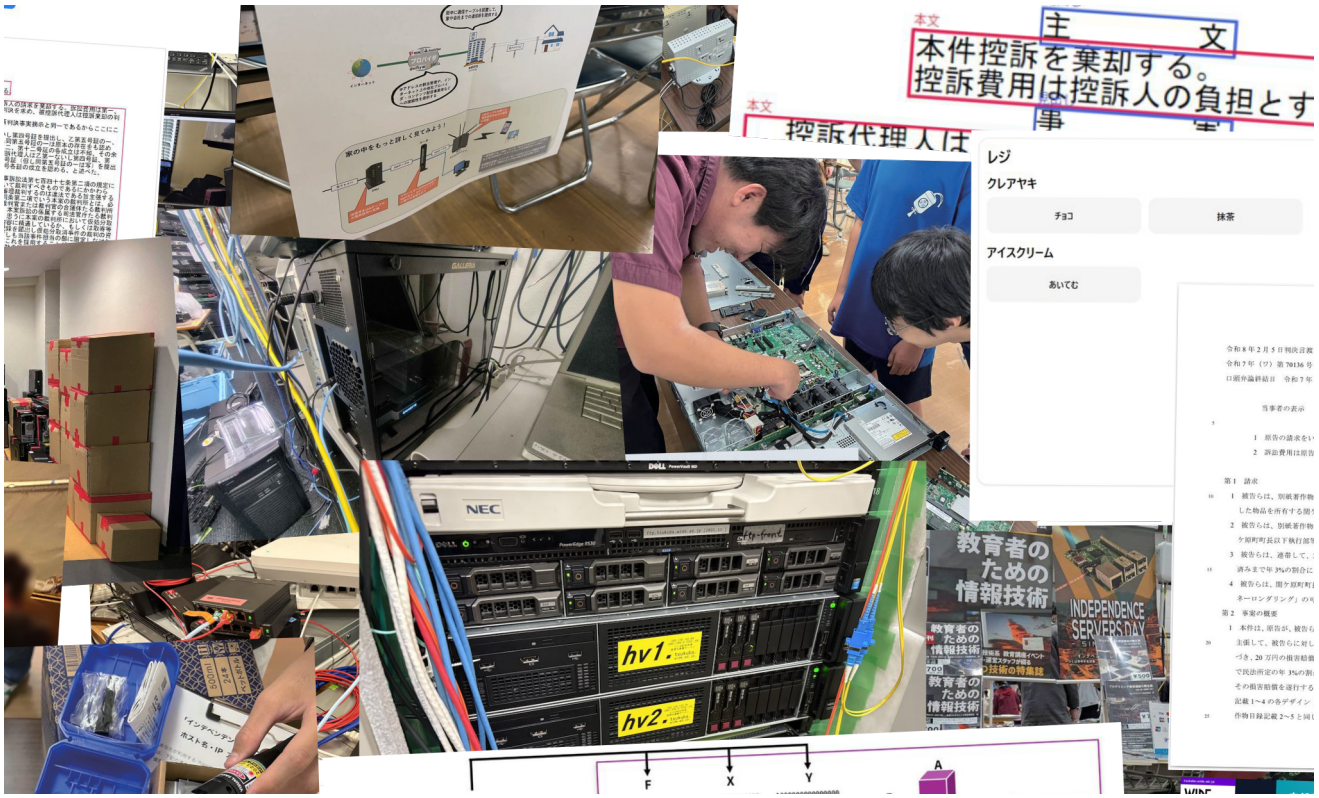
国立大学法人 筑波大学 情報学群 情報科学類

産学間連携推進室



2025年度
活動成果報告書

"AC-Room" Journal 2025

目　次

情報科学類 産学間連携推進室の概要 …… 4

中学生向け産学官連携 ICT 教育イベント開催の総括 …… 8

DPDK ベース高速ソフトウェアファイアウォールの設計・実装・性能評価 …… 29

判決文 PDF ファイルの本文領域抽出モデル …… 41

イベント型店舗運営に適用可能なカスタマイズ性を有する販売管理システムの開発 …… 49

GEMMul8 における Ozaki Scheme II INT8 DGEMM のデータフロー最適化 …… 55

Zig 言語の普及に関する活動 …… 67

Wide Project つくば NOC 運用報告 …… 69

Media over IP 技術に関する取り組み・研究 …… 77

空間 ID の集合演算の基礎理論の構築 …… 97

情報機器資源リユースプロジェクト活動報告 …… 107

奥付 / 裏表紙 …… 118

情報科学類 産学間連携推進室の概要

情報学群 情報科学類 間瀬 太陽（s2410334@u.tsukuba.ac.jp)

2026 年 5 月 20 日

1　概要

筑波大学 情報学群 情報科学類 産学間連携推進室（以下、当研究室という）は、主に情報科学類生による研究開発活動の支援を目的として学生主体の研究室であり、工学系学系 F 棟 3F230 室の割り当てを受け活動を行っている。

通常、情報科学類生は 4 年次における研究室配属まで、学内に研究拠点を持つことが無い。そこで当研究室では 3 年次以下で研究活動を行いたい学生に対し、学内での研究開発拠点を提供することで活動を支援し、そのような学生が互いに切磋琢磨し合える環境を作ることを目的に各種活動を行っている。また、研究開発した成果を外部にアウトプットする機会も提供し、この研究室から大学発ベンチャー起業や、優秀な技術者が排出される流れを創造することも目的に活動している。

当研究室は 2014 年度より情報学群情報科学類に移管され、同室の活動に置いて正式な支援及び予算の提供を情報科学類より受けている。これは、現在の多岐に亘る活動に加え、今後のより柔軟な活動を目指す上で大きな礎になっている。

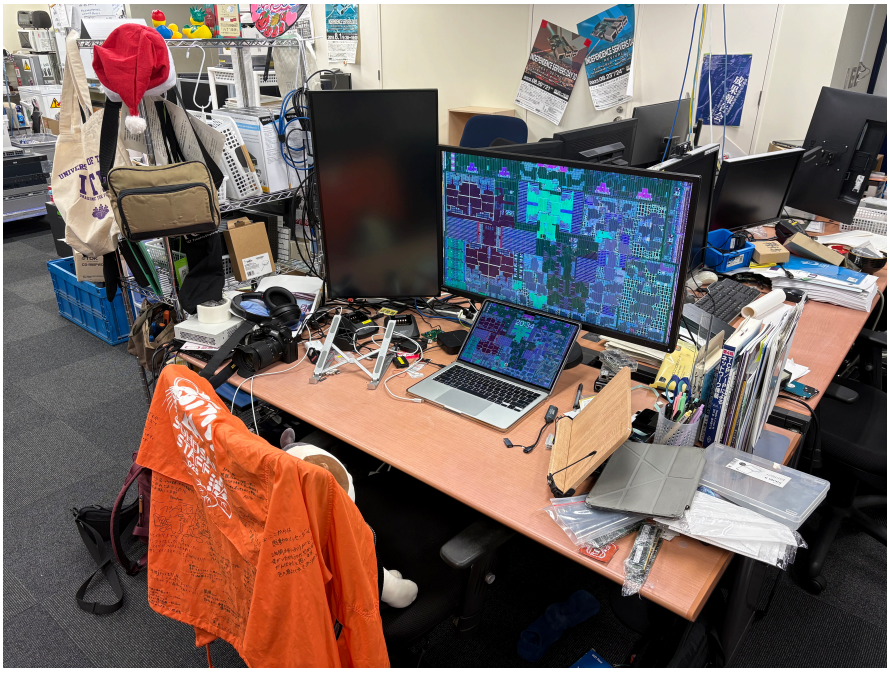


図 1: 間瀬のデスク 2026/04/21 撮影

2　目標

2025 年度において当研究室は、今までと引き続き以下の項目を目標として各種活動を行った。

- 個人または複数人でプロジェクトを立案遂行し、問題点を認識し、その問題を解決する一連の流れを経験することによって総合的な問題解決能力および開発能力の向上を図る。
- 前年度の成果をもとに研究開発活動を継続的に行い、より長期間にわたるプロジェクトの遂行を通じて、より大規模なプロジェクトを遂行する能力の向上を図る。

3　目標に対する評価

2025 年度において当研究室の活動について、上述した目標に対する評価を述べる。

当研究室においては、様々な活動を行うにあたり、その基礎となる考え方として「問題解決型開発」を取り入れている。これは、既存の技術などについて問題を提起し、その問題を解決するためにソフトウェア等の実装を行うというものである。各活動において、それぞれ問題点を提起しその問題を考察し解決するための実装を行い、問題の解決に挑んでいる。

このように、一連のプロセスに沿って開発できる統合的な開発能力を育成することに重点を置いた点については、今後長期間活動を行っていくにあたり必須であると考えられ、評価できると思われる。

4　組織構成について

本章においては、当研究室の組織構成について記載する。

4.1　略歴

産学間連携推進室は 2007 年 5 月頃から活動を開始した。 当時は、活動拠点として第三エリア 3B 棟 3B408 号室を利用していた。当時の活動内容は、主にメンバー個人が各々興味のある分野について、個人の裁量にて何かしらの活動を行うというものであり、組織としてまとまって活動を行うことはなく、活動拠点を個人が個人の目的に利用するという形態が取られていた。

また、厚生労働省が行っている特定健康診査に関するデータ入力ソフトウェアの開発をオーダーメイド創薬株式会社（現、株式会社 LOTUS）およびソフトイーサ株式会社との産学連携活動の一環として行った。 このソフトウェアは現在、全国各地の診療所等で利用されている。

その後、第三エリア 3B 棟の改修工事に伴い活動を一時的に休止した。その期間に現在の活動拠点である第三エリア工学系学系 F 棟 3F230 号室に利用開始準備を進め、新たな体制での組織運営を行うための基盤の整備を行った。3F230 号室が利用可能になった後、各種インフラ整備および運営体制の整理が行われた。

更に 2011 年から毎年春に研究開発成果の報告会を学内にて行った。成果報告会はプロジェクトが目指す目標や問題点、それに対する解決方法などに関して実際に行った実装の開設などを中心に行われた。また、2023 年には情報科学類の主催する夏のオープンキャンパスにてもスライド発表を行い、来訪した高校生らに対し研究の紹介等を行っている。

4.2 人員構成

当研究室は、学年主体の組織である。 現在では以下の学生が所属しており、互いに様々な分野で活動を行っている。

- 大学院 システム情報工学研究科 博士後期課程：1 人
- 大学院 システム情報工学研究科 情報理工学位プログラム 博士前期課程：3 人
- 情報科学類 4 年：2 人
- 物理学類 3 年：1 人
- 情報科学類 3 年：1 人
- 情報科学類 2 年：1 人

5 他組織との連携

5.1 WIDE Project

産学間連携推進室は 2007 年 11 月に筑波大学で初めて WIDE Project の活動拠点となった。 その一環として、部屋内に構築した NOC (Net-work Operation Center) においてネットワーク分野の様々な研究開発活動を行っている。

5.2 ソフトイーサ株式会社

産学間連携推進室は、 筑波大学発ベンチャー企業であるソフトイーサ株式会社から様々な支援を受けており、研究開発活動に必要な計算機リソース等をはじめ様々な機材の貸与を受けている。また、実験用途として IPv6 ネットワークのコネクティビティを提供するなど、相互に支援を行っている。

5.3 輝日株式会社

産学間連携推進室のメンバーが立ち上げた筑波大学発ベンチャー企業である。 ネットワーク機器の提供や技術的なアドバイスなどの支援を受けているほか、共同でネットワーク関連の研究および実証実験、教育活動などを行っている。

6　活動概要

当研究室における活動の基準は、各個人の興味にある。 所属するメンバーがやりたい分野について様々な研究開発活動を自由に行うことでさらなる技術力向上を目指している。また、複数のメンバーが協力して分野をまたがった研究開発を行うことで、様々な知見を得て成果につなげることも目的としている。

具体的なプロジェクトとしては以下のものがある。

6.1　産学官連携プロジェクト

- 中学生向け産学官連携 ICT 教育イベント「インデペンデンス・サーバー・デイ」（つくば市総合教育研究所・OPEN プロジェクト）
- 「テキストベースのプログラミングに挑戦してみよう！」（つくば市総合教育研究所・ソフトイーサ株式会社）

6.2　産学連携プロジェクト

- VPN トンネルソフトウェアの活用に関する研究および公開実験（ソフトイーサ株式会社）
- 固定 IP アドレス割り当て型サービス（インターリンク株式会社・ソフトイーサ株式会社）
- モバイルデータ回線のボンディング技術（輝日株式会社）
- セキュア公衆無線 LAN に関する研究及び実証実験（輝日株式会社）
- 小・中学生向けプログラミング講座開催支援（輝日株式会社）

6.3　他組織連携プロジェクト

- WIDE プロジェクト筑波大学 NOC の運営（WIDE プロジェクト）
- 国内向けパブリックミラーサーバの運営（WIDE プロジェクト）
- 情報機器資源のリユース・リサイクルに関する支援（筑波大学システム情報系技術室）

6.4　学生活動プロジェクト

- 居住空間での業務用電子計算機の運用に関する研究
- モバイル回線を組み合わせた頑健な管理用サーバの構築
- 法律文書の自動解析
- 超軽量な量子通信エミュレータの実装
- 人為的ミス・いたずらの削減を主眼に置いた飲食店における注文システムの開発
- 子音情報のみを用いた日本語文字入力手法の提案と実装
- 持続可能な AC 部屋のネットワーク・システム構成に関する考察と実装

中学生向け産学官連携 ICT 教育イベント開催の総括

～ サーバ構築実習イベント「インデペンデンス・サーバー・デイ」の継続的な開催 ～

2026 年 5 月 23 日 筑波大学情報科学類 産学間連携推進室 活動成果報告会
ICT 教育イベントプロジェクト代表　根本 晃輔、　副代表　服部 真吾
協力：産学間連携推進室 関口亞聖、北野尚樹、松田心杏、細島涼雅、中野あおい、
間瀬太陽、齋藤智郎

概要

産学間連携推進室においては、2016 年度より茨城県つくば市総合教育研究所[1]、および筑波大学発ベンチャー企業数社[2]等と協力し、茨城県つくば市の中学生を対象とした実践的な産学官連携 ICT 教育イベント「インデペンデンス・サーバー・デイ」を運営している。本イベントは、各受講生に固有の物理サーバ（Raspberry Pi に代表されるシングルボードコンピュータ 1 台）とグローバル固定 IPv4 アドレス 1 つを配布し、受講生にネットワーク・サーバの動作原理、構築や運用の方法、運用上重要なセキュリティ対策等を学んでもらうものである。受講生は本イベントの開催期間中だけでなく、イベントの後も割り当てられたサーバを継続して自由に運用することができる。本イベントの目的は、高度情報化社会に顕著な「クラウド化」の傾向の中で、特に情報技術・IT インフラ技術に強い興味を持つ生徒らに対し、自主学習の難しい「物理的なサーバの動作や運用」の学習機会を提供し、更に高度な情報技術の学習に向けての橋渡しを狙うことにある。本報告においては、2025 年に開催した「インデペンデンス・サーバー・デイ 10」の開催報告、また 2026 年に開催予定の「インデペンデンス・サーバー・デイ 11」に向けての取り組みについて述べる。

---

[1] つくば市教育局内の組織で、教育における情報技術の導入・運用を専門的に扱う組織。物理的な施設はつくば市大形 1333-1（旧大形小学校）に所在する。

[2] 2026 年 4 月時点での協力組織は、ソフトイーサ株式会社（代表取締役 登 大遊 氏・筑波大学情報学類 2003 年度 AC 入学)、筑波大学 OPEN プロジェクト（同)、輝日株式会社（代表取締役 佐藤 大哲 氏・筑波大学情報科学類 2016 年度入学)、太平電算（代表 根本 晃輔・筑波大学情報科学類 2013 年度 AC 入学）である。また、2026 年度は筑波大学発ベンチャーに限らない協力組織の拡充を予定している。本イベントの開催に当たり、各組織にはネットワーク資源・各種機材・運営学生への謝金等の協力を頂いている。

１．序論

2020 年度の小学校教育における「プログラミング教育」の必修化を契機に、近年、若年層へのプログラミング教育に対する注目・関心が急速に高まりつつある。
特に 2010 年代後半以降においては、官・民の別を問わず、多様な難易度の若年層向けプログラミング・情報技術教育イベントが開催されている。「セキュリティ・キャンプ」の知名度が向上し、応募者が年々増加傾向にあること、小中高生を対象とする「未踏ジュニア」が 2016 年にスタートしたことなどは特に記憶に新しい。

１－１．高度な技術の学習機会の必要性

さて、官民の主体を問わず開催されている「プログラミング」に関連するイベントについて、簡易なビジュアルプログラミング環境である Scratch[3]や、そのコード記述量の少なさ、機械学習・AI 分野などの耳目を集めやすいテーマを扱いやすいことを題材とした Python によるテキストプログラミング、ビジュアルプログラミングを介して物理的なパーツを動作させオリジナルのロボットなどを設計することのできる LEGO MINDSTORMS[4]、小中学生に高い人気を誇るサンドボックスゲーム環境を基盤として簡易なプログラミングに習熟できる Minecraft などの環境を用いて簡易的なプログラミングを学ぶようなものについては、さまざまな企業・団体により、日本各地で同様のものが開催されている。若年層向けの公立小中学校における「プログラミング教育」の運用、または部活動における「プログラミング」の取り組みの拡大もあり、小中学生の児童・生徒らにとって、簡易なプログラミングを学習するための土壌が成立しつつある。

しかし、これらの「世間一般における取り組み」のほとんどは、あくまで特定のプラットフォームの範囲内で Scratch / Python などの環境に閉じた・ごく簡易なプログラミングのみを取り扱うものであったり、運営主体が異なっても特定の教材・教具を用いたワンパターンなものばかりであったり、果ては「プログラミング未経験者可」として講師の求人がなされているものがあったりと、イベント・講座内で技術の学習が閉じており、情報技術全体の学習への発展性のないものと言えるだろう。

---

[3] 一般にはキャラクター（画像）の動きをプログラミングするものであるが、発展させてゲームを制作したり、動画作品のようなものを制作したりすることが可能である。

[4] 2000 年代初頭から同名のシリーズが展開されているが、現在販売されているものは当時より高性能・高機能なシステムとなっている。また、一部の教育機関が実際に教育の場に取り入れているなど、活用事例も豊富になってきている。

それらより高度な内容に当たる「テキストベースの自由なプログラミング」や「サーバ・ネットワークの構築実習」「オリジナルのサービスの設計や構築・保守」を取り扱う、若年層を対象とし、なおかつ気軽に参加できるようなイベントは、2026 年 4 月現在においても、従前同様、ほとんど存在しない[5] 取り組みであると言える。

また、学校教育などの場における「プログラミング」の内容をほとんど理解してしまっている児童・生徒らにとっては、学校や各種のイベントが取り扱う内容より高度なことに取り組んでみようとしても、それを牽引してくれる、あるいは困った時に相談できる「IT 関連の正しい知識・技術を持った大人」を周囲に見つけることが難しく、結果として高度な知識・技術に触れる機会を失っているという現状が挙げられる。

学校・地域の特性如何では、学校に掲示されたポスター等から高度な技術イベントの存在を知るようなことさえも難しいケースも多い。「インデペンデンス・サーバー・デイ」の広報ポスターにも同様のことが言えるが、国・関係機関が運営するようなイベントのポスターでさえ、学校には渡るが実際に掲示すらされない、というケースも散見される。このような背景から学校の支援・環境づくりに期待ができない場合、保護者に IT 関連の知識があり、各家庭において教育ができることが最善であるが、IT 技術に習熟した保護者の全体数はさほど多くはないと推測される。

中学生・高校生といった若年層がサーバ構築やネットワークに関する専門的な技術を学ぶための教育イベントの例としては、サーバ・ネットワーク等が強く関連する「情報セキュリティ」の高度な学習を主題とし，毎年 8 月に独立行政法人情報処理推進機構（IPA）によって開催されている「セキュリティ・キャンプ[6]」が存在する。しかし、セキュリティ・キャンプは全国の 22 歳以下の学生・生徒を対象としたものであり、年齢に基づく対象層は大学生・一部の大学院生までと幅広いことから、参加者全体に占める中学生の割合は非常に低い。また、参加に当たっては応募課題の回答内容による書類審査の過程も存在するため、中学生・高校生にとってはやや敷居の高い環境となっている面が否めない。

---

[5] 既存の「社会人向けプログラミングスクール」が小中学生向けに門戸を開くケースもあるが、一例として 24 回のオンライン授業で 80 万円以上もの高額な代金が発生する講座などが多く、参加に当たり家庭の資本力のハードルが大きく存在すること、また同種のプログラミングスクールは講師の技術力および指導力に疑義があること、該当のスクールについて IT 技術者からの悪評が目立つことなどから、現実的な選択肢ではないといえる。

[6] 情報処理推進機構「セキュリティ・キャンプ」>「セキュリティ・キャンプ事業紹介」
https://www.ipa.go.jp/jinzai/security-camp/about.html

なお、「セキュリティ・キャンプ」に積極的に参加している中学生・高校生は存在するものの、特に近年において、その絶対数は少ないものである。例えば 2018 年の開催回においては全体の参加者 85 名中、高等専門学校の参加者が 7 名、高等学校の参加者が 4 名、中学校の参加者が 1 名、中等教育学校の参加者が 1 名であった。高等専門学校の学生の学年構成は不明であるが、その全てを高等学校相当の年齢と考えても、高等学校以下の生徒の参加者は 85 名中 13 名に留まるものといえる。

「セキュリティ・キャンプ」は、情報セキュリティの第一線に立つ技術者がそれぞれの専門分野の講義を行う、極めて高度な内容を扱うイベントとして知られている。情報技術に興味・関心がある中学生・高校生にとっては、先に述べたように、いきなり「セキュリティ・キャンプ」等の高度なイベントに応募することのハードルは相応に高いものとなるであろう。また、応募課題に対して回答するための知識・技術を身に着けるための場が身近にある中学生・高校生は決して多くない。現段階で高度な知識を有さない人材の「育成」よりも、既にある程度の知識を有している人材の「発掘」に偏重している種々のイベントは、「興味がある」だけで参加できるような状況とは程遠い。

当人がおかれた環境に強く左右されない、より広範な若手の情報技術者の育成という観点を考慮すれば、最初から「セキュリティ・キャンプ」に参加できるような知識・技術を持ち合わせているような生徒でなくとも、将来的にこのような先進的なイベントへの参加を見据えた上で、受講生が自らの知識・技術を高めていけるような場が求められていることは、論を俟たない。

このように、児童・生徒らにとって実践的な学習機会が不足している、より詳細に言えば「学校教育における情報技術の学習に満足していないが、セキュリティ・キャンプのような高度なイベントに参加するための知識・技術を有しているレベルには至っていない」児童・生徒らが「より高度な学習機会にアクセスするための、学校教育より高度で実践的な知識・技術を得る」ための場がないという状況は、将来にわたって各種の技術を維持・活用するための高度な IT 人材の育成という観点から、大きな損失を招きうるのではないかと考えられる。

１－２．クラウド化の傾向、物理的なネットワーク技術の学習機会の減少

近年の情報化社会に顕著な「クラウド化」の傾向は、情報技術を専門としない一般的なユーザにも、容易かつ安価に大規模なコンピュータ・ネットワーク・サーバを取り扱うことを可能にした。

古くは 1990 年代最末期～2000 年代初頭頃から活用されているような「レンタルサーバ[7]」がその例となるが、2010 年代に入って以降、AWS や Google Cloud Platform などの「クラウドコンピューティングサービス[8]」が急速に知名度を獲得し、商用の大規模なサービスに用いられている例、またオンプレミスで構成されたシステムがクラウド環境に移行した例も数多く存在する。

確かにこれらのサービスを用いることで、何らかの Web サイト、あるいは Web アプリケーションを手っ取り早く公開したいユーザにおいては、本来は大変複雑な知識・技術や物理リソースを必要とするような「ネットワーク・サーバ」に関する構築の操作を最低限とし、その複雑かつ本質的な部分に関する作業をレンタルサーバ・クラウド事業者に転嫁することによって、Web サイト、あるいはアプリケーションの開発そのものに自らの開発リソースを集中させることができるだろう。

しかし視点を変えれば、これらの技術を用いることで、ユーザはネットワーク・サーバ環境そのもののハードウェア・ソフトウェア的な動作について詳細に理解することなく、環境を利用するために必要な表層的な知識・操作方法のみを習得すれば良い状況が生まれてしまうと言えよう。したがって、これらの技術の発展は、それぞれの物理的な環境を動作させるための複雑な実装にユーザが直接触れる機会を減少させるものと成り得るのである。いわゆる「レンタルサーバ」の時代から連綿と続く「クラウドコンピューティングサービス」に関連する諸技術の発展以降、情報技術に強い興味や関心を持つ子供らにとって、物理的なサーバの動作や、インターネットとの直接の繋がりを知覚する機会は次第に失われつつある。

１－３．ICT 教育イベント「インデペンデンス・サーバー・デイ」の運営

このような背景を踏まえ、産学間連携推進室 ICT 教育プロジェクトでは、2016 年以降、グローバル IP アドレスと物理サーバを用いたサーバ構築・運用技術を学ぶ産学官連携 ICT 教育イベント「インデペンデンス・サーバー・デイ」を、毎年 1 回、継続的に運営している。

---

[7] 一例として、さくらインターネット株式会社による「さくらのレンタルサーバ」は、年額 1,500 円ほどから利用できるなど、低価格で利用しやすいことで有名である。商業用途に利用できるエディションも提供されている。

[8] これらのクラウドコンピューティングサービスは、利用したい時に利用したいだけの資源を都度課金にて利用できるという手軽さから、個人用途から商業用途まで幅広く用いられている。

本イベントは、つくば市総合教育研究所における通年の ICT 教育イベント「キッズプロジェクトプログラミング講座」の一環として実施されているものであり、「インデペンデンス・サーバー・デイ」は小中学生の夏休み期間に開催されるイベント「つくばプログラミングフェスタ」内の 1 講座として開催されているものである。

本イベントにおいては、将来の優れた IT 人材となりうる子供らに、サーバの基本的な構築方法や動作原理、セキュリティなど各種の先進的な問題について学習する環境を提供することを目標としている。

以降の章においては、本イベントの通算 10 回目の開催に当たり、2025 年 8 月に開催した「インデペンデンス・サーバー・デイ 10」に関する報告、2016 年以降の各回イベントにおける実施形式・講習内容の変遷や考察、また 2026 年に開催が予定されている「インデペンデンス・サーバー・デイ 11」に向けての取り組みについて述べる。

2.「インデペンデンス・サーバー・デイ 10」の開催報告

2－1. 2025 年度開催「インデペンデンス・サーバー・デイ 10」の概要

産学官連携 ICT 教育イベント「インデペンデンス・サーバー・デイ 10」は、2025 年 8 月 20 日・21 日の 2 日間、両日とも 9:30 - 16:00 につくば市総合教育研究所において「つくばプログラミングフェスタ 2025」の 1 講座として開催され、つくば市の中学生 10 名が参加した。本年度は事前のポスター・パンフレットによる広報が従前通り実施できたことから、応募総数は例年通りの水準で 44 名となった。

本イベントは 2018 年以降、事前のポスター・パンフレットなどでの広報に注力している。特に 2022-2023 年度においてはつくば市の全中学生の人数の 1% に当たる 50-60 名ほどの応募を記録したことから、本活動が広く周知されている様子を窺うことができ、広報によって特に大きな成果を得られていることは間違いないことである。例年の状況として、申し込み開始当日に申し込みを行う受講希望者、（自己申告によるが）既に自宅でサーバを構築・運用した経験のある受講希望者が応募者に多くみられるなど、極めて意欲的な受講生が多いように思われる。

「インデペンデンス・サーバー・デイ 10」の開催告知に利用したリーフレットは、次頁掲載の通りである。本リーフレットは、産学間連携推進室所属の学生のデザインにて、また協力組織の資金負担にて製作され、つくば市内の公立中学校の生徒全員に配布された。

図 1 :「インデペンデンス・サーバー・デイ 10」のリーフレット

２－２.「インデペンデンス・サーバー・デイ 10」の主催・後援・協力組織

本イベントの主催組織は、つくば市総合教育研究所である。後援組織である筑波大学情報科学類には、講座において利用する機材の貸与や費用の支援を受けているほか、産学間連携推進室の学生が本イベントの直接的な運営に関与している。協力組織である筑波大学 OPEN プロジェクトおよびソフトイーサ株式会社・輝日株式会社には、同様に機材の貸与や費用の援助、グローバル IP アドレスなどの提供を受けている。

本イベントの実施に当たっては、ソフトイーサ株式会社によってつくば市総合教育研究所の実習室内に光ファイバ線（フレッツ光）の引き込みが行われ、筑波大学 OPEN プロジェクトの技術によって、フレッツ光を経由して OPEN プロジェクトのネットワークを利用できる状態としている。このようにして構築されたネットワークは、本イベントの開催当初より現在に至るまで、「インデペンデンス・サーバー・デイ」において受講生が構築したサーバを運用するために継続して活用されている。

また、本イベントは産学間連携推進室内の「情報機器資源リユースプロジェクト」を中継し、筑波大学システム情報系技術室より、受講生が遠隔操作の実習・調べ物などで一時的に利用するための、情報機器資源リユースプロジェクトによってリユースされた Windows ノート PC 8 台（図 2 にその一部を示す）、ディスプレイ 13 台の貸与を受けている。

図 2：情報機器資源リユースプロジェクト・技術室経由で貸与を受けた Windows ノート PC

更に、イベント当日に受講生の実習の補佐をするスタッフとして、開催当初より筑波大学情報科学類の学類生有志、情報メディア創成学類の学類生有志、情報理工学位プログラム（旧・コンピュータサイエンス専攻）大学院生有志の協力を受けている。

また、本年度は前年にあたる 2024 年 7 月に「日本ネットワーク・オペレーターズ・グループ JANOG54 Meeting」内「若者へのネットワーク技術継承 BoF」において「インデペンデンス・サーバー・デイ」の取り組みを広報する機会が存在したことから、当該の BoF に関連する熱心な議論を経て、本年度は上述の筑波大学関係者・協力企業各社のスタッフに加え、当該講演への参加者である BGPNAP 小坂様の協力・セキュリティに関する実践的な講義の御担当を頂くことができた。

[Day2] 2024年7月4日(木)

開催時間	開催場所	タイトル
10:00～11:30	会議室 201+202	TS-OPS(時刻同期) BoF
10:00～11:30	会議室 205	若者へのネットワーク技術継承BoF
10:30～11:30	会議室 206	地域NOG BoF
11:40～12:20	会議室 206	無線LAN認証連携BoF

図 3：JANOG54 BoF 一覧より「若者へのネットワーク技術継承 BoF」

改めて、当該 BoF へのお誘いを頂いた株式会社ミラパス 小島様、2025 年度の「インデペンデンス・サーバー・デイ」に引き続きのご参加を頂いた BGPNAP 小坂様をはじめ、関係の皆様方にはこの場を借りて、厚く御礼を申し上げる次第である。

２－３．対象とする受講生について

本イベントの対象は、つくば市内に在学する中学生で、2025 年 8 月 20 日および 8 月 21 日両日 9:30～16:00 の時間帯の全てを通して対面形式のイベントに参加できる、以下のスキルと意欲を持つ生徒とした。

- パソコン操作（キーボード、インターネット検索）ができること。
- 「TCP/IP」、「ポート番号」などの用語の意味・概念が少しわかること。
- インターネットサーバー構築に興味があること。

２－４．講習内容について

本イベントでは、参加者のそれぞれに小型の SBC である Raspberry Pi 3B を 1 台ずつ配布し、Raspberry Pi 上の Raspberry Pi OS（debian ベース、旧称 Raspbian）で動作する Apache Web サーバを用いた基本的なサーバ構築手法と、HTML・CSS を用いた簡易な Web サイトの作成方法に関する講義・実習を行った。講習の様子を収めた写真を、下記図 4 に示す。

図 4：「インデペンデンス・サーバー・デイ・10」開催の様子

下記に、講座において取り扱った内容の概略を箇条書きで示す。

- サイバーセキュリティ専門家による、公開サーバのセキュリティの講義・諸注意
- コンピュータの動作原理および OS の基本的な概念、および Linux の基礎についての講義と、ターミナルを用いたコマンドによるディレクトリ・ファイル操作の実習
- グローバル IP アドレスとホスト名の設定を実習形式で行ったのち、ping、traceroute、nslookup などのネットワークコマンドの操作方法に関する実習
- Raspberry Pi OS への Apache のインストールと基本的な設定、およびサーバ上への HTML コンテンツの配置
- HTML と CSS の記述による Web サイト制作の実習
- 実際にサーバをラックに設置したのち、サーバへの SSH・SCP による遠隔接続の実習、および遠隔操作方法に関する説明
- 自由な質疑応答コーナー「なぜなにコーナー」
- 認定エシカルハッカーによるセキュリティの心得講座

また、本年度は上記の講座の流れに加え、下記に示すさまざまな体験コーナー・説明コーナーを設け、休み時間やイベントの終了後などに、受講生らが積極的な体験を伴って情報技術に触れていくことのできる取り組みを行った。いずれのコーナーも多くの受講生が実際に体験を行っており、受講生の低レイヤ・ネットワーク技術に対する強い興味・関心が感じられた。

- LAN ケーブル成端体験コーナー
- 光ケーブル成端体験コーナー（ソフトイーサ株式会社による）
- 1U サーバ分解体験コーナー（産学間連携推進室 服部氏・関口氏による）
- フレッツ光関連機材見学「インターネットが家に来るまで」の紹介コーナー

これらの取り組みのうち「1U サーバ分解体験コーナー」は従前より行ってきたものであるが、他の取り組みは 2024 年度より行っているものである。以降において、その様子を示す写真をいくつか掲載する。

図 5：LAN ケーブル自作（成端）体験コーナーの様子

図 6：LAN ケーブルの成端に挑戦する受講生（’24 撮影）

図 7：光ケーブル自作（成端）体験コーナーにおける、
可視光を用いた光ファイバケーブルの構造の提示・光信号の伝送の実演（’24 撮影）

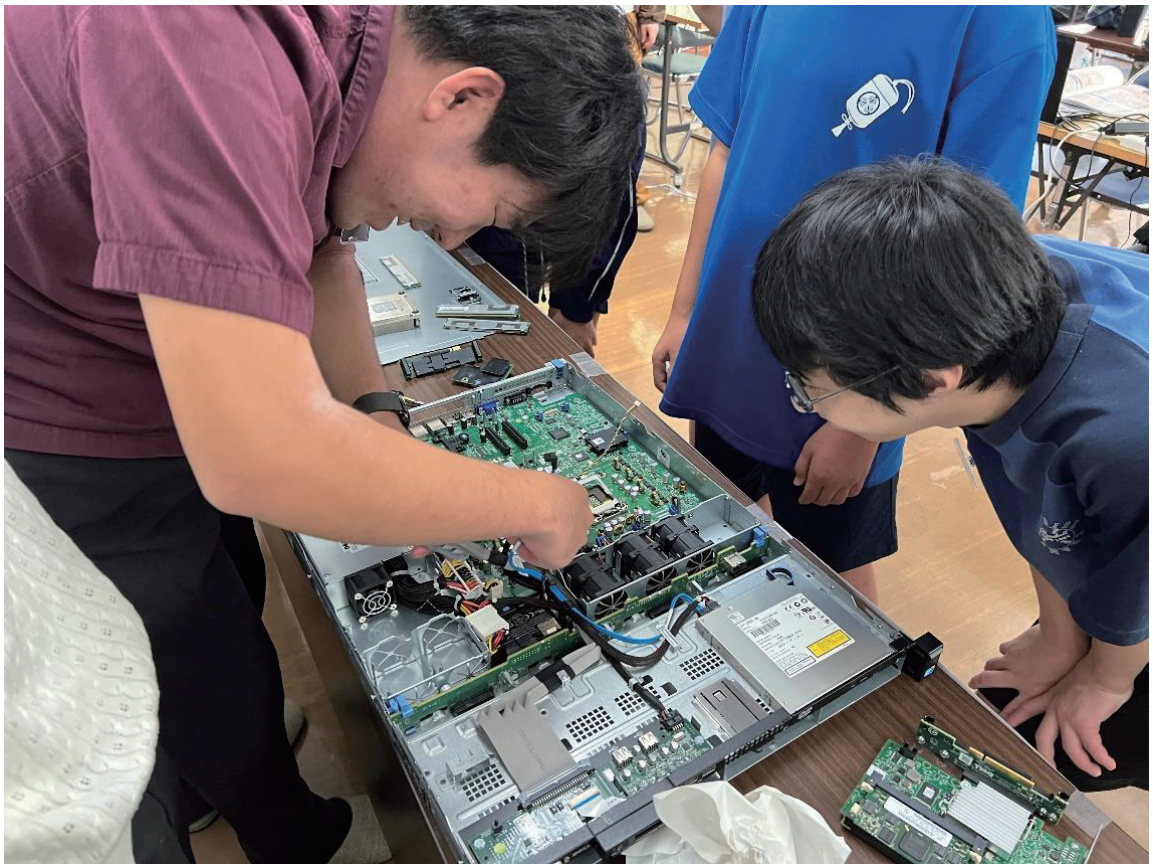


図 8：毎年恒例となった、1U サーバ分解体験コーナー
（分解する機材の提供は、情報機器リユースプロジェクト・システム情報系技術室による）

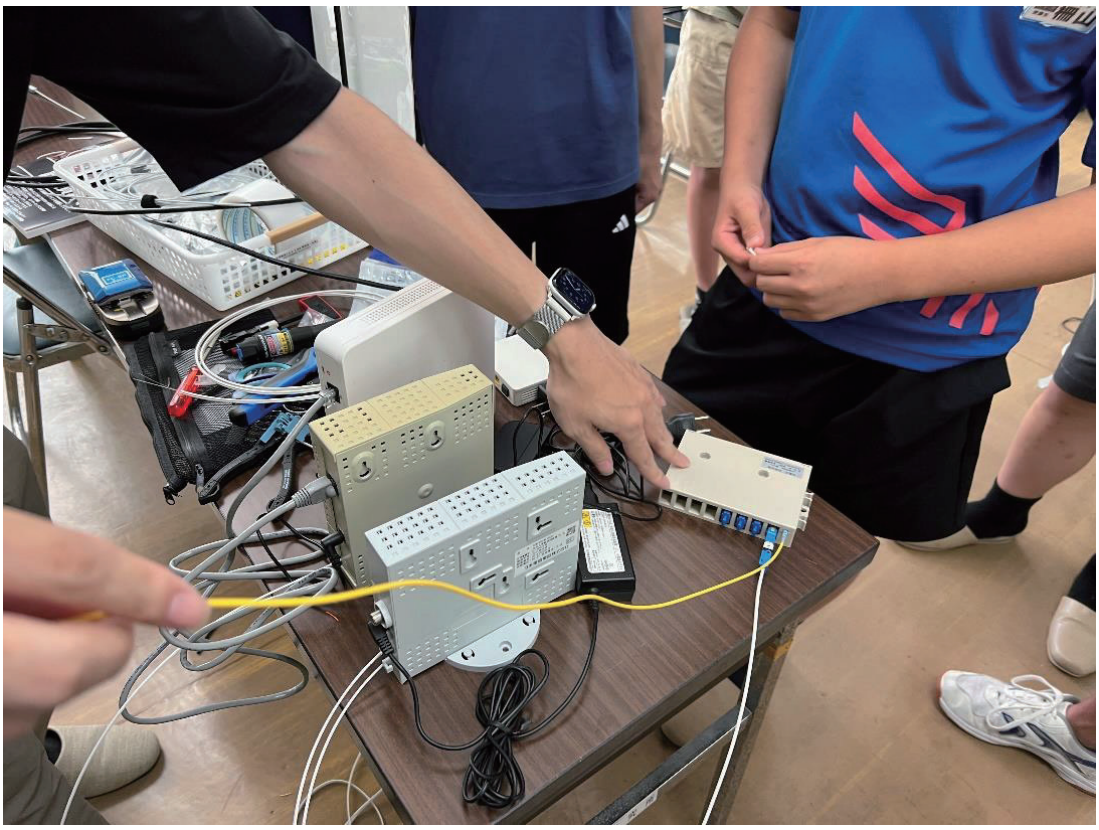


図 9：フレッツ光関連機材見学コーナー

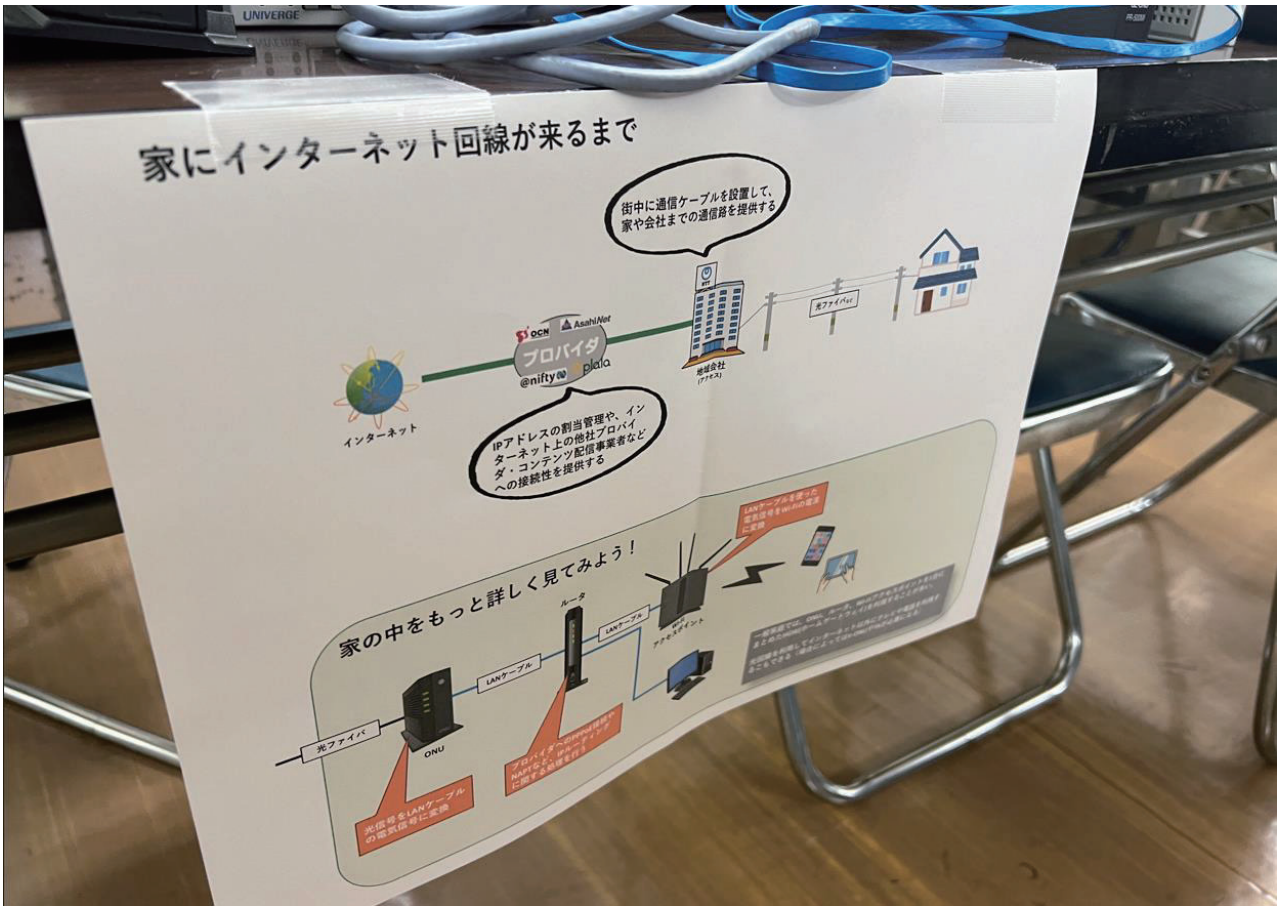


図 10：フレッツ光関連機材見学コーナー掲示の「家にインターネット回線が来るまで」

なお、当日の講習においては、それぞれについてオリジナルのスライドを作成し学習用の教材とした。「コンピュータの動作原理および OS の基本的な概念、および Linux の基礎についての講義と、ターミナルを用いたコマンドによるディレクトリ・ファイル操作の実習」は細島氏が、「グローバル IP アドレスとホスト名の設定を実習形式で行ったのち、ping、traceroute、nslookup などのネットワークコマンドの操作方法に関する実習」は間瀬氏が、「HTML と CSS の記述による Web サイト制作の実習」は松田氏が、「認定エシカルハッカーによるセキュリティの心得講座」は BGPNAP 小坂氏が、それぞれ教材作成・講義の実施を担当した。また、それぞれの分野の専門書を受講生らに配布し、事後の学習に役立ててもらえるように配慮した。

イベントで受講生らが利用した Raspberry Pi はイベント以後もつくば市総合教育研究所内に設置し、受講生が遠隔ログインを行って引き続き実習を行えるようにした（図 11 にその写真を示す）。また、単純なサーバの構築手法を解説するだけでなく、近年の中学生の関心の強い話題を用いてサーバ・ネットワークについて受講生により強い関心を持ってもらうため、図 12 のようなスライドを用いて「なぜなにコーナー」と称する自由な質疑応答コーナーを設け、参加者の中学生から寄せられたさまざまな疑問・質問に講師陣が回答した。

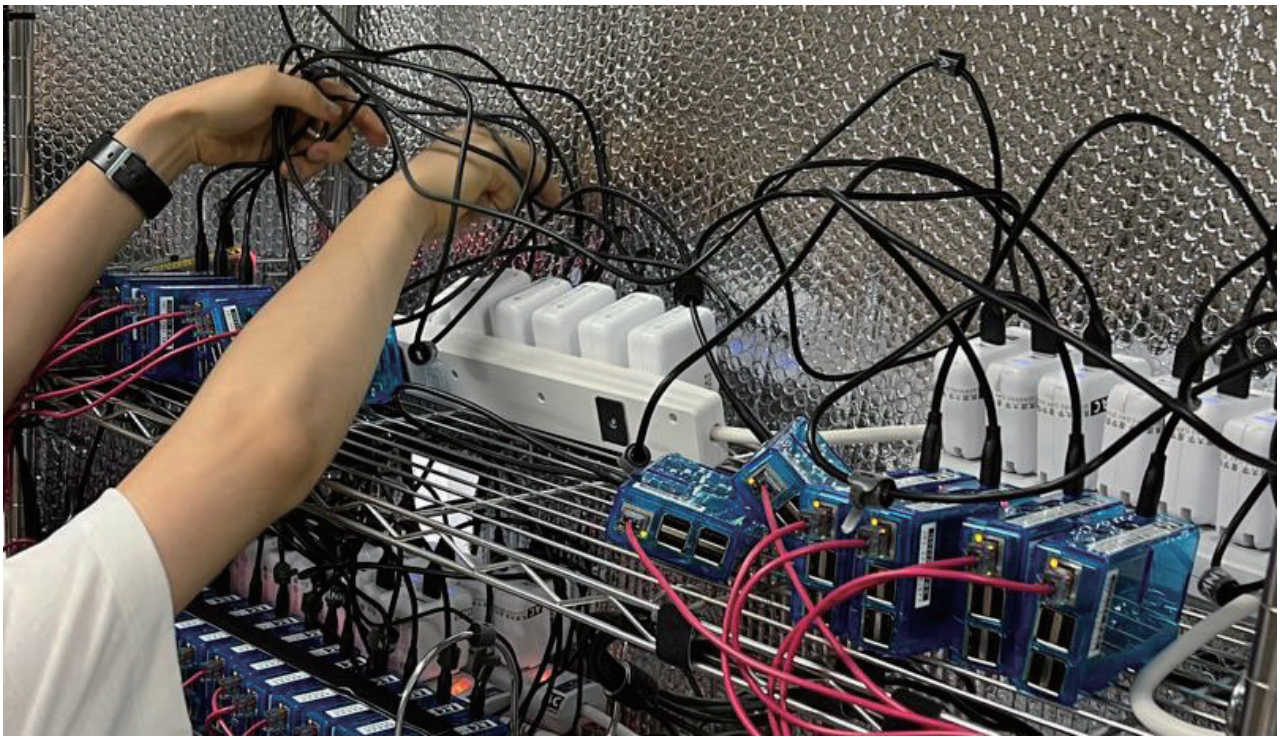


図 11：「インデペンデンス・サーバー・デイ・10」において受講生が設置したサーバ

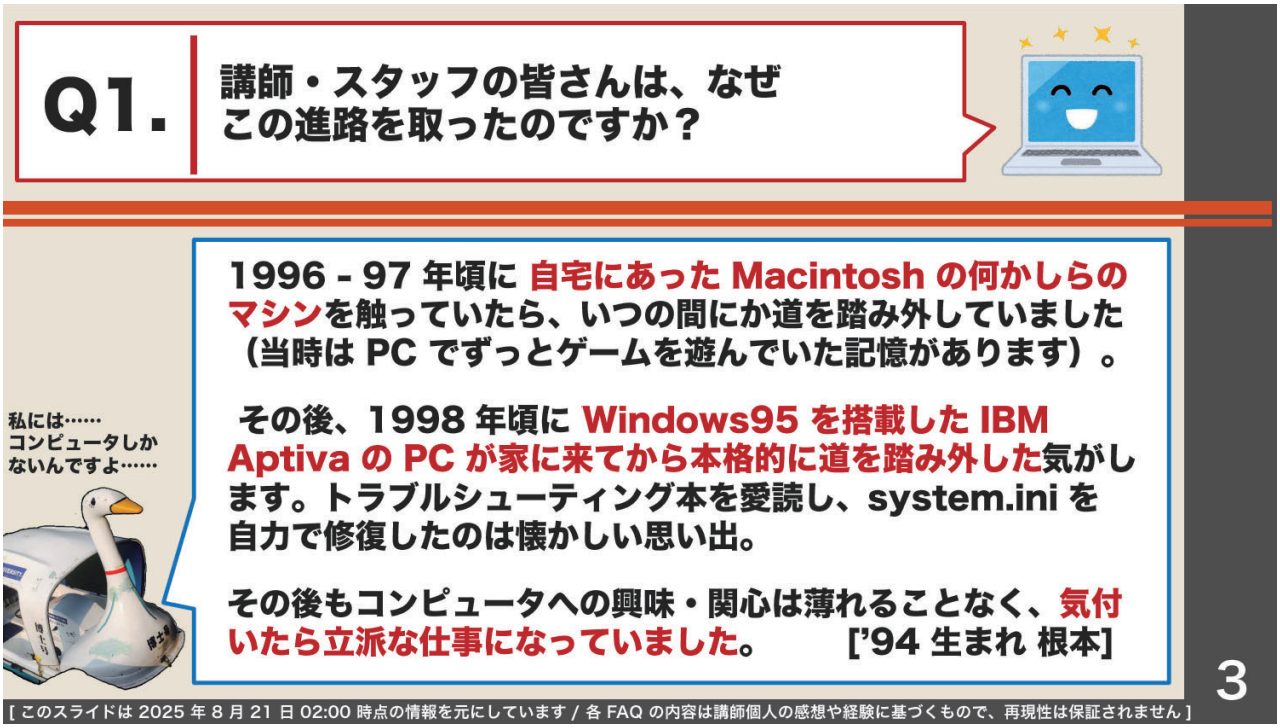


図 12：「なぜなにコーナー」における質疑応答の例

また、本イベントはサーバの継続的な運用の実習の場としても活用できるようにするため、運用上の問題や技術的な質問などに回答するための連絡の場である Discord を設けるなど、受講生らの興味・関心を長期的に満たすことのできるような運営を行っている。

図 13 に、Discord において実施された受講生とのやり取りの例を示す。

図 13：Discord における受講生との継続的なやり取り

２－５．イベント実施を踏まえて

本イベントにおいては、各回の受講生に対して「講座の難易度」や「扱った問題の難易度」「講座への感想」などの項目によるアンケート調査を実施している。「インデペンデンス・サーバー・デイ・10」においても、本イベントは中学校におけるコンピュータ（技術家庭科）の授業を易しいもの・物足りないものと感じる受講生らにとって、学校の授業では扱わないような幅広い事項を 2 日で扱うというややペースの速い・ごく実践的な実習[9]であり、ある程度高い難易度でありながらも、楽しみながら実践的な内容を理解してもらえるイベントとなったと考えられる。

2025 年度は、資料の完全形での製本という受講生・スタッフ双方にとって大きな意義のある取り組みを達成したほか、筑波大学関係外の技術者による講演をいただく場を設けたことにより、情報通信業界全体での若手技術者育成への取り組みの足掛かりとしての大きな試みを行うことができた。また、前年に引き続き整備した、受講生が休み時間にさまざまな「ネットワーク技術における低レイヤの体験」を行える環境は、本年も極めて有効に活用され、今後の「インデペンデンス・サーバー・デイ」においても同様のコーナーを設けることの意義が示さ

[9] 筑波大学情報科学類において開設されている授業である「コンピュータリテラシ」を基準として考えれば、大学生が数週間かけて学習するような内容を 2 日でこなす内容となっていると言えるだろう。また、一般的な IT 企業の新人研修（情報系の強力なバックグラウンドを持たない新人に対する技術研修）として考えるのであれば、少なくとも 2-3 週間以上をかけて、少しずつ慣らすように扱っていかなければならないような内容である、とも言えるだろう。

れたほか、「インデペンデンス・サーバー・デイ」級の規模でなくとも、手軽に情報通信技術に触れてもらえるようなイベント・出張講座の開催に向けた足掛かりを築くことができたものとも思われる。

３．2026 年度の「インデペンデンス・サーバー・デイ」

2026 年度も「インデペンデンス・サーバー・デイ」は実施の方向にて調整を進めている。無事に実施となった場合、2026 年度は通算 11 回目の開催となる。

成果発表会当日：2026 年 5 月 23 日時点においては、その詳細が完全に確定している状態ではないが、2026 年の開催規模については、2025 年と同様のもの（定員 10 名・応募状況により 12 名まで受け入れ・2 日間開催）となる予定である。本イベントはつくば市をはじめ、関係組織からの期待も大きく、十分な広報を行った環境の下では極めて高い応募者数の水準を維持していることから、2025 年までの「インデペンデンス・サーバー・デイ」の開催によって得られた知見：たとえば「なぜなにコーナー」の効率的な運用、スタッフ・受講生間の Slack（に限らないが）コミュニティでの関係性の醸成、サーバ・ハードディスクなどの解体コーナーをはじめとした「体験コーナー」の運用等を引き続き実施・発展させることを目標としていきたいと考えている。

また、講座の本筋はもちろんのこと、「インデペンデンス・サーバー・デイ 10」における各種の体験コーナーのような、講座から派生したさまざまな「興味を惹く」活動にも目を向けてもらえるよう、引き続き取り組みを進めていく所存である。

４．総括と今後の課題

「インデペンデンス・サーバー・デイ」は、2026 年度以降も継続的に開催し、2016 年度～2025 年度と同様、産学間連携推進室のメンバーが各種講義・体験コーナーの企画、講習資料の作成や当日の運営に関与する予定である。「インデペンデンス・サーバー・デイ」の総括、および今後の本イベントの継続開催や、他種の発展的な ICT 教育イベントを行う上での課題について、次の通り記す。

４－１．「インデペンデンス・サーバー・デイ」の総括

「インデペンデンス・サーバー・デイ」は、過去 10 年に亘る開催実績、およびその中で積み上げられてきた多様なノウハウの下、昨今の情報化社会において、地方自治体と地元の大学・

ベンチャー企業、そして「インデペンデンス・サーバー・デイ 9/10」では若年層の技術者の育成に強い興味・関心を有する学外の技術者までもが関与するという、産学官連携の理想的な形の下、高い技術・知識を持つ意欲的な中学生に対し、より高度かつ実践的な技術・知識を学べる有意義な「情報技術者教育」の場を提供することができたと考えられる。先述のように、直近回の開催においては広報面での大きな問題が生じたものの、その中でも定員を充足するだけの応募があったこと、また総じて受講者の満足度が高かったことから、大きな成功を収めたと言える。

一方で、現在の本イベントは 2 日の中にさまざまな要素を詰め込む形の開催であることから、内容の取捨選択や講義速度の調節について、引き続き緻密な計画が必要となることは間違いのないことである。本イベントは先に述べた通り、今年度以降も継続して開催する予定であるが、引き続き講座で取り扱う情報量の精査・検討を行い、「受講生の理解」を第一に据えた適切な進度で実習を進められるような改善を行っていく予定である。

また、本イベントの受講希望者数、および受講生へのアンケートにおいて、前年度に引き続き、「つくば市の中学生」の実践的な IT 技術を取り扱うイベントへの参加の意欲が非常に高いことが分かった。アンケートの回答内容や、イベントの Web サイトへのインターネットコミュニティ上の反応などから、サーバ構築・運用について取り扱う本イベントの定期的な開催、また地域や年齢層などの規模を拡大しての開催だけでなく、以前産学間連携推進室が運営していた中学生向けイベントである「テキストベースのプログラミングに挑戦してみよう！」に代わり得るような一般的なプログラミング言語を用いたプログラミングの実習・アプリケーションの制作や、OS やネットワークの詳細な事項について掘り下げて取り扱うような多種の実践的なイベントを開催することが、将来活躍するであろう高度な IT 人材の育成に強く求められているのではないかと考えられる。

４－２．他種の講座の開設について

「インデペンデンス・サーバー・デイ」、またかつて開催していた「テキストベースのプログラミングに挑戦してみよう！」の両講座における受講生向けのアンケートを通じ、情報技術に高い関心を持つ中学生から、引き続き多様な実践的技術を取り扱う講座の開催を望む声が挙がっていることが分かっている。

また、著者は中学生向けに筑波大学が実施している「夏休み自由研究お助け隊」内の講座「Arduino を使ってプログラムしてみよう」におけるメイン講師・教材作成担当を 9 年間務めているほか、「つくば SKIP アカデミー」において同系統の講座である Arduino プログラ

ミング講座を運営した経験もあること、また 2017 年頃より一般企業における技術研修を業務として行っていることなどから、高度な意欲を持つ中学生向けの ICT 教育に関する知見が豊富に蓄積されている状態にある。

2026 年度においては「インデペンデンス・サーバー・デイ」の継続開催を軸におきつつ、つくば市総合教育研究所をはじめとする各関係組織・組織内外の協力者と連携しながら、より幅広い参加者を募ることのできる他種の講座の開催・「インデペンデンス・サーバー・デイ」の内容の洗練・マテリアルの整備という 2 つの目標を設定し、本活動をより発展的なものとしていくことを検討している。

新規に開設する講座の具体的な候補としては、下記のようなものが挙げられる。

● 「情報機器資源リユースプロジェクト」によって回収された廃棄 PC を活用し、PC の分解や組み立ての実習を行い、併せて PC を構成するさまざまな部品について解説する短時間・単発の講座を開催する。短時間であることから気軽に参加できることが想定されるほか、低予算で繰り返し実施でき、親子での参加等も可能であるため、実施までのハードルが極めて低い。また、現在産学間連携推進室が学内向け（新入生向け）に行っているイベントの知見の応用が可能である。

● 「openfab 創房」において設置されている機材を用いた、デジタルなものづくりを体験できる講座を開催する。システム情報系技術室の支援を得られる状況にあるほか、「夏休み自由研究お助け隊」での内容に近い実習形式となるため、こちらも実施までのハードルが低く、なおかつ一般的な中学生の興味を惹ける内容ともなっている。また、筑波大学の設備を活用する講座となることから、大学自体の取り組みの広報にも繋げることが可能である。

● 簡易に取り扱えるマイコンボードである Arduino を活用し、ブレッドボードを用いた簡易な電子工作と組み合わせ、触って遊べるおもちゃ・ゲームを制作する。ある程度限られた予算・受講期間であっても、早押しゲームや記憶ゲームなど、受講生の記憶に残るような楽しい作品を制作することが可能である。性質上、先述の「夏休み自由研究お助け隊」と近い内容となることが想定される[10]。

---

[10] 「夏休み自由研究お助け隊」では 2.5 時間の枠内で、Arduino を用いた LED の点灯から、スイッチと LED、ブザーを用いた記憶ゲームの制作までを取り扱っている。なお、一部に筑波大学情報科学類開設科目である「情報科学基礎実験」の内容と共通している部分があり、この点でも中学生にとっては応用的な内容を扱っていると捉えることが可能である。

● Java 言語や C# 言語などを活用し、Scratch で実装するようなキャラクターの描画を伴うプログラムを記述する。Scratch を既に体験したことのある生徒に受講してもらうことにより、Scratch などで習得したプログラミングの手法が、実際の開発に用いられるプログラミング言語でも活用できることを学習してもらうことを目標とする。また、場合によっては、産学間連携推進室関係者が開発したオープンソースのゲームエンジン Altseed を活用し、簡易なゲームに拡張することも検討する。

● Raspberry Pi の GPIO ピンを活用し、各種のセンサなどの部品と連携したデバイスを制作する。Raspberry Pi の性質上、インデペンデンス・サーバー・デイで取り扱っているような単体でのサーバの構築が可能であるため、例えば「アクセスすると部屋の温度が実測値で表示されるサーバ」など、複数の要素を組み合わせた成果物を仕上げることが可能である。ただし、内容の都合上 1-2 日での実施が難しいことが想定されるため、この内容での開催に当たっては開催日程やその対象についてよく検討する必要があるものと思われる。

４－３．講座開催に伴う知見の共有について

「インデペンデンス・サーバー・デイ」の開催を通して得られた知見については、産学間連携推進室のイベント運営者有志により「プログラミング教育講座の舞台裏」および「教育者のための情報技術」という自費出版書籍に取りまとめを行い、定期的に日本最大の同人誌即売会「コミックマーケット」にて刊行している。その様子を図 13 に示す。

図 13：「プログラミング教育講座の舞台裏」などの頒布の様子

「プログラミング教育講座の舞台裏」は、2019 年 8 月に創刊し、2026 年 4 月現在時点では Vol.06 まで、および「教育者のための情報技術」というスピンオフ書籍も Vol.03 までが出版されている。Vol.01 から Vol.06、およびスピンオフ書籍 Vol.03 までを含めた現在までの累計発行部数は、現時点で 1000 部ほどに上る。「コミックマーケット」会場ではコンピュータ技術者、教育者、また情報技術教育に携わる方々によって多く購入されているようである。

５．学会発表等に関する情報

2018 年春時点までの「インデペンデンス・サーバー・デイ」に関する総括は、2018 年 3 月 13 日 - 2018 年 3 月 15 日に早稲田大学にて開催された 情報処理学会 第 80 回全国大会にて、学生セッション 6ZF-06「サーバ構築と運用の実習を通じた中学生対象の実践的な ICT 教育イベントの試み」として口頭発表を実施した。

また、2020 年春時点までの「インデペンデンス・サーバー・デイ」に関する総括は、2020 年 3 月 14 日にオンラインにて開催された 情報処理学会「コンピュータと教育研究会」第 154 回研究発表会にて、学生セッション[3] 「サーバ構築・長期運用の実習を主題とする中学生向け ICT 教育イベントの継続的実践とその成果」として口頭発表を実施し、学生奨励賞を受賞した。情報技術教育に関する研究会において、本イベントの概況、およびその成果を発表できたことは、社会的にも大きな意義のあることであると思われる。

2020 年夏以降の「インデペンデンス・サーバー・デイ」に関しては、2020 年～2022 年までの新型コロナウイルス感染拡大の影響下でのイベント開催形態の変遷、また 2023 年度以降の本イベントの発展・拡大等も踏まえ、今後の学会発表に向けた準備を行っている。

６．謝辞

「インデペンデンス・サーバー・デイ 10」の開催に当たっては、関係各所の多くの皆様のご協力・ご支援を頂きました。「インデペンデンス・サーバー・デイ」を継続して主催して頂いております つくば市総合教育研究所 の皆様、ご後援を頂いた 筑波大学情報科学類 関係教職員の皆様、ご協力を頂いた 筑波大学 OPEN プロジェクトのご担当者の皆様、ソフトイーサ株式会社の皆様、輝日株式会社の皆様に、心より御礼を申し上げます。

また、本講座に当日のスタッフとしてご協力いただいた、産学間連携推進室のメンバーの皆様、筑波大学大学院・筑波大学の有志の皆様、そして BGPNAP 小坂様に、厚く感謝を申し上げます。

DPDK ベース高速ソフトウェアファイアウォールの設計・実装・性能評価

筑波大学情報学群情報科学類産学間連携推進室成果報告書

202520657 服部真吾

2026 年 4 月

1 はじめに

近年、ネットワークの高速化に伴い、100 Gbps 級の回線においてもワイヤレートに近い速度でパケットフィルタリングを行う需要が高まっている。従来のカーネルベースのパケット処理では、システムコールオーバーヘッドやコンテキストスイッチにより、高スループット環境での性能が制限される [1], [2]。 DPDK に代表されるカーネルバイパス型フレームワークは、ポールモードドライバによるユーザ空間パケット処理により、この制限を克服する [3]。

ファイアウォールの設計においては、「何を検査するか」がスループットを根本的に規定する。現在のインターネットトラフィックの 75% 以上が TLS/SSL 暗号化されており [4]、この割合は増加の一途を辿っている。暗号化トラフィックに対して DPI（Deep Packet Inspection）のパターンマッチングは本質的に機能せず、IPS（Intrusion Prevention System）がペイロードを検査する

にはTLSインスペクションが必要となる。しかしTLSインスペクションは計算コストが極めて高く、セキュリティアーキテクチャ内の複数のツールが個別に復号・検査を行う場合、その性能影響は累積的となる [5]。100GbE環境においてはスループットとセキュリティの間に深刻なトレードオフが生じる。暗号化されたペイロードに対してDPI/IPSのためにスループットを犠牲にしても、得られるセキュリティ上の効果は限定的である。この現実を踏まえると、100GbE環境における合理的な設計は、**ヘッダベースの5-tupleフィルタリングと全フローロギング**に特化し、スループットを最大化するアプローチである。5-tupleに基づくACLフィルタリングは暗号化の影響を受けず、フローログは事後的なインシデント分析・フォレンジックの基盤となる。

本稿では、Intel DPDKを用いてConnectX-4 100GbE NIC [6] 上で動作するL2透過型ソフトウェアファイアウォール（dpdk-fw）を設計・実装し、その性能を評価する。主な貢献は以下の通りである：

1. 3種のACLエンジン（rte_aclトライ、ハッシュテーブル、線形走査）の分類性能比較（ルール数10〜100,000）
2. アブレーションスタディによる最適化手法の個別寄与の定量評価
3. UDP小パケットPPS計測により、ACLプレフィックス長・ルール数・フロー数の複合影響を定量評価
4. RSSマルチコアスケーリング（4ワーカで2.27倍）およびダブルバッファリング動的ルール更新の設計・評価

2 関連研究

ソフトウェアによるパケット処理の高速化は、Click [7] のモジュラルータに始まる。Dobrescuら [8] はRouteBricksで汎用x86上35 Gbpsの転送を実証し、Rizzo [9] はnetmapで10 GbEワイヤレートを単一コアで達成した。DPDKはこれらと同様のカーネルバイパスをポールモードドライバで実現し、最高のスループットを達成する [1]。VPP [10] はバッチ（ベクトル）単位処理で命令キャッシュ局所性を最大化する設計であり、本研究も同様のバッチ処理を採用している。

パケット分類はN-tupleマッチング問題として定式化され [11]、HiCuts [12] / HyperCuts [13] 等の決定木、TSS [14] 等のハッシュベース手法が提案されている。本研究が採用するrte_acl [15] はstride 8のマルチビットトライ（DFA）を構築し、ルール数に対してO(1)の分類速度を持つ。

NFVの文脈では、ClickOS [16] が5 MB VMで10 Gbpsを実証し、bpf-iptables [17] がeBPF/XDPでiptablesのスケーリング問題を解決した。ただしXDPベースの処理でもDPDKのような完全なカーネルバイパスには及ばない [2]。Open vSwitch [18] のmegaflow cacheは初回パケットのみ完全分類を行い、以降はキャッシュヒットで高速処理する設計であり、本研究のフローキャッシュもこの思想に基づく。

DDoS防御の文脈では、Gatekeeper [19] がDPDKベースの初のオープンソースDDoS防御システムとして、フロー単位のレート制限とポリシーエンジンを実装し、BGP連携による地理分散配置で100Gbps NICでの本番運用実績を持つ。FastNetMon [20] はsFlow/NetFlowベースの検知に特化し、ミティゲーション自体はBGP Blackholeにより上流ルータに委任する設計である。これらのシステムはフロー識別やポリシー判定に計算資源を投じる設計であるのに対し、本研究はDPI/

IPS を意図的に排除し、5-tuple ACL フィルタリングとフローロギングのみに特化することで、単一サーバ上で最大のスループットを追求する点で設計思想が異なる。

3 設計と実装

3.1 システム構成

3 台のサーバ（cluster-b: 送信、cluster-d: FW、cluster-c: 受信）で構成する。 cluster-d 上の dpdk-fw が VLAN 100/200 タグスワップにより L2 透過転送を行い、ACL ルールにマッチしたパケットを DROP する。

リスト 1: ネットワーク構成

項目	仕様
NIC	Mellanox ConnectX-4 100GbE (mlx5_pci)
CPU	Intel Xeon (2 ソケット, 最大 3.6 GHz)
DPDK	23.11.4
OS	Ubuntu 24.04 LTS (Linux 6.8)
Hugepage	2 GB (1024 × 2 MB)

表 1: ハードウェア・ソフトウェア環境

3.2 パケット処理パイプライン

パケット処理は 64 パケット単位のバースト処理で構成される（図 1)。 VLAN フィルタ → プリフェッチ → フローキャッシュ参照 → ACL バッチ分類（ミス分のみ）→ 結果適用 → フローロギング → VLAN スワップ → TX の順に処理する。

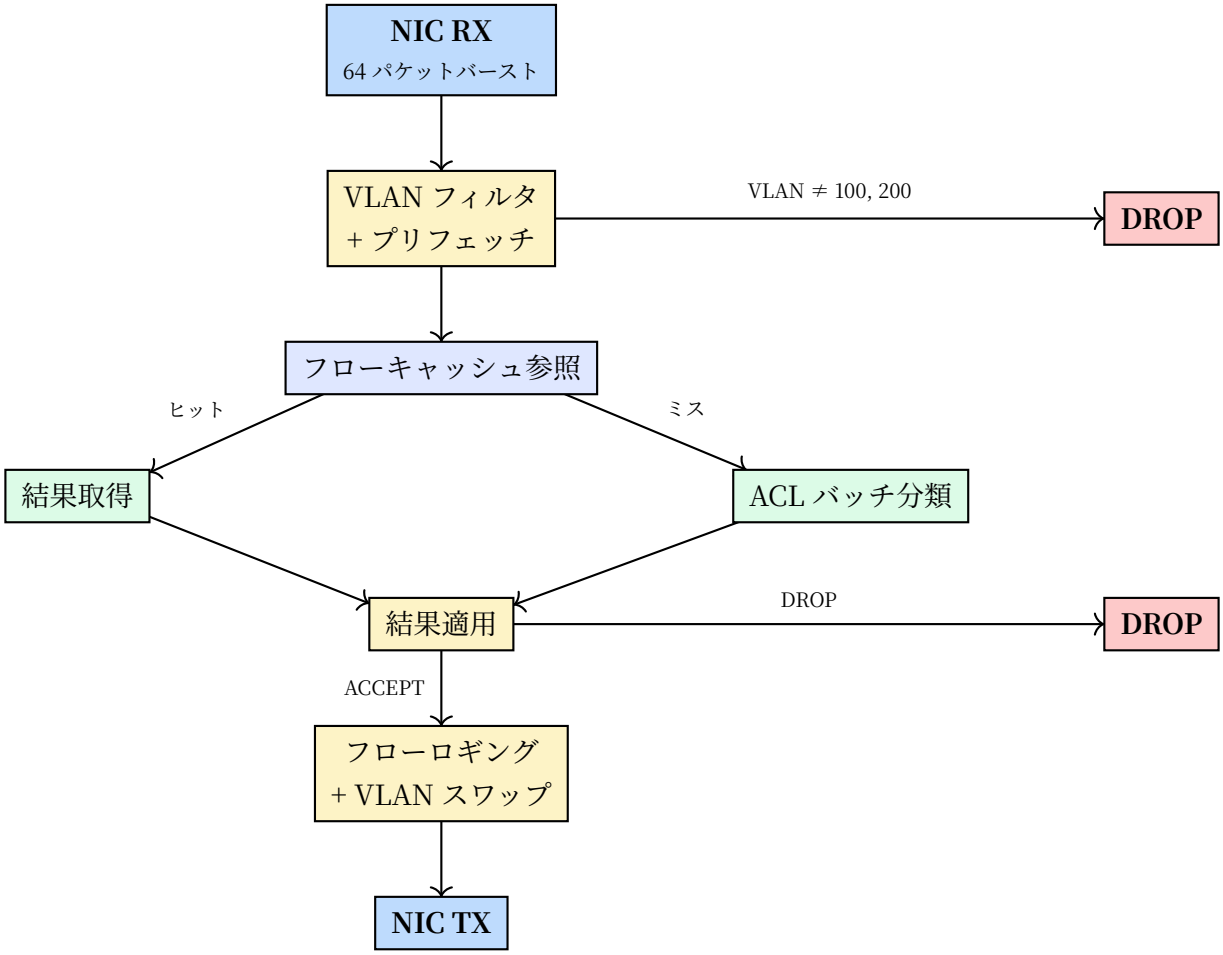


図 1: パケット処理パイプライン

3.3 ACL エンジン

関数ポインタによる Strategy パターンで 3 種の ACL エンジンを切り替え可能とした。

エンジン	計算量	対応ルール	実装
rte_acl	O(1)	prefix, range, wildcard	DPDK トライ + SIMD (scalar/SSE/AVX2)
hash	O(1)	exact-match のみ	rte_hash (Jenkins hash)
linear	O(n)	全形式	優先度降順の逐次走査

表 2: ACL エンジン比較

3.4 フローキャッシュ

Open vSwitch [18] のフローキャッシュ設計を参考に、4-way set-associative 方式（CRC32c SSE4.2 ハッシュ、65,536 セット × 4 エントリ = 262,144 エントリ）を実装した。1 セット = 4 エントリ × 16B = 64B = 1 CPU キャッシュラインに収まる設計である。 2 パスのバッチプリフェッチ（Pass 1: ハッシュ計算+プリフェッチ発行、Pass 2: ルックアップ）により L3 キャッシュミスペナルティを隠蔽する。

3.5 マルチコアとルール更新

RSS（Receive Side Scaling）によるマルチコアパケット処理を実装した。 各ワーカ lcore は独立した RX/TX キュー、mbuf プール、フローテーブル、統計カウンタを保持し、ロックフリーで動作する。

ダブルバッファリング動的ルール更新：C11 `_Atomic` ポインタの `atomic_exchange` により、パケット転送中に ACL コンテキストを無停止でスワップする。ビルド中もワーカは旧コンテキストで転送を継続し、スワップ後はフローキャッシュを一括クリアして新ルールでの再分類を強制する。

4 チューニング

開発過程でシングルストリーム TCP スループット（iperf3 P=1）を 24 Gbps から 32.9 Gbps に改善した。ビルド最適化・CPU ガバナー・NUMA ソケットの 3 項目は開発初期に同時適用したため個別の寄与は分離していないが、これらの合計が改善の大部分を占める。

項目	変更前	変更後	効果
ビルド最適化	`-O0 -g`	`-O3 -flto`	※
CPU ガバナー	`schedutil` (1.9 GHz)	`performance` (3.6 GHz)	※ (周波数比 1.9 倍)
NUMA ソケット	socket 0 (コア 0-1)	socket 1 (コア 8, NIC 同居)	※
EMPW/CQE 圧縮	デフォルト	`txq_mpw_en=2`, `rxq_cqe_comp_en=1`	TX/RX 効率化
バーストサイズ	32	64	バッチ処理効率向上
mempool キャッシュ	128	512	コアローカル高速化

表 3: 主要チューニング項目。※の 3 項目は同時適用（合計 24→32.9 Gbps、個別寄与未分離）

コードレベルでは、`__rte_always_inline` によるホットパスインライン化、`__rte_cache_aligned` によるキャッシュラインアラインメント配置を施した。パケットデータは RX から TX まで mbufポインタのみで受け渡し、コピーを排除した。各ワーカは独立した RX/TX キュー・mbuf プール・フローテーブルを保持し、ワーカ間でロックを用いない設計とした。

5 性能評価

5.1 基本スループットと Linux Bridge 比較

iperf3 による TCP スループットを計測した（30 秒 × 5 トライアル、並列ストリーム P=1/4/16/32）。

設定	P=1	P=4	P=16	P=32
direct（FW なし）	34.9	72.7	72.7	72.7
DPDK FW（100 ルール）	32.9	72.8	72.7	72.8
Linux bridge	8.0	30.1	48.6	51.3

表 4: スループット比較 (Gbps)

DPDK FW は direct 通信と同等のスループットを達成し、Linux bridge 比で約 1.4 倍（P=32）の性能を示した。ACL エンジン種別・ルール数（10〜100,000）・最適化パラメータのいずれも iperf3 スループットに検出可能な影響を与えなかった。これは TCP スタックが律速要因であり、ACL 処理のオーバーヘッドが計測不能なほど小さいことを意味する。

5.2 ACL 分類 PPS

TCP 律速により ACL の性能差がスループットに現れないため、NIC I/O を介さないローカルベンチマーク（`test_acl_perf`）で ACL 分類 PPS を直接計測した。TCP/UDP ヘッダを持つ合成パケットをフロー数分生成し、バッチサイズ 32 で ACL エンジンに繰り返し投入して分類速度を測定する。

ルール数	1 フロー	64 フロー	1K フロー	16K フロー
rte_acl (scalar)				
10	31.8 M	47.1 M	46.7 M	38.3 M
1,000	32.8 M	48.6 M	43.3 M	37.3 M
10,000	33.0 M	48.8 M	48.6 M	39.6 M
linear				
10	57.3 M	49.4 M	29.0 M	21.4 M
1,000	1.2 M	1.1 M	1.1 M	1.1 M
10,000	0.03 M	0.03 M	0.03 M	0.03 M

表 5: ACL 分類 PPS (Mpps) — ルール数 × フロー多様性

rte_acl はルール数に対して O(1)であり、10〜10,000 ルールで 31〜49 Mpps で安定した。linear は O(n)走査のため 10,000 ルールで 0.03 Mpps まで低下した。

5.3 UDP 小パケット PPS 評価

DPDK ベース pktgen により 74B UDP パケット（100 万フロー）を最大レートで送信し、実ネットワーク上の PPS を計測した。

設定	TX Mpps	RX Mpps (定常)	RX Mpps (平均)	Violations
rte_acl 10 ルール	15.73	7.38	5.40	0
rte_acl 1K ルール	16.39	6.79	5.01	0
rte_acl 10K ルール	16.31	6.21	4.58	0
linear 1K ルール	16.18	0.66	0.49	0

表 6: pktgen 計測結果（74B UDP、100 万フロー、30 秒）

全設定で violations=0 であり、高 PPS 負荷下でも ACL 正当性が維持された。 単一コアでの FW 最大処理能力は約 6.9 Mpps（74B UDP）であり、送信レート制御実験により 6 Mpps 以下では転送効率 98% 超を確認した。

5.4 ACL プレフィックス長とルール構造の影響

pktgen（74B UDP、100 万フロー）を用い、プレフィックス長{/16, /20, /24, /32} × ルール数{1K〜50K}を計測した。/16 の 1K ルールで 7.31 Mpps であり、ACL DROP がほぼ発生しないためこの値が単一コアの実質的なパススルー上限に近い。

プレフィックス	1,000	5,000	10,000	50,000
/16	7.31	7.30	7.32	7.33
/20	5.06	5.08	5.09	5.08
/24	6.77	6.33	6.21	5.08
/32	7.07	7.01	6.92	6.63

表 7: プレフィックス長×ルール数 (RX 定常 Mpps)

/16 はルール数 50 倍増でも性能不変（7.3 Mpps）、/24 は 25% 低下（6.8→5.1 Mpps）。50Kルールでは/16 と/24 で 44% の性能差が生じた。/20 ルールを等価な/24 に分割する構造変換は、ルール数が約 250 倍に膨張し 8〜16% の性能低下を招いた。DFA 性能はプレフィックス構造よりルール総数に強く依存する。

5.5 フロー数×ルール数の複合影響

pktgen（74B UDP）でフロー数{100〜1M} × ルール数{10〜10K}の 20 組み合わせを計測した。

フロー数	r=10	r=100	r=1K	r=10K
1,000	7.04	7.18	7.47	7.58
10,000	7.42	7.46	7.94	7.19
100,000	7.87	7.72	7.34	6.67
1,000,000	7.35	7.27	6.80	6.23

表 8: フロー数×ルール数マトリクス (RX 定常 Mpps)

フローキャッシュはフロー数 100K 以下で有効に機能し、ルール数の影響を大幅に緩和する。100 万フロー環境ではルール数 10K で r=10 に対して 15% の性能低下が生じた。

5.6 ACL DROP/ACCEPT 正当性評価

5 サブネット 10IP アドレスを用い、8 種のテストケースで 52 件の個別検証を実施した。全検証でクライアント側（ping/iperf3）の出力に基づき期待通りの結果を確認した。

テスト	シナリオ	検証内容	検証数	結果
T1	全 ACCEPT	5 サブネット＋クロス＋逆方向の全通過	9	9/9
T2	ホスト DROP	10.0.0.10/32 のみ遮断、他サブネット影響なし	6	6/6
T3	サブネット DROP	10.1.0.0/24 遮断、宛先マッチも確認	6	6/6
T4	宛先ホスト DROP	10.0.0.20/32 宛の異サブネットからの遮断	5	5/5
T5	優先度オーバーライド	高優先 ACCEPT が低優先 DROP を上書き	7	7/7
T6	プロトコル別制御	ICMP 通過・TCP 5201 遮断の共存	4	4/4
T7	複合ルール	/16・/24・/32・ポート遮断の同時適用	8	8/8
T8	全 DROP	5 サブネット＋逆方向の全遮断	7	7/7

表 9: ACL DROP/ACCEPT テスト結果（全 52 件 PASS）

5.7 マルチコアスケーリング

RSS（Receive Side Scaling）によるマルチキュー分散を実装した。送信側は単一コア pktgen で約 15.5 Mpps（74B UDP、100 万フロー）を最大レートで送出する。初期実装では NIC TX パスの DMA 読み出し競合により 2 ワーカで性能が 20% 悪化したが、mlx5 ドライバの `txq_inline_min=128` による TX WQE インライン化で解決した。

構成	FW RX	FW TX	NIC imissed	SW Drop	Receiver
1 worker	7.1 Mpps	7.0 Mpps	278M	0	7.07 Mpps
2 workers (inline)	14.2 Mpps	8.6 Mpps	44.8M	163M	8.89 Mpps
4 workers (inline)	16.5 Mpps	15.9 Mpps	5.9M	0	15.30 Mpps

表 10: マルチコアスケーリング結果（送信側 15.5 Mpps、74B UDP、1K ルール、30 秒）

1 ワーカ構成では単一 RX キューの NIC 受信能力が約 7.1 Mpps に制限され、送信側 15.5 Mpps との差分（約 278M パケット）は NIC ハードウェアが破棄する（imissed）。FW ソフトウェアには 7.1 Mpps しか到達しないため、TX 7.0 Mpps で処理可能であり SW Drop=0 となる。

2 ワーカ構成では 2 RX キューにより NIC 受信能力が 14.2 Mpps に向上するが、TX 処理能力（8.6 Mpps）が追いつかず、差分の約 5.5 Mpps × 30 秒 ≈ 163M パケットが FW ソフトウェア側で廃棄される。

4 ワーカ構成では 4 キュー合計の TX 処理能力（15.9 Mpps）が送信側のレート（約 16.5 Mpps）とほぼ均衡し、SW Drop=0 を達成した（単一コア比 2.27 倍）。送信側をマルチコア化すれば FW 側にさらに負荷をかけることが可能であり、4 ワーカの上限は未到達である。

2 ワーカでの SW Drop 発生は、NIC のマルチキュー TX におけるキュー当たり処理能力の低下に起因する。74B の小パケットでは、TX キュー数の増加に伴い NIC 内部の WQE 処理や Completion Queue アクセスの競合が増加し、キュー当たりの TX 処理能力が低下する。

構成	TX キュー数	キュー当たり TX	合計 TX
1 worker	1	7.0 Mpps	7.0 Mpps
2 workers	2	4.3 Mpps	8.6 Mpps
4 workers	4	4.0 Mpps	15.9 Mpps

表 11: TX キュー数とキュー当たり処理能力（`txq_inline_min=128` 適用済み）

1 キューから 2 キューへの増加でキュー当たり TX 処理能力が 39% 低下（7.0→4.3 Mpps）し、合計 TX も RX の 14.2 Mpps に追いつかない。4 キューではキュー当たり 43% 低下（7.0→4.0 Mpps）だが、合計 15.9 Mpps で送信側とほぼ均衡する。再実験（同一条件、別日実施）でもキュー当たり 4.1 Mpps（2 キュー）、3.9 Mpps（4 キュー）と一致する結果が得られ、この傾向が再現性のある NIC ハードウェア特性であることを確認した。

5.8 動的ルール更新

更新内容	更新前 PPS	ビルド時間	PPS 低下	violation	更新後 PPS
100 → 1,000	7.3 Mpps	26.7 ms	7% (1 秒)	0	6.9 Mpps
1,000 → 1,010	6.9 Mpps	30.7 ms	4% (1 秒)	0	6.9 Mpps
1,000 → 10,000	6.9 Mpps	231 ms	26% (1 秒)	0	6.7 Mpps

表 12: 動的ルール更新の PPS 影響（74B UDP、100 万フロー）

全ケースで更新後の violation 増加はゼロであり、ダブルバッファリングによる ACL スワップの正当性を確認した。

5.9 アブレーションスタディ

ベースライン（全最適化 ON、フローキャッシュ OFF）から各最適化を 1 つずつ無効化し、個別の寄与を定量評価した（1 ワーカ、1K ルール、100 万フロー、74B UDP）。

構成	TX (Mpps)	差分
全最適化 ON（ベースライン）	6.76	—
バッチ ACL OFF	5.13	−24.1%
プリフェッチ OFF	6.76	−0.0%
フローキャッシュ追加 (※)	6.63	−1.9%
TX WQE インライン OFF	7.02	+3.8%
全最適化 OFF	4.66	−31.1%

表 13: アブレーションスタディ結果（主要構成のみ）

バッチ ACL 分類が最大の寄与 (-24.1%) を持つ。※ フローキャッシュはベースラインでは OFF である。rte_acl の O(1)分類が十分高速なため、キャッシュを追加すると逆にオーバーヘッドで 1.9% 低下する。 TX WQE インラインは単一ワーカでは逆効果（+3.8%）だがマルチワーカでは不可欠であり、ワーカ数依存の最適化特性を発見した。

6 考察

6.1 TCP 律速と PPS 計測の必要性

iperf3 による評価（計 73 設定）では全設定で P=32 時に約 72 Gbps で飽和し、ACL 性能差が検出されなかった。 これは TCP スタックが律速要因であり、ACL 分類性能の評価には PPS 計測が不可欠であることを示す。

6.2 フローキャッシュの限界

フローキャッシュを 4-way set-associative + CRC32c に全面改修し、ヒット率は 0%→16% に改善したが、スループットは±0.7% の範囲に留まり改善が見られなかった。 rte_acl の O(1)分類が十分高速であり、NIC/PCIe 律速環境ではキャッシュの効果が顕在化しない。フローキャッシュのオーバーヘッドが ACL バイパスの恩恵を上回り、3% の性能低下を招く結果となった。

6.3 マルチコアと NIC TX 最適化

74B 小パケットでのマルチコアスケーリングでは、NIC の DMA 読み出し競合がボトルネックとなった。`txq_inline_min=128` による TX WQE インライン化が不可欠であり、ソフトウェア最適化だけでなく NIC ドライバの devarg 調整の重要性を実証した。

6.4 実運用への示唆

74B UDP パケット 6.9 Mpps は約 4.1 Gbps 相当であるが、ジャンボフレーム（9,000B MTU）の TCP トラフィックでは同一 Mpps で約 500 Gbps 相当に対応する。iperf3 実験で 72 Gbps を達成したことと合わせ、実用的な TCP トラフィック環境ではパケットロスなく動作する処理能力を有している。

7 おわりに

本稿では、DPDK ベースの L2 透過型ソフトウェアファイアウォール（dpdk-fw）を設計・実装し、ConnectX-4 100GbE 環境で性能評価を行った。TLS/SSL 暗号化が主流となった現在、ヘッダベースの 5-tuple フィルタリングと全フローロギングに特化したアプローチは、スループットとセキュリティ可視性を両立する合理的な選択である。 主な知見は以下の通りである：

1. **ACL 処理は 100GbE 環境でボトルネックにならない**：rte_acl はルール数に対して O(1)であり、10〜10,000 ルールで 31〜49 Mpps で安定
2. **ACL ルールは正確に動作する**：8 テスト・52 件の検証が全て PASS
3. **DPDK FW は Linux bridge 比で約 1.4 倍**（P=32 で 72.9 vs 51.3 Gbps）
4. **バッチ ACL が最大の最適化効果**（-24.1%）、プリフェッチは 74B パケットでは効果なし
5. **プレフィックス長が DFA スケーリングを決定**：/16 は 50K ルールでも性能不変、/24 は 25% 低下
6. **4 ワーカで 2.27 倍のスケーリング**：TX WQE インライン化が不可欠
7. **ダブルバッファリング動的ルール更新**：violation 0 で無停止切替、1K ルールは 27 ms で完了

今後の課題として、送信側マルチコア化による 4 ワーカ以上のスケーリング検証、ConnectX-5 以降での TX インライン自動最適化、IPv6 ACL サポートが挙げられる。 また、ftp.tsukuba.wide.ad.jp 等におけるアクセスコントローラ・ロガーとしての実践投入を計画しており、実トラフィック環境での長期運用評価を行う予定である。

参考文献

[1] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, と G. Carle, 「Comparison of Frameworks for High-Performance Packet IO」, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), IEEE, 2015, pp. 29–38. [Online]. 入手先: https://www.net.in.tum.de/fileadmin/bibtex/publications/papers/gallenmueller_ancs2015.pdf

[2] D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak, と G. Carle, 「Performance Implications of Packet Filtering with Linux eBPF」, 30th International Teletraffic

Congress (ITC 30), IEEE, 2018, pp. 209–217. [Online]. 入手先: https://www.net.in.tum.de/fileadmin/bibtex/publications/papers/ITC30-Packet-Filtering-eBPF-XDP.pdf

[3] DPDK Project, 「DPDK Programmer's Guide: Overview」. 参照: 2026 年 2 月 17 日. [Online]. 入手先: https://doc.dpdk.org/guides/prog_guide/overview.html

[4] E. Papadogiannaki と S. Ioannidis,「Acceleration of Intrusion Detection in Encrypted Network Traffic Using Heterogeneous Hardware」, Sensors, vol. 21, no. 4, p. 1140, 2021, doi: 10.3390/s21041140.

[5] Cato Networks, 「Traditional Firewalls Can't Keep Up with the Growth of Encrypted Traffic」. 参照: 2026 年 2 月 17 日. [Online]. 入手先: https://www.catonetworks.com/blog/traditional-firewalls-cant-keep-up-with-the-growth-of-encrypted-traffic/

[6] NVIDIA/Mellanox Technologies,「ConnectX-4 Lx EN Adapter Card — Product Brief」. 参照: 2026 年 2 月 17 日. [Online]. 入手先: https://network.nvidia.com/files/doc-2020/pb-connectx-4-lx-en-card.pdf

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, と M. F. Kaashoek, 「The Click Modular Router」, ACM Transactions on Computer Systems, vol. 18, no. 3, pp. 263–297, 2000, [Online]. 入手先: https://pdos.csail.mit.edu/papers/click:tocs00/paper.pdf

[8] M. Dobrescu ほか, 「RouteBricks: Exploiting Parallelism to Scale Software Routers」, 22nd ACM Symposium on Operating Systems Principles (SOSP 09), ACM, 2009, pp. 15–28. [Online]. 入手先: https://www.sigops.org/s/conferences/sosp/2009/papers/dobrescu-sosp09.pdf

[9] L. Rizzo, 「netmap: A Novel Framework for Fast Packet I/O」, USENIX Annual Technical Conference (ATC 12), USENIX Association, 2012, pp. 101–112. [Online]. 入手先: https://www.usenix.org/system/files/conference/atc12/atc12-final186.pdf

[10] L. Linguaglossa, D. Rossi, S. Pontarelli, D. Barach, D. Marjon, と P. Pfister, 「High-Speed Data Plane and Network Functions Virtualization by Vectorizing Packet Processing」, Computer Networks, vol. 149, pp. 187–199, 2019, [Online]. 入手先: https://nonsns.github.io/paper/rossi19comnet.pdf

[11] P. Gupta と N. McKeown, 「Algorithms for Packet Classification」, IEEE Network, vol. 15, no. 2, pp. 24–32, 2001, [Online]. 入手先: https://cse.sc.edu/~srihari/reflib/GuptaIN01.pdf

[12] P. Gupta と N. McKeown,「Classifying Packets with Hierarchical Intelligent Cuttings」, IEEE Micro, vol. 20, no. 1, pp. 34–41, 2000, [Online]. 入手先: https://ieeexplore.ieee.org/document/820051/

[13] S. Singh, F. Baboescu, G. Varghese, と J. Wang, 「Packet Classification Using Multidimensional Cutting」, Proceedings of ACM SIGCOMM, ACM, 2003, pp. 213–224. [Online]. 入手先: https://cseweb.ucsd.edu/~susingh/papers/hyp-sigcomm03.pdf

[14] V. Srinivasan, G. Varghese, S. Suri, と M. Waldvogel, 「Packet Classification Using Tuple Space Search」, Proceedings of ACM SIGCOMM, ACM, 1999, pp. 135–146. [Online]. 入手先: http://yuba.stanford.edu/~nickm/papers/Sigcomm99.pdf

[15] DPDK Project, 「Packet Classification and Access Control — DPDK Programmer's Guide」. 参照: 2026 年 2 月 17 日. [Online]. 入手先: https://doc.dpdk.org/guides/prog_guide/packet_classif_access_ctrl.html

[16] J. Martins ほか, 「ClickOS and the Art of Network Function Virtualization」, 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), USENIX Association, 2014, pp. 459–473. [Online]. 入手先: https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-martins.pdf

[17] S. Miano, M. Bertrone, F. Risso, M. V. Bernal, Y. Lu, と J. Pi, 「Securing Linux with a Faster and Scalable Iptables」, ACM SIGCOMM Computer Communication Review, vol. 49, no. 3, pp. 2–17, 2019, [Online]. 入手先: https://dl.acm.org/doi/10.1145/3371927.3371929

[18] B. Pfaff ほか, 「The Design and Implementation of Open vSwitch」, 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), USENIX Association, 2015, pp. 117–130. [Online]. 入手先: https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf

[19] C. Doucette, Q. Chen, と J. M. Smith,「Gatekeeper: A High-Performance DDoS Defense System using Programmable Packet Processing」, Proceedings of the 2018 Symposium on SDN Research (SOSR '18), ACM, 2018, pp. 1–2. doi: 10.1145/3185467.3190972.

[20] P. Odintsov,「FastNetMon: Very Fast DDoS Analyzer with sFlow/NetFlow/IPFIX/SPAN Support」. 参照: 2026 年 4 月 21 日. [Online]. 入手先: https://github.com/pavel-odintsov/fastnetmon

判決文 PDF ファイルの本文領域抽出モデル構築用データセットの作成

筑波大学 情報学群 情報科学類 産学間連携推進室
情報理工学位プログラム 202620608 北野尚樹 [s2620608@u.tsukuba.ac.jp]

1 はじめに

前年度まで「法律文書の自動解析」として法令・条例・判決文などの法律文書に対して読み替え規定文や略称定義文、他の法令などへの参照表現について抽出するプロジェクトに取り組んでおり、特に法令や条例に対して重点的に取り組んでいた。本年度は、今までターゲットとしつつもあまり取り組むことができていなかった判決文について取り組んだ。

2 判決・判決文・判例とは

判決とは、裁判所で行われている個々の裁判の判断結果であり、判決文とはその判断が書かれた文書のことである。

判決文は主に、主文と事実と理由から構成される。主文とは、判決の要である「本件控訴を棄却する」などの判断結果である。事実は、訴訟の際に提起された様々な証拠等のうち、判決を出すにあたって整理され裁判所によって事実と認定されたものごとである。理由は、事実をもとに主文の内容に至った裁判所の判断理由である。

これらの判決のうち、最高裁判所が出したものなどが判例となり、その後の裁判や法律実務において参考にされる[*1]。

3 判例の重要性と既存の判例集

法律実務においては法令の条文だけではなく判例も重要な資料として用いられる。

例えば法律実務で取り組む事例とその事例に関連する条文について、似た事例についてすでに判例がある場合、実際に裁判を起こしたとしても下級裁判所もその判例と同じように条文を解釈して事例に適用する可能性が高くなる。そのため、裁判を起こして裁判所の判断を仰ぐことなくこの判例を参考にして実務を進めるのが確実で早いということとなる。

このように、判例には結果の予測可能性を高めるという性質や条文の解釈の仕方を示すものという性質がある。そのため、判例は法律実務において欠かすことのできない重要なものである。

このような実務上の需要に応えるために、判例を収集したものが複数ある。

---

*1 地裁や高裁が出した判決であっても、その判決に対して不服として行われた上告が最高裁判所によって棄却された場合は、実質的に地裁や高裁の判決が判例として扱われうる。

古くは判例タイムズ社が 1948 年から発行している判例タイムズという雑誌や判例時報社が 1953 年から発行している判例時報という雑誌が有名である。これらは最高裁を始めとした将来判例として実務で参考にされそうな重要な判決を解説とともに紹介するものである。

近年では電子的な判例データベースも登場しており、前述の判例タイムズのアーカイブや、Westlaw などの商用判例データベース、各出版社が内部で保有している判例データベースがある。また、裁判所も裁判所自身が重要であると判断した判決文を公開している[*2]。

4 判決データベースの需要の高まりと民事裁判情報の活用の促進に関する法律の施行

近年の生成 AI の発展および法律実務のデジタル化に伴い、大規模な判決データベースの需要が急速に高まっている。前述の判例データベースは弁護士や会社の法務部や大学が契約し、人が自力で検索して読むことを前提とした PDF のスキャンデータなどである。

しかし、現在需要が高まっているものは生成 AI をファインチューニングしたりプロンプトに与えて出力結果の精度を向上させるためのものや、関連する判例を自動で検索するための検索モデルの精度向上を目的とするものである。そのため、従来のものよりも取り扱いのしやすいテキストファイルで量も膨大なものである。

この需要の食い違いについて国も認識しており、リーガルテック業界の働かけにより、2026 年 1 月に民事裁判情報の活用の促進に関する法律（令和七年法律第四十九号）が施行され、民事裁判の判決文がテキストファイルで入手できるデータベースを国の責務で整備するための制度が用意されるという解決のための進展が見られた。

なお、この判決データベースは主にリーガルテック業界の民間業者が商用のために利用することが想定されるものであり、また、データベースの整備のためには判決文中の人物等について匿名化処理を行う必要があるなどコストがかかることから、制度維持のためにデータは有償で提供される方針となっている。

5 オープンな判決データベース整備の重要性

前述のとおり、民事裁判情報の活用の促進に関する法律によって整備される予定の判決データベースは有償利用であり、既存の民間判決データベースも利用は有償である。

しかし、本来であれば法令の内容がどのように解釈され運用されるのかを国民が把握できるようにするのが法治国家のあるべき姿であり、そのためには判例に国民が容易にアクセスできる環境があるべきである。

現状、最高裁判所のサイトで運用されている裁判例検索がそれに最も近い環境となっている。しかし、このサイトでは重要な判決文を PDF ファイルで閲覧することができるだけであり、収録されている大量の判決がもたらす判例としての価値は十分に提供できていない。

そのため、これらの大量の重要な判決文を収録して国民が自由に分析することができるオープンなデータベースを整備することが重要である。

---

*2　https://www.courts.go.jp/hanrei/index.html

6 オープンな判決データベース整備に向けた取り組み

前年度まで「法律文書の自動解析」プロジェクトの一つとして、2023 年に「判例の自動収集およびテキストデータ化」としてオープンな判決データベースの整備に取り組んだ [*3]。この取り組みで最高裁判所が公開している判決情報をスクレイピングで取得するソフトウェアと得られた判決文のテキストファイルはすべて GitHub でオープンなライセンスのもと公開している。なお、判決文は著作権法 13 条第 3 号の規定により著作物とならないため、判決文のテキストファイルを再配布しても問題は無い。

- 判決情報スクレイピングソフトウェア（MIT ライセンス）：https://github.com/japanese-law-analysis/listup_precedent
- 判決文テキストファイル（CC0 ライセンス）：https://github.com/japanese-law-analysis/data_set/tree/master/precedent

2023 年に公開した際に本データベースは一定の反響があり、リポジトリに star が 52 個ついている他、複数の方から利用した旨の報告があり、目的とした国民による法情報の自由な活用に資することができていると評価している。

しかし、2023 年に作成した本データベースには、判決文の PDF ファイルをテキストファイルに変換するプログラムの精度が低いという課題を抱えていた。そこで、本年からはこの問題を解消し、より万全なデータベースを構築するための取り組みを始めた。

7 判決文 PDF ファイルの難しさ

最高裁判所の公開している判決文 PDF ファイルを機械可読な形に変換するに当たり、主に 2 つの難しさがあった。

7.1 行番号

判決文は判決を出した裁判所ごとにフォーマットが少しずつ異なる。そのフォーマットのうち、行番号を入れるものが存在しており、少なくない判決において行番号が見られる（図1）。この行番号を、見出しや箇条書きを表す行頭の数字と間違えずに除去することは難しく、度々処理に失敗していた。

7.2 画像の取り扱い

知財関連の判決にはまれに画像が含まれることがある（図2）。従来の変換の際はこれらの画像の情報は完全に欠落しているが、機械学習などの用途にデータベースを用いる場合に備えて`<image>`などのマスクデータに置換しておくことが望ましいと考える。

8 判決文 PDF ファイルの本文領域抽出モデルの構築

前節で述べた判決文 PDF ファイル特有の難しい問題に対処して機械可読な形式に変換するために、本プロジェクトでは判決文 PDF ファイル中の本文領域や画像領域を認識す

---

*3 https://www.ac-room.org/public/%E6%88%90%E6%9E%9C%E5%A0%B1%E5%91%8A%E6%9B%B820240526.pdf

るモデルを作成し、そのモデルを用いて本文領域だけの文章を OCR 技術でテキストファ

令和 8 年 2 月 5 日判決言渡　同日原本領収　裁判所書記官

令和 7 年（ワ）第 70136 号　著作権侵害による不法行為　損害賠償請求事件

口頭弁論終結日　令和 7 年 12 月 4 日

判　　決

当事者の表示　　別紙当事者目録記載のとおり

主　　文

1　原告の請求をいずれも棄却する。

2　訴訟費用は原告の負担とする。

事 実 及 び 理 由

第 1　請求

1　被告らは、別紙著作物目録記載 1～4 のデザインを複製し、同デザインを利用した物品を所有する関ケ原町町長以下執行部等の頒布を防ぐ監督・確認を行え。

2　被告らは、別紙著作物目録記載 1～4 のデザインを利用した物品を所有する関ケ原町町長以下執行部等の破棄が完了するまで監督・確認を行え。

3　被告らは、連帯して、20 万円及びこれに対する令和 5 年 10 月 26 日から支払済みまで年 3%の割合による金員を支払え。

4　被告らは、関ケ原町町長以下執行部等の「印刷業者」を介した架空支払等「マネーロンダリング」の可能性に相当する不法行為の監督・管理を行え。

第 2　事案の概要

1　本件は、原告が、被告らの公権力の行使によって原告の著作権が侵害されたと主張して、被告らに対し、国家賠償法（以下「国賠法」という。）1 条 1 項に基づき、20 万円の損害賠償及びこれに対する令和 5 年 10 月 26 日から支払済みまで民法所定の年 3%の割合による遅延損害金の支払を求める（請求 3）と共に、その損害賠償を遂行するために必要な行為であると主張して、別紙著作物目録記載 1～4 の各デザイン（以下「本件各デザイン」という。なお、甲 1 の別紙著作物目録記載 2～5 と同じものである。）を関ケ原町（以下「町」ともいう。）の

1

図 1：行番号を含む判決文の例


（別紙）

引用商標

阿夫利大山

5　指定商品：

第３３類「日本酒、洋酒、果実酒、中国酒、薬味酒」

図２：画像を含む判決文の例（令和７年（行ケ）第１００６６号 審決取消請求事件より）

イルに変換する手法を取ることにした。

本文領域抽出モデルはYOLOなどの物体検出モデルを応用して作成することとした。判決文の１ページごとを画像化したものを入力とし、本文領域が行番号領域、見出し領域などを判定して出力するものである。

本年度では、図3・4のように、アノテーション用ソフトウェアを作成し、モデル構築のための学習データを作成している途中である。

本アノテーションソフトウェアでは手動で矩形と対応するラベルを定め、 最終的にYOLOで学習するための次のようなデータを生成する機能を持つ。

```
1 0.159836 0.092186 0.086885 0.017394
0 0.277049 0.116537 0.318033 0.026670
1 0.164754 0.134510 0.086885 0.016234
0 0.495902 0.221477 0.749180 0.162339
1 0.163115 0.307865 0.080328 0.015074
0 0.495082 0.366423 0.747541 0.097404
```

これは一番左から順に次の意味を持つ。

- ラベルの区別
  - `0`：本文領域
  - `1`：見出し領域
  - `2`：ページ番号領域
  - `3`：行番号領域
  - `4`：画像領域

    - 5：表領域
    - 6：ヘッダー
    - 7：フッター
- 矩形の中心位置の x 座標
- 矩形の中心位置の y 座標
- 矩形の横幅
- 矩形の縦幅

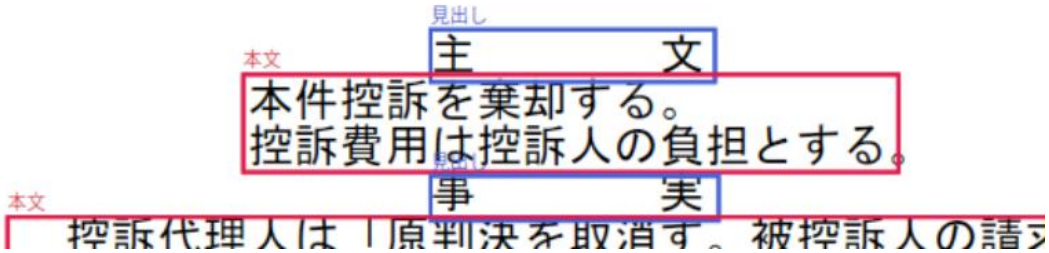


図 4：アノテーション結果

9 おわりに

本年度では、 オープンな判決データベース作成を最終的な目標として、 判決文 PDF ファイルの本文領域抽出モデル構築用データセットを作成した。 現時点ではモデルを構築するために十分な量を確保できていないため、 引き続きデータセットを充実させるために作業を進めていく。

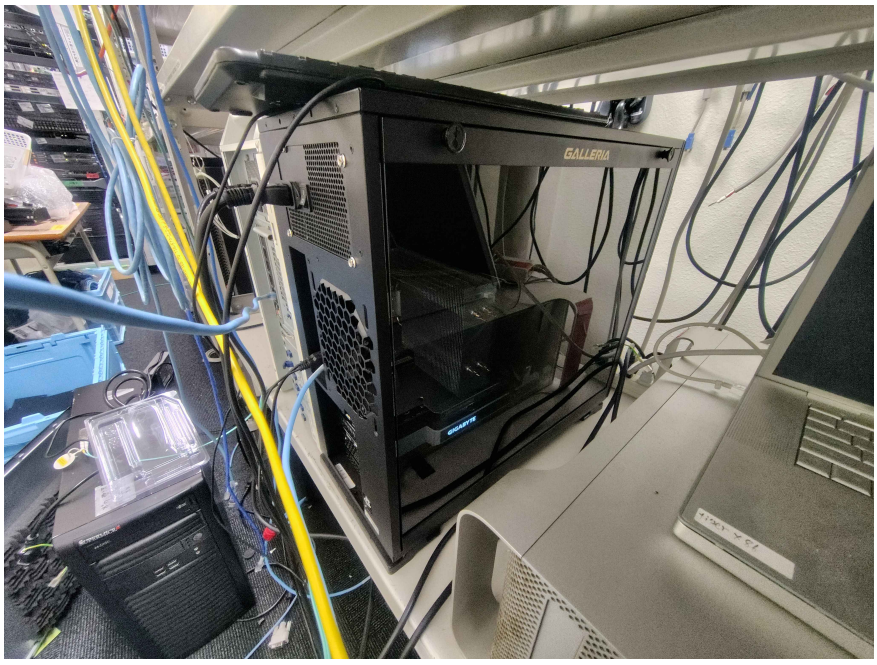


図 5：産学間連携推進室のサーバールーム内に設置している GPU サーバー

図３：判決文 PDF ファイルの本文領域抽出モデルのためのアノテーションソフトウェア

Face 等で公開することを考えている。モデルの訓練には産学間連携室内のサーバールーム内に設置されている GPU サーバー（図5）で行うことを考えている。

イベント型店舗運営に適用可能なカスタマイズ性を有する販売管理システムの開発

情報学群 情報科学類 中野 あおい（s2312794@u.tsukuba.ac.jp）

2026 年 4 月 25 日

概要

本報告書では, 注文管理, 会計処理, レジ・提供口連携, ならびにリアルタイム状態共有を目的として開発している POS アプリケーションについて述べる. 本システムは, フロントエンドに React Router ベースの Web UI, バックエンドに Go / Gin, データベースに PostgreSQL を用い, 注文情報や運用状態を WebSocket を通じて各画面へ即時反映する構成をとる. 本アプリケーションは, 所属サークルの学園祭店舗向けに共同開発された既存リポジトリを出発点とし, その構成や知見を継承しながら, 特定運用に依存する部分を整理して一般化することを目的として再構成を進めている. また, 使用者がシステムをカスタマイズしやすいように, ドキュメントを整備することも目指している. 本稿では, 実装方法, 先行研究との関連, 現時点での進捗, および今後の展望について述べる.

1 はじめに

POS（Point of Sale）システムとは, 商品の販売や注文の発生時点において, その内容を記録し, 会計処理や売上管理, 注文管理へと接続する情報システムである. 飲食店や小売店では, 単なるレジ機能にとどまらず, 注文情報の共有, 在庫把握, 提供状況の管理, 売上集計など, 店舗運営を支える基盤として利用されている. 特に飲食提供の現場では, 注文受付, 会計, 調理・提供の各工程が密接に関係しており, POS システムの設計は業務効率やオペレーションの正確性に大きく影響する.

近年, 飲食物の提供を伴うイベントや小規模店舗において, 現場の運用に即した柔軟な注文・会計システムが求められている. 一般的な POS システムは多機能である一方, 模擬店等の運用形態に対して過剰であったり, システムを起動する手順が複雑であったり, 画面構成や操作フローの変更が難しい場合がある. そのため, 現場ごとの役割分担や注文処理手順に合わせた専用システムを構築する意義は大きい.

本アプリケーションは, 注文受付, 会計, 提供管理, 運用状態共有を支援する POS システムとして開発を進めている. もともとは, 所属サークルの学園祭店舗向けに共同開発された既存リ

ポジトリ[1] が存在しており, 本研究ではそのリポジトリをフォークしたうえで, 特定の店舗運用や一時的な要件に依存していた実装を見直し, より汎用的に利用できる構成へと整理・再設計している.

本システムでは, 運用環境としてラップトップ PC とタブレット端末を用い, できるだけ小規模な構成で完結することを目指している. 具体的には, ラップトップ PC を Docker ホスト兼サーバとして用い, 同一ローカルネットワーク内のタブレットまたは別端末からアクセスする構成を想定している. このとき, 外部クラウドや既存店舗ネットワークへの依存を減らし, 周囲のネットワーク環境の影響を受けにくい, 自己完結的な運用を実現することが重要である.

したがって, 本研究は既存システムの実践的知見を継承しながら, アーキテクチャの再編, 機能の整理, UI の改善を通じて, 一般化された POS システムへ発展させる試みである. システムの使用者が自分の運用に合わせてカスタマイズしやすいように, ドキュメントの整備やコードの構造化も重要な目標としている. また, 本ソースコードを GitHub 上で公開することで, 同様のニーズを持つ他の現場や開発者が参考にできるようにすることも目指している.

本報告書では, 関連する先行研究, 実装方法, 現時点での成果, そして今後の課題について整理する.

2 先行研究

注文管理や会計支援を目的とした POS システムは, 商用システムに加え, 学生団体や小規模運営組織によって独自に開発された事例も多い. 特に, 学園祭店舗やイベント出店のような短期的・限定的な運用においては, 自らの業務に適した POS を内製する試みがしばしば見られる. また, 不特定多数の利用者を想定した汎用的な Web 型 POS も存在し, ブラウザ上で利用可能な手軽さを特徴としている.

一方で, これら既存システムには課題もある. 一部のシステムでは専用機器や特殊な周辺機器を必要とし, 導入や構築の負担が大きい. また, 汎用的な Web サービス型の POS は利用開始が容易である反面, ログインを前提とした利用形態や, 画面遷移・入力方式・役割分担などが固定的であり, 利用者側で柔軟に変更しにくい場合が多い. そのため, 現場ごとに異なる運用手順や人的配置を反映したシステムとしては不十分である場合がある.

[1] https://github.com/cafeore-tkb/cafeore-pos/

そこで本研究では, ノート PC とタブレット端末程度の汎用機器のみで構成でき, ローカルネットワーク内で完結して運用可能であり, なおかつ現場ごとの要求に応じて UI や操作導線を調整しやすい POS システムの実現を目指す. これは, 導入容易性とカスタマイズ性を両立させることで, 既存の学生開発 POS や汎用 Web 型 POS の課題を補完する試みとして位置付けられる.

3 実装方法

3.1 システム全体構成

本システムは, フロントエンド, バックエンド, データベース, リアルタイム通信機構から構成される. フロントエンドは React Router を用いた Web アプリケーションとして実装し, バックエンドは Go 言語と Gin フレームワークを用いて REST API および WebSocket エンドポイントを提供する. データベースには PostgreSQL を利用している.

利用者の役割としては, 主にキャッシャー画面と提供画面が存在する. キャッシャー画面では商品の追加, 割引適用, 受領額入力, 注文送信を行う. 提供画面では注文状況の確認や客の呼び出し, 提供状態の切替を行う. これらの画面間では, 注文情報を WebSocket 経由で同期し, 更新内容を即時に反映する.

本システムの基盤には, 学園祭店舗向けに共同開発された既存リポジトリを用いている. しかし, そのまま流用するのではなく, 所属サークル依存の特定の構成や実装を見直し, 役割ごとの責務分離, 共通モジュールの整理, API と画面ロジックの再設計を行った. これにより, 元リポジトリで得られた知見を活かしつつ, 別の運用環境にも適用しやすい構造を目指している. 既存リポジトリに機能としてあった「オーダーストップ機能」や「ラベルプリンタとの連携機能」, 「ダッシュボード画面」についても, ON / Off の切り替えで使用できるようにしたいと考えている.

また, 運用構成としては, ラップトップ PC 上でアプリケーションサーバとデータベース, フロントエンド開発環境を動作させ, 同一ネットワーク内の別端末から利用する形を想定している. このため, サーバ機能をラップトップ PC 単体で十分に提供できるよう, 構成や処理負荷を抑えた軽量な実装が求められる. したがって, 本研究では機能追加だけでなく, 限られた計算資源で安定運用できる構成の整理も設計上の重要課題としている.

3.2 フロントエンド実装

フロントエンドでは, React の関数コンポーネントとフックを用いて画面を構成している. 商品の一覧やマスターデータの取得には SWR ベースのキャッシュを採用し, 同一データへの重複アクセスを抑えつつ, 複数コンポーネント間での共有を可能にしている.

キャッシャー画面については, フォーク元リポジトリ従来のキーボード中心の操作フローから, タッチ操作を前提とした UI への移行を進めている. 具体的には, 商品を大きなボタンとして分類表示し, 注文中の商品一覧を同画面内に常時表示することで, 操作対象を直感的に把握できる構成を採用している. また, 備考, 割引番号, 受領額, 会計確認といった操作を, 少ない画面遷移で完了できるよう調整している. また, フォーク元リポジトリの機能としてあったラベルプリンタ連携機能も, 使用可能である.

3.3 バックエンド実装

バックエンドでは, 注文作成, 商品・商品種別追加・取得, コメント追加, 商品提供ステータス更新などの API を実装している. 注文登録時には, リクエスト内容をバリデーションした後, 注文本体, 注文商品, コメント情報を DB に保存する. その後, 保存された最新データを読み直してレスポンスとして返却し, 加えて WebSocket ハブを通じて接続中クライアントに更新を配信する.

フォーク元リポジトリの機能としてあったオーダーストップ機能についても同様に, 状態更新 API で DB へ保存を行い, その後リアルタイム通知を配信する構成である. これにより, 複数端末で同一の運用状態を共有できる.

3.4 データモデルと役割分離

共通モジュールでは, 注文, 商品, 商品種別などに関するフロントエンド向けのエンティティ, 補助的な型定義, API アクセス処理, 変換関数, 各種フックを集約している. これにより, 画面実装に必要なデータ操作や状態管理を一箇所にまとめ, 再利用しやすい構成としている.

一方で, フロントエンドとバックエンドのインタフェースとして用いる API の入出力仕様は OpenAPI により記述し, それをもとにクライアント側で利用する型を生成している. これにより, API 仕様を共有しながら実装できる構成としている.

また, ユーティリティ関数と React Hook の責務は明確に分離するようにしている. 具体的には, フックはコンポーネントまたは別のフックのトップレベルでのみ呼び出し, 通常の変換関数や計算関数には必要なデータを引数として渡す構成とした. これにより, Hook の不正利用に伴う実行時エラーを防ぎ, 保守性を高めている.

4 進捗

現時点までに, バックエンドでは商品, 商品種別, 注文, コメント, 商品提供状況を扱う基本的な API 群を整備した.[2] データベースとして PostgreSQL を用い, 注文情報の保存および再取得が可能になっている. また, WebSocket ハブを用いた注文情報や提供状況のリアルタイム配信機構についても基礎部分を実装している.

フロントエンドでは, 元となったリポジトリの画面から, キャッシャー画面の UI を再構築した. 注文一覧, 商品一覧, 会計処理, コメント追加, 提供表示など, 主要機能は概ね動作する段階にある.

図 1 は, キャッシャー画面の変更例である. 左の旧 UI は, キーボード中心の操作を前提とした構成である. 一方, 右の新 UI は, タッチ操作を前提とし, 商品ボタンや注文中アイテム一覧を一画面内に配置することで, より直感的な操作が可能な構成となっている. 今後は, この新 UI をベースにさらに改善を進める予定である.

図 1: キャッシャー画面の UI 変更例（左：旧 UI, 右：新 UI）

本開発は, 学園祭店舗向けに共同開発された既存リポジトリをフォークして進めているため, 初期段階では既存コードの読解と依存関係の整理に多くの時間を要したが, その過程で現場運用に基づく実践的な設計意図を把握することができた. これにより, 特定の運用に依存する部分を抽象化し, より一般的な構成へと再設計することが可能になった.

[2] https://github.com/Lailai0477/tmp-pos

現時点でデータのやり取りや UI については基礎的な部分は整っているが, エラー画面が一瞬だけ見えるなど不安定な部分があり, 安定運用に向けた調整が必要である. また, カスタマイズ性を高くするため, カスタマイズ設定を可能にするページや, 設定の DB 保存などの実装を進めている.

5 今後の展望

今後は, 使用者がシステムを使用可能な状態になる, またシステムのカスタマイズをしやすくするためのドキュメント整備を進める.

ネットワーク構成について, 改善の余地がある. ローカルネットワーク上の複数端末からフロントエンドおよびバックエンドに安定して接続できるよう, ホスト公開設定やポート転送, API ベース URL の環境変数化などを進め, 実運用に近い形で検証できる環境を整備したい.

加えて, システムの保守性向上や, 開発者のため, コンポーネント分割, 共通処理の整理, API スキーマの見直しも継続する予定である. 将来的には, 利用ログの分析や操作時間の計測を行い, 業務効率改善の定量的評価につなげたい.

6 おわりに

本報告書では, 注文受付と提供管理を目的とした POS アプリケーションについて, システム構成, 実装方法, 先行研究との位置付け, 現時点での進捗, および今後の展望を整理した. 本システムは, Web 技術を基盤として, 役割別画面構成とリアルタイム同期を組み合わせることで, 現場運用に適した柔軟な注文管理を実現しようとするものである.

現段階では, 主要な注文処理とデータ同期の基礎実装は整っており, 今後はタッチパネル UI の改善や運用状態共有の洗練を進めることで, より実践的なシステムへ発展させる見込みである. 最終的には, 特定現場に最適化された軽量 POS として, 運用効率や操作性の向上に寄与することを目指す.

GEMMul8 における Ozaki Scheme II INT8 DGEMM のデータフロー最適化

筑波大学 情報科学類 4 年
202311951 細島涼雅 [s2311951@u.tsukuba.ac.jp]

概要– 近年の GPU では、低精度演算の性能が重点的に強化されている。一方で、数値シミュレーションをはじめとする科学技術計算の分野では、依然として FP64 の精度が重要である。このような背景から、高精度演算を低精度演算器でエミュレートする技術が重要になる。GEMMul8 は Ozaki Scheme II に基づき、INT8 や FP8 の行列演算器を用いて高精度 GEMM を実現するライブラリである。本研究では、比較的安価なコンシューマ向け GPU である NVIDIA GeForce RTX 3060 Ti 上で GEMMul8 の INT8 DGEMM 経路を対象に、性能測定と高速化を行った。実行時固定費の削減、GEMM 前カーネルの融合、GEMM 後処理の選択的融合により、`OS2-fast-9` で約 23.4% の性能向上を達成した。

1 はじめに

1.1 背景

近年の GPU アーキテクチャは、生成 AI や大規模言語モデルの需要拡大に伴い、低精度演算性能を重視する方向へ発展している。特に FP16、BF16、INT8、FP8 といった低精度データ型は、学習・推論におけるメモリ帯域、消費電力、演算スループットの観点から重要であり、Tensor Core のような行列演算器ではこれらの低精度演算を高い性能で実行できる。一方で、倍精度浮動小数点数（FP64）演算性能はデータセンター向け GPU では重視されるものの、コンシューマ向け GPU や AI 向けアクセラレータでは相対的に制限されている場合が多い。

代表的な GPU の理論ピーク性能を 表 1 に示す。比較対象として、HPC 向け GPU の代表例である NVIDIA H100 SXM、プロフェッショナル・ワークステーション向け GPU である NVIDIA RTX PRO 6000 Blackwell Workstation Edition、および本研究で用いるコンシューマ向け GPU である NVIDIA GeForce RTX 3060 Ti を取り上げる。

H100 SXM は 34 TFLOPS の FP64 性能、および 67 TFLOPS の FP64 Tensor Core 性能をもつ。一方で、RTX PRO 6000 Blackwell Workstation Edition は FP32 性能が 126.0 TFLOPS と高いものの、FP64 性能は 1.97 TFLOPS に留まる。（なお、この GPU の FP64 性能は未公表であるため、NVIDIA の Blackwell アーキテクチャ資料に示された、GB202 GPU の FP64 性能が FP32 の 1/64 に制限されるという情報に基づいて算出した。[1]）これらの値から、データセンター向け GPU とコンシューマ向け GPU の間には、FP64 演算能力に大きな差があることが分かる。

GPU	FP64 理論ピーク性能	FP32 理論ピーク性能	INT8 Tensor 理論ピーク性能	FP8 Tensor 理論ピーク性能
H100 SXM	34 / 67 TC TFLOPS	67 TFLOPS	1979 / 3958 TOPS	1979 / 3958 TFLOPS
RTX PRO 6000 Blackwell Workstation Edition	約 1.97 TFLOPS	126.0 TFLOPS	1007.6 / 2015.2 TOPS	1007.6 / 2015.2 TFLOPS
GeForce RTX 3060 Ti	約 0.25 TFLOPS	約 16.2 TFLOPS	約 130 / 260 TOPS	非対応

表 1: 各 GPU 理論ピーク性能比較

a / b は通常時/疎性利用時の値を表す。H100 SXM の FP64 理論ピーク性能は、通常の FP64 演算/ FP64 Tensor Core の値を併記した。RTX PRO 6000 Blackwell の FP64 理論ピーク性能は、GB202 の FP64 性能が FP32 の 1/64 であることから算出した。RTX 3060 Ti の FP32、FP64、INT8 Tensor 理論ピーク性能は、公開仕様と GA10x アーキテクチャ資料に基づく概算である。 [1]–[5]

このように、近年の GPU では FP64 性能と低精度 Tensor Core 性能の間に大きな非対称性がある。特に、コンシューマ向け GPU やワークステーション向け GPU では、FP64 演算性能は HPC 向け GPU に及ばない一方で、AI 向けに強化された INT8 や FP8 の行列演算器は高い理論性能を持つ。

しかし、科学技術計算や数値シミュレーションでは、FP64 は依然として重要である。流体計算、構造解析、分子シミュレーション、固有値問題、線形方程式求解などでは、丸め誤差の蓄積や条件数の影響により、単純に低精度演算へ置き換えることは難しい。そのため、低精度演算器の高いスループットを活用しつつ、FP64 相当の精度を実現する手法には大きな実用的価値がある。

このような背景のもと、高精度演算を複数の低精度演算の組み合わせとして実現する手法が注目されている。Ozaki Scheme は、高精度な行列積を複数の低精度行列積へ分解し、それらの結果を合成することで高精度 GEMM を実現する手法である。さらに Ozaki Scheme II では、中国剰余定理に基づく整数剰余表現を用いることで、INT8 や FP8 の行列演算器を用いた高精度 GEMM エミュレーションを可能にしている。[6]

GEMMul8 は、この Ozaki Scheme II に基づき、INT8 および FP8 バックエンドを通じて SGEMM、DGEMM、CGEMM、ZGEMM のエミュレーションを提供するライブラリである。これは、AI 向けに発展した低精度行列演算器を、FP64 を必要とする科学技術計算へ応用するための具体的な基盤であるといえる。[7], [8]

一方で、Ozaki Scheme II に基づく高精度 GEMM エミュレーションでは、入力行列のスケーリング・分割、複数回の低精度 GEMM、中国剰余定理に基づく累積、スケーリングの復元など、通常の GEMM には存在しない処理が必要となる。そのため、低精度 GEMM 自体が高速であっても、前処理・後処理・カーネル起動・メモリアクセスに伴うオーバーヘッドが、全体性能を制限する可能性がある。したがって、Ozaki Scheme II を実用的な高精度計算手法とし

て活用するには、アルゴリズムのみならず GPU アーキテクチャを踏まえた実装最適化が重要である。

本研究では、GEMMul8 の CUDA / INT8 DGEMM 経路に着目し、比較的安価なコンシューマ向け GPU である NVIDIA GeForce RTX 3060 Ti 上で Ozaki Scheme II に基づく FP64 相当 GEMM エミュレーションの高速化を検討する。最適化を通じて、FP64 性能が限定的な GPU において低精度演算器を高精度計算へ活用する可能性を評価する。

1.2 Ozaki Scheme II による高精度 GEMM エミュレーション

本研究で対象とする DGEMM は、倍精度行列 A, B, C に対して、概略として

$$C \leftarrow AB + C$$

を計算する処理である。通常の DGEMM ではこの積和計算を FP64 演算器で直接実行する。一方、Ozaki Scheme II では、入力行列を低精度または整数行列演算器で扱える形に変換し、複数回の低精度 GEMM の結果を合成することで、高精度な GEMM をエミュレートする [6]。

Ozaki Scheme II の基本的な流れは、まず入力行列 A および B に対してスケーリングを行い、対象とする整数表現に収まるように値域を調整する。次に、複数の互いに素な法 p_i に対する剰余表現へ変換する。各法 p_i に対して、低精度または整数行列演算器を用いて行列積を計算し、その結果を中国剰余定理（Chinese Remainder Theorem、以下 CRT と表記）に基づいて合成する。最後に、スケーリングの復元を行うことによって元の FP64 スケールに対応する結果を得る。

この方式では、FP64 演算器を直接用いる代わりに、INT8 や FP8 の行列演算器を複数回利用する。そのため、低精度 GEMM 本体の演算性能が高い GPU では、FP64 性能が限定的であっても高い実効性能を得られる可能性がある。一方で、通常の DGEMM には存在しない入力行列のスケーリング・分割、複数回の低精度 GEMM、CRT に基づく累積、スケーリングの復元といった処理が必要になる。また、高精度化のために用いる法の数（`num_moduli`）を増やすほど、精度や表現可能な範囲は向上するが、低精度 GEMM の実行回数や前後処理のコストも増加する。

GEMMul8 は Ozaki Scheme II に基づいて実装されており、INT8 および FP8 行列演算器を通じて、SGEMM、DGEMM、CGEMM、ZGEMM のエミュレーションを提供する [8]。本研究では、このうち CUDA / INT8 バックエンドによる DGEMM 経路を対象とする。特に、低精度 GEMM 本体は cuBLASLt によって実行されるため、本研究の最適化対象は主に GEMM の前後に存在するスケーリング・分割、CRT に基づく累積、スケーリングの復元、およびカーネルの起動にかかるオーバーヘッドである。

1.3 本研究の目的

本研究の目的は、GEMMul8 の CUDA / INT8 DGEMM 経路を対象として、Ozaki Scheme II に基づく高精度 GEMM エミュレーションを GPU 上でさらに高速化することである。特に、前節で述べた前処理・後処理および実行時固定費の削減を図る。また、FP64 性能が限定的な

コンシューマ向け GPU である NVIDIA GeForce RTX 3060 Ti 上で、INT8 Tensor Core を用いた DGEMM エミュレーションがどの程度有効であるかを評価する。

2　対象実装と性能上の課題

本章では、GEMMul8 の INT8 DGEMM 経路の実行フローを整理し、本研究で最適化対象とする処理を明確にする。Ozaki Scheme II では低精度 GEMM 本体だけでなく、その前後にあるデータ変換や累積処理も性能に影響するため、まず既存実装のどこにコストが生じるかを確認する。

GEMMul8 の INT8 DGEMM 経路は、概略として以下のような流れで実行される。

1. 入力行列 A および B に対して、スケーリングを行い、INT8 行列演算器で扱える領域へ変換する。
2. 複数の法に対応する剰余表現へ変換する。
3. 各剰余に対して cuBLASLt を用いた INT8 GEMM を実行し、得られた結果を CRT に基づく累積によって合成する。
4. スケーリングの復元を行い、FP64 相当の出力行列を得る。

この実行フローでは、計算時間の中心は各剰余に対する INT8 GEMM である。一方で、Ozaki Scheme II では複数回の INT8 GEMM を実行するため、その前後にある処理も無視できない。特に、行列サイズが十分に大きい場合でも、これらの前処理・後処理におけるメモリアクセスやカーネルの起動は大きく性能に影響を与える。

2.1　最適化対象

GEMMul8 の既存実装では、低精度 GEMM 本体は主に cuBLASLt に委譲される。cuBLASLt は高性能な GEMM 実装を提供する一方で、GEMM カーネルの内部はブラックボックスであり、CRT に基づく累積やスケーリングの復元を GEMM エピローグに直接融合することは難しい。そのため、本研究では cuBLASLt による INT8 GEMM 本体は維持しつつ、GEMM の前後に存在する処理を主な最適化対象とした。

具体的には、以下の 4 種類のコストに着目することとした。

- cuBLASLt 記述子やヒューリスティックの構築など、API 呼び出しに伴う固定費
- 測定用プロファイリングに伴う同期コスト
- スケーリングや剰余化などの GEMM 前カーネル群
- `conv_hi2mid`、CRT に基づく累積、スケーリングの復元などの GEMM 後カーネル群

3　実装した最適化とその評価

本章では、各最適化の目的と実装方針、およびその評価結果を述べる。

3.1 評価環境

評価環境を 表 2 に示す。本研究では、FP64 性能が限定的なコンシューマ向け GPU である NVIDIA GeForce RTX 3060 Ti を用いた。

項目	内容
GPU	NVIDIA GeForce RTX 3060 Ti
GPU アーキテクチャ	`sm_86`
実行環境	WSL + Docker + CUDA コンテナ
対象ライブラリ	GEMMul8
対象経路	CUDA / DGEMM / INT8 バックエンド
主な測定サイズ	`4096 x 4096 x 4096`
CRT の法の数	`num_moduli = 9`
測定反復	`warmup = 30`, `mainloop = 30`

表 2: 評価環境

3.2 実行時固定費の削減

3.2.1 実装方針

GEMMul8 の既存実装では、低精度 GEMM 本体の実行に cuBLASLt を用いている。cuBLASLt を利用する際には、行列サイズ、データ型、転置の有無、リーディングディメンションなどに基づいて記述子を構築し、実行アルゴリズムのヒューリスティックを取得する必要がある。これらは GEMM 本体の演算ではないが、Ozaki Scheme II では複数の剰余に対して低精度 GEMM を繰り返し実行するため、cuBLASLt 呼び出し周辺の固定費が全体性能に影響しうる。

そこで、同一条件で繰り返し利用される cuBLASLt 記述子およびヒューリスティックをキャッシュし、毎回の構築を避けるようにした。（`cached`）さらに、内部プロファイリング用の同期を除去し、段階別時間測定を行わない、実運用に近い実行経路を用意した。（`cached-notiming`）

3.2.2 評価結果

4096 四方行列において、元実装、`cached`、`cached-notiming` を比較した結果を 表 3 に示す。

実装	`OS2-fast-9` TFLOPS	`OS2-fast-9` time [s]	`OS2-accu-9` TFLOPS	`OS2-accu-9` time [s]
`baseline`	6.031525	0.02278677	5.509434	0.02494611
`cached`	6.087550	0.02257706	5.584585	0.02461042
`cached-notiming`	6.694198	0.02053106	6.218286	0.02210238

表 3: Baseline から固定費を削減した場合の性能

cached による記述子およびヒューリスティックのキャッシュ単体の効果は限定的である。一方、cached-notiming は baseline に対して、OS2-fast-9 で 10.99%、OS2-accu-9 で 12.87% の TFLOPS 改善を示した。

3.2.3 小括

この結果は、既存実装に含まれる段階別時間測定や同期処理が、実運用に近い実行では無視できないオーバーヘッドになっていたことを示している。特に GPU カーネルは本来非同期に実行されるため、測定用の同期が実行時間に影響する。以降の評価では、実運用に近い性能を見るため、必要に応じて notiming 条件を用いる。

3.3 GEMM 前カーネルの融合

3.3.1 実装方針

Ozaki Scheme II では、低精度 GEMM を実行する前に、入力行列 A, B を低精度または整数表現で扱えるように変換する必要がある。GEMMul8 の INT8 DGEMM 経路では、この前処理にスケーリングや剰余化に関する複数の CUDA カーネルが含まれる。これらのカーネルが分かれている場合、カーネル起動の固定費に加えて、中間結果をグローバルメモリに書き出し、次のカーネルで再度読み込む必要がある。

そこで fused 実装では、fast 経路における A 側の compute_sftA_kernel と scalingA_kernel を融合した。これにより、スケール係数の計算とスケーリングの適用を単一のカーネル内で行い、中間データのグローバルメモリ転送量とカーネル起動回数の削減を狙う。さらに fused-all 実装では、accu 経路を含む A 側の量子化処理の融合を追加した。

3.3.2 評価結果

4096 四方行列における OS2-fast-9 の結果を 表 4 に示す。

実装	OS2-fast-9 TFLOPS	合計時間[s]	量子化[s]
cached	5.981786	0.02297624	0.002871404
fused	6.324913	0.02172978	0.002318149

表 4: GEMM 前カーネルの融合による量子化段階の改善

fused では、量子化段階が 0.002871404 s から 0.002318149 s に短縮され、19.27% の削減となった。全体でも OS2-fast-9 の TFLOPS は 5.981786 から 6.324913 に改善し、5.74% の向上が得られた。

3.3.3 小括

この結果から、GEMM 前処理に含まれるスケーリングや剰余化は、低精度 GEMM 本体と比べれば小さいものの、全体性能に影響するだけの割合を占めていることが分かる。一方で、GEMM 本体の割合が大きいため、量子化段階の局所的な削減率がそのまま全体改善率になる

わけではない。したがって、GEMM 前カーネルの融合は有効であるが、単独で大きな高速化を得るには限界がある。

3.4 GEMM 後処理の選択的融合

3.4.1 実装方針

低精度 GEMM の後には、各剰余に対応する GEMM 結果を高精度な結果へ合成する処理が必要となる。GEMMul8 の INT8 DGEMM 経路では、概略として `conv_hi2mid` により GEMM 出力を中間表現へ変換し、その後 CRT に基づく累積と逆スケーリングを行う。この GEMM 後処理も複数のカーネルに分かれており、グローバルメモリ転送量とカーネル起動オーバーヘッドの観点から最適化の余地がある。

まず `fused-post` 実装では、`conv_hi2mid` と CRT に基づく累積を全面的に融合する実装を試作した。この方式では、中間表現への書き出しと再読み込みを削減できる可能性がある。一方で、全面的な融合では行列全体のアキュムレータを保持する必要があり、特に `double` や `double2` のアキュムレータを用いる場合にはメモリ使用量とメモリ転送量が増加する。

この課題を踏まえ、`fused-hybrid` 実装では、小さな `C_mid` 表現を維持しつつ、最後の剰余のみを最終的な逆スケーリングに直接吸収する方式を実装した。さらに `fused-hybrid2` では、最後の 2 剰余を最終的な逆スケーリングに吸収する実装を試した。本研究では、これらを GEMM 後処理の選択的融合と位置づける。

3.4.2 評価結果

`fused-notiming` と `fused-hybrid-notiming` の比較を 表 5 に示す。

実装	`OS2-fast-9` TFLOPS	`OS2-fast-9` time [s]	`OS2-accu-9` TFLOPS	`OS2-accu-9` time [s]
`fused-notiming`	7.367131	0.01865570	6.538773	0.02101907
`fused-hybrid-notiming`	7.440213	0.01847245	6.700887	0.02051056

表 5: GEMM 後処理の選択的融合による性能改善

`fused-hybrid-notiming` は `fused-notiming` に対して、`OS2-fast-9` で 0.99%、`OS2-accu-9` で 2.48% の改善を示した。改善幅は大きくないが、GEMM 後処理のデータフロー変更により、実際に全体性能が改善することを確認した。

一方、最後の 2 剰余を融合する `fused-hybrid2` は、時間測定を有効にした条件ではわずかに良化したものの、実運用に近い時間測定なしの条件では `fused-hybrid` を明確には上回らなかった。

3.4.3 小括

GEMM 後処理の融合では、単純にカーネル数を減らせば高速化するとは限らない。`fused-post` のような全面融合では、中間表現への書き出しを減らせる一方で、行列全体のアキュムレータによってメモリ転送量が増加し、性能が悪化する可能性がある。

一方、fused-hybrid のように小さな中間表現を維持しながら一部の剰余のみを融合する方式は、小さいながらも実用的な改善を示した。このことから、GEMM 後処理では、カーネル数の削減だけでなく、中間表現のサイズ、アキュムレータの配置、メモリ帯域を考慮した選択的な融合が重要である。

3.5 数値一致性

性能改善に加えて、各実装が数値結果を変更していないことを確認するため、精度確認の一括試験を実行した。 cached、fused、fused-all、fused-post、fused-hybrid、fused-hybrid2、fused-hybrid-vlast の数値行は一致した。

抽出した数値行の SHA-256 は以下で共通であった。

```
ae685d4fe5691a4a03e743e560abf53f37aec299493cb2504a6ecd23e5a5faa3
```

したがって、今回の最適化は、少なくとも本精度確認試験の範囲では、既存 GEMMul8 の数値結果を変更していない。

4 考察

本章では、前章の評価結果に基づき、RTX 3060 Ti における Ozaki Scheme II の有効性、カーネル融合の効果と限界、および cuBLASLt バックエンドの制約について考察する。

4.1 RTX 3060 Ti における Ozaki Scheme II の有効性

本研究の結果から、RTX 3060 Ti のような FP64 性能が限定的なコンシューマ向け GPU においても、INT8 Tensor Core を用いた Ozaki Scheme II による DGEMM エミュレーションは高い実効性能を示すことが確認できた。

RTX 3060 Ti のネイティブ FP64 DGEMM 性能は、本研究の測定では約 0.25 TFLOPS 程度であった。一方で、GEMMul8 の CUDA / INT8 バックエンドを用いた Ozaki Scheme II 経路では、元の baseline の OS2-fast-9 で 6.031525 TFLOPS、最適化後の fused-hybrid-notiming で 7.440213 TFLOPS を達成した。

この結果は、FP64 演算器を直接用いるのではなく、INT8 Tensor Core を用いて高精度 GEMM をエミュレートすることで、FP64 性能が低い GPU でも高い実効スループットを得られる可能性を示している。特に、コンシューマ向け GPU やワークステーション向け GPU では FP64 演算性能が制限される一方、AI 向けに低精度行列演算器の性能は高く設計されている。このようなハードウェア特性は、Ozaki Scheme II のように低精度演算器を高精度計算へ転用する手法と相性がよい。

ただし、本研究で示した TFLOPS は、通常のネイティブ DGEMM と完全に同一の意味を持つものではない。Ozaki Scheme II は複数回の低精度 GEMM と CRT に基づく累積によって高精度結果を得るエミュレーション手法であり、その性能は法の数、入力データの性質、前処理・後処理の実装、数値精度要件に依存する。したがって、ネイティブ FP64 DGEMM との比

較では、単純なピーク性能だけでなく、必要な精度、再現性、適用可能な問題設定を含めて評価する必要がある。

4.2 カーネル融合の効果と限界

GEMM 前カーネルの融合では、`compute_sftA_kernel` と `scalingA_kernel` を融合することで、量子化段階を 19.27% 削減し、全体でも 5.74% の性能改善を得た。この結果から、Ozaki Scheme II の前処理であるスケーリングと剰余化は、低精度 GEMM 本体に比べると小さいものの、全体性能に影響する十分な割合を占めていることが分かる。

この改善は、カーネル起動回数の削減と、中間データのグローバルメモリへの書き出し・再読み込みの削減によるものと考えられる。特に GPU では、演算そのものが高速であるほど、周辺処理のメモリ転送量や起動オーバーヘッドが相対的に目立つ。Ozaki Scheme II のように、複数の剰余に対して同様の処理を繰り返すアルゴリズムでは、このような前処理カーネルの最適化が全体性能に効く。

一方で、GEMM 前処理の融合の改善幅には限界もある。低精度 GEMM 本体の計算時間が全体の大きな割合を占めるため、量子化段階を大きく削減しても、その効果は全体では薄まる。したがって、GEMM 前処理の融合は有効な最適化であるが、単独で大きな性能向上を得るというより、他の固定費削減や GEMM 後処理の最適化と組み合わせることで効果を発揮する。

GEMM 後処理の融合では、より慎重な設計が必要である。`fused-post` のように `conv_hi2mid` と CRT に基づく累積を全面的に融合する方式は、カーネル数や中間表現の往復を減らせる可能性がある。しかし、行列全体の `double` / `double2` アキュムレータを保持する必要があり、結果としてメモリ転送量が増加し、4096 四方行列では性能低下につながった。

一方、`fused-hybrid` では小さな `C_mid` 表現を維持しつつ、最後の剰余のみを最終的な逆スケーリングに吸収することで、`OS2-fast-9` で 0.99%、`OS2-accu-9` で 2.48% の改善を得た。改善幅は小さいが、GEMM 後処理のデータフロー変更が実際に全体性能へ寄与することを確認できた点は重要である。

この結果から、GEMM 後処理の融合では、単にカーネル数を減らすのではなく、アキュムレータのデータ型、中間表現のサイズ、グローバルメモリ転送量、メモリ帯域を含めた設計が重要であるといえる。特に、全面融合よりも選択的融合やタイルまたはブロック単位の累積の方が、実装上の見通しがよいと考えられる。

4.3 cuBLASLt バックエンドの限界

本研究では、低精度 GEMM 本体には既存の cuBLASLt バックエンドを用いた。cuBLASLt は高性能な GEMM 実装を提供するため、GEMM 本体の性能を容易に得られるという利点がある。一方で、GEMM カーネル内部はブラックボックスであり、Ozaki Scheme II 固有の後処理を GEMM エピローグに深く融合することは難しい。

この制約は、GEMM 後処理の融合の限界として現れる。cuBLASLt を使い続ける限り、INT8 GEMM の出力は一度グローバルメモリ上の中間表現として扱う必要があり、その後

に conv_hi2mid、CRT に基づく累積、逆スケーリングを別カーネルとして実行することになる。そのため、GEMM 本体と CRT に基づく累積を一体化したデータフローを実現するには限界がある。

より大きな性能改善を狙う場合、CUTLASS などを用いて独自の INT8 GEMM バックエンドを実装し、GEMM エピローグに剰余変換や部分的な CRT 累積を組み込む必要がある。これにより、GEMM 出力をグローバルメモリに書き出す前に一部の後処理を行える可能性がある。ただし、この方向はバックエンドの再実装に近く、cuBLASLt の最適化済みカーネルと同等の性能を維持する必要があるため、実装コストと検証コストは大きい。

したがって、cuBLASLt バックエンドを維持する場合は、GEMM 本体の外側で可能な固定費削減、GEMM 前処理の融合、GEMM 後処理の選択的融合が現実的な最適化対象となる。一方で、GEMM 本体まで含めた本格的な融合を目指す場合は、CUTLASS バックエンドへの移行が次の大きな研究課題となる。

4.4 本研究の限界

本研究にはいくつかの限界がある。

第一に、評価環境が RTX 3060 Ti に限定されている点である。RTX 3060 Ti は FP64 性能が低く、INT8 Tensor Core を用いたエミュレーションの有効性を確認する対象として適している。しかし、H100、A100、RTX PRO 6000 Blackwell、Ada Lovelace 世代や Blackwell 世代の GeForce / RTX PRO GPU では、Tensor Core 性能、メモリ帯域、FP8 対応、cuBLASLt の挙動が異なる。そのため、本研究の最適化が他の GPU でも同様に有効であるかは、追加評価が必要である。

第二に、本研究では cuBLASLt バックエンドを維持しており、GEMM 本体のカーネル内部には手を加えていない。そのため、GEMM エピローグと CRT に基づく累積を一体化するような深い融合は実現していない。本格的な GEMM 後処理の融合の効果を確認するには、CUTLASS などを用いた独自バックエンドの実装が必要である。

第三に、数値一致性の確認は精度確認の一括試験に基づくものであり、すべての入力分布や問題設定に対する数値的性質を保証するものではない。本研究で実装した最適化は、少なくとも評価した範囲では既存 GEMMul8 の数値結果を変更していない。しかし、Ozaki Scheme II を実アプリケーションへ適用する際には、入力行列のスケール、条件数、法の数、求める精度に応じた追加検証が必要である。

5 まとめ

本研究では、GEMMul8 の CUDA / INT8 DGEMM 経路を対象として、Ozaki Scheme II に基づく高精度 GEMM エミュレーションのデータフローを最適化した。RTX 3060 Ti 上での評価により、INT8 Tensor Core を用いた DGEMM エミュレーションが、FP64 性能の低い GPU においても高い実効性能を示すことを確認した。

実行時固定費の削減、GEMM 前カーネルの融合、GEMM 後処理の選択的融合を組み合わせることで、単発呼び出しに近い条件でも性能を改善できた。`OS2-fast-9` では、元の `baseline` の 6.031525 TFLOPS から `fused-hybrid-notiming` の 7.440213 TFLOPS へ改善し、約 23.4% の性能向上を達成した。

これらの結果から、Ozaki Scheme II の実装では低精度 GEMM 本体だけでなく、その前後にあるデータ変換、累積、スケーリング復元、起動固定費を含めたデータフロー設計が重要であるといえる。

6　今後の課題

今後の課題は大きく三つある。

第一に、評価対象を複数の GPU へ広げる必要がある。本研究では RTX 3060 Ti を用いたが、H100、A100、RTX PRO 6000 Blackwell などでは、FP64 性能、低精度 Tensor Core 性能、メモリ帯域、cuBLASLt のアルゴリズム選択が大きく異なる。したがって、本研究で有効であった最適化が他の GPU でも同様に有効であるかを確認する必要がある。

第二に、cuBLASLt バックエンドの外側にとどまらない最適化を検討する必要がある。本研究では GEMM 本体は cuBLASLt に委譲したため、GEMM エピローグと CRT に基づく累積を深く融合することはできなかった。より大きな性能改善を狙うには、CUTLASS などを用いた独自バックエンドを実装し、GEMM 出力をグローバルメモリへ書き出す前に一部の後処理を組み込む方向が考えられる。

第三に、数値的な検証を拡張する必要がある。本研究では精度確認の一括試験により実装間の数値一致性を確認したが、入力分布、条件数、法の数、対象アプリケーションに応じた精度評価は今後の課題である。

参考文献

[1] NVIDIA, NVIDIA RTX Blackwell GPU Architecture, [2025].

[2] NVIDIA, NVIDIA H100 Tensor Core GPU, URL: https://www.nvidia.com/en-us/data-center/h100/ Accessed on 2026.04.25.

[3] NVIDIA, NVIDIA RTX PRO Blackwell GPU Architecture, [2025].

[4] NVIDIA, GeForce RTX 3060 Family, URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3060-3060ti/ Accessed on 2026.04.25.

[5] NVIDIA, NVIDIA Ampere GA102 GPU Architecture, [2020].

[6] K. Ozaki, Y. Uchino, and T. Imamura, Ozaki Scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique, [2025].

[7] Y. Uchino, K. Ozaki, and T. Imamura, High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines, Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis, 1824/1831 [2025].

[8] RIKEN-RCCS, GEMMul8: GEMM emulation using INT8/FP8 matrix engines based on the Ozaki Scheme II, URL: https://github.com/RIKEN-RCCS/GEMMul8 Accessed on 2026.04.25.

Zig 言語の普及に関する活動

筑波大学 情報科学類 4 年
202311951 細島涼雅 [s2311951@u.tsukuba.ac.jp]

1　はじめに

Zig 言語は、高いパフォーマンスとシンプルな設計を特徴とする一方で、現時点ではまだ広く普及しているとは言い難いプログラミング言語である。新しい言語を普及させるためには、実際に手を動かして試せる環境が重要であるが、一般的な開発用途では、セットアップの手間や学習コストが障壁となることも少なくない。

そこで私は、「競技プログラミングを通じて Zig を普及させる」という方針を出発点として、Zig をより多くの人に試してもらうための環境整備や関連ツールの開発に取り組んだ。

2　活動の概要

2.1　AtCoder における Zig 環境のアップデート

競技プログラミングでは、オンラインジャッジによって即座にプログラムの正誤を確認できるため、初学者でも気軽に Zig を試すことができる。また、競技向けのライブラリが充実すれば、実用的な場面での利用も広がる可能性があると考えられる。

このような目的のもと、国内最大規模の競技プログラミングコンテストサービスである AtCoder において、Zig 言語サポートの強化および関連ライブラリの開発に取り組んだ。その成果は言語アップデートに反映され、実際の提出環境として利用可能となった。

AtCoder では Zig 0.10.1 が提供されていたが、Zig の開発は活発に進んでおり、最新バージョンと大きく差があった。 そこで、他のユーザと協力し、環境のアップデートに取り組み、現在は 0.15.1 が提出環境として利用可能である。

2.2　ac-library の Zig 版の開発とメンテナンス

競技プログラミングにおいては、高速なデータ構造やアルゴリズムを提供するライブラリが不可欠である。AtCoder では C++向けに ac-library（AtCoder Library）が提供されている[1]が、Zig には同等の標準ライブラリが存在しない。そこで、ac-library の Zig 版を作成し、競技プログラミングに適した形で提供している。

C++版 ac-library を参考にしつつ Zig 向けに最適化、競技プログラミング以外の場面でも汎用的に使えるような設計と実装を行った。

実際に、このライブラリを利用可能にすることを提案し、最新環境では提出時に利用できるようになっている。

また、継続的にメンテナンスもしており、最新の Zig v0.16.0 への対応などを進めている。

2.3　Zig 言語のミラーサービスの提供

室員でありかつ WIDE プロジェクト つくば NOC の間瀬とともに、Zig 言語のミラーサービスを実験的に提供している。

zig.tsukuba.wide.ad.jp において利用可能な環境を構築し、国内外から多くのアクセスを受けた。

現在は一部不具合により公式ミラーリストから外れてしまっているが、今後は原因究明を進めるとともに、継続的なサービス提供を目指す。

3　今後の展望

本活動では、AtCoder における Zig 環境のアップデートと ac-library の Zig 版の開発を通じて、競技プログラミングを Zig の普及手段とする取り組みを行った。しかし、Zig のさらなる普及のためには、競技プログラミングに限らず、より広い分野への展開が重要である。

継続した AtCoder における Zig の環境のアップデートと、ac-library の Zig 版のメンテナンスを行っていきたいと考えている。特に、Zig の開発は活発であり、今後も仕様変更や最適化が進むことが予想されるため、AtCoder の Zig 環境を最新の状態に保つことは、言語の競技プログラミングにおける利便性を維持する上で重要である。

さらに、競技プログラミング以外の分野でも Zig の普及を促進することを目指す。 Zig は汎用プログラミング言語であり、様々な分野に応用することができる。 その応用についてのチュートリアルを作成することで、多くの人が Zig に興味を持つことができ普及に貢献できるのではないかと考えている。

Zig は、シンプルな構文、強力な `comptime`、明示的なエラーハンドリングなどの特徴を持ち、システムプログラミングや競技プログラミングの分野で高い可能性を秘めている。今後は、これらの特徴を活かした活動を継続し、Zig の普及に貢献していきたい。

参考文献

[1]. AtCoder. 「AtCoder Library (ac-library)」. GitHub, 2026 年 4 月 25 日取得, https://github.com/atcoder/ac-library

WIDE Project つくば NOC 運用報告

理工情報生命学術院　システム情報工学研究　情報理工学位プログラム　博士前期課程　関口　亞聖（s1611390@coins.tsukuba.ac.jp)
理工情報生命学術院　システム情報工学研究群　情報理工学位プログラム　博士前期課程　服部　真吾（s2111559@u.tsukuba.ac.jp）
情報学群　情報科学類　中野　あおい（s2312794@u.tsukuba.ac.jp）
情報学群　情報科学類　間瀬　太陽（s2410334@u.tsukuba.ac.jp）

1　つくば NOC とは

つくば NOC は筑波大学術情報メディアセンター支援の元、情報科学類産学間連携推進室内サーバー室にて運用している NOC (Network Operation Center、ネットワークを構成する機材等を収容する施設) である。

本 NOC では通常の大学ネットワークから独立した別系統のネットワークを独自に運用しており、大学ネットワークでの運用や実施が難しいプロジェクトや実験 (不特定多数との間で大量のトラフィックを発生させるもの、大学外の一般に向けたサービスの運営、ハニーポットを用いたものなど) を実施している。また、この過程においてネットワーク運用に関する知見や技術の習得を目的としている。特につくば WIDE Project はこのつくば NOC を使用した主要な研究活動である。

2　つくば WIDE に関して

このつくば WIDE は 2007 年 11 月から産学間連携推進室に場所をお借りし、運用している WIDE Project の拠点の 1 つである。WIDE Project は複数の大学など教育機関や企業の研究所などにより構成されており、大規模ネットワークの運用・研究のほか、安定した高品質なネットワークを一般の WIDE BB 利用者や研究者等に提供することを目的とした活動を行っている。つくば WIDE では、この WIDE Project の一部として、つくば地域において WIDE バックボーン（東京大手町）と接続された NOC を運用している。2014 年には CTF シリーズを開催することで新たに情報セキュリティ分野での活動が行われた。つくば WIDE では現在、以下のようなプロジェクトを行っている。

2.1 パブリックミラーサービスの提供

主に日本国内向けに Linux OS (Ubuntu, Debian, fedora など) や各種ソフトウェア (Apache, GCC, Vim など) の配信を行うミラーサーバーである。

30TB 超のストレージアレイを構成し、イメージなどの提供を行っている。おおむね 1 日あたり 289 万件のリクエストがある。昨今は LLM 等によるアクセスが増え、大幅な増加になった。

つくば WIDE において、このプロジェクトは最も大きなトラフィックの発生を伴うプロジェクトである。

2.2 学内向け中規模ネットワークの運用

産学間連携推進室のほか、学術情報メディアセンター、情報科学類 WORD 編集部、新城研究室にまたがる学内中規模ネットワークを運用している。このネットワークは学内ネットワークとは独立した実験環境が必要な場合や NOC ではない各組織の部屋内でサーバーを運用したい場合などに使用している。また、このネットワークの運用を通してネットワーク運用に関する技術や新しい知見の習得を目指している。

3 構成

3.1 対外線接続

つくば NOC と WIDE バックボーンとの間はレイヤ 2 で接続している。接続は筑波大学学術情報メディアセンター内のアクセススイッチと WIDE プロジェクトバックボーンとの間では SINET の L2 接続サービスを、学術情報メディアセンター内と産学間連携推進室内を結ぶ通信は学内ネットワークの光ファイバーをお借りして接続している。

なお、学内ネットワークの借用と SINET の L2 接続サービスの利用は学術情報メディアセンターの多大な支援により実現しているものである。

3.2 経路

3.2.1 アドレスレンジ

IP アドレスは以下の範囲のアドレスを使用している。

- IPv4 アドレス: 203.178.132.64/26
- IPv6 アドレス: 2001:200:0:7C00::/56, 2001:200:1C8::/48

3.2.2 経路交換

上記の対外線に関する記載の通り、筑波大学及び SINET とは直接的にルーティングを行える箇所は存在せず、経路交換は行っていない。経路交換は WIDE バックボーンとの間でのみ行っている。ルーティングプロトコルは IPv4 接続においては OSPF を、IPv6 接続においては OSPFv3 を用いている。

4 新規メンバー

ここ最近でつくば WIDE NOC メンバーとして、産学間連携推進室外の高橋（人文学類 2 年）・臼井（情報科学類 2 年）・御園（情報科学類 2 年）が新しく加入した。

近年、Web 系などのフロントエンジニアばかりが着目されているように感じるが、インターネットを支えるバックエンド・インフラに興味を持ってくれたことは喜ばしいことである。

ノウハウの引き継ぎを継続して行うとともに、今後も若者へのリーチ活動にも力を注いでいかなければならない。

5　新しい Web サイト

間瀬によってつくば WIDE 向けのウェブサイトがデザイン・実装され、2026 年 2 月頃に公開された[1]。

ここには WIDE Project つくば NOC で運用するサービスを始めとした各種つくば NOC のお知らせ・広報や、新しく WIDE Project に参画したい筑波大学関係者向けの情報も掲載している。

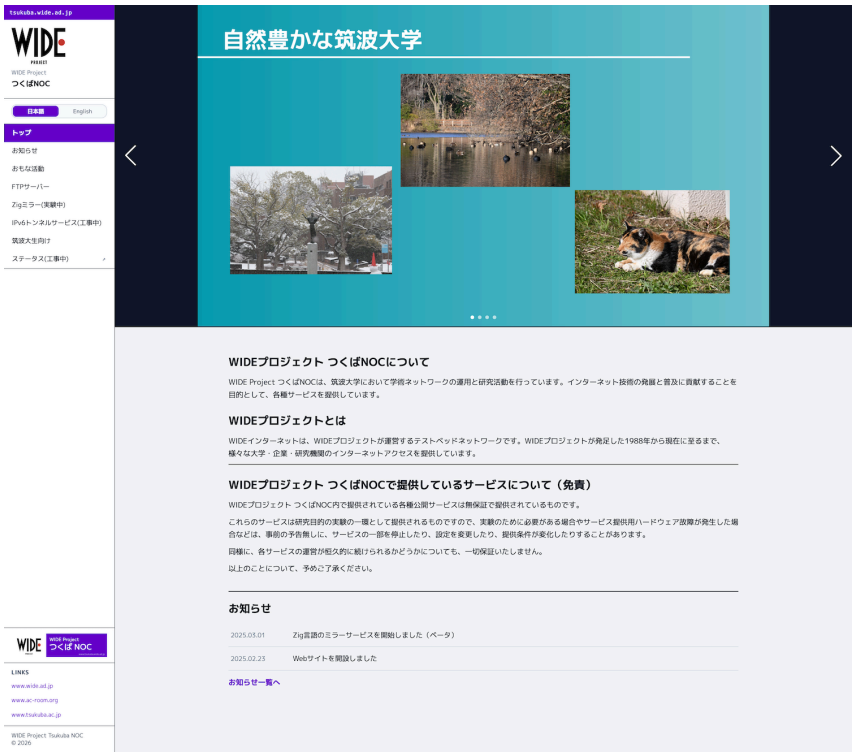


図 1: 2026/05/19 時点での https://www.tsukuba.wide.ad.jp のスクリーンショット

6　機材リプレース

つくば WIDE では、2025 年 11 月頃より、「式年遷宮」として、もうかなり古くなってしまった機材のリプレース・基盤ソフトウェアの入れ替え作業等を実施している。原状、まだまだ旧来のコア設備に依存した状態での構築が続いているが、完全な移行を目指し、新しく入ったメンバー等と共に引き続き努力する。

[1] https://www.tsukuba.wide.ad.jp でアクセスできる。

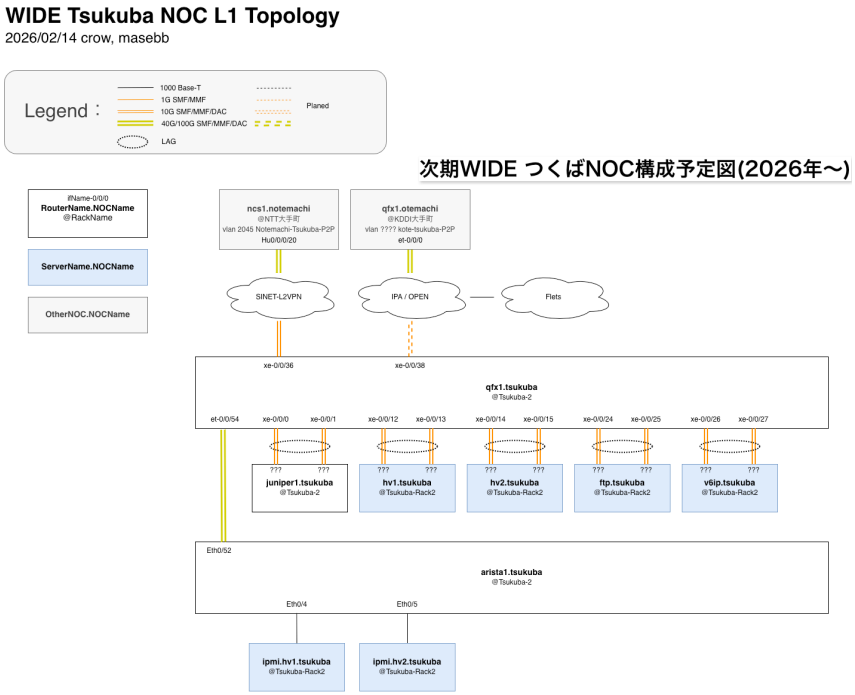


図 2: 次期 WIDE つくば NOC L1 構成予定図

「式年遷宮」にあたっては、現時点で以下の機材等を常日頃からお世話になっているソフトイーサー株式会社、産学間連携推進室・つくば WIDE 服部よりご提供・ご貸し出し頂いている。

- ソフトイーサ様
  - NEC Express5800/R120h-2M × 2 台（ハイパーバイザマシン）
  - その他必須品・テプラ等消耗品
- 服部
  - Juniper Networks NFX 250（ルータ）
  - Juniper Networks QFX 5100（40G イーサネットスイッチ）
  - Arista Networks DCS-7050TX-64（40G イーサネットスイッチ）
  - 40G DAC 等

このタイミングで近代の WIDE Project バックボーン共通の付番ルールなどその他風習に従う等の追従、WIDE Project バックボーンの一員として多くの NOC が使っている WIDE 共通基盤上の監視基盤での監視の実施等をする等の予定がある。

図 3: 導入された 2U サーバ二台（hv と書かれているもの）

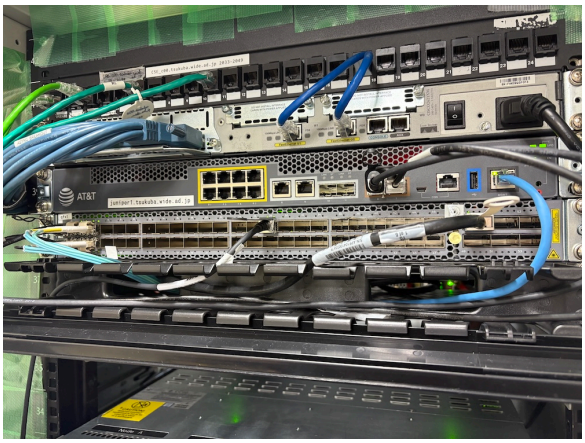


図 4: 導入されたルータ・40G スイッチ

7　WIDE Project 内での活動

近年、つくば NOC メンバーは WIDE メンバーの一員として、WIDE メンバー内で毎年数回、各会場で行われる WIDE Camp へ参加し、最新規格等・動向に関する情報収集・つくば NOC に関する発信等（参考: 図 6）を積極的に行っている。ただ会議に参加するだけではなく、Camp 内でネットワークを構築し、参加者にインターネット接続を提供したり、それらを生かして会場内で最先端の規格等に関する研究を行う Camp-NET の構築への参加も行っている[2]。

Camp-NET に関しては、2026 年度の NET 長を間瀬が務めている[3]。

図 5: 2026 年 2 月に行われた WIDE Camp-NET のホットステージ（慶應義塾大学 湘南藤沢キャンパスで実施）

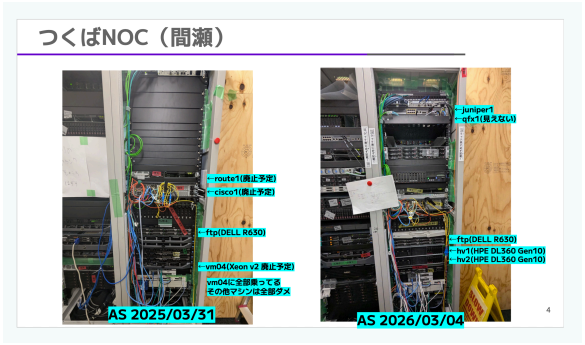


図 6: つくば NOC の近況についての発表資料（一部）

[2]画像 図 5 の構築には筑波大から中野・間瀬・高橋が参加した。
[3]筑波大からの Camp-NET 長は初めてである。

8　その他 : つくば NOC の今後

8.1　筑波大学学内との直接通信

筑波大学学内での研究活動でもつくば NOC のパブリックミラーサービスを利用しているという声を時々聞く。しかし、現状、つくば NOC で行っているその他のサービスも含めて、通信先は学内にあるにも関わらず、筑波大学学内からの通信はつくばから出て、大手町のデータセンターを一度経由し、つくばへ戻ってきている。

これは双方の他所行きの通信を圧迫するだけではなく、遅延の増加も招き非常に非効率である。そこで筑波大学と経路交換を行い、学内で完結可能な通信は学内で完結させたいと考えている。これにより、当パブリックミラーで提供しているソフトウェアを利用している研究ではより高速に環境構築やソフトウェア更新などができるようになり、通信速度に研究活動が律速される機会が減少し、研究支援につながると考えている。

8.2　対外回線の広帯域化・それに耐えうるソフトウェアルータの開発

今回の機材リプレースにより、NOC 内 LAN では 40GbE を利用できる環境が整いつつある。今後は、筑波大学学術情報メディアセンターとの接続についても 40Gbps 以上へ広帯域化し、NOC 内外の帯域差を縮小していきたい。

一方で、リプレース後も対外接続を収容するルータは Juniper Networks NFX 250 であり、最大接続帯域は 10Gbps に留まる。さらに、100Gbps 級の回線を収容しフルルートを扱える商用ルータは、価格・入手性の面からつくば NOC で容易に導入できるものではない。

そこで、汎用サーバと高速 NIC を組み合わせたソフトウェアルータにより、広帯域な対外接続を扱う方法を検討・開発したい。専用機材に頼り切るのではなく、汎用ハードウェアとソフトウェア実装で課題を解くことは、つくば NOC らしい実験的な取り組みであると考えている。

8.3　旧来サービスの復刻・新規サービスのホスト等

新規サービスホストの実例として、Zig 言語のコミュニティミラーサービスを開始した。これは、Zig Software Foundation がバイナリ配布基盤を AWS からセルフホストへ移行した一方で、日本国内からのダウンロードが遅いという声が周囲の利用者からあり、それならばつくば NOC でミラーを提供できるのではないかと考えたことがきっかけである。現在は `zig.tsukuba.wide.ad.jp` にて実験的に提供しており、公式のコミュニティミラー一覧にも一時掲載されていた[4]が、引き続き各種 Zig ツールから利用可能なミラーとして運用している。

[4]現在（2026/05/20）、一覧から外れており原因等を調査中である。

また、従来から運用しているFTPミラーサービスについても、各プロジェクトがCDNへ移行する流れはあるものの、前述のリクエスト数・トラフィック統計[5]にも表れている通り、現在でも十分な需要がある。ストレージ容量にはまだ余裕があるため、今後も需要のあるコンテンツを追加し、より多くのソフトウェア配布を支援していきたい。

さらに、単一拠点でのミラーに留まらず、他リージョンへの分散配置や、地域性を考慮した配信の仕組みも検討したい。

加えて、かつてつくばNOCで運用されていたグローバル・固定IPv6アドレス割当型トンネル接続実験サービスである`v6ip.tsukuba.wide.ad.jp`についても、復刻を検討したい。現在は一般家庭やクラウドでもIPv6接続性が得やすくなった一方で、固定IPv6アドレスやトンネル接続を用いた実験環境には依然として価値がある。つくばNOCらしい実験サービスとして再び提供できる形を模索したい。

[5] https://ftp.tsukuba.wide.ad.jp/status/ にて閲覧可能

Media over IP 技術に関する取り組み・研究

情報学群 情報科学類 間瀬 太陽（s2410334@u.tsukuba.ac.jp)

概要

本稿では、筑波大学の学園祭である雙峰祭における複数拠点配信と自営ネットワーク構築を背景として、Media over IP 技術、特に非圧縮映像を扱う SMPTE ST 2110-20 のソフトウェア実装を検証した。100GbE NIC を搭載した 2 台の汎用 PC を用い、ST 2110 対応 FFmpeg fork、OpenVisualCloud Media Transport Library、NVIDIA Rivermax を対象に、4:2:2 8bit 映像の送受信を試用した。その結果、通常のソケット処理に近い実装では 4K 以上で実時間性に限界が見られた一方、DPDK や Rivermax を用いる実装では 8K 60p 条件でも高い到達可能性を確認した。本稿は、学園祭等の学生主体の運用で広帯域ネットワークを活用し、将来的にソフトウェアベースの映像制作基盤へ発展させるための基礎検証として位置づける。

1　Media over IP 技術とは

Media over IP とは、主にテレビ局等の映像業界で用いられる用語で、その名の通り映像・音声・字幕・タリー・機器制御など、従来は個別の専用ケーブルで扱われていたものを我々が普段使うような IP ネットワークで伝送しようとする技術の俗称である。従来の映像制作では HDMI や SDI(同軸ケーブル)・RS-232C 等の専用インターフェースが中心であり、大規模な設備になると、電話交換機と同様の回線スイッチング設備が用意され、それを介してメディアをやりとりすることもある。

これに対して Media over IP では、信号はネットワーク上のフローとして扱われるため、送信元と受信先の関係をソフトウェア的に切り替えたり、マルチキャスト等によって複数の受信先へ同時に分配したり、離れた場所にある機器同士を同一の制作系として統合したりしやすい。

とくに近年は、リモートプロダクション、分散制作、ソフトウェアベースの柔軟な映像処理といった要求の高まりにより、Media over IP の重要性が増している。

また、4K・8K コンテンツの需要に関しては今までの映像伝送のスタンダードであった SDI ケーブルに限界が来ているという点も挙げられる。4K 映像は 12G-SDI という規格が誕生したことによって 1 本の SDI ケーブルでやり取りすることが可能である。ただ、高品質・太い SDI ケーブルを使う必要等があり、非常に扱いづらい。

図 1 は Blackmagic Design 社が販売している 8K スイッチャー（映像切換器）の I/O 部分である。この機種では 8K 入力を最大 10 系統、8K 出力を 6 系統備えている。ただ、スイッチング映像の入出力だけで 64 個もの大量の同軸ケーブル端子が搭載されている通り、SDI ケーブルで 8K 映像をやり取りする場合、

一般的にクアッドリンク 12G-SDI という SDI ケーブルを 4 つ束ねて帯域を確保する方式が用いられるという非効率性がある。

図 1: ATEM Constellation 8K の背面 I/O [1] より引用

その点、光ファイバー等を用いたものであれば SDI を用いた最大 12Gbps の伝送よりもより高速・広帯域の通信を細い光ファイバケーブル 1 本で通せたりするため、利点がある。また、データセンター等で一般的に使われている光モジュール・高速 Ethernet スイッチが使えるため、システム導入費用も一段と抑えられる。

本稿では、基本的に映像・音声の伝送を中心に取り扱っていく。Media over IP と一口に言っても、その実現方式は一様ではない。

代表的なものとしては、比較的汎用的な環境でも導入しやすい圧縮系の方式と、SDI ケーブルの代替として用い、放送設備などで高い品質と厳密な同期を重視する非圧縮系の方式に大別できる。前者の例として NDI ・ SRT 、古典的なもので言えば RTMP 等のプロトコルがあり、IP ネットワーク上で映像・音声伝送を行いながら、圧縮によってネットワーク帯域の消費を抑えられるため、ソフトウェアや多様な機器との接続性を確保しやすい。通常用途では前者を用いるが、以下の背景より本稿は後者について研究をするものとなっている。

2　本研究の背景（筑波大学 学園祭での取り組み）

本研究の背景には、筑波大学の学園祭である雙峰祭における映像制作およびインターネット生配信の運用がある。その運営は学園祭実行委員会によって担われており、筆者はその中の情報メディアシステム局に所属していた。

また、本研究は、産学間連携推進室からの機材貸与や技術協力を受けながら、学生主体の現場で広帯域ネットワークと映像伝送技術を実際に扱った取り組みでもある。そのため本章では、後段の ST 2110 検証に至る前提として、雙峰祭で構築した配信基盤と自営ネットワークの位置づけを述べる。

雙峰祭におけるインターネット生配信の取り組みは古く、2004 年から学生主体で継続されてきた。配信サイトを独自に構築・ホストし、年ごとに機能や運用方法を改良してきた経緯があり、必要な仕組みを自ら設計・実装しながら発展してきた点に特徴がある。2024 年度には著作権料等の事情を背景として、それまで直近で用いていた YouTube 配信をやめ、独自プラットフォームによる配信へ移行した。

2025 年度の雙峰祭では、前夜祭・本祭 1 日目・本祭 2 日目（後夜祭を含む）の 3 日間にわたり、ステージ企画が行われている時間帯を通しでインターネット生配信が実施された。今年度の運用における大きな変更点は、最大 4 映像の同時配信を行った点にある。従来はメインステージである UNITED ステージの配信に注力していたが、2025 年度は残る 2 つのステージについても配信対象とし、さらに学園祭のステージ以外の雰囲気も伝え、集客等に資するための移動生配信も行った。これにより、単一会場の映像を安定して配信するだけでなく、複数拠点の映像を並行して扱いながら運用全体を成立させることが求められるようになった。

単純に考えれば必要となる労力は 3 倍から 4 倍に増加する一方で、人員は同じ割合では増えていない。さらに、筑波大学のキャンパスは広く、各ステージ間の距離も離れているため、各会場に UNITED ステージと同様の人数をそのまま配置する運用には限界がある。

2025 年度の雙峰祭の配信運用では、複数会場・複数映像を少人数で扱うための省力化と、配信基盤全体の効率化が強く求められていた。そこで大学のネットワーク・自営ネットワークを組み合わせ、それらを用い Media over IP 技術を多用することによって大幅な省力化に成功した。

具体的には以下のような技術等を使用した。

- NDI を用いた映像の伝送
  - 全ステージの映像を 30Mbps〜数百 Mbps のレートで安定して伝送した。
  - 詳細は 図 2 を参照。
- IP による PTZ（パン・チルト・ズーム）カメラの本部からの制御 (図 5)
  - これらによって本部から遠く、1 カメラ構成で十分である「会館ステージ」「1A ステージ」の配信時現場完全無人化に成功した。
- 高速・汎用ファイルサーバの運用 (図 6)
  - メインステージに 3 つあるカメラの REC ファイル転送をネットワーク経由で行うようにした。
  - 昨年度までは、SD カードを徒歩によって 150m ほどある本部まで運んでいた。
  - これらの映像は学園祭最終日に流される「ダイジェスト映像」を制作するために用いられる。制作陣は Windows・macOS を使っているため、これらから接続しやすい SMB サーバを選択した。
- モバイル回線ボンディング機器（LiveU）を用いた移動配信の実現
  - 商用ソリューション（LiveU）をできるだけ安価に我々のフローに乗せた。
  - LiveU 側サーバから送られるデータを一度さくらインターネット上の VPS で受信し、再度送信・テロップ等を付加し再送信。
- SIP プロトコルを利用した IP 電話
  - 広大なキャンパス内における即時性・確実性があるコミュニケーション手段として用いられた
  - デジタル簡易無線でのやり取りができないシーン・屋内にいる協力外部団体との頻繁なやり取り等で活躍した

これらによって、人員は移動配信・基本的に設営・撤収時を除き、本部・UNITED ステージ間のみに存在することとなり、非常に人員管理がシンプルになった。

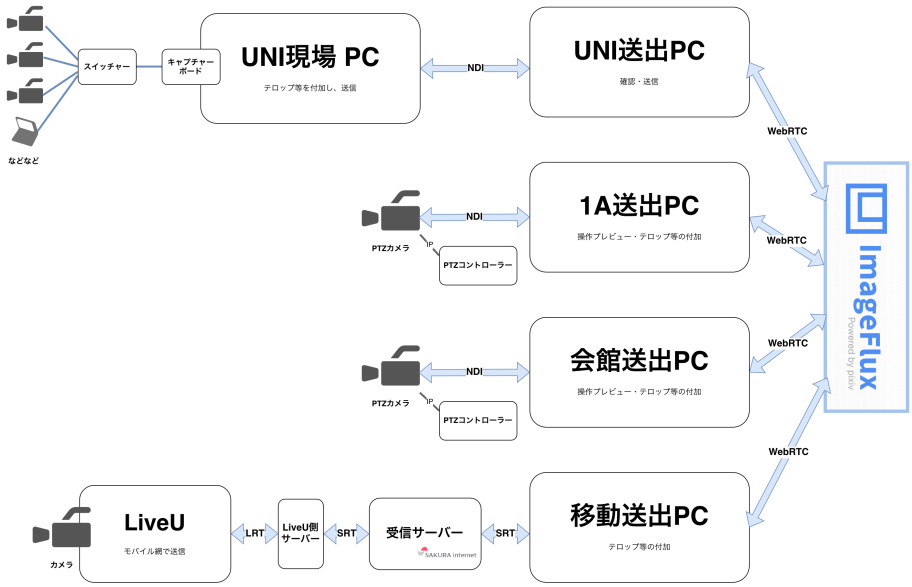


図 2: 今年度の映像・音声に関するフロー図

図 3: UNITED ステージ(メインステージ)の現場

図 4: UNITED ステージ(メインステージ)にある 40G スイッチ等設備

図 5: 1A ステージ・会館ステージ(PTZ カメラ)の操作卓

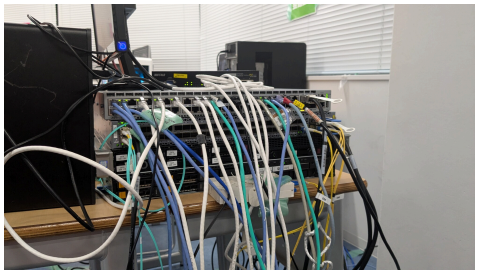


図 6: 本部に設置されたインフラ設備

2.1　自営ネットワークについて

本年度の雙峰祭では、複数ステージの映像を本部に集約し、配信・収録・監視をまとめて扱うために、学園祭実行委員会の情報メディアシステム局が大学ネットワークを活用しつつ、広帯域が必要となるところには L1 レベルで大学ネットワークから独立した自営ネットワークを構築した。

具体的な構成としては、本部教室に基幹スイッチ、配信 PC、ストレージサーバ・汎用サーバ等を集約し、各ステージや学内ネットワーク設備までを光ファイバーで接続した。大学既設の光ファイバー網を借用しつつ、本部までの一部区間は学生側で敷設・成端を行い、学園祭期間中の制作系として利用できる物理網を用意した。(図 7)

屋外の UNITED ステージへは本部と 40G-LLR トランシーバを用いて 40G リンクで接続した。試験的に産学間連携推進室の WIDE Project ラックにも同様の 40G リンクを構成した。広帯域の利用による既存大学ネットワークへの影響を避けるため、大学コアスイッチまで L1 レベルで接続し、そこに SFP+LR トランシーバーによる 10G リンクで接続した。

その他ステージ（1A ステージ・大学会館ステージ）においては大学の情報コンセント（1G）を用い、上記大学コアスイッチを経由し映像等を受け取った。

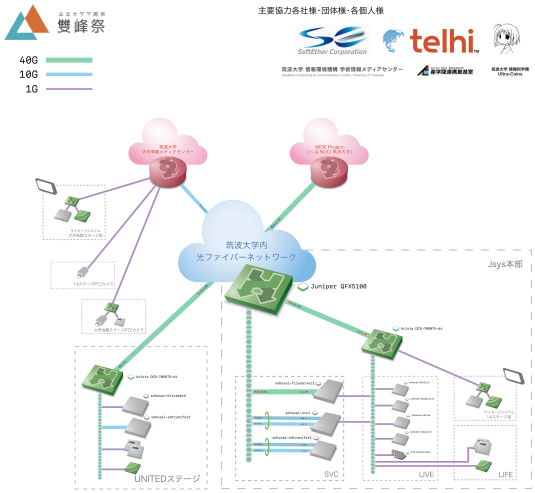


図 7: 全体ネットワーク図

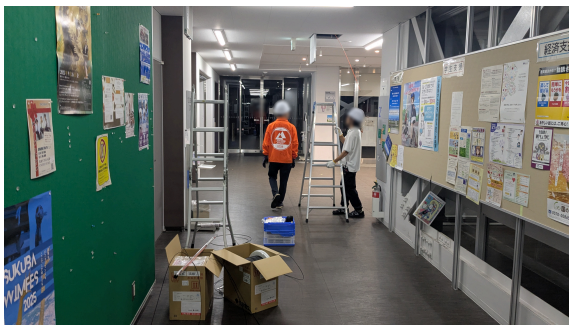


図 8: ファイバー敷設工事の写真

このように構築した自営ネットワークだが、圧縮系の Media over IP 技術を使用しているのもあり、ファイル転送等にしか最大 40Gbps にもなるネットワークをフル活用できていなかった。

そのため、「映像伝送に圧縮系を用いるのではなく、あえて広帯域を利用する非圧縮系を用いてみたい」というのが本研究の始まりである。また、内部の映像に NDI 等の圧縮方式を用いると、最終的な

配信時に再度エンコードされるため、複数回、特に初段での符号化・復号化に伴う画質劣化が生じうる[2][1]。非圧縮系を用いることは、この問題の解決にもつながり、より高品質な映像の配信にもつながる。

非圧縮系については、かねてより存在は知っていたが、前述の通り、消費する帯域等の関係上、汎用PC上で、学生・アマチュアの立場から扱う先行事例を十分に確認できておらず、また変換をする機器であるエンコーダ・デコーダ等[2]の専用機材は非常に高価である。

これを、前述のような汎用PCの上に高速NICを搭載するだけで済む、操作も簡易なソフトウェアを作りたいと考え、次項の研究を行った。

[1] ここではNDIそのものではなく、圧縮済み映像を再度圧縮する一般的な問題として参照している。
[2] ゲートウェイボックスとも呼ばれる

3　汎用 PC 上で動作する ST 2110 ソフトウェア既存実装の試用と比較

3.1　SMPTE ST 2110 規格について

SMPTE ST 2110 規格群があり、映像は ST 2110-20、音声は ST 2110-30、時刻同期は ST 2059、その他補助データは ST 2110-40 といった形で、それぞれを独立したストリーム・規格として扱い、共通の時刻基準に基づいて同期させることで、制作設備を IP 化するための基盤となっている。

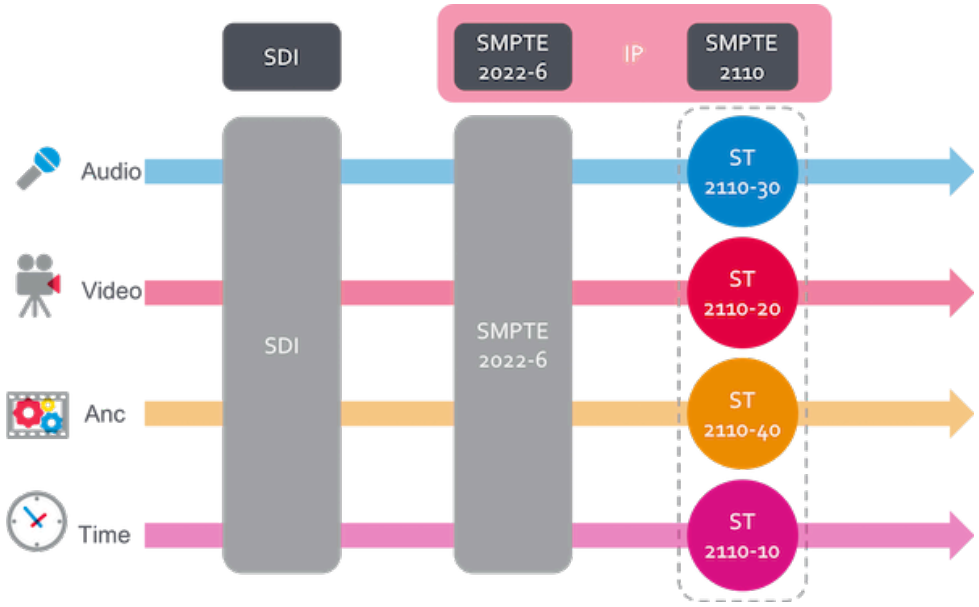


図 9: 映像の IP 化に関する SMPTE 規格の遍歴 [3] より引用

一方で、ST 2110 を実際のシステムとして安定に成立させることは容易ではない。特に ST 2110-20（映像）については RTP パケットによる非圧縮映像の伝送によって常に高い帯域と高い PPS 処理性能を要求する点にある。非圧縮映像では 1 ストリームあたりの通信量が大きく、しかも多数のパケットを短時間に連続して処理しなければならないため、単にネットワークに接続できるだけでは不十分である。

そのため現在の ST 2110 ベースの運用は、FPGA・ASIC 等で実装され、IP コンバータ、マスタークロック、モニタリング装置、ハードウェアスイッチャーなどを組み合わせた、比較的大規模で非常に高価な構成として実現されることが多い。

3.2　非圧縮映像の送信・受信に関するソフトウェア実装の課題点

本稿で主に扱う対象は、このうち ST 2110-20 のソフトウェア実装である。ここで主眼となるのは、まず非圧縮の ST 2110-20 映像をソフトウェアで正しく送受信できることである。本稿で特に難所となるのは、汎用 PC が非圧縮映像ストリームを安定して受け取り、また送り出せるかという点にある。ホスト側の NIC、OS、メモリ帯域、ユーザ空間の実装、さらには後段の映像処理系まで含めて、全体がその通信量とパケットレートに安定して耐えられなければ実用にはならない。

ST 2110-20 では映像を非圧縮のまま RTP ストリームとして扱うため、圧縮系の Media over IP と比べて必要帯域は大きくなる。4:2:2 8bit の場合、2 画素を 4 byte で表すため、1 画素あたり 16 bit となる。したがって、映像そのもののデータ量だけでも $W \times H \times f \times 16$ bit/s の帯域が必要となる。表 1 にまとめている。

表 1 の PPS[3]計算では MTU を 9000 byte とし、IPv4 ヘッダ 20 byte、UDP ヘッダ 8 byte、RTP 固定ヘッダ 12 byte、RFC 4175 の最小ペイロードヘッダ 8 byte を差し引いた 8952 byte を 1 パケットに格納できる映像データ量としている。

形式	純粋な映像データ量	PPS 概算 (MTU 9000 時)
Full HD 30p	0.995 Gbps	約 13.9 kpps
Full HD 60p	1.991 Gbps	約 27.8 kpps
4K 30p	3.981 Gbps	約 55.6 kpps
4K 60p	7.963 Gbps	約 111.2 kpps
8K 30p	15.925 Gbps	約 222.4 kpps
8K 60p	31.850 Gbps	約 444.7 kpps

表 1: 4:2:2 8bit 非圧縮映像における純粋な映像データ量と PPS の目安

表 1 に示す通り、4K 60p では映像データだけで 8 Gbps 近い帯域と、MTU 9000 のジャンボフレームを用いた場合でも、8K 60p では 1 ストリームで 40 万 pps を超える処理が必要になる。問題はこれらが瞬間的なのではなく、持続的に続くものであるということである。このため、ソフトウェア実装では NIC の公称帯域だけでなく、受信キュー、割り込み処理、メモリコピー、ユーザ空間への受け渡し、後段の映像処理まで含めた全体の設計が問題となる。

一般的な汎用 OS では、NIC が受信したパケットをドライバやカーネルのネットワークスタックが処理し、ソケット等を通じてユーザ空間へ渡す。高 PPS では割り込み、メモリコピー等の負荷が支配的になり、通常のソケット API では通常処理もネットワークに関する処理もままならなくなってしまう。

このため、汎用 PC で高 PPS・広帯域通信を扱うために、DPDK 等に代表される高速パケット入出力フレームワークを用いたソリューションを用いる必要がある。

3.3 ST 2110-20 既存実装の試用と比較

ここでは、汎用 PC 上で ST 2110-20 を扱う既存実装を試用し、前節で述べた高 PPS・広帯域処理にどの程度対応できるかを比較する。比較対象として、CPU 処理経路込みの参考実装である FFmpeg をパッチしたもの、DPDK を用いた OpenVisualCloud Media Transport Library、ST 2110 向けに特化した NVIDIA Rivermax を取り上げる。

[3]Packet per Second

3.4 比較対象

3.4.1 FFmpeg をパッチしたもの

1 つ目は、ST 2110 対応 FFmpeg fork である cbcrc/FFmpeg [4] を用いた実装である。 本検証ではコミット番号 `3d4f71d3853ad17ea465d1e3eb4fe86c8b2400c2` のものを使用した。

この実装は 7 年前ほどで更新等が途絶えているが、ある程度単純なものとして非常に参考になる。本稿では、特に高 PPS・広帯域等への対策フレームワーク等を用いていない実装の限界を確認するための参考対象として扱う。 なお、ST 2110-20 の送出には対応していないため、送信側には GStreamer の `videotestsrc` と `rtpvrawpay` を用いた。実際の送信・受信フローは後述する。

3.4.2 OpenVisualCloud Media Transport Library

2 つ目は、Intel の OpenVisualCloud Media Transport Library（以下 MTL）である。MTL は DPDK を用いた高速パケット入出力を前提とし、ST 2110 などのメディア伝送を汎用 PC 上で扱うためのライブラリである。現在も頻繁にメンテナンスがなされている。DPDK によって ST 2110 パケット受信・送信のバースト的に行うようになっている。

本検証では OpenVisualCloud Media Transport Library [5] の main ブランチ（不安定版）のコミット番号 `58eaa08ade27d1f6fe5a162c0a2f6110f305f041` のものを使用した。

MTL は様々なエコシステムも作られており、FFmpeg のプラグインとしてのものもあったり、GStreamer の Element としてのものもあったりする。だが、巨大なコードベースから構成される FFmpeg・GStreamer の内部のパイプラインでの律速等が起きる可能性を否定できないため、基本的にはフレームワークを用いる際のサンプル実装[4]から本検証用に一部変更したものを使用した。

また、テストされた推奨 NIC として Intel E830・Intel E810 が挙げられているが、今回の検証では ConnectX シリーズを用いた。

MTL を製品等に組み込み、商用ソリューションとして販売している例は、残念ながら探した限り見つからなかった。

3.4.3 NVIDIA Rivermax

3 つ目は NVIDIA Rivermax Media Library [8] である。Rivermax はライセンス制のクローズドソース SDK である。ConnectX-5[5]以降の ConnectX NIC・BlueField DPU を前提とした高性能なパケット処理基盤であり、ST 2110 を含むメディア over IP 用途に特化した実装として位置づけられる。こちらも、ネットワークスタックや通常のソケット API ではなく、NIC ドライバの独自の API を用いて NIC のバッファ領域に

[4]送信サンプル [6] および受信サンプル [7] を参照。
[5]ConnectX-5 は EOL とされている。だが、少なくとも`v1.80.24`時点では動作を確認できている。

ユーザランドから書き込むことができ、バースト的にパケットを送信することで高 PPS・広帯域の映像伝送を扱う。

また、DPDK に依存、つまり Linux に依存している MTL とは異なり、Windows に正式対応しているのもポイントである。

本検証では試用版ライセンスを利用し、`v1.80.24` を利用した。

性能面では高い一方で、対応 NIC や SDK に依存するため、オープンなソフトウェア実装として改造・発展させる対象というよりは、ST 2110 に特化した商用系実装の到達点を確認する位置づけである。NVIDIA Rivermax は商用製品にも多く組み込まれており、映像業界的にも市民権を得つつあると思われる。

具体的には生放送などの時に複数のカメラ等の映像ソースを切り替えるスイッチャーとして Panasonic Connect の「KAIROS」（図 10 図 11）、映像の確認等に使用するマルチビューワーで株式会社 芙蓉ビデオエイジェンシーの UMV シリーズ（図 12）等が挙げられる。

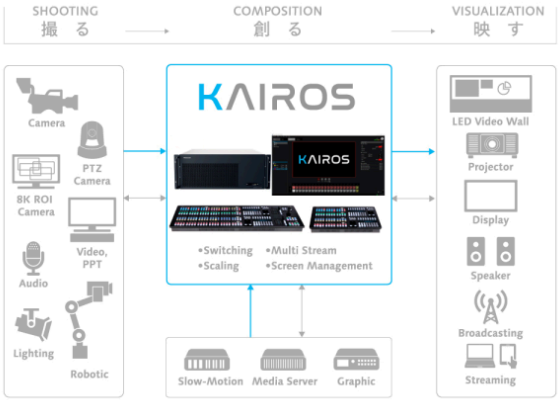


図 10: KAIROS の製品外観 [9] より引用

図 11: KAIROS Core 2000 Mainframe [10] より引用

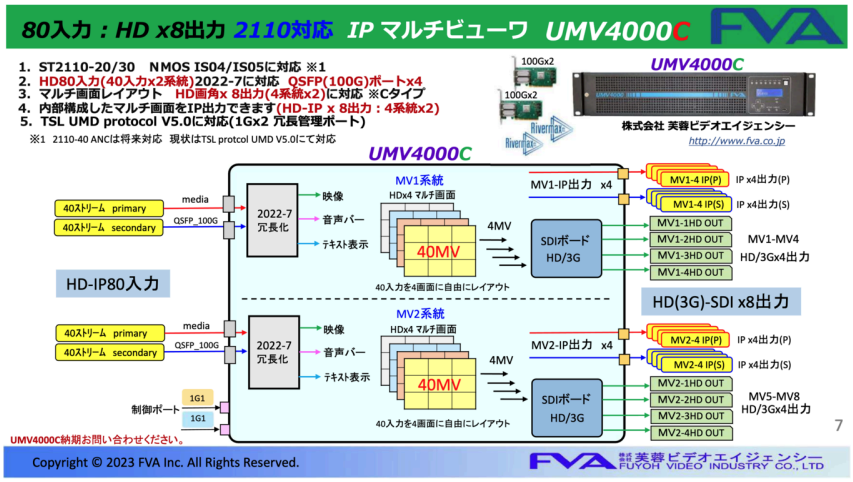


図 12: UMV シリーズの製品外観 @fva-umv-catalog より引用

特に Panasonic Connect の KAIROS は 図 11 を見れば分かる通り、汎用サーバの形をしており、PCIe ボードが搭載される箇所に NVIDIA 社製の GPU と思われるもの・SDI の入出力端子・そして ConnectX と思われるものが接続されている。これらによって今までの FPGA・ASIC 等で実装されてきた従来スイッチャーの代替をするというものである。

今までの FPGA 等によって実装されてきたスイッチャーの欠点として、映像を重ねられる場所が決められている・映像に施せる処理が限られている・対応入力・出力解像度以外の映像を受け付けられないなどの欠点があった。だが、ソフトウェアベース・汎用 PC ベースにすることによって、GPU パワーが許す範囲で様々な映像を自由な位置に重ねたり、処理を柔軟に追加したりでき、入出力解像度の自由度も高くなる。とくに後者が重要で、出力解像度が自由であることによってスタジアムの巨大 LED パネル・様々なサイネージ等の映像をリアルタイムの映像から作り出すことができ、それらに採用されているという事例もある。これは、ソフトウェアベース化の利点が表れている例である。[6]

3.5 比較条件

本稿では、主に 4:2:2 8bit の非圧縮映像を対象とし、受信した映像をディスクへ保存しない 60 秒試験を中心に比較する。そのため、基本的にはメトリクスベースの比較となっており、視覚的な評価等は行っていない。また、ソフトウェアによってメトリクス指標が異なる。

したがって本節では、細かな品質の優劣ではなく、各実装が参考としてどの程度の解像度・フレームレートまで実用的に扱えるか、また今後のソフトウェア実装として使いやすいかを中心に比較する。

[6]ただ、ハードウェア的に作られているものと比べて安定性は欠けると思われる。

3.6 評価観点

評価観点は、導入のしやすさ、必要なハードウェア・ソフトウェア条件、4K 60 以上での実時間性、8K への拡張性、実装・改造のしやすさ、今後発展させやすいかである。

なお、相互運用性については 1. 商用で使われている送信機・受信機等を正としたリファレンス実装がない 2. 上記ソフトウェアもそこまで隠蔽されているものではなく、物によっては自前で RTP ヘッダを組み立てる必要なもの等もあるため、本稿ではあまり比較せず、今後の課題とする。

3.7 試験環境

試験は、100GbE NIC を搭載した 2 台の Linux ホストを直結し、片方を送信側、もう片方を受信側として行った。主な環境を 表 2 に示す。

項目	端末 1（主として受信側として利用）	端末 2（主として送信側として利用）
OS	Ubuntu 24.04.4 LTS	Ubuntu 24.04.4 LTS
CPU	Intel Core i9-9900K	Intel Core i9-9900K
RAM	約 46 GiB	約 62 GiB
試験用 NIC	ConnectX-5	ConnectX-6 Dx
HugePages	2 MiB: 6144 1 GiB: 12	2 MiB: 2048 1 GiB: 6

表 2: 検証に用いた 2 台の PC の概要

MTL と Rivermax については、1 GiB HugePages を有効にした状態の結果を後述の表へ反映している。1 GiB HugePages についてだが、2 MiB HugePages のみでは MTL の高解像度条件でメモリ確保周りの警告が出ており、1 GiB HugePages を有効にしたという経緯がある。4K・8K となってくると映像自体を HugePages に載せるのは難しいと判断し 2 MiB HugePages のみにしていたが、以下で述べる通り 1 GiB HugePages は MTL に対して非常に有効だった。

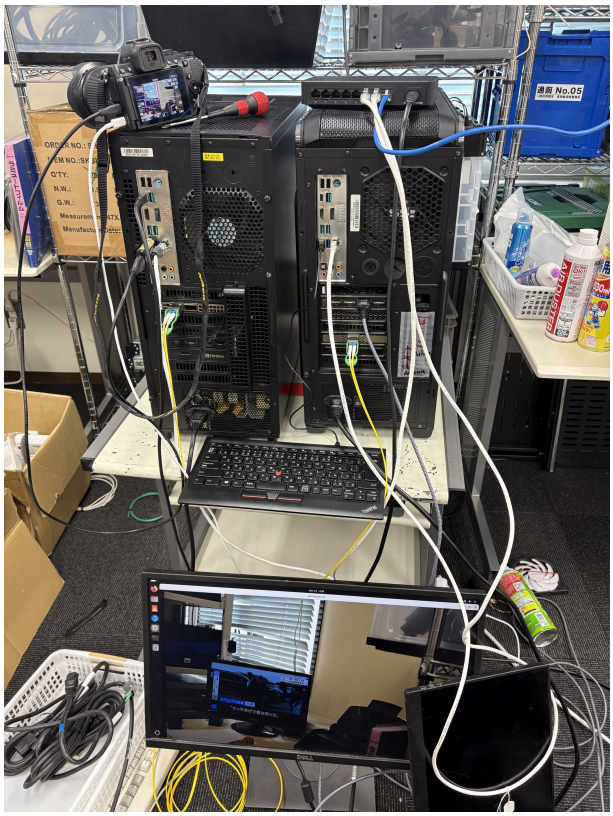


図 13: 検証環境外観

3.8　送信素材生成・送受信フロー

本検証では、いずれも「ST 2110-20 相当の映像を送信し、受信側で一定時間取り切れるか」を見るという点では共通している。一方で、送信前のテスト映像の作り方は実装ごとに異なる。FFmpeg をパッチしたものは GStreamer がテスト映像を逐次生成し、そのまま RTP の非圧縮映像として送出する。これに対して MTL と Rivermax では、送信前に FFmpeg で raw 素材を作成し、それを送信プログラムへ渡す構造になっている。

このため、同じ解像度・フレームレートの検証であっても、送信側で比較している層は完全には同じではない。FFmpeg をパッチしたものでは、テスト映像生成と RTP 化を含んだ経路を見ている。一方、MTL と Rivermax では、事前に生成した raw を送信プログラムへ渡しているため、送信時のテストパターン生成負荷は切り離されている。また、送信元の絵柄も完全には一致しておらず、FFmpeg をパッチしたものでは GStreamer の `videotestsrc`、MTL と Rivermax では FFmpeg の `testsrc2` を用いている。

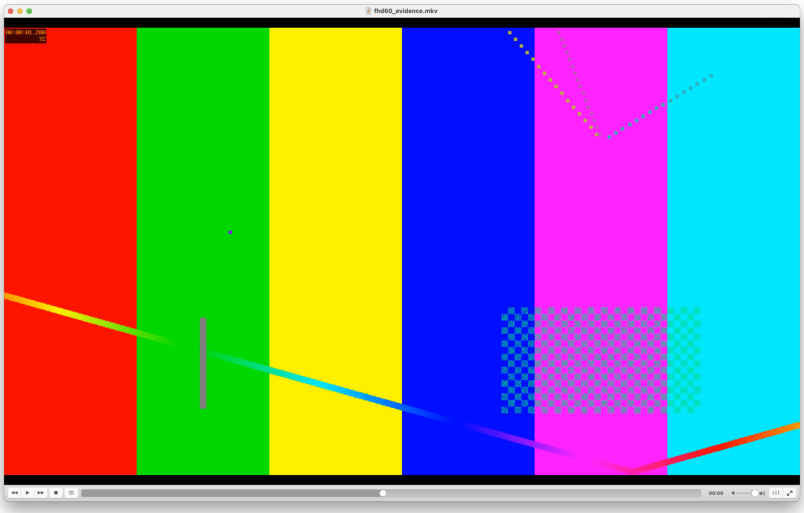


図 14: FullHD 60p の testsrc2 を MTL で伝送・保存した例

3.8.1 FFmpeg をパッチしたもの

FFmpeg をパッチしたものの検証では、送信前に raw ファイルを作らない。送信側では GStreamer がテスト映像を逐次生成し、それを `rtpvrawpay` で RTP の非圧縮映像としてマルチキャスト送出する。

送信側の代表的なコマンドは次の形である。

```
gst-launch-1.0 -q rtpbin name=rtpbin \
  videotestsrc is-live=true horizontal-speed=2 ! \
  video/x-raw,width=${width},height=${height},framerate=${fps}/1,format=UYVY ! \
  rtpvrawpay pt=102 ! queue ! \
  rtpbin.send_rtp_sink_0 rtpbin.send_rtp_src_0 ! queue ! \
  udpsink host=239.0.0.0 port=5005 multicast-iface=${iface}
```

受信側では、ST 2110 対応パッチを当てた FFmpeg fork を用いた。

この FFmpeg は、検証用スクリプト内で生成した SDP を標準入力から受け取り、保存せずに `null` 出力へ捨てる。これにより、ディスク書き込みではなく受信・復号処理側の到達可能性を確認した。

```
cat "${sdp_file}" | ~/cbcrc-FFmpeg/ffmpeg \
  -protocol_whitelist file,udp,rtp,pipe \
  -smpte2110_timestamp 1 \
  -f sdp -i pipe:0 -t ${duration_sec} \
  -f null -
```

3.8.2 MTL

MTL では、FFmpeg プラグインではなく、サンプル実装を基にした送受信プログラムへ raw 素材を渡す経路を扱った。送信素材は、事前に FFmpeg の `testsrc2` から raw として生成する。10 bit 4:2:2 では `yuv422p10le`、8 bit 4:2:2 では `uyvy422` を用いた。

raw 生成は概ね次の形で行った。

```
ffmpeg -f lavfi -i testsrc2=size=${video_size}:rate=${fps} \
  -vf format=${raw_pixel_format} \
  -frames:v ${source_frames} \
  -f rawvideo "${source_raw}" -y
```

送信側は、生成済み raw を `--tx-file` で渡して起動する。

```
sudo -E mtl_direct_bench_tx \
  --port ${tx_port} --sip ${tx_sip} --tx-ip ${rx_ip} \
  --udp-port ${udp_port} \
  --width ${width} --height ${height} --fps ${fps} \
  --queues ${tx_queues} --frames ${sender_frames} \
  --profile ${bench_profile} \
  --tx-file "${source_raw}"
```

受信側は、同じ解像度・フレームレート・プロファイルを指定し、保存なしの場合は `--output none` として起動する。

```
sudo -E mtl_direct_bench_rx \
  --port ${rx_port} --sip ${rx_sip} --rx-ip ${tx_ip} \
  --udp-port ${udp_port} \
  --width ${width} --height ${height} --fps ${fps} \
  --queues ${rx_queues} --frames ${frames} \
  --profile ${bench_profile} \
  --output none
```

3.8.3 Rivermax

Rivermax の検証でも、送信前に FFmpeg で繰り返し再生用の raw 素材を生成した。本稿の比較では MTL と同様に FFmpeg の `testsrc2` から `uyvy422` の raw を作成し、それを `media_sender` に渡した。

raw 生成は MTL と同じ形で行った。Rivermax では `raw_pixel_format` を `uyvy422` とし、`source_frames` は `fps * loop_seconds` とした。`loop_seconds` は通常 1 秒または 5 秒であり、送信時にはこの短い raw 素材を元に、後述の `--max-media-units` で指定したフレーム数まで送信する。

```
ffmpeg -f lavfi -i testsrc2=size=${video_size}:rate=${fps} \
  -vf format=${raw_pixel_format} \
  -frames:v ${source_frames} \
  -f rawvideo "${source_raw}" -y
```

送信側では NVIDIA の `media_sender` に SDP と raw ファイルを渡す。60 秒試験では、`--max-media-units` に渡す値を `fps * 60` として指定した。

```
media_sender \
  -s "${video_sdp}" \
  -m --register-memory \
  --max-media-units $((fps * 60)) \
  -f "${source_raw}"
```

受信側の保存なし試験では、SDP を指定して `media_receiver` を起動し、raw 保存は行わない。受信側では欠落数、RTP ヘッダ異常数、パケット数から推定されるフレームレートを確認した。

```
media_receiver \
  --sdp-file "${video_sdp}" \
  -i 192.168.0.1 \
  --memory --register-memory \
  -k ${receiver_packets} \
  --timestamp 0 \
  --header-size 20 --data-size 7680
```

3.9 試用結果

整理すると、結果は 表 3 のようになる。各実装で取得できる指標が異なるため、FFmpeg をパッチしたものでは FFmpeg ログのフレーム数と処理速度、MTL を用いたプログラムでは受信フレーム数、Rivermax では受信欠落の有無とフレームレートを示している。

映像については以下のとおりである。

- 4:2:2 8bit 各解像度
- 保存なし
- 各実装のテストパターンを 60 秒伝送する

形式	FFmpeg をパッチしたもの	MTL	Rivermax
Full HD 30p	1800/1800 speed 1x	受信 1800/1800 約 30.018 fps	受信欠落 0 約 30.003 fps
Full HD 60p	3600/3600 speed 1x	受信 3600/3600 約 60.017 fps	受信欠落 0 約 60.011 fps
4K 30p	1800/1800 speed 0.698x	受信 1800/1800 約 30.017 fps	受信欠落 0 約 30.004 fps
4K 60p	3600/3600 speed 0.348x	受信 3600/3600 約 60.020 fps	受信欠落 0 約 60.010 fps
8K 30p	932/1800 speed 0.101x	受信 1800/1800 約 30.021 fps	受信欠落 0 約 30.004 fps
8K 60p	928/3600 speed 0.0683x	受信 3600/3600 約 60.021 fps 送信時刻遅れ 1	受信欠落 0 約 56.526 fps

表 3: 4:2:2 8bit・保存なし・60 秒条件を中心とした実装比較。MTL を用いたプログラムは本稿で扱う片方向の結果のみを示す

FFmpeg をパッチしたものでは、Full HD 30p / 60p は 60 秒でも実時間で処理できた。4K 30p は要求フレーム数までは届くが、処理速度は 1.0x を下回る。4K 60p もフレーム数だけなら届くものの、処理速度はさらに低下した。8K では処理速度が遅いためジッターバッファが溢れ、破棄された結果要求フレーム数まで届かず、CPU 側の映像処理経路が明確に律速となった。

MTL を用いたプログラムは、Full HD 30p から 8K 60p まで要求した受信フレーム数を取り切っている。フレーム活動区間で見ると、各条件とも公称フレームレート付近で動作していた。送信したパケットが次に送るべきフレーム時刻に対して遅れている状態は、8K 60p 時に 1 回のみ発生している。本検証の範囲では、この発生頻度は結果全体を左右するほどではなかった。

Rivermax も高い結果を示した。表に示す範囲では受信欠落は見られず、Full HD 30p から 8K 30p までは目標フレームレート付近まで到達している。一方で、8K 60p では受信欠落は 0 だったものの、安定区間の推定フレームレートは約 56.5 fps に留まり、公称の 60 fps には届かなかった。

3.10 考察

FFmpeg をパッチしたものは参考実装として用い、実際、Full HD 以上では実時間処理が難しくなっている。CPU 等にかかる負荷も決して無視できるものではないが、通常のソケット処理に近い実装としては一定の性能を示している。なお、10 秒ほどであれば 4K 30p である程度動作することも確認している。また、FFmpeg は受信後、エンコードをかけての保存などが非常にやりやすく、また高速である。そのため、他のフレームワークから送出された ST 2110 パケットをエンコード・保存する等の用途である程度使えそうであるということも分かり、引き続き検証用途で使える可能性を秘めていた。

MTL を用いたプログラムは、少なくとも本検証で取得したメトリクス上では、商用ソリューションである Rivermax と近い水準に達していた。特に 8K 60p 条件では、MTL は要求フレーム数を取り切った一方、Rivermax は欠落 0 のまま推定フレームレートが約 56.5 fps に留まった。想定よりも差が小さく、今後の実装方針を考える上で重要な結果となった。

Rivermax は、全条件で受信欠落 0 という点では安定した結果を示した。ただし 8K 60p では公称フレームレートに届いておらず、今後はこの差が送信側、受信側、計測方法のいずれに由来するかを切り分ける必要がある。また、今回メトリクスとして出せてはいないが、MTL 等と比べて、Rivermax の方が有意にロードアベレージが低いという利点もある。

また、繰り返しになるがいずれの結果もメトリクスの値による比較であることに留意し、本項は参考程度としていただきたい。

4　今後の方針

ST 2110 関連ソフトウェアの検証について、学園祭が 10 月末にあり、その後今に至るまで学習をしながらの検証だったため、上記検証にはまだ多くの制約が残る。上記の基礎的検証についても続けつつ、まだまだ未検証のものが大量にある。大きく以下のものがあると考える。

1. 相互運用性についての確認
   - 何らかの機会でリファレンス実装のようなものと接続ができたらと考えている。
2. 動画の保存・視覚的評価
   - 現状、視覚的な評価をする際は`/dev/shm` に大量の受信した生データを保管し、後から FFmpeg 等で変換する等を行っている。この方式では 4K 60p〜の生データ書き込みが追いつかないという問題がある。
   - (H.264 等ではない軽量な)圧縮等も活用しつつ、将来的には大容量・高速ディスクを入手し、そこへの記憶も検討する。
3. NIC から GPU への GPUDirect RDMA 機能を使う
   - Rivermax の場合、GPUDirect RDMA 機能を用いるのだがこれが一般的に販売されている GeForce シリーズでは対応しておらず、ワークステーション向けの比較的最新版である NVIDIA RTX A4000 等の接続が必要とされているが、そのようなグラフィックボードは持ち合わせていない。
   - MTL に関しても Intel 製 GPU に限定される等、厳しいものがある。
   - だが、これが実現できないと高速な表示等ができず、また CPU にも負荷がかかり、超低遅延性なども失われる。
4. 音声（ST 2110-30）に関しても行う（PTP 同期 ST 2110-10 が必要）
   - ST 2110-30 の中核となる AES67 プロトコルへの理解が必要
5. マルチストリーム（映像）送信・受信の検証
6. NMOS によるメタデータやり取り・SDP 自動やり取り

- NMOS と呼ばれるプロトコルによって NDI と同じように ST 2110 機器の発見・登録が自動化できる。Sony 社によるオープンソース実装もある [11]。

将来的には、KAIROS のような ST 2110 等を汎用 PC で収容するソフトウェアベーススイッチャーを作成したいと考えている。

また、学園祭や外部のイベント等でこれらの検証から得られたソフトウェアでの実証実験等が行えたらなとも考える。

5 謝辞

本研究に必須である ConnectX 等の NIC を寛大にも私に貸し出してくださった、産学間連携推進室 服部氏に感謝する。また、学園祭の取り組みにあたっては先の服部氏に加え、学園祭実行委員会 情報メディアシステム局の皆様・学術情報メディアセンター・同 佐藤先生・ソフトイーサ株式会社様・輝日株式会社・情報科学類 産学間連携推進室・情報科学類 Ultra-Coins・学生生活課 等から機材貸出から各種交渉まで様々な面でお世話になった。この場をお借りして感謝申し上げます。

6 参考文献

[1] Blackmagic Design, 「ATEM Constellation 8K」. [Online]. 入手先: https://www.blackmagicdesign.com/jp/products/atemconstellation8k

[2] T. Grajek, J. Stankowski, D. Karwowski, K. Klimaszewski, O. Stankiewicz, と K. Wegner, 「Analysis of video quality losses in the homogenous HEVC video transcoding」. [Online]. 入手先: https://arxiv.org/abs/1702.07548

[3] Net Insight,「What is SMPTE 2110 and NMOS All About?」. [Online]. 入手先: https://netinsight.net/blog/what-is-smpte-2110-and-nmos-all-about/

[4] Canadian Broadcasting Corporation,「cbcrc/FFmpeg」. [Online]. 入手先: https://github.com/cbcrc/FFmpeg

[5] Open Visual Cloud, 「Media Transport Library」. [Online]. 入手先: https://github.com/OpenVisualCloud/Media-Transport-Library

[6] Open Visual Cloud, 「tx_st20_pipeline_sample.c」. [Online]. 入手先: https://github.com/OpenVisualCloud/Media-Transport-Library/blob/8a5893b1d2b2a2d860ddea4371528f6a52633619/app/sample/tx_st20_pipeline_sample.c

[7] Open Visual Cloud, 「rx_st20_pipeline_sample.c」. [Online]. 入手先: https://github.com/OpenVisualCloud/Media-Transport-Library/blob/8a5893b1d2b2a2d860ddea4371528f6a52633619/app/sample/rx_st20_pipeline_sample.c

[8] NVIDIA, 「NVIDIA Rivermax SDK」. [Online]. 入手先: https://developer.nvidia.com/rivermax-sdk

[9] Panasonic Connect,「KAIROS」. [Online]. 入手先: https://connect.panasonic.com/jp-ja/products-services/proav_it-ip-platform

[10] Panasonic Connect, 「KAIROS Core 2000 Mainframe AT-KC20M1G」. [Online]. 入手先: https://connect.panasonic.com/jp-ja/products-services/proav_it-ip-platform/lineup/at-kc20m1g

[11] Sony, 「nmos-cpp」. [Online]. 入手先: https://github.com/sony/nmos-cpp

空間 ID の集合演算の基礎理論の構築

筑波大学情報科学類 2 年生 齋藤智郎

Abstract– ドローンの飛行経路管理や地下埋設物の管理の効率化に向けて，独立行政法人情報処理推進機構（IPA）のデジタルアーキテクチャ・デザインセンターは，ボクセル（3 次ブロック）を用いて空間を管理する地理空間情報規格「空間 ID」の策定を進めている．この規格は，空間の重複判定を論理演算に帰着できるという利点を持つが，国家規模の大規模な空間データを高密度に扱う際，標準的な立方体ボクセルの集合ではデータ量および計算量が指数関数的に増大するという課題がある．本研究では，各次元で独立した解像度を保持する「拡張空間 ID」と，空間の二分木構造をビット列として符号化した独自の「V-Bit 表現」を提案する．V-Bit を順序付き Key-Value Store のインデックス検索と組み合わせることで，従来は要素数 N に対して線形 $O(N)$ の計算時間を要していた空間的な包含関係の検索を，対数時間 $O(\log N)$ またはズームレベルに比例する $O(Z)$ の計算量で実行可能とした．

1 研究の社会的背景

ネットで注文した荷物が翌日に届く便利な日常は，深刻なドライバー不足によって崩れ去ろうとしている．また，災害大国である日本においては，孤立した地域へ即座に救援物資や医薬品を届ける手段が不足している．これらの問題を根本から解決する切り札が，ドローンや空飛ぶクルマといった低空モビリティ技術である．低空空域を輸送路として活用すれば，地上の状況に左右されることなく，迅速な輸送が可能となる．

空飛ぶクルマやドローンといったハードウェアは高度に発達し，一部ではすでに実用化されているが，無数の機体が安全に飛び交う社会はいまだ実現していない．その最大の原因は，現在の日本の行政の空域管理システム(Dips2.0)が空を「2 次元の平面」として扱っていることにある．

図 1: Dips2.0 で飛行予約を行う様子

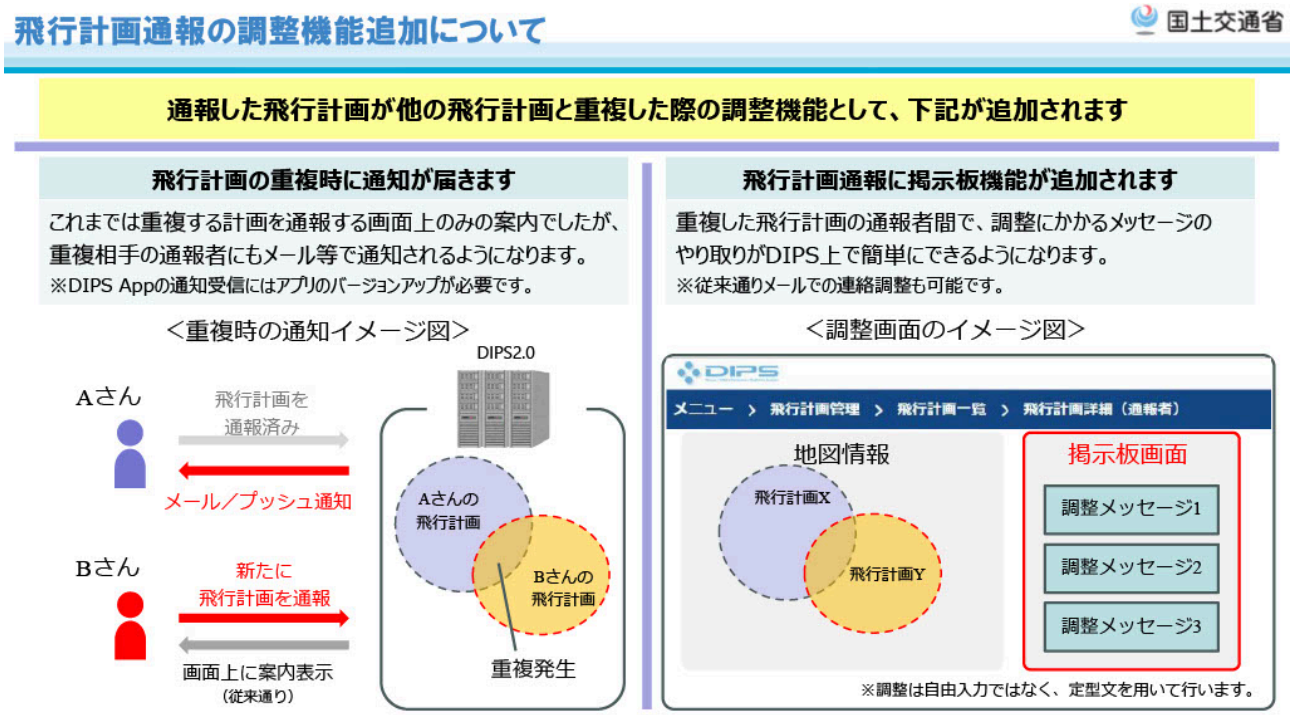


図 2: 飛行範囲が重なった場合のマニュアル

2 研究の目的

上述の課題を解決し，低空モビリティが安全に飛行する社会を実現するためには，最適な航路の策定および動的な空域管理が不可欠である．これを達成するには，天候や障害物，飛行制限区域といった 3 次元空間に紐づく膨大な情報をフィルタリングし，統合的に処理する技術が求められる．本研究の目的は，IPA（独立行政法人情報処理推進機構）が策定を進める「空間 ID」を用いて，この処理基盤を構築することである．空間 ID を用いることで，空間を構成する各ボクセル（3 次元のブロック）に対して，ズームレベルを示す z，高さを示す f，平面座標を示す x, y からなる `{z}/{f}/{x}/{y}` の形式で一意の ID が付与され，効率的な情報統合が可能となる．

2.1 空間 ID とは

IPA のデジタルアーキテクチャ・デザインセンターが策定を進める「空間 ID」は，Web メルカトル図法で表された地球をボクセルで再帰的に分割していく地理空間情報規格である．具体的には，ズームレベルが 1 上がるごとに各次元が 2 分割され，1 つのボクセルが 8 つのより小さなボクセルに細分化される構造を持つ．

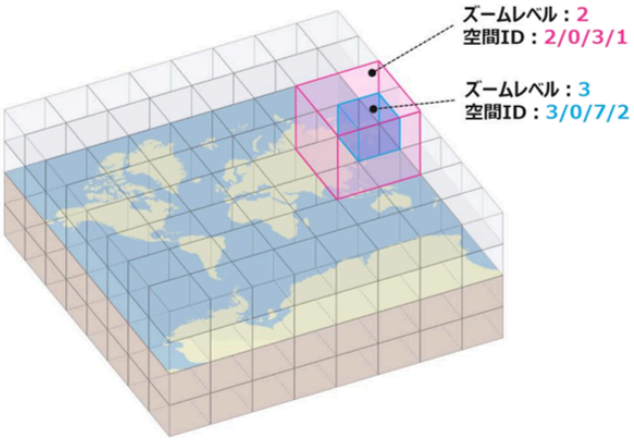


図 3: 空間 ID のイメージ

2.1.1 水平方向の分割(x, y インデックス)

Web 地図で標準的に用いられる Web メルカトル図法（XYZ タイル）に準拠し，極域を除く地球全体（西経 180 度～東経 180 度，南緯約 85 度～北緯約 85 度）をズームレベル 0 と定義する．ズームレベルが 1 上がるごとに，東西・南北にそれぞれ 2 分割され，計 4 分割されていく．

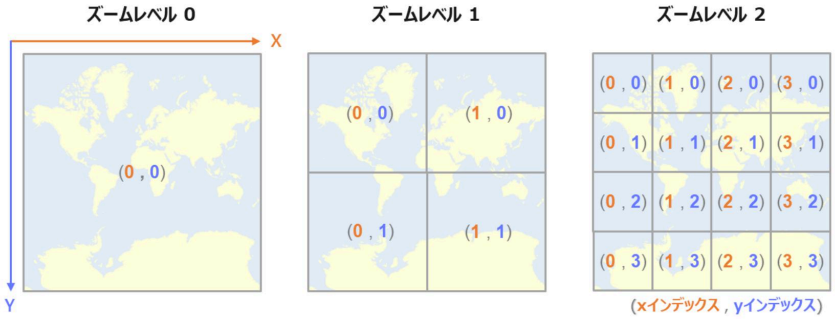


図 4: 水平方向の分割方法

2.1.2 鉛直方向の分割(f インデックス)

まず，地球の重力による等電位面である「ジオイド」を高さの基準（標高 0m）と定め，これを平坦な面とみなして 3 次元座標上にボクセルを配置する．この基準面を境界として，高さ方向にはプラスの領域（標高 0m～約 3.3 万 km）とマイナスの領域（標高-約 3.3 万 km～0m）が存在する．各領域の全体をズームレベル 0 とし，ズームレベルが 1 上がるごとに高さが 2 等分されていく．この定義により，既存の標高データを持つ地物情報をそのまま空間ボクセルへ紐付けることが可能となる．

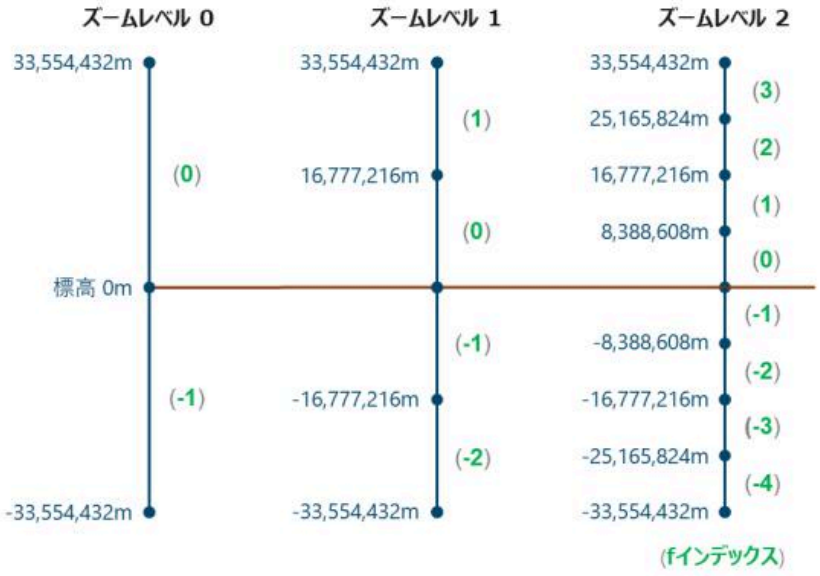


図 5: 垂直方向の分割方法

2.1.3 空間 ID の優れている点

こうした再帰的な分割構造により，空間 ID は極めて柔軟な解像度を表現できる．また，常に立体的な空間の範囲を示すことができるため，空域管理などの用途において非常に高い有用性を発揮する．既存の規格と比較し，具体的に以下の点で優れている．

- 任意の粒度を表現：天候（1km 四方）や障害物（10cm 四方）など，扱う情報の性質や用途に応じて任意の粒度を表現できる．
- 重複判定が容易： 領域の重複判定を複雑な幾何学的計算ではなく，単純な演算に帰着させることができるため，シンプルである

2.2 空間 ID 集合を用いた演算

空間 ID の集合は，特定の属性（例：風速 5m/s 以下，天候=晴れ）や利用条件（例：飛行禁止区域，既存の予約航路）を満たす空域を定義するために用いられる．異なる属性を持つ二つの空間 ID 集合 A と B が存在する場合，それらに対して和集合，積集合，差集合といった論理演算を行うことが可能である．

これにより，「(属性 A かつ 属性 B) − (飛行禁止区域) − (既存の予約航路)」のように，条件に飛行可能空域を算出できる．この演算処理が高速かつ大規模に実行可能になれば，本研究の目的であるドローンの航路策定に大きく寄与する．

したがって本研究は，この空間 ID 集合の論理演算を，社会実装に耐えうる実用レベルの技術へと昇華させることを目標とする．

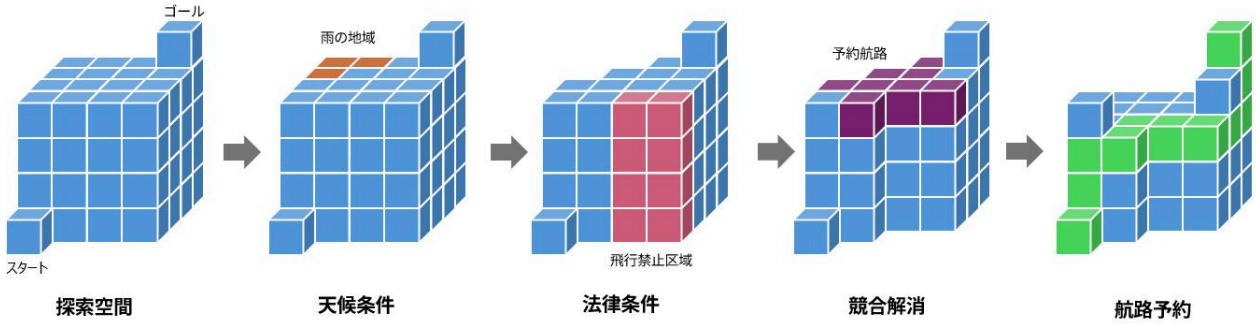


図 6: 集合演算を用いた航路確定のイメージ

3 課題

上述の空間 ID を用いた論理演算を実装するにあたり，標準の空間 ID をそのまま扱うと以下の 3 つの課題に直面する．

3.1 いびつな形状や微細な構造の表現に伴うデータ量の膨張

現実世界のオブジェクトは，標準の空間 ID が前提とする完全な立方体の形状とは大きく乖離している．例えば「飛行禁止区域」は，高度方向には極めて高く，水平方向には狭い柱のような形状となる場合が多い．また，建造物などの複雑な障害物を表現する際には，高いズームレベルの空間 ID を用いる必要がある．これらを標準の空間 ID のみで表現しようとすると，微小なボクセルを大量に積み上げて空間を埋め尽くす必要があり，結果としてメモリ消費量と処理コストが膨張する．

3.2 ズームレベルの異なる空間 ID 同士の演算によるデータ量の膨張

扱う空間情報によって，要求される適切なズームレベルは異なる．例えば，天候データは 1kmメッシュ（ズームレベル15 相当）の粗さで十分であるが，建造物などの障害物情報には 1m 程度（ズームレベル 25 相当）の精度が求められる．これら解像度の異なるデータ同士で，単純に大きなズームレベルに合わせて集合演算を行おうとすると，例えば，ズームレベルの差が「10」ある場合，8^{10}個もの空間 ID が発生し，現実的な時間が不可能となる．

3.3 大規模データの取り扱い

全国規模の空域情報や無数のドローン航路といった大規模データを扱う場合，全データをメモリ上に展開して処理を行うことは現実的ではない．したがって，ディスク上のデータ構造と親和性が高く，限られたメモリ空間でも高速に機能する処理方式が求められる．

4 提案手法

4.1 拡張空間 ID とその演算

定義 1（拡張空間 ID） 通常の空間 ID が縦・横・高さすべて同じズームレベルで分割されるのに対し，拡張空間 ID は「各次元が独立したズームレベルを持つ空間領域」である．

拡張空間 ID の定義により，飛行禁止区域や巨大な建物といった大量の空間 ID の塊を，少数の拡張空間 ID の集合として表現できる．この効果は数学的にも検証できる．ある 1 辺に沿って N 個連続する空間 ID がある場合，拡張空間 ID を用いれば最大でも 2log2N 個のブロックに圧縮される．これを 3 次元の各軸に適用することで，元々 FX Y 個のボクセルが必要だった空間領域であっても，8 (log2 F)(log2 X)(log2 Y)個以下の拡張空間 ID へと圧縮できる．これにより「いびつな形状や微細な構造の表現に伴うデータ量の膨張」が解決される．

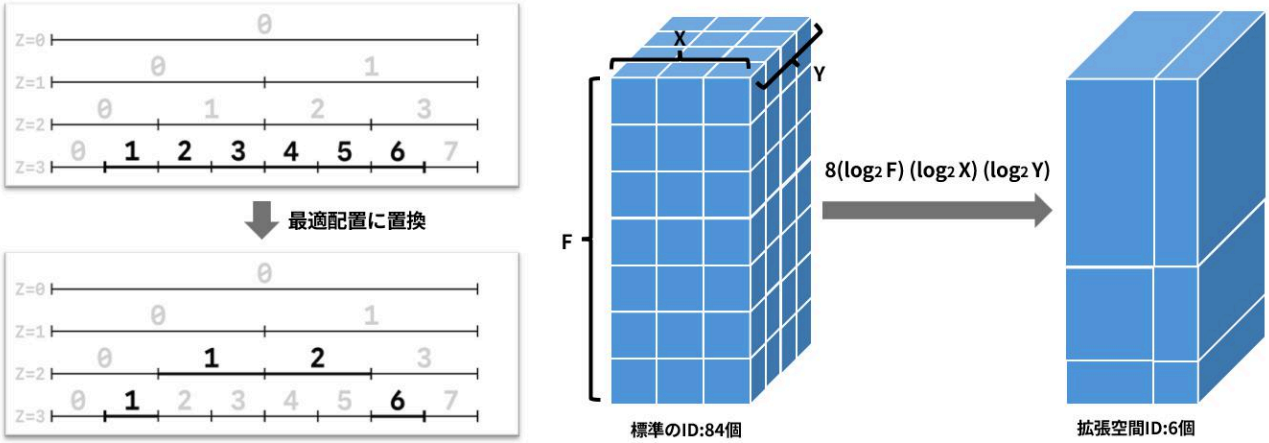


図 7: 拡張空間 ID を用いたデータの圧縮

4.1.1 セグメント

定義 2 (セグメント(Segment)) 空間 ID を構成する各次元において，特定のズームレベル値 z およびいずれかの次元のインデックス値 i の組 $S = (z, i)$ として定義されるデータ構造．任意の次元のインデックス値のことを便宜上，iと表す．

また，2 つのセグメント$A = (z_A, i_A)$と$B = (z_B, i_B)$の間の関連性を以下のように定義する．

定義 3 (等価なセグメント(Equal Segment)) セグメント A と B について，ズームレベル値およびインデックス値が共に等しいとき，両者は等価であると定義し，$A = B$ と表記する．

数式的には以下の条件と同値である．

$$A = B \Leftrightarrow z_A = z_B \wedge i_A = i_B \tag{1}$$

定義 4 (上位セグメント(Ancestor Segment)) セグメント A と B について，$z_A < z_B$ であり，かつ A の表現する区間が B の区間を包含する場合，A は B の上位セグメントであると定義し，$A \supset B$ と表記する．

数式的には以下の条件と同値である．

$$A \supset B \Leftrightarrow z_A < z_B \wedge i_A = \left\lfloor \frac{i_B}{2^{z_B - z_A}} \right\rfloor \tag{2}$$

また，このときのズームレベル値の差 $N = z_B - z_A$ を用いて，A を B の「N階層上位のセグメント」と呼称する．

定義 5 (下位セグメント(Descendant Segment)) セグメント A と B について，$z_A > z_B$ であり，かつ A の表現する区間が B の区間に包含される場合，A は B の下位セグメントであると定義し，$A \subset B$ と表記する．

数式的には以下の条件と同値である．

$$A \subset B \Leftrightarrow z_A > z_B \wedge i_B = \left\lfloor \frac{i_A}{2^{z_A - z_B}} \right\rfloor \tag{3}$$

また，このときのズームレベル値の差 $N = z_A - z_B$ を用いて，A を B の「N階層下位のセグメント」と呼称する．

定義 6 (セグメント間の階層差) 関連のあるセグメント A と B について，セグメント A を主体，セグメント B を客体として見たときのズームレベル値の差 $z_A - z_B$ を，セグメント A から見た B との階層差と定義する．

定義 7 (兄弟セグメント(Sibling Segment)) 非等価なセグメント A と B について，それぞれの 1 階層上位のセグメントが同一のセグメント C である場合，これらを兄弟セグメントであると定義する．

数式的には，以下の条件を満たす C が一意に存在することと同値である．

$$(A \neq B) \wedge (C \supset A) \wedge (C \supset B) \wedge (z_C = z_A - 1 = z_B - 1) \tag{4}$$

4.1.2 拡張空間 ID の 1:1 演算定義

まず，拡張空間 ID の 1:1 の演算定義について議論する．その後その定義を多:多に拡張し，集合演算を実現する．

定義 8 (積(Intersection)) 2 つの重なりのある拡張空間 IDE_1 および E_2 に対し，重なりがある部分の拡張空間 ID を抽出する操作を定義する．

E_1 および E_2の各次元のセグメントのうち，下位のものを抽出することで解を得る．

定義 9 (差(Difference)) 2 つの重なりのある拡張空間 ID E_1 および E_2 に対し，E_1 を主体，E_2 を客体として比較したときに，E_1 が表現する範囲のうち E_2 によって覆われない部分を抽出する操作を定義する．

まず，E_1 の各次元のセグメントを基準として階層差を求める．各次元における階層差を δF，δX，δY とする．

もし $\delta F < 0$ の場合，F_2 の兄弟セグメントを求める．F_2の 1 つ上位のセグメントが F_1 と等価でない場合は，そこへ移動し，また兄弟セグメントを求める．1 上位のセグメントにF_1 と等価なセグメントが現れるまでこの操作を再帰的に繰り返す．この過程で現れた$-\delta F$個の兄弟セグメントを，$F_1', F_2', F_3'...$とし，$-\delta F$個の拡張空間 ID の組 $(F_1', X_2, Y_2), (F_2', X_2, Y_2), (F_3', X_2, Y_2)...$がこの操作で得た拡張空間 ID である．$\delta X < 0$ の場合および $\delta Y < 0$ の場合についても，それぞれ X 次元，Y 次元に対して同様の操作を行い，記録したセグメントを$X_1', X_2', X_3'...$及び$Y_1', Y_2', Y_3'...$とする．ただし，記録する拡張空間 ID は，すでに操作を終えた次元に関してはE_1のものを参照し，$-\delta X$個の拡張空間 ID の組$(F_1, X_1', Y_2), (F_1, X_2', Y_2), (F_1, X_3', Y_2)...$及び，$-\delta Y$個の拡張空間 ID の組$(F_1, X_1, Y_1'), (F_1, X_1, Y_2'), (F_1, X_1, Y_3')...$を得る．最終的に，階層差が負となる全次元での操作において得た拡張空間 ID の組を，解として得る．

その個数は，各次元の階層差の中から負であるものを選び，足し合わせたものに-1 をかけた数に一致する．例えば，$\delta F < 0, \delta X < 0, \delta Y > 0$であれば，求める組の個数は$-(\delta F + \delta X)$個である．どの次元から操作を行うかによって差分の表現は異なるが，表される範囲と，拡張空間 ID の組の個数は常に一致する．

定義 10（和 (Union)） 2 つの拡張空間 ID E_1 および E_2 が表現する空間的範囲の間に，重なりまたは包含関係が存在する場合に，それらが表す空間的範囲を保持したまま，重複や包含関係を含まない拡張空間 ID の集合へ変換する操作を定義する．

まず，E_1 の各次元のセグメントを基準として階層差を求める．各次元における階層差を δF，δX，δY とする．重複を解消するには，E_1を主体，E_2を客体と考える方法と，E_1を客体，E_2を主体と考える方法の二つがある．主体客体が定まったら，客体はそのまま返し，客体はそのまま返し，主体は差分を求める操作を用いて，客体との差分を求めて返せばよい．差分を求める操作において，どちらを主体とすれば拡張空間 ID の個数がより少なくなるかを比較するには，次式で定義される値の符号を考えればよい．

$$k = \delta F + \delta X + \delta Y \tag{5}$$

$k > 0$ならば，E_1を，$k < 0$ならばE_2を主体とし，差分を求める操作を行う．$k = 0$ならば，どちらを主体としても良い．求めた差分と客体が，本操作によって得られる拡張空間 ID の集合である．本操作の結果として得られる符号化空間 ID の集合は，元の E_1 および E_2 が表現する空間的範囲と完全に一致し，かつ集合内の任意の 2 要素が互いに重なりのない状態となる．

例えば，$\delta F < 0, \delta X < 0, \delta Y > 0$であれば，$E_1$を主体としたとき，求める組の個数は$-(\delta F + \delta X) + 1$個であり，$E_2$を主体としたときは，$\delta Y + 1$個であるから，その大小は k の符号で判別出来る．

これらの定義により，「ズームレベルの異なる空間 ID 同士の演算によるデータ量の膨張」の問題が解消した．

4.2 拡張空間 ID の集合演算

これらの 1:1 における演算定理を，要素が複数ある多対多の演算に拡張することは定義的には可能である．しかし，この演算をそのまま実運用に適用することは現実的ではない．例えば，特定の空域における「安全な領域（集合 A）」と「飛行禁止区域（集合 B）」の差集合を取ろうとした場合，集合 A の各要素に対して，集合 B の「すべての」拡張空間 ID と重なりがあるかを総当たりで検証する必要が生じる．この方式では，計算量が集合の要素数 N に対して線形 O(N) で増加してしまう．国家規模のシステムにおいて，この線形計算のオーバーヘッドは致命的である．つまり，大規模な空間 ID の集合演算を実用化するための真の課題は，「重なりを持つ拡張空間 ID をいかに早く相手の集合から発見するか」という課題に収束する．

4.2.1 V-Bit を用いたセグメントの表現

集合演算について発生する課題③を，拡張空間 ID の各次元のセグメントに対し独自の符号化処理を施し，検索アルゴリズムを高速化することで解決する．我々はこの符号化方式を「V-Bit」と呼称している．空間 ID は，ズームレベルが上がるごとに空間を 2 分割していく二分木構造を持つ．V-Bit ではこの特性に着目し，各階層の分岐情報を 2 ビットの「階層ビット対」として表現する．1 つ目のビットがその階層のデータが有効かを示す有効 Bit，2 つ目のビットが空間のどちらへ分岐したかを示す分割 Bit である．セグメントを符号化する際は，ルート階層から目的のズームレベルまでの階層ビット対を順番に並べて固定長のバイト配列に格納し，末尾の余った階層はすべて「0（無効）」でパディングする．

図 8: V-Bit の例

4.2.2 V-Bit と順序付きインデックスページを用いた検索

V-Bit 表現の最大の強みは，B+Tree や LSM Tree などの順序付きインデックスと親和性が高い点にある．V-Bit でエンコードされた拡張空間 ID を順序付きインデックスに格納することで，関連あるセグメントをシンプルな方法で抽出することができる．

- 等価なセグメント:完全一致検索を行うだけであり，O(log N) の計算量で完了する．
- 下位のセグメント:V-Bit の性質により下位セグメントは必ず「自身の Bit 列をプレフィックス」として持つようにメモリ上に連続して配置される．そのため，順序付きインデックスの範囲検索を用いることで，下位セグメントを O(log N+K)（K は取得要素数）の計算量で一括取得できる．
- 上位のセグメント:自身の V-Bit のうち，最下層の 2Bit を 00 に置換するだけで，親セグメントを生成できる．これをルート階層まで繰り返して完全一致検索するだけでよいため，ズームレベル Z に比例した O(Z)で上位セグメントを検索できる．実用上，ドローンの障害物判定に必要な 10cm 解像度でも Z=30 であり，Z の最大値はたかだか 30 程度に収まる．

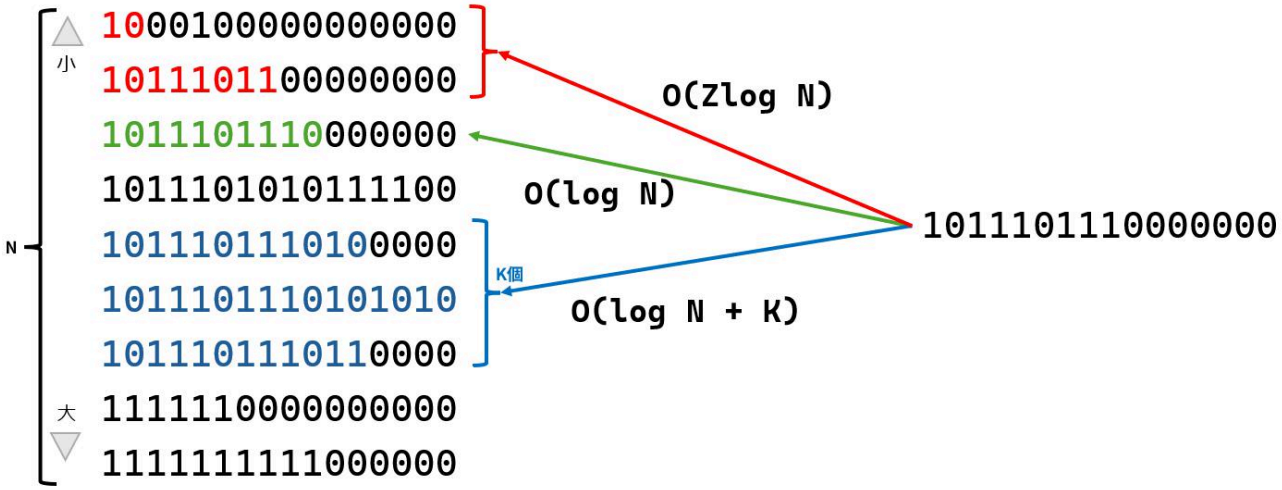


図 9: V-Bit の検索オーダー

4.2.3 拡張空間 ID の集合演算

上記のアルゴリズムを各次元に独立して適用することで，対象となる拡張空間 ID と空間的な重なりを持つ候補を高速に絞り込むことができる．すべての次元において共通して「関連あり（等価・上位・下位）」と判定された拡張空間 ID 群のみを取得し，それらに対して前述の 1:1 の論理演算（積・差・和）を適用する．これにより，総当たり検証（全探索）を完全に排除することができ，大規模な空間データ集合に対する演算処理を実用的な計算時間内で完了させることが可能となった．

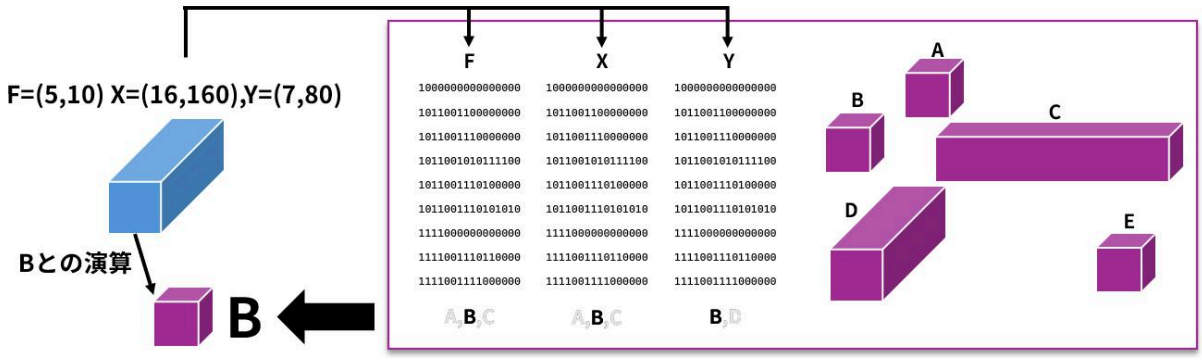


図 10: V-Bit を用いた検索の概観

5 成果物

5.1 空間検索アルゴリズムの体系化

本研究の主要な学術的成果は，これまで計算量の爆発により困難とされていた「大規模な 3 次元空間データにおける集合演算」を，数学のアプローチを用いて体系化したことである．具体的には，以下の 4 つの理論的基盤を確立した．

1. **拡張空間 ID と演算定理の定式化：** 次元ごとに独立した解像度を持つ「拡張空間 ID」および「セグメント」というデータ構造を新たに定義した．さらに，それらを用いた積・差・和の論理演算手順を定式化し，特に差集合演算時において発生する空間ブロック数の上限が各次元の階層差の合計 $|Z_f + Z_x + Z_y|$ に収束することを数理的に証明した．これにより，空間の断片化によるデータ量膨張の課題を原理的に解決した．
2. **V-Bit 符号化手法の確立：**空間 ID が本質的に持つ空間の二分木構造に着目し，各階層の分岐情報を固定長のビット列で表現する「V-Bit 符号化」を考案した．これにより，多次元の空間データを 1 次元のシンプルなバイト配列として表現する手法を確立した．
3. **非線形計算量による検索アルゴリズムの構築：**V-Bit 表現と B+Tree などの汎用的な順序付きインデックスを統合することで，空間的な包含関係（等価・上位・下位）の判定を，プレフィックス検索や範囲検索といった極めて軽量な操作に帰着させた．従来，要素数 N に対して線形探索 $O(N)$ が不可避であった総当たり判定を完全に排除し，対数オーダー $O(\log N)$ やズームレベルに比例する $O(Z)$ の計算量で目的の空間領域を抽出するアルゴリズムを構築した．
4. **任意の多次元空間への高い拡張性：** 従来の幾何学的な衝突判定アルゴリズムは 3 次元に特化していることが多く，動的モビリティに不可欠な「時間（4 次元）」を追加すると計算モデルの再構築が必要となる．しかし，本研究のセグメントと V-Bit に基づく演算モデルは，各次元を完全に独立して処理する直交性を持つ．そのため，X, Y, 高さ(F)という既存の 3 次元に「時間(T)」やその他の属性を加えて多次元空間 $(F, X, Y, T, ...)$ へと次元を増やした場合でも，アルゴリズムの基本構造を一切変更することなく適用できる．差集合演算時のブロック発生数の上限も各次元の階層差の総和 $\sum|Z_i|$ として一般化されるため，将来的な「4 次元時空間情報」の利活用に向けた極めて拡張性の高い処理基盤となる．

この体系化により，特定のマイナーなデータ構造に依存することなく，汎用的な Key-Value Store や分散データベースを用いて，大規模データを処理可能な仕組みの基礎を開発した

5.2 拡張空間 ID の集合演算をメモリ上で実装したライブラリの配布

本研究で提案した拡張空間 ID のデータ構造，V-Bit を用いたセグメントの符号化手法，および高速な集合演算アルゴリズムを Rust 言語を用いて実装し，コアライブラリ「Kasane-Logic」としてオープンソース公開した．本ライブラリは，Rust の公式パッケージレジストリである crates.io (`https://crates.io/crates/kasane-logic`) にて配布している．また，開発者が容易に本アルゴリズムを組み込んで応用アプリケーションを構築できるよう，Docs.rs にて API ドキュメントを充実させ，各演算定義の振る舞いや空間 ID に関する型定義を網羅的に記載している．

5.3 空間 ID プレビューツール「Madori」の開発

本研究で開発した演算アルゴリズムの正確性を評価し，生成された 3 次元構造を地理空間上で直感的に把握するため，Web ベースの空間 ID プレビューツール「Madori」を構築した．

空間 ID は抽象的な数値インデックスの集合であるため，特に幾何学的な投影歪みの補正や，複雑な集合演算（差集合等）の結果を検証する際，視覚的なフィードバックが不可欠である．本ツールは，TypeScript をベー

スに，GPU を活用した大規模地理空間データの描画に特化したフレームワーク「deck.gl」を採用している．これにより，標準的な Web ブラウザ上で数万個規模のボクセルをリアルタイムにレンダリングし，3 次元的な空間構造を俯瞰することを可能にした．

ツールの主要な機能は以下の通りである．

1. **レンダリング：**`z/f/x/y` 形式の空間 ID の直接入力，または JSON 形式によるインポートに対応し，空間 ID を正確に描画する．
2. **属性情報の可視化：**各空間 ID に付与された属性値（風速，電波強度，飛行禁止フラグ等）に基づき，ボクセルを任意の色で塗り分ける機能を実装した．これにより，リスク領域の空間的分布を視覚的に解析できる．

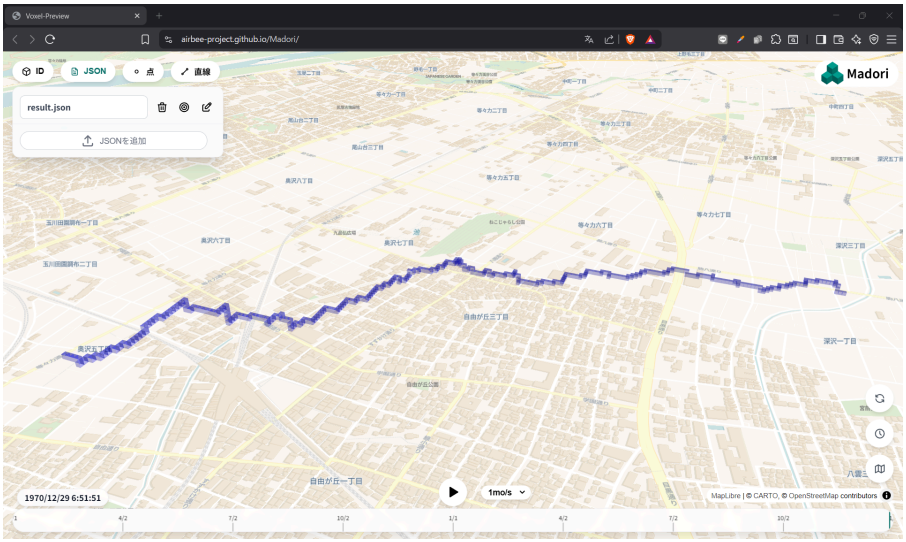


図 11: 空間 ID をプレビューする様子

6 研究において工夫した点

6.1 数学的証明を重視した大規模化に耐えうるアプローチ

ヒューリスティックな最適化に頼るのではなく，データ量の圧縮効果や最悪計算量について数学的な裏付けを持たせてアルゴリズムを設計した点に工夫がある．例えば，差集合演算時に発生する空間ブロック数の上限を $|Z_f + Z_x + Z_y|$ と厳密に定義し，空間の断片化が指数関数的に爆発しないことを保証している．これにより，国家規模へのスケールアウトを見据えた際にも，システムが破綻しない理論的基盤を構築した．

6.2 テスト駆動開発

空間演算はエッジケース（境界での接平面，完全包含，一部重複など）が非常に多く，実装の些細なバグが致命的なデータの不整合を引き起こす．そのため，実装前に想定される重なり方の全パターンに対する単体テストを網羅的に記述するテスト駆動開発を徹底した．また，すべてのテストは人間の手で書かれており，最近の OSS で散見される「AI がテストとコードを両方書いて，パスをさせる」という意味不明な状況ではなく，テストケースに意味を持たせ，その演算が正しいかどうかを確認している．

6.3 ベンチマーク駆動開発による継続的なパフォーマンス保証

空間演算アルゴリズムを国家規模の社会インフラとして実用化する上で，データ増大に対する計算量のスケールアウトは最重要課題である．そこで，理論的な計算量オーダー（$O(\log N)$など）が実際の実装においても確実に達成・維持されているかを検証するため，継続的インテグレーション環境に自動ベンチマークを組み込む「ベンチマーク駆動開発」を実践した．

特筆すべき点として，既存のベンチマークライブラリでは，データサイズ N に対する計算量のスケーリング特性や，複数コミット間にわたるパフォーマンスの長期的な変動を厳密に追跡・可視化することが困難であった．この開発上の障壁を乗り越えるため，アルゴリズムの計測に特化した独自のベンチマーク用クレート「Memori」を新たに開発した．本クレートを用いて日々のコード変更が演算速度に与える影響を計測し，その推移を GitHub Pages 上で継続的に可視化することで，意図しないパフォーマンスの劣化を未然に防ぐ，極めて堅牢なパフォーマンス保証体制を構築した．

7 対外的な取り組み

7.1 LINKS ハッカソン

2025 年 11 月に国土交通省が主催したハッカソン「LINKS: POWER of DATA x DATA 2025」に友人らと共にチームで出場した. 本研究の成果の前段となる技術を活用し, ドローンがリスクの低い空域の飛行を計画するソフトウェアを開発した.

この取り組みは, 計算機科学的な理論の優位性を直感的なアプリケーションとして具現化した技術力, およびドローンの社会実装を阻む課題を解決する公益性の高さが審査員から高く評価され,「オーディエンス賞」および「審査員特別賞」のダブル受賞を果たした.

7.2 筑波大学国際産学官連携本部からの特許出願

本研究が確立した「拡張空間 ID と V-Bit を用いた非線形計算量による空間検索アルゴリズム」は, 既存の 3 次元空間情報処理が抱えていた計算量爆発のボトルネックを根本から解決するものであり, 空域管理 (UTM) や自動運転分野において極めて高い産業的価値を有する. この成果について, 計算科学研究センターの建部修見教授に報告を行った結果, そのアルゴリズムの革新性とスケーラビリティが高く評価された.

現在, 同教授からの症例を受け, 筑波大学国際産学連携本部の全面的なバックアップ (出願資料の技術的精査, 弁理士事務所との連携, 費用支援等) の下, 本アルゴリズムに関する特許を出願中である.

謝辞

本研究の遂行および本稿の執筆にあたり, 多大なるご支援とご指導を賜りました多くの方々に, 心より感謝申し上げます.

まず, 本研究の根幹を成す「空間 ID」という革新的な概念と仕様を策定され, 次世代の空間情報基盤に向けた有用なガイドラインを公開してくださっている独立行政法人情報処理推進機構 (IPA) デジタルアーキテクチャ・デザインセンターの皆様に深く敬意と感謝を表します.

また, 本研究で開発した演算アルゴリズムに対し, 高性能計算の観点から極めて有益なご助言を賜り, その産業的価値を見出して特許出願に向けた力強いご推薦とご指導をいただきました, 筑波大学計算科学研究センターの建部修見教授に厚く御礼申し上げます. さらに, 特許出願に係る複雑な実務手続きや知的財産の保護において, 多岐にわたる手厚いサポートと全面的なバックアップを提供してくださった筑波大学国際産学連携本部の皆様に, 心より感謝申し上げます.

最後に, 日頃より活発で有意義な技術的議論を交わし, 研究を支えてくれた室員の皆様に, 深く感謝いたします.

参 考 文 献

1) 4 次元時空間情報利活用のための空間 ID ガイドライン (1.1 版), 独立行政法人情報処理推進機構 デジタルアーキテクチャ・デザインセンター, URL: https://www.ipa.go.jp/digital/architecture/Individual-link/nl10bi000000377d-att/4dspatio-temporal-id-guideline.pdf Accessed on 2025.12.30.

2) 飛行計画通報の調整機能追加について, 国土交通省, URL: https://www.mlit.go.jp/common/001732680.pdf Accessed on 2025.12.30.

情報機器資源リユースプロジェクト 活動報告

～ 「まだ使える」機器の有効な活用を目指して ～

2026 年 5 月 23 日 筑波大学情報科学類 産学間連携推進室 活動成果報告会
情報機器資源リユースプロジェクト 第 8 代統括作業総責任者 北野 尚樹
協力：産学間連携推進室 根本晃輔、関口亞聖、服部真吾、松田心杏、細島涼雅、
中野あおい、間瀬太陽、齋藤智郎

1．序論

1－1．本プロジェクトの背景

筑波大学 システム情報系エリアにおいては例年、毎年 2 回（6 月および 12 月）の頻度で、図 1 のような各種の大型物品[1]の一斉廃棄が行われている。その中でも高度な情報機器資源であるコンピュータについては、図 2 のように、毎回の物品廃棄で約 250 台～300 台が廃棄されている。

図 1：システム情報系エリア物品一斉廃棄の様子

[1] 什器類、家電類、実験機器類、情報機器類を指す

図 2：2024 年 6 月に廃棄されたコンピュータ類

これらの廃棄されたコンピュータについては、確かに Intel Pentium4 世代や Intel Core 2 世代、あるいは第 6 世代以前の Core i シリーズの CPU を搭載したコンピュータなど、既に研究の場での現用が難しいもの[2]が殆どである。しかし一部には、たとえば第 7-8 世代以降の Intel Core i 世代のコンピュータなど、パーツを交換・増強すれば現用に耐えうる性能でありながら、単純な経年・入替のため廃棄されたものや、修理・交換可能な部品の一部のみが故障したために廃棄されている比較的新しいコンピュータも存在している[3]。それらは部品を適切に交換・修理したり、性能向上を図れるパーツを追加したりすることで、ある程度の期間、再度利活用することが可能になる。

このように、故障したコンピュータを積極的に買い換え、故障したコンピュータを廃棄することのできる研究室がある一方で、筑波大学における一部の研究科・専攻においては、高価な実験機材を購入するためにコンピュータに十分な予算を設けることができず、やむ

---

[2] 既に製造から 12 年以上、Core 2 世代では約 20 年が経過しており、近年のソフトウェアの要求性能に満たない性能の機器が多いこと、および長期の使用による修復困難な故障などが発生しているものが多いことから、現用が難しいものと考えられる

[3] 中にはマザーボードの致命的な故障など重大なものも含まれているが、OS レベルの障害や一部の交換可能なパーツの故障などが発生した際に、その原因を詳細に特定することなく廃棄されているものも少なくないと推測される

を得ず Intel Core 2 Duo ～ 初代 Core i シリーズ 世代などの古いコンピュータを限られた台数で利用している研究室が存在している。また、研究室における急激な人数の増加に対し、予算が限られているなどの理由によりコンピュータの新規導入が難しいという状況や、留学生のために短期的に利用できるコンピュータが必要になるという状況にある事例もみられるほか、2020～2021 年頃には新型コロナウイルス感染症の拡大に伴う在宅での学修・研究を行うために緊急で貸出用の PC が必要となった研究室なども存在した。

筑波大学の物品廃棄において廃棄される物品のうち、コンピュータ以外の一般の備品については、教職員が閲覧可能なシステム上で次の利用者を募集するなどのリユースの仕組みが構築され、実際に運用されている。

しかし、性能や経年の状況が外観から判断できず、限られた情報からでは詳細な状況を判断しづらいコンピュータについては、その仕組みの枠の中で取り扱うことが難しい状況にある。更に、コンピュータを廃棄・譲渡する際にはデータの厳重な消去が必要になるが、その方法については教職員間でも徹底がなされておらず、そのような徹底が不十分な状況下で安易な廃棄・譲渡を行うことは、時として情報漏洩などの重大な事故を招きうるものとなる。

１－２．本プロジェクトの目的

このような諸課題を解決するための情報機器資源のリユースの試み、および付随的な目的として情報記録媒体の確実な廃棄の補助・物品一斉廃棄の支援を行う本プロジェクトは、システム情報系技術室の要請の下、2015 年 6 月に根本（産学間連携推進室所属）らにより発足した。そして、2026 年 4 月現在に至るまで、規模を随時拡大しながら運用が続けられている。

本プロジェクトは、筑波大学システム情報系技術室の指揮・管理の下で、物品の一斉廃棄の際に廃棄されるコンピュータを適切に修理・再構成し、システム情報系技術室によって上述のようにコンピュータを必要としている研究室や各種組織などに分配することで、情報機器資源のリユースを行うことを目的としている。またリユースの過程の中に、廃棄されるべき情報記録媒体の確実な破壊・廃棄、および再利用される情報記録媒体におけるデータの確実な削除を組み込むことで、物品廃棄に際し発生しうる情報漏洩などのインシデントを未然に防ぐことも付随的な目的としている。

２．プロジェクトの概要

２－１．活動の概略

「情報機器資源リユースプロジェクト」は、システム情報系技術室の要請・協力の下、情報科学類産学間連携推進室内に設置されているプロジェクトである。本プロジェクトにおいて、産学間連携推進室のメンバーはコンピュータの修理・再構成などの技術的部分を主に扱っているため、本稿ではその担当範囲における活動について述べる。

本プロジェクトは 2015 年 6 月（発足時） ～ 2025 年 12 月度の全 21 回の物品廃棄において、デスクトップパソコン・ディスプレイ・ノートパソコンを併せ、総計 900 台以上のリユースを行った。集計が完了している 2021 年 6 月度までの内訳を表 1 に示す。

表１：リユースの内訳。2020/06 は新型コロナウイルス感染拡大の影響により中止。

Data	Desktop	Laptop	Display	Sum
15/06	15	10	15	40
15/12	15	0	15	30
16/06	23	0	20	43
16/12	16	2	34	52
17/06	32	21	39	92
17/12	12	2	5	19
18/06	30	20	10	60
18/12	28	15	9	52
19/06	35	5	12	52
19/12	17	2	11	30
20/06	-	-	-	-
20/12	25	5	14	44
21/06	16	5	17	38

本プロジェクトは現在、2025 年 12 月の物品一斉廃棄で搬出された機材類においても同様の作業を進めており、2026 年 6～7 月中には再利用可能となったデスクトップパソコン・ディスプレイ・ノートパソコンをシステム情報系技術室に引き渡す予定である。

２－２．工程と運用

本プロジェクトにおけるコンピュータの修理・再構成の工程においては、部品が欠損・故障しているコンピュータから性能の高い再利用可能な部品（CPU・メモリ・グラフィックカード等）を取り出し、状態の比較的良好なコンピュータにそれらの部品を移植することで、修理・性能向上を行う形をとっている。また、液晶ディスプレイなどの表示機器についても、動作確認を行った上で、ある程度の解像度があるものや経年していないものについては形式の取りまとめを行い、機器のリストと合わせ技術室に引き渡す形をとっている[4]。

現在の本組織の運用形態は、産学間連携推進室メンバーの 9 名および情報科学類・情報メディア創成学類・大学院生の有志学生 2 名からなる本プロジェクトのように少人数な組織であっても、十分に運用可能なものである。なお、活動の様子を記録した写真を図 3 ・図 4 に示す。図 3 はディスプレイ等各種機器の状態確認、図 4 は専用に構築した機器を用いてハードディスクのデータ消去を行っている様子である。

図３：ディスプレイ等各機器の動作確認を実施している様子

[4] なお、2017 年 12 月以降の表中の「ディスプレイ」の数値は高解像度のものや、形式の取りまとめを行っているものに限っている。高解像度でないもの、あるいは若干の経年があるものの利用が可能であるディスプレイについてもリユースを行っているが、作業の簡易化のために取りまとめを行っていないため、この表中には記載していない。

図４：ハードディスク処理作業の様子

３．リユースの事例

本プロジェクトの活動実績を示す上での参考として、実際にこれまでのプロジェクトの進行の中で行ってきたコンピュータの修理・再構成の例の一部を示す。

３－１．ノートパソコンの液晶画面の入れ替え

搬出された同型のノートパソコン 2 台のうち、1 台は液晶ディスプレイ部分が破損していたものの筐体をはじめとする他部分には損傷がなく、もう 1 台はタッチパッド・DVD ドライブ部をはじめとする筐体部に大きな歪みがあったものの、液晶ディスプレイ部分には損傷がない状態であった。

そこで、後者のノートパソコンから液晶ディスプレイ部を取り外し、前者のノートパソコンに移植することで液晶ディスプレイの交換作業を行い、無事に利用できる状態に回復させた。なお、後者のパソコンに搭載されていた取り外し可能な他のパーツのうち、メモリとハードディスクについては取り外しを行い、他のノートパソコンへの移植を行った。

３－２．エアダスターや無水エタノールなどを用いた分解清掃

搬出されたデスクトップパソコン・ノートパソコンのいずれについても、基板上・ファン内部などの各部位に蓄積した埃などについては、エアダスターや無水エタノールを用いた清掃を行っている。また、ノートパソコンについては消毒用エタノールを用いた清掃を行い、次の利用者が快適に利用できるような環境を整えている。

特に、デスクトップ Mac 環境などの特殊な環境の動作を維持するためのパーツは絶対数が少なく、供給の少ない部品を清浄しながら利用する必要がある。中には煙草の脂と思われるものが付着しているものもあったが、この分解清掃を行うことで外見・動作とも問題のない状況に仕上げることが出来た。

３－３．パーツ世代・電力量などを加味した各種パーツ等の増設や高性能化

搬出されたデスクトップパソコンについては、例えば CPU をより上位のものにアップグレードできるものや、更に大容量のメモリを搭載できるものなどが存在する。それらのパソコンについては、アップグレード可能なパーツがあれば、電力量や想定用途などのさまざまな要素を加味した上で、再利用の際に支障が出ない範囲でのアップグレードを施している。

４．今後の展望と課題

本プロジェクトにおける今後の活動の展望、および現状の課題に関して以下の通り記す。

４－１．継続的な活動に向けて

本プロジェクトは 2026 年度の物品廃棄においてもコンピュータの修理・再構成を実施しており、今後も活動を継続していく予定である。特に 2022-2025 年度の活動においては、特殊工具を含めた資材の大幅な拡充等を行ったことにより、前年度に引き続き作業効率や作業速度を向上させること、および作業手順のマニュアル化を進めることで作業の属人化・負担の集中を防ぎ、今後も継続して活動ができるプロジェクト運営を行うこと、の 2 点を重点におく形で活動を進めることができた。本プロジェクトは創設より 8 代に亘り代替わりを続けながら活動を継続しているが、大学組織に特有の「メンバーの入れ替わり」があっても持続可能な、なおかつ人々に必要とされる組織を構築していく上では、本年度の成果には非常に大きな意義があるといえる。

４－２．他エリアへの拡大に向けて

筑波大学においては、システム情報エリアに限らず、さまざまな学域・エリアにおいて物品の一斉廃棄と、それに伴うコンピュータ類の搬出が行われている。それらのコンピュータ類にも、システム情報系物品廃棄において搬出されているコンピュータ類と同様、まだ十分に利用可能な製品や、修理および再構成を行えば数年間は利用可能な製品が含まれており、これらをシステム情報系の物品廃棄における本プロジェクトと同様の形でリユースすることができれば、当該エリアにおける情報機器資源の効率的な運用や、比較的高性能なコンピュータを活用した研究の推進に役立つことが考えられる。またもちろん、情報技術に明るくない学域・エリアにおいては、本プロジェクトの副次的な効果である HDD/SSD などの記録媒体の安全な処理を推進していくことで、より安全に情報記録媒体の廃棄を進めていくこともできるだろう。

エリアを跨いだ活動には現状下で困難が多いと思われるが、このような活動をより広範な形で推進していくことにより、大学内における経年した旧型機器の入れ替えの推進による研究活動の効率化や、昨今の大学・各種研究機関において非常に大きな問題となっている予算の逼迫の解決、廃棄機材からの情報漏洩への対策に対する一つのアプローチとなるのではないかと思われる。

４－３．知見の一般共有

本プロジェクトの副次的な効果として、コンピュータの組み立てやトラブルシューティング、およびそれらに関連する知見が豊富に得られることが挙げられる。例えば、以下のような知見を得ることができる。

- PC パーツの世代の見分け方や、同一世代内での性能の素早い見分け方[5]
- PC ケース内の整線や、カスタマイズの行いやすいパーツの配置方法
- CPU やメモリのトレイなど、余剰パーツを収納するための特殊な部材の入手先[6]

このような一般的な知見について、プロジェクト内部のメンバーだけでなく、Web サイトなどを活用して外部にも積極的に公開していくことは、本プロジェクトそのものの広

---

[5] 中には外見上非常に見分けづらいパーツが存在するが、型式などが記載されている部分を探せばすぐに世代を特定することのできるパーツ（メモリなど）も存在する。

[6] たとえば、CPU トレイは国内に取扱いがなく、需要がニッチなため ebay などによる海外通販を利用するしかない。しかし、その海外通販もあまり知られていない。

報・発信の観点からも望ましいことである。このような情報の共有は 2024 年度までにあまり力を入れて実施できなかったことでもあるが、2025 年度はこれらの情報の集約・公開に向けた準備を一定程度進行することができたため、2026 年の活動においてはこれらの知見を広く公開できるよう、更なる活動の推進が必要であると考えられる。

それだけでなく、例えば、コンピュータの組み立てを行ったことがないが、それに興味のある小〜中学生を対象に、コンピュータのパーツの役割を説明しながら実際に組み立てていく / 分解を実演する講座を開設するなど、ワークショップ形式で情報発信を実施していくことも選択肢の 1 つとなるだろう[7]。

４－４．廃棄機材の他プロジェクトにおける活用

2021 年頃より現在に至るまで、本プロジェクトにより、情報記録媒体について安全な処理を施した状態の廃棄機材を「インデペンデンス・サーバー・デイ」における「サーバ解体コーナー」、産学間連携推進室の主催する情報科学類生向け PC 分解講座・ネットワーク講座等に活用する取り組みがなされている。

特に毎年 4 月に筑波大学情報科学類の新入生を対象として実施している「PC 分解講座」は、開催場所である 3C213 ラウンジが満員となるほどの盛況である。

図４：毎年 4 月に情報科学類新入生を対象に開催している「PC 分解講座」(2026/04)

[7] 「夏休み自由研究お助け隊」や「インデペンデンス・サーバー・デイ」などとの協調実施も視野に入りうる。

このような取り組みは、物理的にコンピュータという機器の内部に触れる機会が減ってきている近年の生徒ら・学生らにとって、またとない学習機会となるであろう。

４－５．今後の活動に向けての課題

現在、この活動は特に PC ハードウェアに関する知識を有する学生によって成り立っている。しかし、昨今のノート PC の高性能化やタブレット型 PC の出現などから、デスクトップ PC を組み立てたことのある学生の割合は年々減少しつつあるように思われる。しかし、研究の場で用いられるコンピュータは、依然としてデスクトップ PC が殆どであろう。

そのような状況下においても滞りなくこの活動を継続させるために、デスクトップ PC の組み立てなどについて興味のある学生へのアプローチを中心に、持続可能なプロジェクト運営に向けての方策の構築に、今後十分に時間をかけて取り組まなければならないと考えられる。2025 年度も 2 名の新入生を本プロジェクトに迎え入れることができたため、新入生に対する技術指導を中心として、今後の活動の持続可能性を高められるような取り組みを進めていくことが求められるものと思われる。

５．謝辞

本プロジェクトを進行するにあたり、システム情報系技術室 雨谷 恵 様をはじめとする、筑波大学内のさまざまな方々、プロジェクトメンバーの皆様、プロジェクト OB/OG の皆様のご支援・ご協力を頂いております。この場をお借りしまして、深く御礼を申し上げます。

2025年度 筑波大学 情報学群 情報科学類

産学間連携推進室 活動成果報告書

発行日：2026年 5月 20日 初版発行
発行者：間瀬 太陽（情報科学類 産学間連携推進室 学生代表）
編集者：間瀬 太陽、根本 晃輔
制作者：情報科学類 産学間連携推進室 所属学生一同