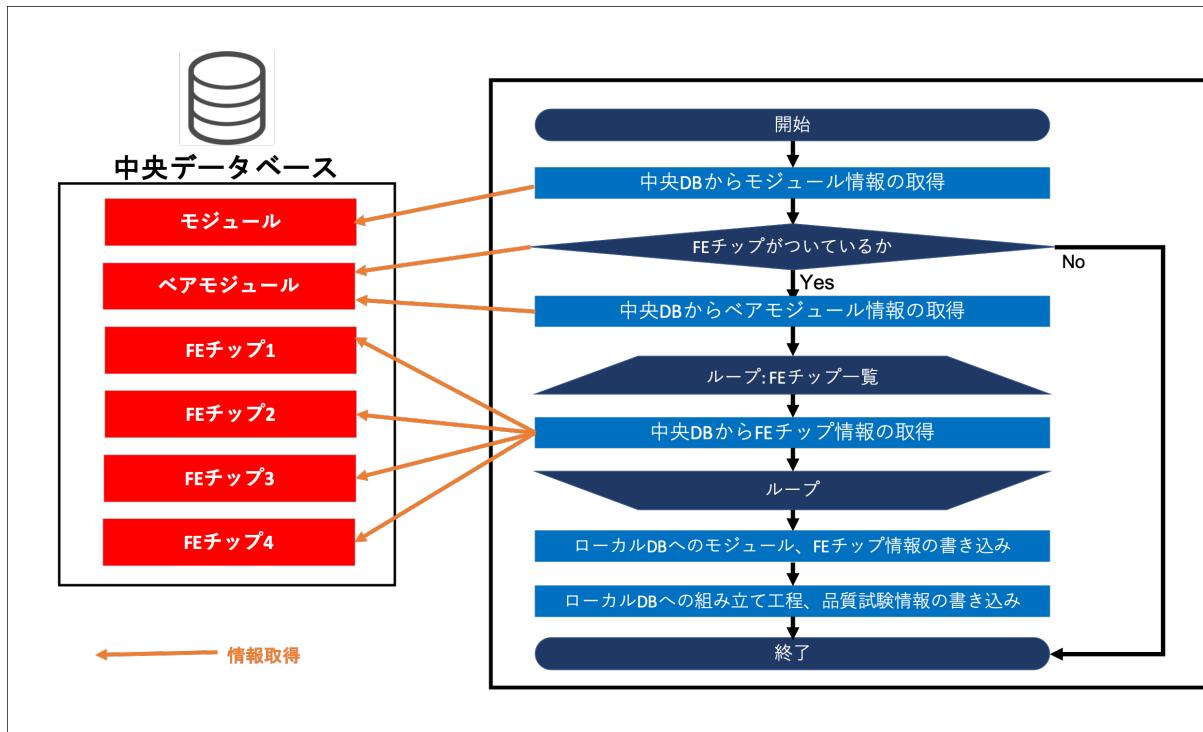


図 7.6 モジュール及び構成部品情報取得のイメージ。図のように処理の流れの中で合計 7 回中央データベースに接続し、モジュール、ペアモジュール、FE チップの情報取得を行っている。

864 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).

865 情報取得のイメージを表 7.6 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
866 合計して 7 回、データベース API を用いて情報取得を行う。

867 結果を表 7.6 に示す。

868 この結果より、各構成部品情報の取得(モジュール、ペアモジュール、FE チップ)の取得にそれぞれ均
869 等に処理時間がかかっていることがわかった。そのため、処理時間を改善するために、この情報取得の回
870 数を減らすアルゴリズムを考える必要がある。

871 7.2.7 改善点の考案と見積もり

872 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。

- 873 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.
- 874 2. FE チップの登録機関を取得しない.
- 875 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.
- 876 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.

877 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。

878 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

879 上述した機関が途中で変更となるような組み立ての流れにおいて、全てのモジュール ID をダウンロー
880 ドする必要はない。中央データベースにはモジュールの現在位置情報を保持しているため、機能実行者と

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 7.6 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).

881 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を改善できる。見積もりとして
882 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。

$$(11 \pm 1) \times \frac{n}{10000} [\text{hour}] \quad (7.3)$$

883 改善案 2: FE チップの登録機関を取得しない

884 表 7.6 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
885 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
886 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。
887 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
888 処理時間の見積もりは、以下になる。

$$\{(4.0 \pm 0.4) - (2.3 \pm 0.2)\} \times 10,000 = (1.8 \pm 0.3) \times 10^4 [\text{sec}] \quad (7.4)$$

$$= 4.9 \pm 0.8 [\text{hour}] \quad (7.5)$$

889 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
890 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
891 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可能性もある。

892 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

893 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
894 7.2 における listComponents によるモジュール一覧取得でその情報を参照することができる。こうする

895 ここで、表 7.6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。この場合処理時
896 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、

$$\{(4.0 \pm 0.4) - (2.9 \pm 0.2)\} \times 10,000 = (1.1 \pm 0.3) \times 10^4[\text{sec}] \quad (7.6)$$

$$= 3.1 \pm 0.8[\text{hour}] \quad (7.7)$$

897 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
898 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
899 が加えられた場合などを管理するのが難しくなる。

900 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

901 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
902 に対して構成要素は取得できない。そのため表 7.6 のようにモジュールごとに中央データベースに接続
903 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うな
904 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ら
905 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けの
906 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。

907

908 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
909 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、必要に応じて変更を加えていく必要が
910 ある。

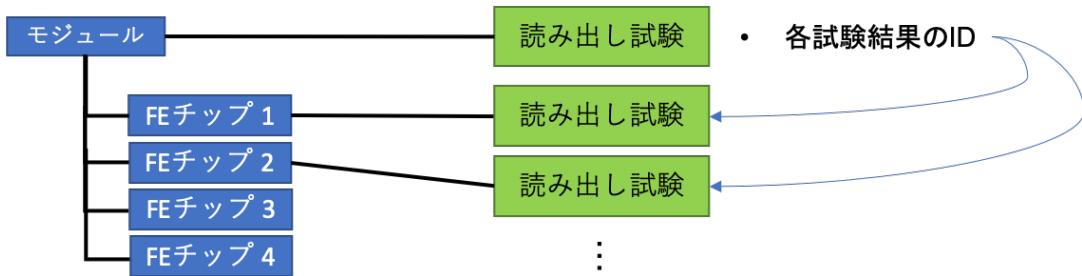


図 7.7 中央データベースにおける読み出し試験結果の構造。YARR の出力ファイル及びローカルデータベースのデータ構造において、読み出し試験結果は全て FE チップに紐つけられている。そのため、図のように中央データベースにおいてもこのデータ構造を保持する形でアップロードを行う。モジュールの結果は各 FE チップの試験結果に対する ID を持つことで紐付けを行っている。

911 7.3 読み出し試験結果のアップロード機能確認と処理時間測定

912 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
913 た。以下で詳細を述べる。

914 7.3.1 アップロードする情報とその構造

915 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。

- 916 ● 試験日時.
- 917 ● モジュール周りの環境温度.
- 918 ● ピクセル解析結果.
- 919 ● 各試験結果データファイル.
- 920 ● 読み出し設定ファイル.
- 921 ● その他設定ファイル (DB、ユーザ、組み立て機関等).

922 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
923 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
924 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。イメージ
925 を図 7.7 に示す。

926 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。

927 7.3.2 処理の流れ

928 アップロード機能における処理の流れのイメージを図 7.8 に示す。流れの中には共通して行われる処理
929 と、各 FE チップに対して行われる処理がある。

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

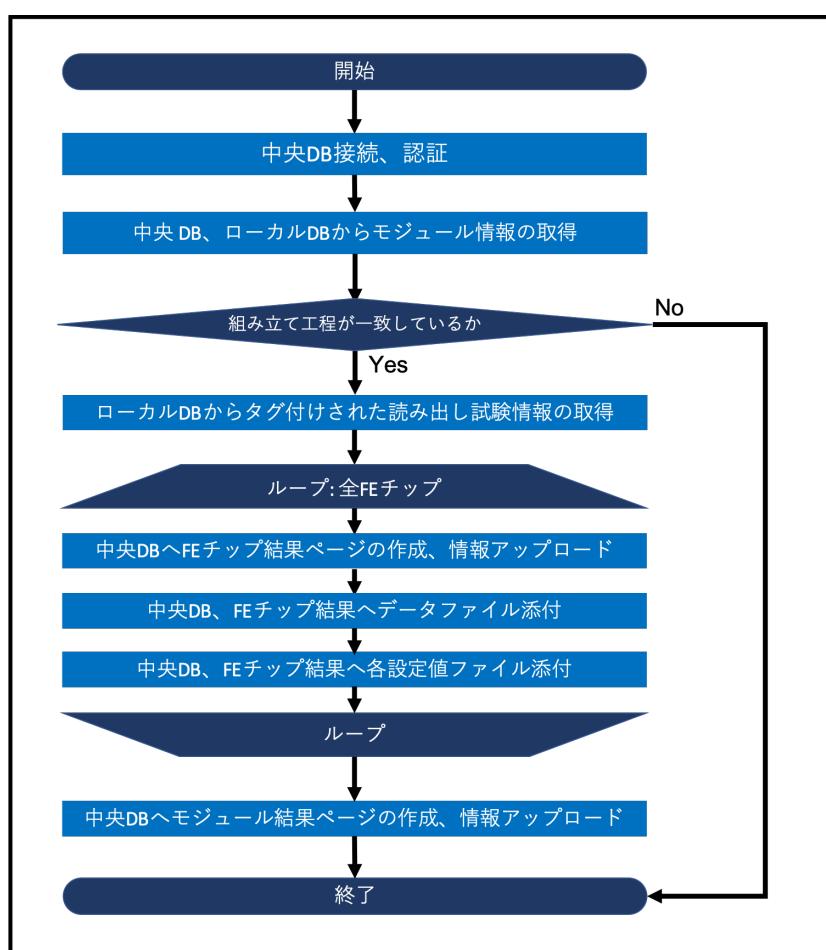
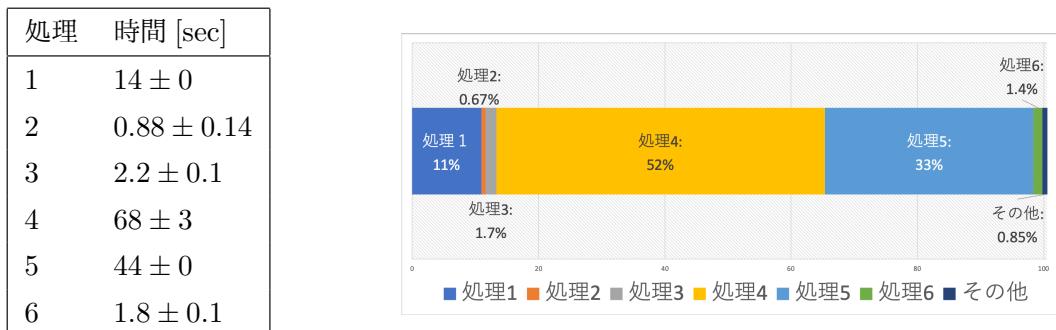


図 7.8 アップロード処理に関する流れのイメージ図。流れの中には FE チップに関するループ構造があり、ここで FE チップの結果生成、結果ファイルのアップロードを行う。最後にモジュールに対する結果生成とアップロードを行う。

930 7.3.3 機能確認と問題点

931 上述した処理のツールを開発し、機能確認と処理時間測定を行った。ここで行ったアップロード及び読
 932 み出し試験の項目は 5 章で行ったデモンストレーションのものと同じとする。その際以下のような問題が
 933 あった。



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。

図 7.9 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、各詳細処理時間を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の時間がかかっていることが分かる。

- 934 ● 処理時間が長くかかってしまった。
 935 ● ファイルの容量制限により、データ容量が 4MB を超えるファイルの添付に失敗した。

936 初めに、問題 1 の改善に向けて、アップロードにかかる時間を測定した。KEK のサーバーを用いて
 937 アップロード処理を 20 回行い、全体でかかる時間を測定した。平均値と標準偏差を測定値と誤差とした。
 938 以下のようにになった。

$$(1.3 \pm 0.0) \times 10^2 [\text{sec}] \quad (7.8)$$

939 ここで処理流れの表 7.8 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。

946 結果を表 7.9 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わ
 947 かった。

948 生産時における見積もり

949 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
 950 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 7.9 を

951 用いて以下のように計算できる。

$$\text{FE チップ処理} : ((2.2 + 68 + 44) \pm \sqrt{(0.1)^2 + 3^2 + 0^2} \times 4 \quad (7.9) \\ = (4.6 \pm 0.1) \times 10^2 [\text{sec}]$$

$$\text{合計} : (4.6 \pm 0.1) \times 10^2 + (14 \pm 0) + (0.88 \pm 0.14) + (1.8 \pm 0.1) = (4.7 \pm 0.1) \times 10^2 [\text{sec}] \quad (7.10) \\ = 7.9 \pm 0.1 [\text{min}]$$

952 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
953 て、データ管理を行うために、データ同期は速やかに行われる必要がある。さらに同期が必要な品質試験
954 結果は、各組み立て工程に読み出し試験以外にも多く存在する。よって現状のアルゴリズムを見直し、処
955 理速度を改善に努める必要があると考えた。

956 7.3.4 改善策

957 ファイル添付に多く時間がかかってしまっている現状を踏まえ、各ファイルに対して添付処理時間の測
958 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。添付する結果デー
959 ティファイル、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、
960 問題点 2 として 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った。

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??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。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。(tex の技術的に caption が複数になってしまう。)

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	

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??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。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。(tex の技術的に caption が複数になってしまふ。)

	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	

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??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。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。(tex の技術的に caption が複数になってしまふ。)

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
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

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??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。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。(tex の技術的に caption が複数になってしまふ。)

	afterCfg_chipCfg.json	Error	7,200	3.5 ± 0.1	
std_totscan	MeanTotMap-0.json	Ok	1,900	4.8 ± 0.1	20±0
	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	18±0
	NoiseOccupancy.json	Ok	1,300	2.5 ± 0.1	
	NoiseMask.json	Ok	1,300	2.4 ± 0.1	
	ctrlCfg_ctrlCfg.json	Ok	0.46	0.57 ± 0.08	
	dbCfg_dbCfg.json	Ok	0.60	0.55 ± 0.06	
	siteCfg_siteCfg.json	Ok	0.033	0.53 ± 0.04	
	userCfg_userCfg.json	Ok	0.14	0.61 ± 0.07	
	scanCfg_std_noisescan.json	Ok	1.4	0.53 ± 0.05	
beforeCfg_chipCfg.json	beforeCfg_chipCfg.json	Error	7,200	2.8 ± 0.2	
	afterCfg_chipCfg.json	Error	7,200	3.5 ± 0.1	

表 7.8 より、データサイズ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。また添付処理を行うオフセットがあることから、threshold scan のように各容量が大きくなくてもファイル数が多いものにはアップロード時間が合計して多くかかることがある。

これらのことと添付処理の失敗をなくすことを考慮に入れ、次のような改善策を考えた。

- 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、圧縮後にアップロードを行う。

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。

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。合計処理時間は以下のようになり、全てのファイルのアップロードに成功した。

$$36 \pm 1[\text{sec}] \quad (7.11)$$

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]
std_digitalscan	std_digitalscan_datafiles.zip	Ok	10	0.77 ± 0.18	1.9 ± 0.2
	std_digitalscan_configfiles.zip	Ok	56	1.1 ± 0.1	
std_analogscan	std_analogscan_datafiles.zip	Ok	46	1.0 ± 0.2	2.2 ± 0.3
	std_analogscan_configfiles.zip	Ok	58	1.2 ± 0.2	
std_thresholdscan	std_thresholdscan_datafiles.zip	Ok	1,500	2.6 ± 0.1	3.5 ± 0.1
	std_thresholdscan_configfiles.zip	Ok	190	0.86 ± 0.08	
std_totscan	std_totscan_datafiles.zip	Ok	730	1.7 ± 0.2	2.5 ± 0.2
	std_totscan_configfiles.zip	Ok	190	0.83 ± 0.15	
std_noisescan	std_noisescan_datafiles.zip	Ok	19	0.56 ± 0.07	1.7 ± 0.1
	std_noisescan_configfiles.zip	Ok	190	1.1 ± 0.1	

生産時における見積もり

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 7.9において、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、

$$\begin{aligned} \text{FE チップ処理} : & ((2.2 + 6.7 + 5.1) \pm \sqrt{(0.1)^2 + (1.1)^2 + (0.3)^2}) \times 4 \\ & = 56 \pm 5[\text{sec}] \end{aligned} \quad (7.12)$$

$$\begin{aligned} \text{合計} : & (56 \pm 5) + (14 \pm 0) + (0.88 \pm 0.14) + (1.8 \pm 0.1) = 73 \pm 5[\text{sec}] \\ & = 1.2 \pm 0.1[\text{min}] \end{aligned} \quad (7.13)$$

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

975 在は改善後の方針を用いたツールを提供している。

976 第8章

977 まとめ

978 8.1 本論文のまとめ

979 CERN にある LHC 加速器を用いて ATLAS 実験が行われている。2025 年より LHC のアップグレードを行う予定であり、これを HL-LHC と呼ぶ。HL-LHC に向けて ATLAS 内部飛跡検出器の総入れ替え
 980 を予定しており、ピクセル検出器は現行のものよりも広い範囲をカバーする。新しく作る検出器は ITk
 981 と呼ばれ、これに向けてピクセルモジュールを世界で 10,000 台生産する予定であり、各モジュールに対
 982 して品質試験を行う。全てのモジュール及び品質試験の結果は中央データベースに保存する。
 983

984 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。各生産現場にて
 985 データ保存、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
 986 機能など、生産時に必要となる諸ツールの開発を行った。またデータベース機能の普及と生産の成功に向
 987 けて、共同生産者及びシステムユーザを対象としたシステムのチュートリアルを行った。多くの議論と
 988 フィードバックを受け、システムの更なる改善に繋げた。チュートリアルを受けて 2020 年 11 月現在、世
 989 界 18 の生産現場でローカルデータベースの導入及び試運転が行われていることを確認し、システム普及
 990 を達成した。

991 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学内実験室にて行った。本番を想定したソ
 992 フトウェア、ハードウェアのセットアップと各ツールの処理実行を達成し、機能が使用可能であることを
 993 確認した。

994 主な開発項目の 1 つ目として品質試験検索機能を開発し、MongoDB 内に新しいコレクションを設ける
 995 工夫により、開発当初に問題となった処理時間の改善に成功した。実際に処理時間の測定を行い、デー
 996 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た。本番を想定した見積もりを行い、
 997 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込みであり、生産時において十分に使
 998 える機能であることを確認した。

999 2 つ目に中央データベースとの同期ツールを開発した。世界的に使われるツールであり、全ての生産現
 1000 場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を KEK、LBL、CERN
 1001 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などを確認し、このサー
 1002 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と考えた。開発した中央
 1003 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た。モジュール情報のダウ
 1004 ナロード機能に関して、モジュール 1 つあたり $4.0 \pm 0.4[\text{sec}]$ の処理時間がかかるなどを確認した。生産
 1005 に向けて処理時間の改善策をいくつか考案し、それぞれについて見積もりを行った。読み出し試験結果の
 1006 アップロード機能に関して、処理時間のボトルネックとなっている箇所を分析し、結果ファイルを ZIP

1007 ファイルにまとめ、圧縮しアップロードを行うという改善を加え、処理時間の改善に成功した。生産時に
1008 おいてモジュール1つに対しての結果アップロード処理時間の見積もりが $1.2 \pm 0.1[\text{sec}]$ であり、生産を
1009 通して使える機能であることを確認した。

1010 8.2 現状と今後の課題

1011 8.2.1 ソフトウェアリリースとユーザサポート

1012 ローカルデータベース開発は複数人のチームで行っている。また、本論文で述べたツールの他に、読み
1013 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などの開発もチームとして行ってい
1014 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ようなアプリケーションスイートを
1015 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を行い、いくつかの機関で全体のシ
1016 ステム及びソフトウェアが使われている現状である。

1017 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
1018 トとしてソフトウェア使用のためのドキュメント[7-1]の作成、整備も行っている。開発者の連絡先や
1019 ロー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
1020 単に問い合わせができる仕組みを整えている。

1021 8.2.2 開発課題

1022 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
1023 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
1024 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
1025 げる。

- 1026 ● 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.
- 1027 ● 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.
- 1028 ● 品質試験結果解析とモジュール選別機能.
- 1029 ● 組み立て工程管理を世界的にサポート.
 - 1030 – モジュールの組み立て工程は各生産現場ごとに異なる。そのため全ての現場における工程を調
1031 査し、それをサポートするシステムとなるように実装する必要がある。

1032 付録 A

1033 ローカルデータベースのチュートリアル
1034 と普及状況

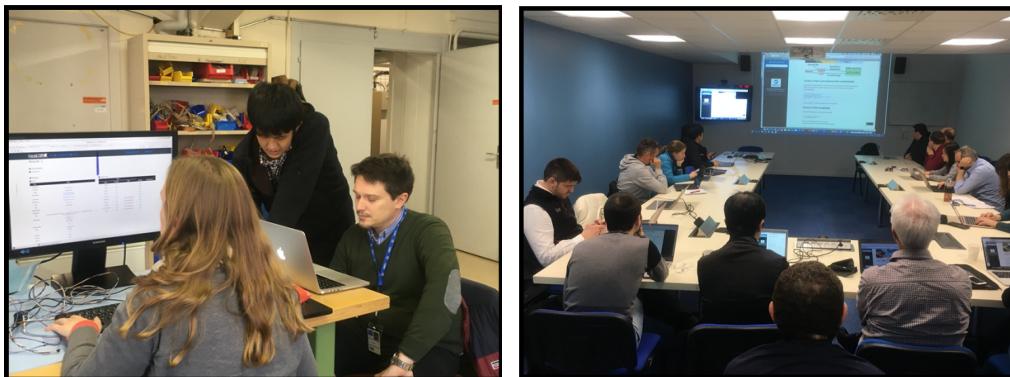


図 A.1 ローカルデータベースシステムチュートリアルの様子。2020 年 2 月に CERN でローカルデータベースシステムのチュートリアルを行った。参加者が実際にシステムの設置、機能実行を行うハンズオンセッション（左図）と、参加者の前で実際に機能の動作をみせ、議論を行うハンズオフセッション（右図）に分けて行った。システムに有益な情報を獲得したと共に、システムの機能普及に成功した。

1035 A.1 チュートリアルと普及状況

1036 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
1037 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。

- 1038 ● 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
1039 ション（2 月 3 日から 6 日まで）
- 1040 ● 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
1041 日）

1042 それぞれのセッションの様子を図 A.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
1043 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。

1044 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
1045 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
1046 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
1047 世界地図を A.2 に示す。

- 1048 ● 高エネルギー加速器研究機構 (KEK), 日本
- 1049 ● 欧州原子核研究機構 (CERN), スイス
- 1050 ● University of Liverpool, イギリス
- 1051 ● University of Oxford, イギリス
- 1052 ● University of Glasgow, イギリス
- 1053 ● Paris-Saclay University, フランス
- 1054 ● パリ第 6 大学, フランス
- 1055 ● フランス国立科学研究中心, フランス
- 1056 ● University of Grenoble, フランス
- 1057 ● University of Gottingen, ドイツ

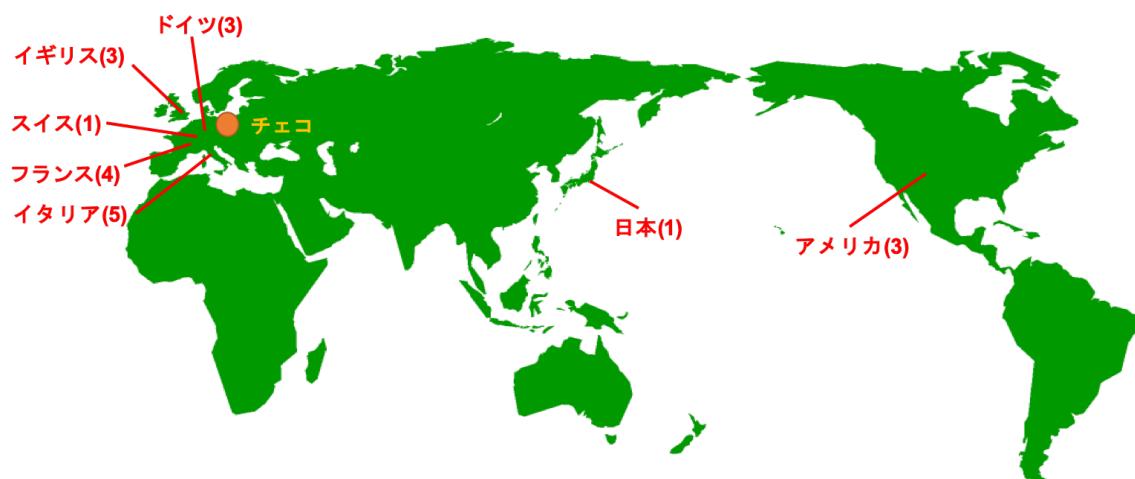


図 A.2 ローカルデータベースシステム導入及び試運転場所。赤文字は設置している地域、括弧内の数字はその地域におけるシステム導入場所の数を示している。2020 年 11 月現在、ローカルデータベースシステムは世界 18 の機関で試験運転がなされている。日本を除いてその多くはヨーロッパとアメリカに位置していることが分かる。

- 1058 ● University of Siegen, ドイツ
- 1059 ● University of Genoa, イタリア
- 1060 ● University of Salento, イタリア
- 1061 ● University of Milan, イタリア
- 1062 ● University of Udine, イタリア
- 1063 ● University of Trento, イタリア
- 1064 ● University of Oklahoma, アメリカ
- 1065 ● Argonne National Laboratory, アメリカ
- 1066 ● Lawrence Berkeley National Laboratory(LBL), アメリカ

1067 付録 B

1068 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

1069

1070 (時間があればもっとちゃんと書きます。) KEK と LBL においてなぜ差が出るのかを考察する。ファ
1071 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 B.1 は KEK から LBL の
1072 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したものであ
1073 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とが分かる。これは TCP 通信において
1074 パケットの送信に輻輳制御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を行
1075 っている。

1076 scp によるファイル送信を KEK->LBL、LBL->KEK の場合に対しておこなった。図 B.2 のように差異
1077 が見られた。Server の spec は同程度。読み書き速度も変わらなかった。輻輳制御アルゴリズム (Cubic)、
1078 Window size と ping(111msec) は変わらなかった。一般的には上りより下りの方が太いと考えると、
1079 KEK の上り network は LBL 上りと比べての方が細いと考えられる。

1080 scp ファイル送信、KEK-Lxplus、LBL->Lxplus 上述したように KEK の上りネットワークは細い。加
1081 えて ping による反応時間が KEK は 170msec 程度なのに対し、LBL は 150msec 程度。ネットワーク上

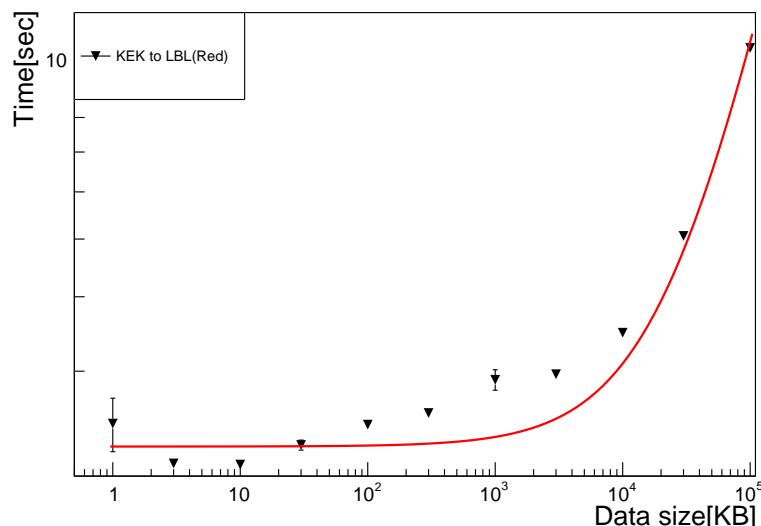


図 B.1 添付するファイルサイズと処理時間の関係

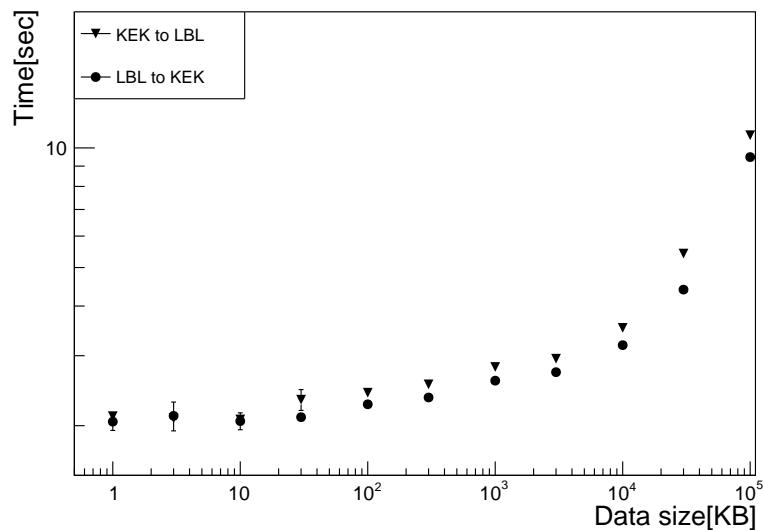


図 B.2 KEK、LBL 間のファイル送信

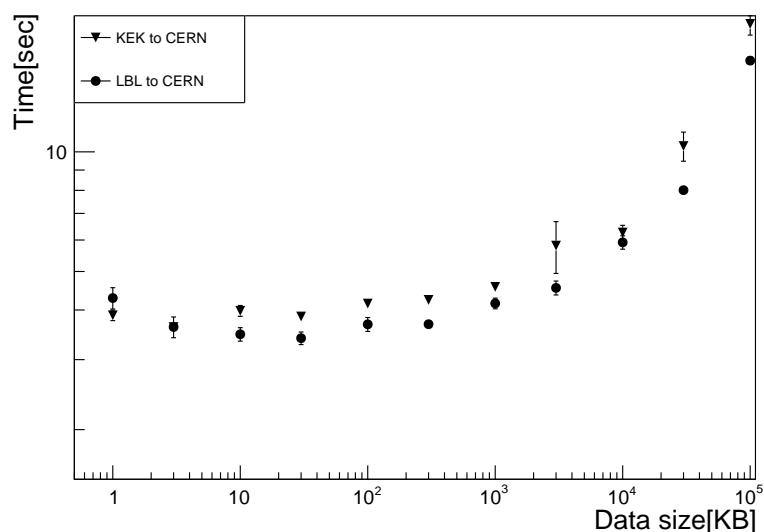


図 B.3 KEK、LBL と CERN 間のファイル送信

1082 の距離差も処理速度影響していると考えられる。

1083 CERN と中央 DB では条件が違うが、上述したことをまとめると KEK と LBL の間で処理時間の際が
1084 生まれる要因は以下であると考えた。

- 1085 • KEK の上りネットワークが遅い
- 1086 • ネットワーク上の距離差があり、KEK の方が差が大きい。

参考文献

- 1087 [1-1] Damerau,H et al. "LHC Injectors Upgrade Technical Design Report". CERN Document server.
 1088 2016-05 <https://cds.cern.ch/record/2153863>, (2020-12)
- 1090 [1-2] Georges Aad et al. "The ATLAS Experiment at the CERN Large Hadron Collider".
 1091 Semantic Scholar. 2008-8 <https://www.semanticscholar.org/paper/The-ATLAS-Experiment-at-the-CERN-Large-Hadron-Aad-Groot/7d771b20731969fe10c267465582ee60e9383db3>, (2020-12)
- 1092 [1-3] ATLAS Collaboration. "Technical Design Report for the ATLAS Inner Tracker Pixel Detector".
 1093 CERN Document Server. 2018-8 <https://cds.cern.ch/record/2285585>, (2020-12)
- 1094 [1-4] ATLAS Collaboration. "Study of the material of the ATLAS inner detector for Run 2 of the
 1095 LHC". CERN Document Server. 2017-7 <https://cds.cern.ch/record/2273894>, (2020-12)
- 1096 [1-5] ATLAS Collaboration. "The upgraded Pixel Detector of the ATLAS Experiment for Run 2 at the Large Hadron Collider". ScienceDirect. 2016-9
 1097 <https://doi.org/10.1016/j.nima.2016.05.018>, (2020-12)
- 1098 [1-6] Apollinari, G;Bjar Alonso, I; Brning, O; Lamont, M; Rossi, L. "High-Luminosity Large
 1099 Hadron Collider (HL-LHC) : Preliminary Design Report". CERN Document Server. 2015-12
 1100 <https://cds.cern.ch/record/2116337>, (2020-12)
- 1101 [1-7] The HL-LHC project. "The HL-LHC project". CERN Accelerating science. 2020-8
 1102 <https://hilumilhc.web.cern.ch/content/hl-lhc-project>, (2020-12)
- 1103 [2-1] "Semiconductor Devices Physics and Technology" . S.M. Sze 著, 南日康夫・川辺光央・長谷川文
 1104 夫訳, 産業図書, 2015 年 3 月第 2 版第 11 刷発行.
- 1105 [2-2] Garcia-Sciveres, Maurice. "The RD53A Integrated Circuit". CERN Document Server. 2017-10
 1106 <https://cds.cern.ch/record/2287593>, (2020-12)
- 1107 [2-3] "Pixel Detectors" . Rossi, L., Fischer, P., Rohe, T., Wermes, N. Springer, 2006-7-8
- 1108 [3-1] Meng, Lingxin. "RD53A Module Testing Document". CERN Document server. 2020-9
 1109 <https://cds.cern.ch/record/2702738>, (2020-12)
- 1110 [4-1] "MongoDB: The most popular database for modern apps". MongoDB, Inc.
 1111 <https://www.mongodb.com/2>, (2020-12)
- 1112 [4-2] "Databases and Collections". MongoDB Manual. <https://docs.mongodb.com/manual/core/databases-and-collections/>, (2020-12)
- 1113 [4-3] "Welcome to Flask". Flask Documentation. <https://flask.palletsprojects.com/en/1.1.x/>, (2020-12)
- 1114 [4-4] "PyMongo 3.11.2 Documentation". PyMongo 3.11.2 Documentation.
 1115 <https://pymongo.readthedocs.io/en/stable/>, (2020-12)

- 1120 [5-1] "E3640A E3649A Programmable DC Power Supplies - Data Sheet". Keysight Technologies.
1121 2018-3 <https://www.keysight.com/jp/ja/assets/7018-06827/data-sheets/5968-7355.pdf>, (2020-
1122 12)
- 1123 [5-2] "XpressK7-160-Gen2". Mouser Electronics. <https://www.mouser.jp/ProductDetail/ReFLEX-CES/XpressK7-160-Gen2?qs=rrS6PyfT74eSJLUPLu1P5g%3D%3D>, (2020-12-22)
- 1124 [5-3] Microchip Technology. "2.7V Dual Channel 10-Bit A/D Con-
1125 verter with SPITM Serial Interface". 秋月電子通商. 2006-08
<https://akizukidenshi.com/download/ds/microchip/mcp3002.pdf>, (2020-12)
- 1126 [5-4] "Raspberry Pi 3 Model B+". RASPBERRY PI FOUNDATION.
<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>, (2020-12)
- 1127 [5-5] "ROOT: analyzing petabytes of data, scientifically.". ROOT Team. <https://root.cern>, (2020-12)
- 1128 [5-6] "InfluxDB: Purpose-Built Open Source Time Series Database". influxdata.
<https://www.influxdata.com>, (2020-12)
- 1129 [5-7] "Grafana: The open observability platform". Grafana Labs. <https://grafana.com>, (2020-12)
- 1130 [5-8] "PySerialComm". GitLab. <https://gitlab.cern.ch/solans/PySerialComm>, (2020-12)
- 1131 [5-9] Renesas Electronics Corporation. "GPIO". RENESAS. <https://www.renesas.com/jp/ja/support/engineer-school/mcu-programming-peripherals-01-gpio>, (2020-12)
- 1132 [5-10] "RD53A Single Chip Card Configuration". CERN twiki. 2018-5-14
https://twiki.cern.ch/twiki/pub/RD53/RD53ATesting/RD53A_SCC_Configuration.pdf, (2020-
1133 12)
- 1134 [5-11] Wielers, Monika. "Pixel production database serial numbering scheme". CERN Document
1135 Server. 2020-12-07 <https://cds.cern.ch/record/2728364>, (2020-12)
- 1136 [7-1] "LocalDB docs". LocalDB docs. <https://localdb-docs.readthedocs.io/en/top/>, (2020-12)

1143 謝辭