観測システムとしての Tomo-e Gozen の開発

大澤 亮
自然科学研究機構国立天文台 JASMINE プロジェクト

Abstract & Table of Contents

東京大学木曽観測所では Tomo-e Gozen による大規模な動画サーベイを実施している.
観測やデータをハンドリングするシステムについて開発をふりかえる.

Tomo-e Gozen の概要

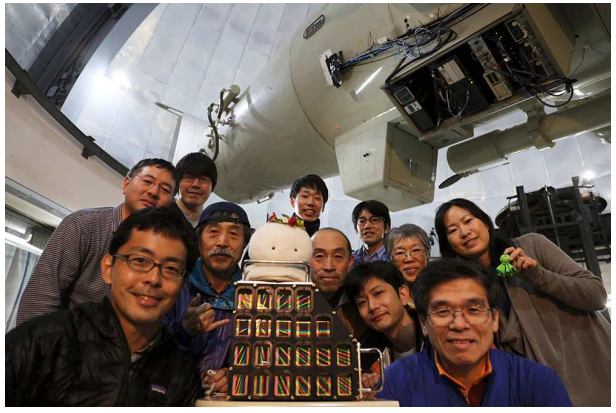
Tomo-e Gozen Development Team

Tomo-e Gozen は多くの人の協力で作られた観測装置.

特に以下の 3 人が主導して主要なコンポーネントを開発.

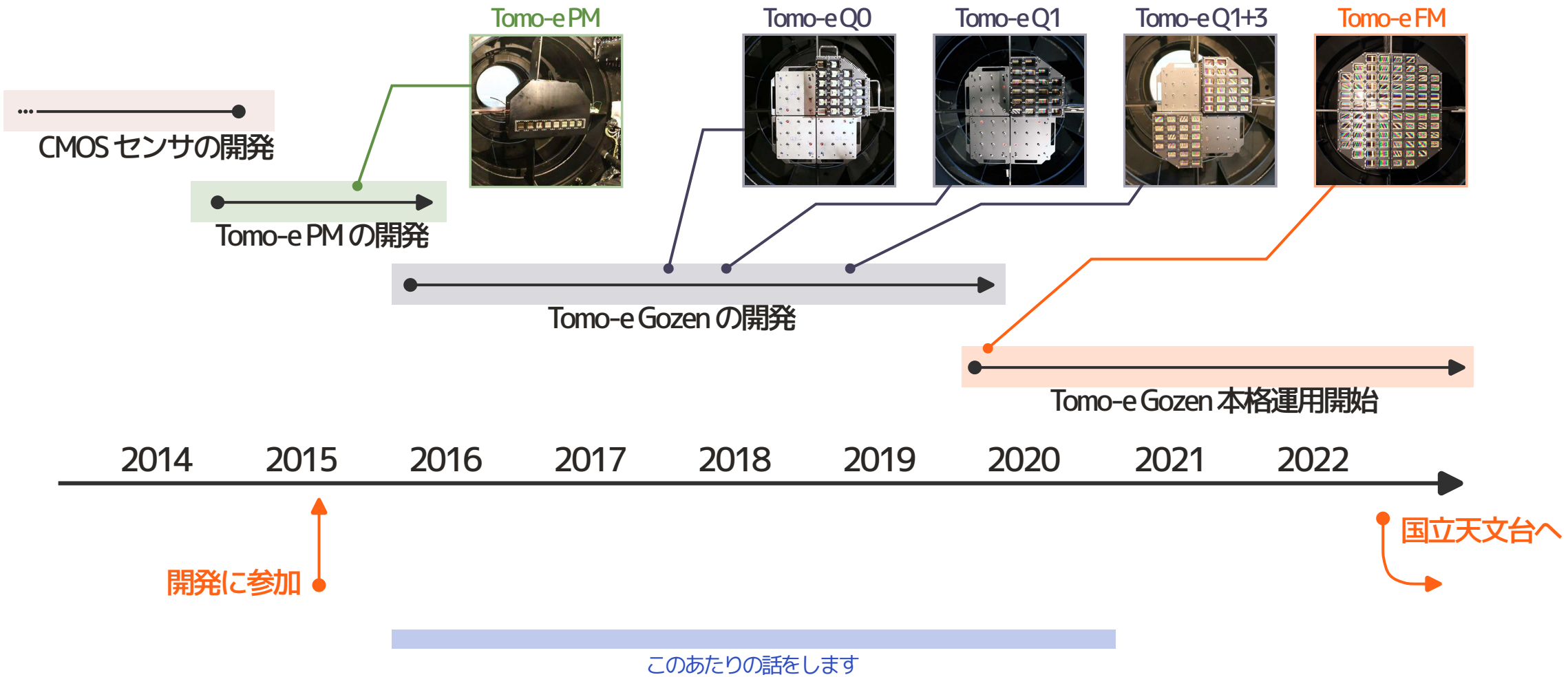
酒向 重行 (東京大学 准教授)	⇨	全体設計・カメラ駆動系
高橋 英則 (東京大学 助教)	⇨	カメラ筐体・望遠鏡 IF
大澤 亮 (東京大学 → 国立天文台)	⇨	データ取得・解析システム

この発表ではここにフォーカスします



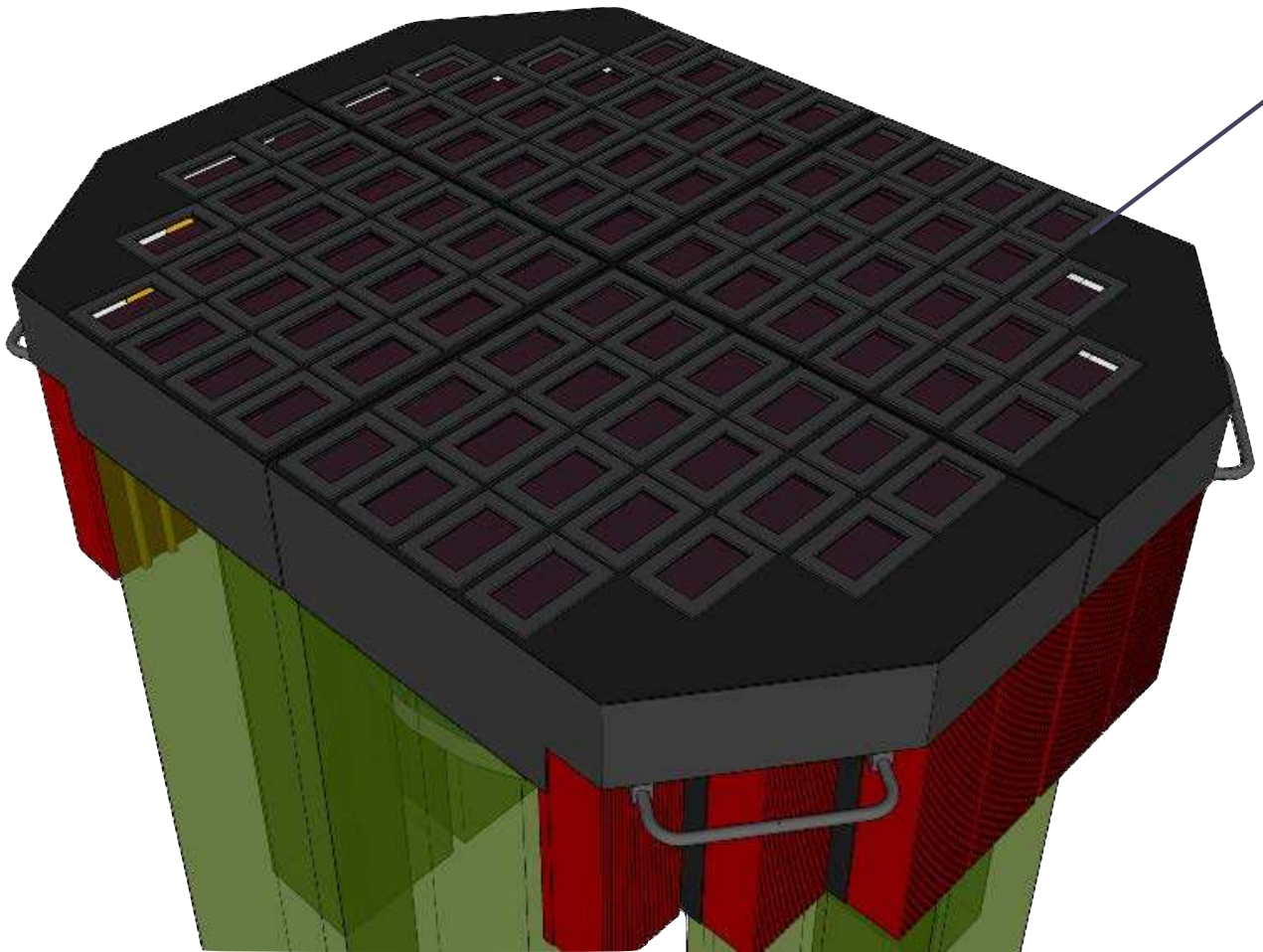
2019-04-23
Tomo-e Gozen 最終ユニット Q4 搭載前の集合写真

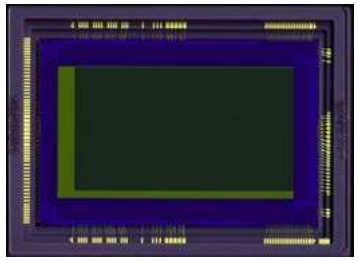
Timeline of Tomo-e Gozen Development



Sako et al. 2018, Proc. SPIE; Kojima et al. 2018, Proc. SPIE

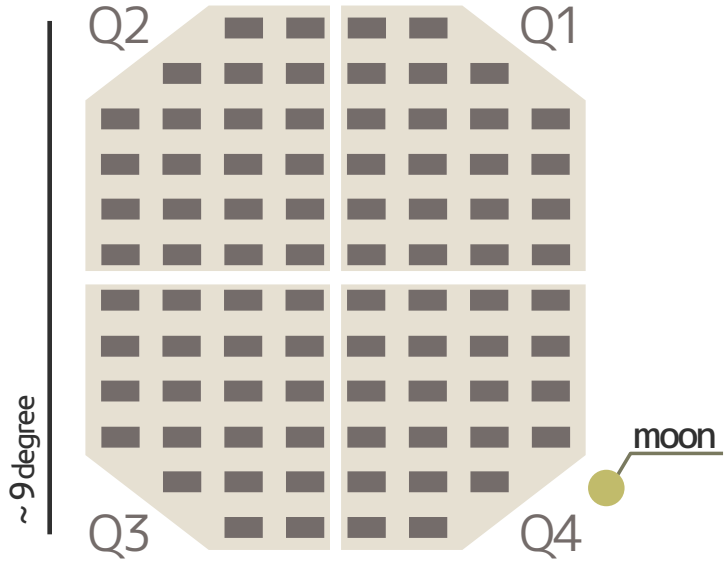
Overview of Tomo-e Gozen Camera





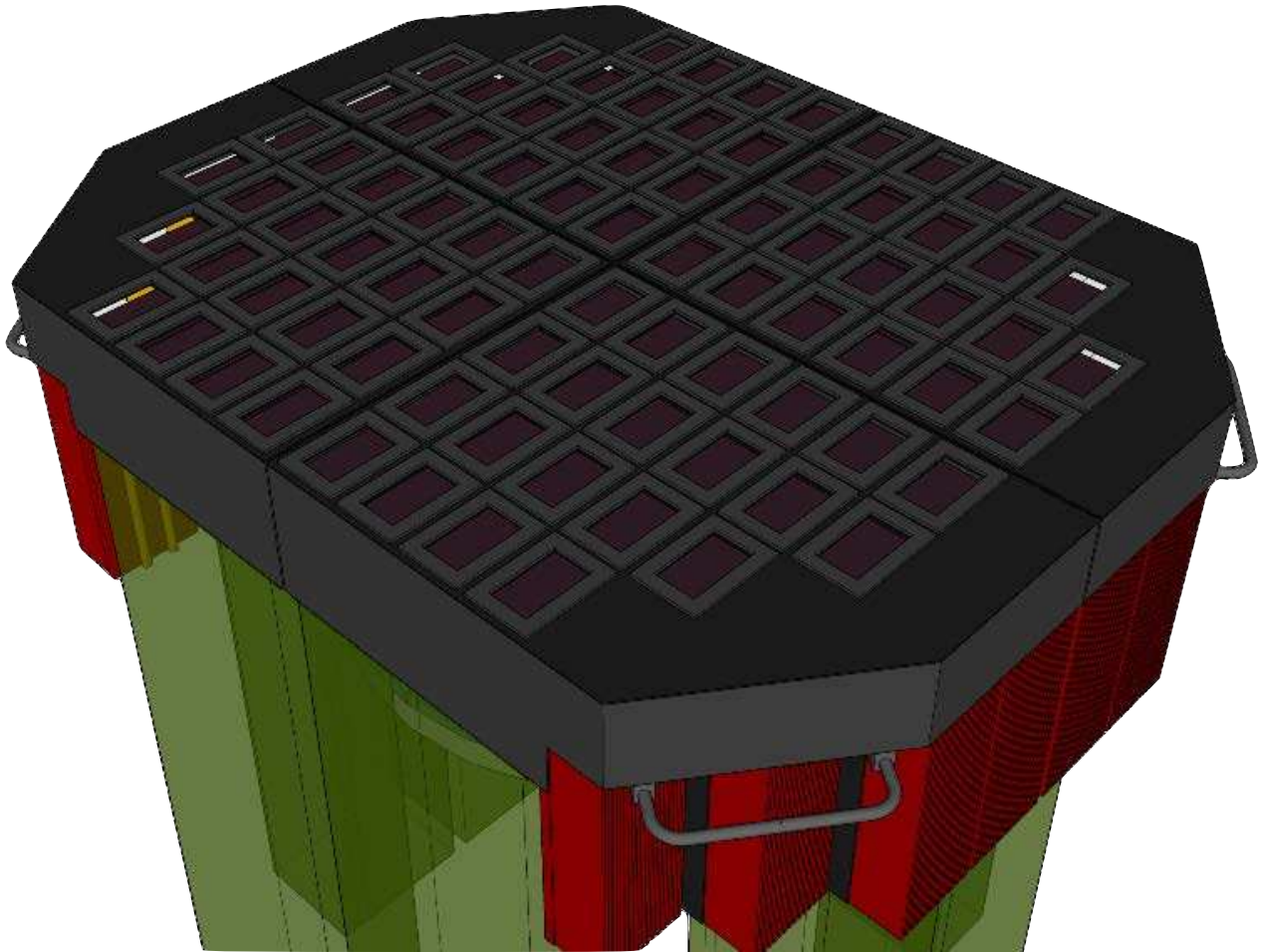
2K×1K CMOS image sensor by **Canon**

84 CMOS sensors covering ~20□°



Sako et al. 2018, SPIE, 107020J

Overview of Tomo-e Gozen Camera



Specifications

Observatory	Kiso Observatory
Telescope	1.0-m f/3.1 Schmidt telescope
Sensor format	2160×1200 pix chip^{-1}
Field of view	39′.7×22′.4 × 84 chips
Pixel scale	19 μm, 1″.189 pix^{-1}
Wavelength	350–700 nm (peak at 500 nm)
Filters	optical broadband (clear filter)
Frame rate	2 fps (max, continuous, full frame)
Read noise	~ 1.9 e$^-$ at 2 fps
Dark current	~ 0.1 e$^-$ sec^{-1} pix^{-1} at 277 K
Well depth	~ 6,400 e$^-$
5σ lim. mag.	~ 18.5 mag. in 0.5 sec exposure

Sako et al. 2018, SPIE, 107020J

Overview of Tomo-e Gozen Science

超新星の発見 (SN 2019cxx, SN 2020hvf, etc.)
wis-tns.weizmann.ac.il/object/2019cxx

地球接近天体の発見 (合計 49 天体)
Kojima+ MPEC 2019-F19; MPS 1049395; Foglia+ MPEC 2019-V87, etc...

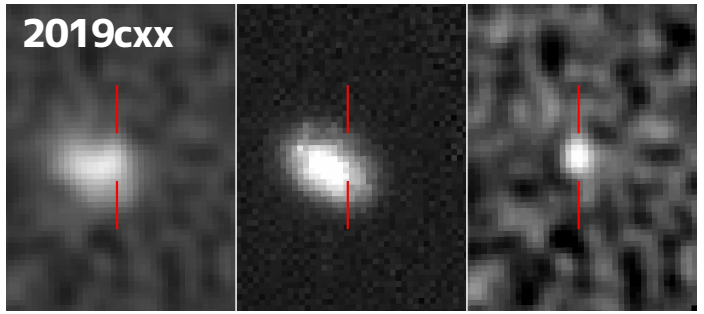
重力波可視光対応天体のフォローアップ観測 (合計 5 イベント)
S190408an (GCN 24064), S190412m (GCN 24113, 24350)
S190426c (GCN 24299), S190930t (GCN 25907)

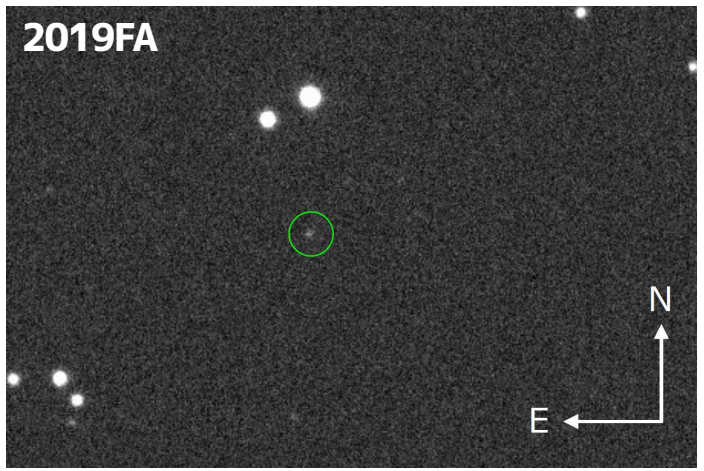
太陽系外縁天体 (50000) Quaoar の掩蔽観測
Arimatsu+, 2019, AJ, 158, 236

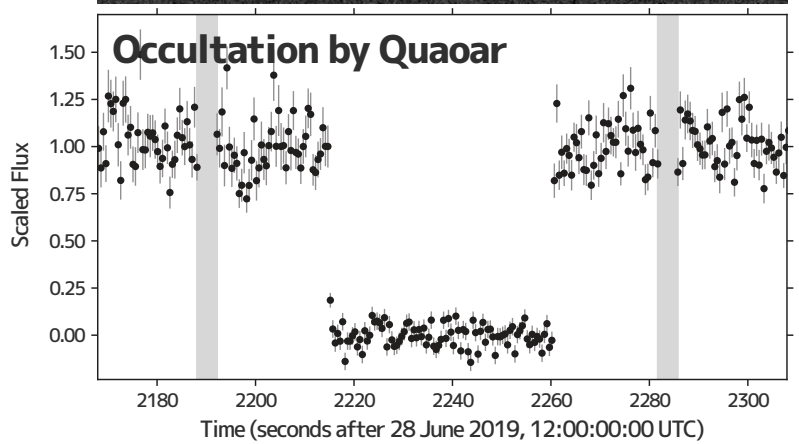
京都大学 MU レーダ & トモエゴゼンによる微光流星同時観測
Ohsawa+, 2020, P&SS, 194, 105011

動画観測による秒スケール変動現象のサーベイ
Richmond+, 2020, PASJ, 72, 3; Aizawa+, 2022, PASJ, psac056

ニュートリノアラート IceCube-170922A 追観測
Morokuma+, 2021, PASJ, 73, 25-43







Development in Kiso Observatory

Tomo-e Gozen の開発は木曽観測所の環境に強く依存

1.05 m 木曽シュミット望遠鏡 ⇨ **赤道儀·装置回転機構なし**

撮像観測のみ (Tomo-e Gozen はフィルタ交換機構もなし)

望遠鏡に温度補償機構 ⇨ **フォーカス再調整はほぼ必要ない**

共同利用観測をやめて**プロジェクト観測主体の運用**

これまでに**自動サーベイ観測·リモート観測**の実施経験あり

できるだけ一般化できそうな
要素に着目するつもりです

⇨ システムのデザインは木曽観測所での運用に合わせたもの
他の装置に転用できるかどうかはケース·バイ·ケース

Policy of Tomo-e Gozen

Tomo-e Gozen の開発ではかなり割り切ったポリシーを採用

観測所のリソースは有限, 運用コストを現実的な範囲に収めることを優先

動画データをすべて保存することは非現実的 ⇨ ある程度の時間をおいて消去する

多少のトラブル･オーバヘッドによるロスは許容する (曇ったものと考える)

全データの精査は不可能 ⇨ クオリティで適度に足切りをすることを許容

⇨ システムの設計もこのポリシーに基づいて省力化している

Tomo-e Gozen のうらがわ

Requirements

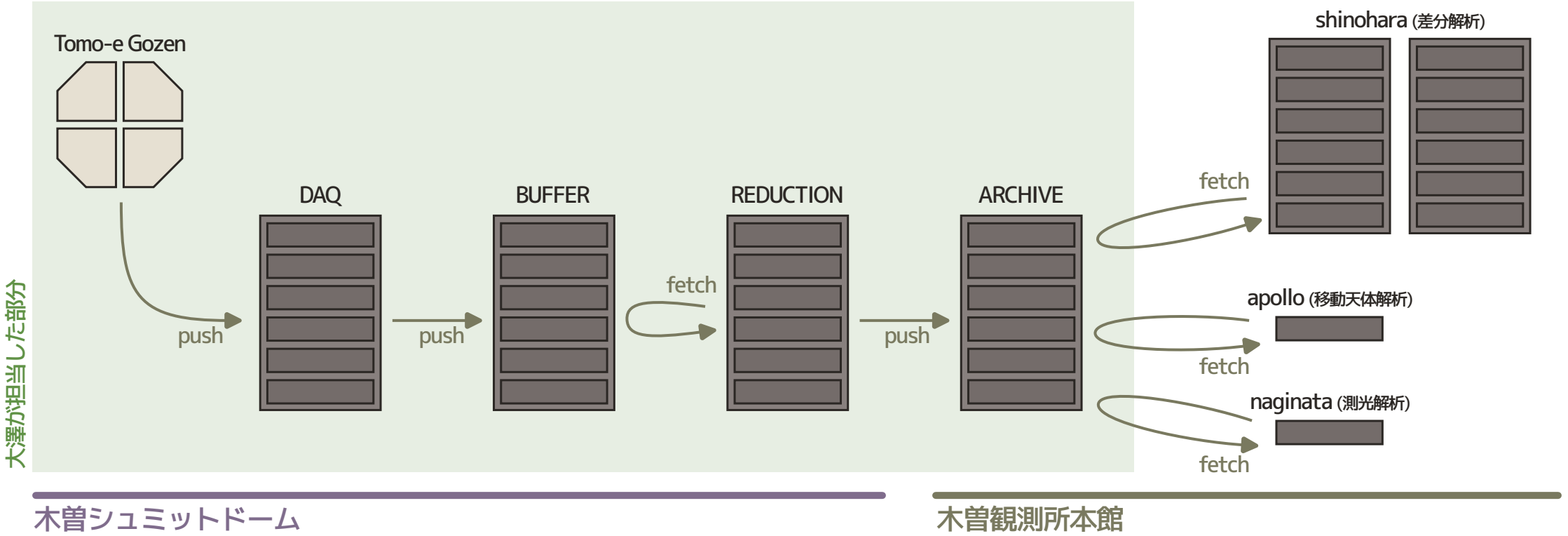
Tomo-e Gozen の開発にあたり重要な要求は以下のとおり

- 一晩に 400 件を超えるサーベイ (+ToO 観測) をハンドリングする
- 最大で ~10 TB/night に達する大量のデータを管理する
- データを自動で解析してすぐにサイエンスに使える状態で共有する
- 木曽観測所で 10 年運用するための体制を整える

これらの視点から Tomo-e Gozen のバックエンドを俯瞰する

Dataflow of Tomo-e Gozen

Tomo-e Gozen を構成する計算機とデータの流れ



Handling of Observations

Tomo-e Gozen の観測に関する主な要求

全天サーベイと領域を絞った high-cadance サーベイが主軸

1 日の観測視野はおよそ 400 pointing × 4 dithering

スケジュールに基づいて効率よくキュー観測を実施する

観測所の天候モニタと連携して自律的に観測を実施する

重力波・GRB・小惑星の観測が突発的にわり込む (優先度に応じて管理)

ネットワーク越しに状態の確認・リモート観測が実施できる

スケジュールの生成は諸隈さん担当
⇨ 動的なスケジュール生成 by 津々木さん

Observation Recipes

観測の内容を「観測指示書」で定義 ⇨ 人間が読める形式で管理

```
Observer: T. Morokuma
Project: All-Sky Survey
```

観測リクエストの責任を明確にする

```
Priority: 0.9400
Time Window: [ '08:00:00.0' ,  '10:30:00.0' ]
```

観測スケジュールに関する要求

```
Operations:
  - SetFocus: 28.13
  - Assert: domeslit_open
  - Pointing: { ra: 19.7498221, dec: 29.7107127 }
  - MirrorCover: open
  - Wait: pointing, mirrorcover_open
  - SetPipeline: [ wcs, stack, neo ]
  - SetParameter: { gain: high, tinteg_sec: 0.5, nframe: 18 }
  - Exposure: J0118+2942_dith1
  - Dithering: [ 0, 1440 ]
  - Wait: pointing
  - Exposure: J0118+2942_dith2
  - Dithering: [ -1980, 0 ]
  - Wait: pointing
  - Exposure: J0118+2942_dith3
  - Dithering: [ 0, -1440 ]
  - Wait: pointing
  - Exposure: J0118+2942_dith4
Comment:
   All-Sky Survey, 120, 2023-12-12T19:23:26.976
```

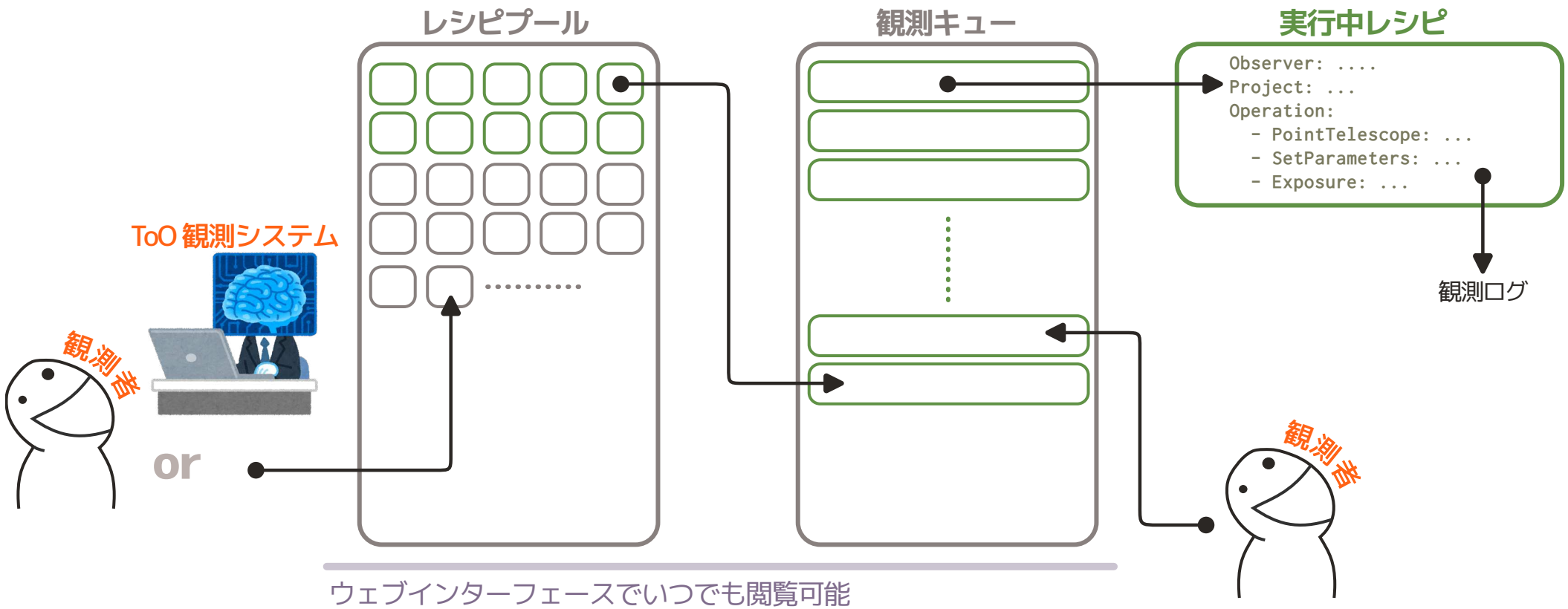
望遠鏡･カメラの動作を時系列で定義する
途中でエラーが発生すると指示書単位でキャンセル
指示書で定義されていない動作は実行できない

Observation Queue System

すべての観測をキューで管理

スケジュールに応じてレシピプールから観測キューへ観測指示書をプッシュ

緊急性の高い観測は優先度で管理 or 観測キューに直接登録



Tomo-e Gozen Queue Status Monitor

Current Schedule (pid: 8611 / standby)

Calibration: 16:50/ Start Observation: 18:20/ End Observation: 05:40

Executing Queue Item (pid: 10883 / lock blocking)

Anonymous Observer | Kiso Observatory
2024-03-07T04:31:23.853041 | d476d83e-5985-571b-96e7-5aabed987a64
Emergency
send message "No recipe assigned"
comment: post a ping message in the log

History

Tomo-e Gozen Scheduler | Kiso Observatory
2024-03-06T20:52:12.021742 | b9a96e0f-f1cc-5678-b108-8fa30a4a775d
comment: Shutdown Tomo-e Gozen System 2024-03-06 20:50:03.140649

Kiso Observatory | Kiso Observatory
2024-03-06T20:50:03.632559 | 109976d0-ebe6-5693-8de0-4a17767961c8
comment: set telescope lock = False

Tomo-e Gozen Scheduler | Kiso Observatory
2024-03-06T08:49:38.786011 | b8492c46-5c9b-5051-a8b6-750a8ef7de18
comment: Scheduled Calibration 2024-03-06 07:40:01.613356

Kiso Observatory | Kiso Observatory
2024-03-06T07:42:49.749577 | 7a3c9a60-22c7-560d-9aa3-f01fc54bc5fa
comment: start up for observation

Kiso Observatory | Kiso Observatory
2024-03-06T07:40:02.268897 | 98a819c8-4804-5627-9ffc-4e5fd59c0704
comment: set telescope lock = False

Tomo-e Gozen Scheduler | Kiso Observatory
2024-03-05T20:52:12.358664 | 3645962a-cd85-5ca7-b3bd-d431bcf7ee05
comment: Shutdown Tomo-e Gozen System 2024-03-05 20:50:02.860099

Kiso Observatory | Kiso Observatory
2024-03-05T20:50:03.402039 | b051a7ae-99e5-540e-b1e4-c7e37f16cec7
comment: set telescope lock = False

Tomo-e Gozen Scheduler | Kiso Observatory
2024-03-05T08:50:32.798646 | ad0c326f-1be4-511d-acb4-58e113e5ebcb
comment: Scheduled Calibration 2024-03-05 07:40:00.511914

Kiso Observatory | Kiso Observatory
2024-03-05T07:43:01.460335 | 1ba76a67-43f6-502e-b980-f84979e958f2
comment: start up for observation

Kiso Observatory | Kiso Observatory
2024-03-05T07:40:01.201629 | 574f309b-e424-5ce4-a077-0b016c0cd3a5
comment: set telescope lock = False

Tomo-e Gozen Scheduler | Kiso Observatory
2024-03-04T20:52:02.656724 | 45aaf204-0d4a-5571-8274-1eb888b498e5
comment: Shutdown Tomo-e Gozen System 2024-03-04 20:50:02.973238

Scheduled Observations (total: 6)

T T T T T T

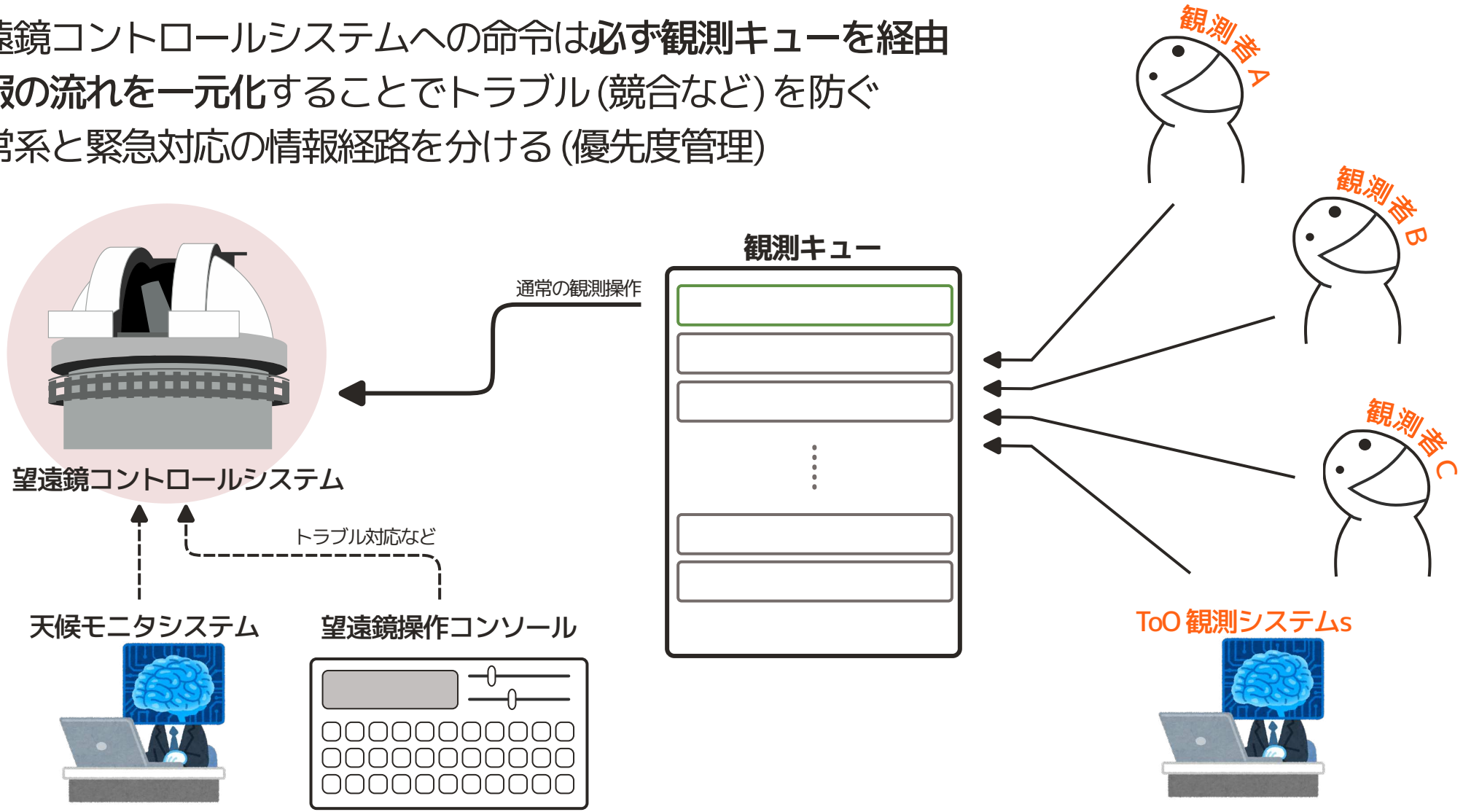
レシピプール

First 10 Recipes in the Queue (total: 0)

観測キュー

Observation Queue System

望遠鏡コントロールシステムへの命令は**必ず観測キューを経由**
情報の流れを一元化することでトラブル(競合など)を防ぐ
正常系と緊急対応の情報経路を分ける(優先度管理)



Handling of BIG Data

Tomo-e Gozen のデータハンドリングに関する主な要求

動画観測による ~20 TB/night (max) のデータを適切に処理する

データの流れが滞るとさまざまな性質の悪い障害が発生する

- トラフィックが増大することによるパフォーマンスのさらなる低下
- プロセスが渋滞することによるメモリ･ストレージの不足 ⇨ プロセスの異常終了
- リングバッファの上書きによるデータの損失 ⇨ 異常なデータの生成

連続的に生成されるデータの洪水に対処するためには...

- データ解析レート > データ取得レートを保証する
- バッファとしてストレージを確保してパフォーマンスのブレを吸収する

Tomo-e Gozen では
こちらのデザインを採用

BIG Data Transfer

カメラ⇨バッファ用ストレージまで

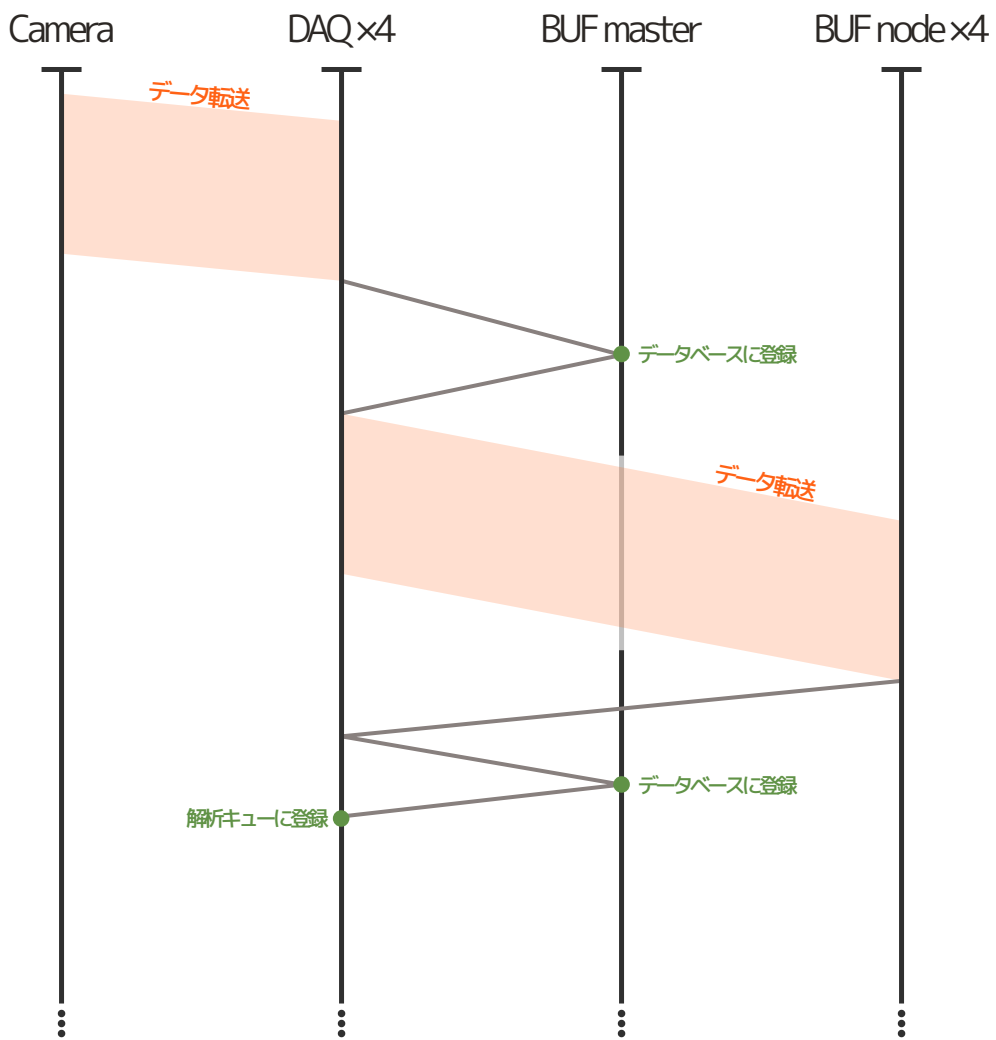
ストレージは 4 台の計算機で構成

データ取得用計算機が特定のカーネルでしか動作が確認できなかったため, 広く使われているストレージシステムを採用せず自前で実装した.

合計で 251 TB の生データを保持

ストレージの I/O に高い性能を要求する
障害耐性についてはそれほど重視していない
RAID10 をベースにしてストレージを構築

データには TTL を設定 ⇨ 自動消去



BIG Data Transfer

データ解析 ⇨ アーカイブまで

バッファ前後で負荷を切り離す

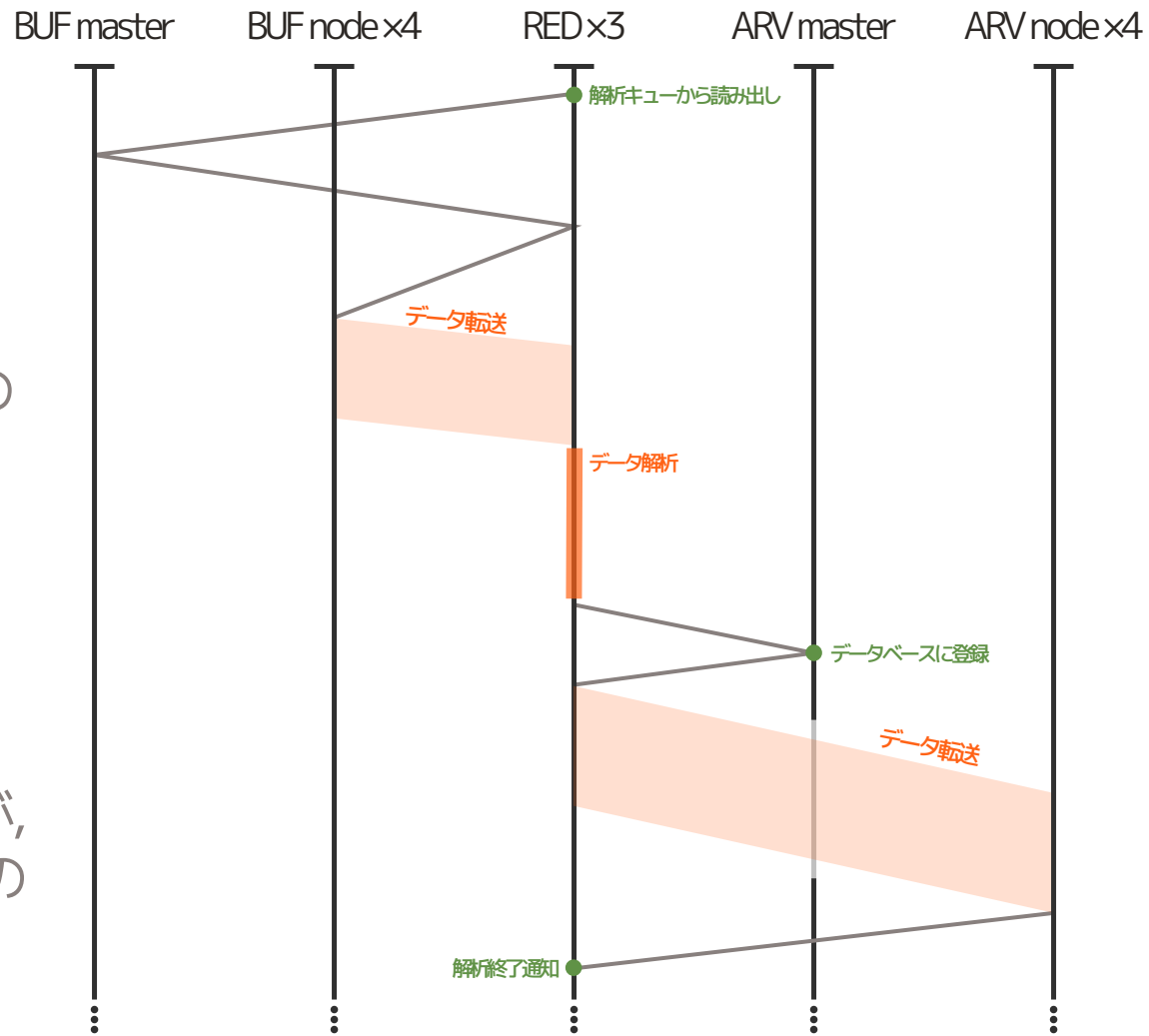
解析キューにはシンプルな rq を採用

リソースを適切に分配できるジョブシステムの採用を検討するべきだった

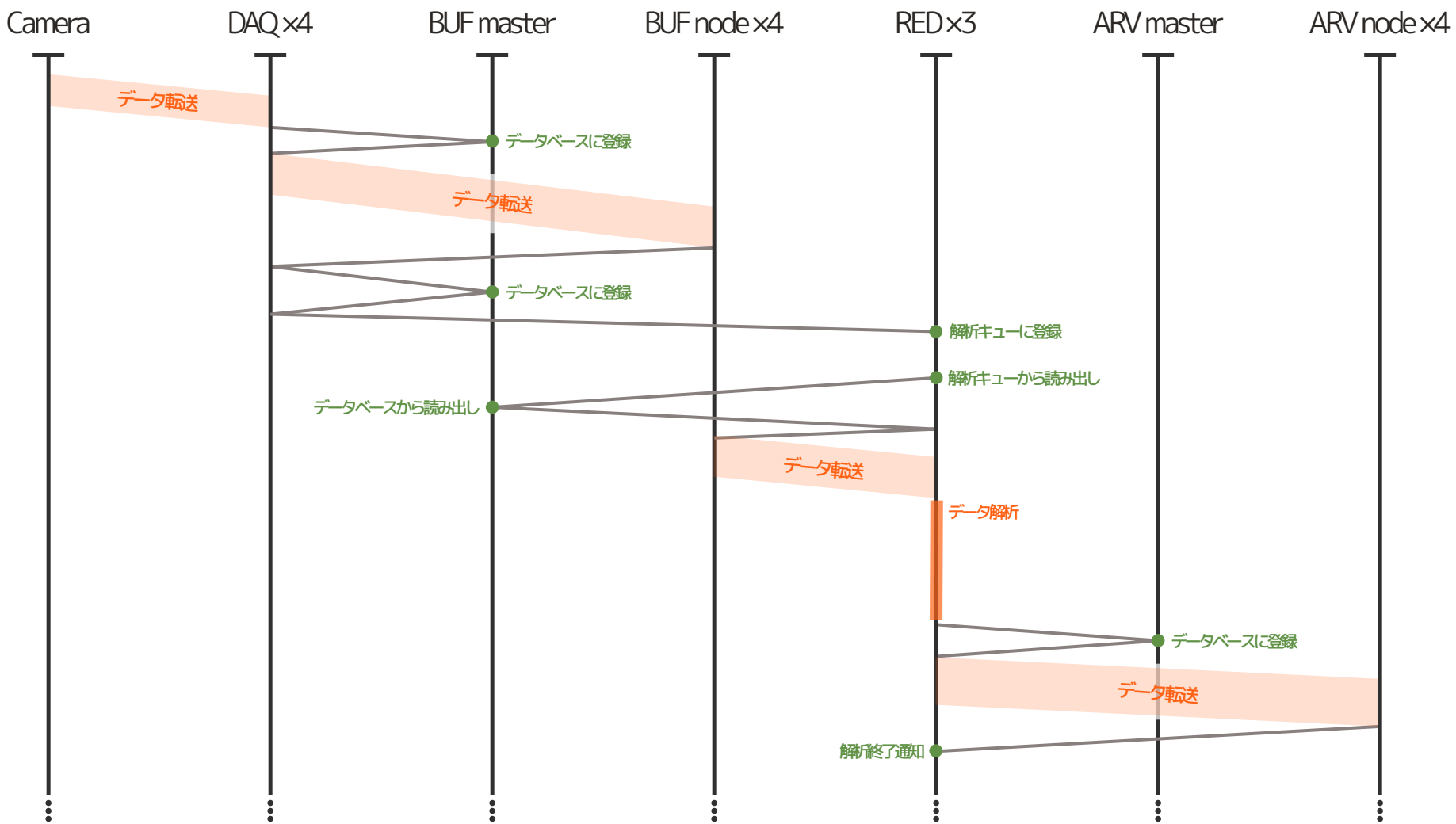
計算機 4 台でアーカイブを構築

合計で 1.4 PB のストレージに相当

バッファ用ストレージのシステムを流用したが，結果的に Lustre などを採用したほうがその後の運用で利点があったように思う



BIG Data Transfer



Struggles with Data Transfer Problems

データ取得系への要請: **バッファへの転送レート > データ取得レート**

- データ取得システムのリングバッファを上書きしてしまいデータが破損する

データ転送を高速化するためのわるあがき

- ストレージの高速化 (複数計算機 & RAID10 による高速化)
- ネットワークの高速化 (10 GbE ×2 のネットワークカードを搭載)
- リソースの管理 (データ取得・転送で異なる CPU を割り当て)
- データ転送の省力化 (非暗号化通信)
- ファイル I/O をできるだけ減らす

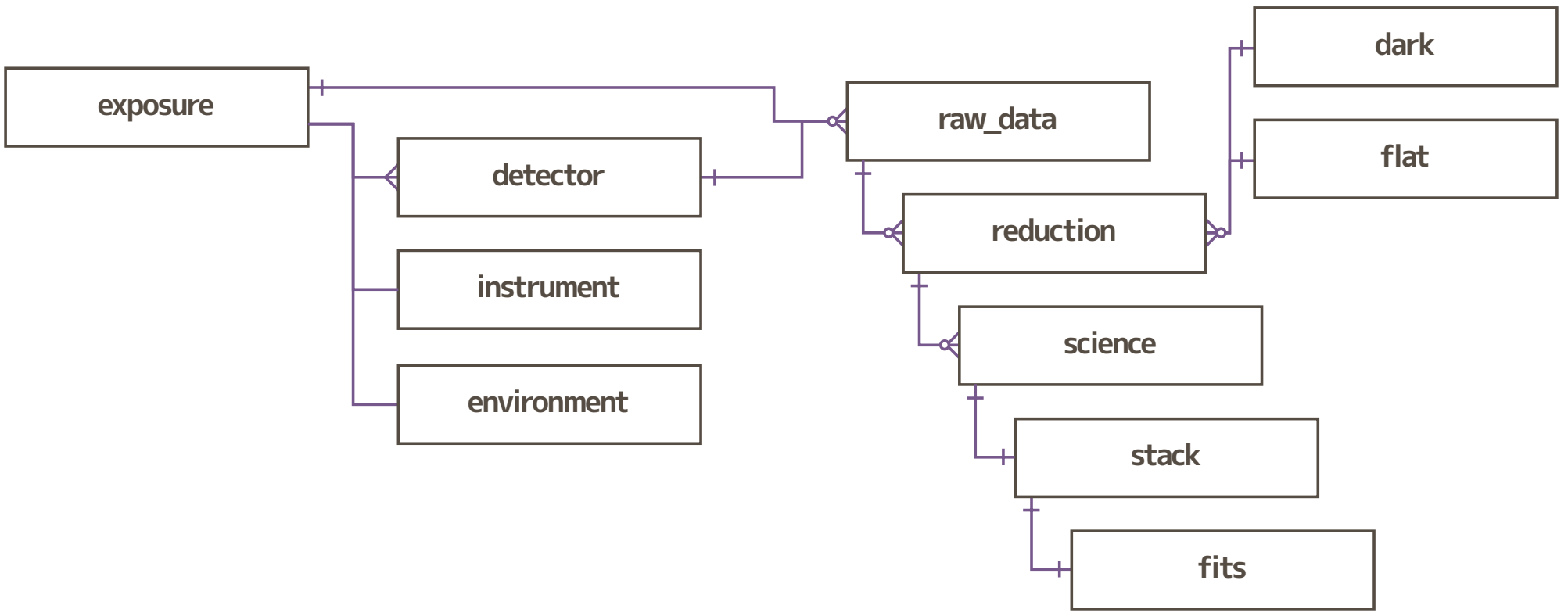
⇨ まだ速度が足りておらず連続的な 2 fps 観測は未達成

Design of Tomo-e Gozen database

データベースの設計 ⇨ 情報の対応関係と更新タイミングでテーブルを定義

一度登録したレコードの更新は避ける (パフォーマンスとトラブル対策)

Tomo-e Gozen データベースの概略



Handling of Data Reduction

Tomo-e Gozen のデータ解析に関する主な要求

- 大量の動画データを自動で解析する (個々の解析に対して人間が介入しない)
- 迅速な追観測・アラート発信 ⇨ データ解析レート ~ データ生成レート
- アーカイブするデータ量を < 1 TB/night 程度に抑える (10 年運用 ~ 1000 日)
- トレーサビリティに配慮 (生データは残せない・解析したコードは辿れるように)
- あやしいデータはできるだけアーカイブしない (後続する解析への信頼性)

Toward Automated Data Reduction

観測指示書で解析内容を定義

stack: 時間方向に平均

wcs: 座標解決（アストロメトリ）

neo: 移動天体解析

cube: 動画データとして保存

```
Operations:
  - SetFocus: 28.13
  - Assert: domeslit_open
  - Pointing: { ra: 19.7498221, dec: 29.7107127 }
  - MirrorCover: open
  - Wait: pointing, mirrorcover_open
  - SetPipeline: [ wcs, stack, neo ]
  - SetParameter: { gain: high, tinteg_sec: 0.5, nframe: 18 }
  - Exposure: J0118+2942_dith1
```

観測開始前にかならずキャリブレーションデータを取得

取得したデータに対応する dark, flat データは事前に取得している状態を正常とする

クオリティチェックをクリアしたデータのみアーカイブする

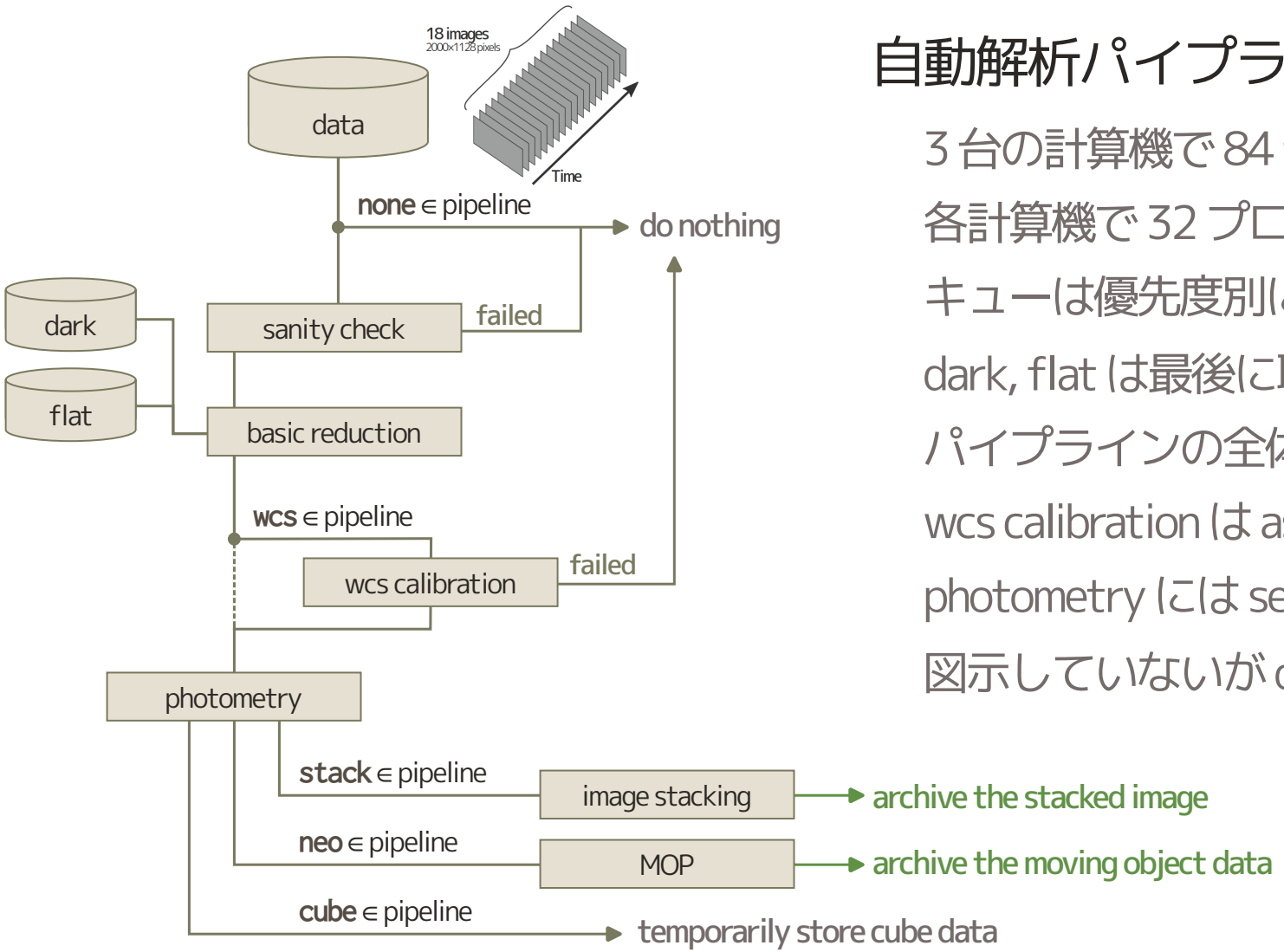
平均カウントが高すぎる ⇨ 視野全体でサチュレーションを起こしている

ノイズが規定値よりも高い ⇨ データ欠損により画像が壊れている

時刻情報・座標が整合的でない ⇨ センサやデータベースに不具合が生じている

アストロメトリ解析の失敗 ⇨ おそらく曇っている

Data Reduction Flowchart



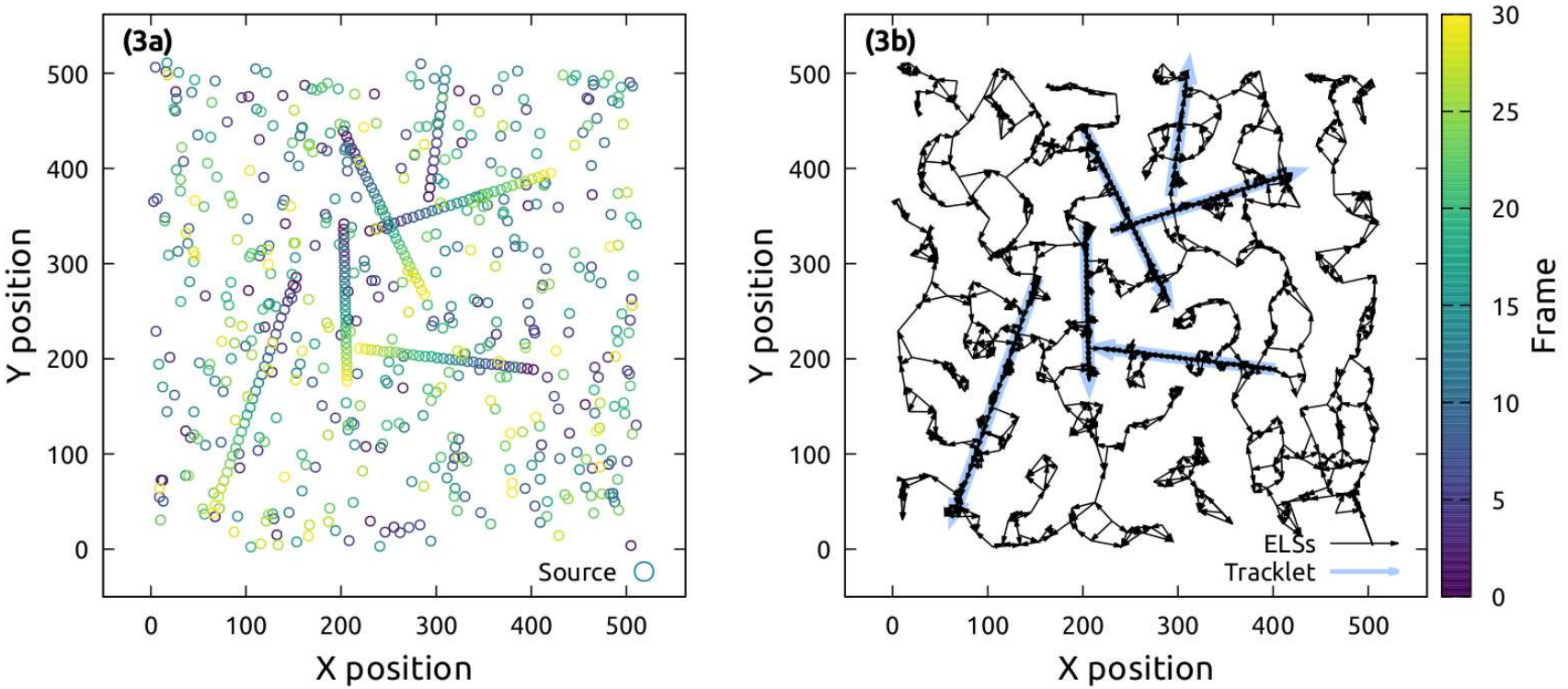
自動解析パイプラインの概要

- 3 台の計算機で 84 台の検出器を分担
- 各計算機で 32 プロセスの worker がキューを逐次処理
- キューは優先度別に 2 系列を用意 (default, express)
- dark, flat は最後に取得したものを使用
- パイプラインの全体は Python で構成
- wcs calibration は astrometry.net を使用
- photometry には sep を使用 (Source Extractor)
- 図示していないが dark, flat 用のルートもある

Moving Object Processing

移動天体を効率よく抽出するアルゴリズムを開発

- 解析対象を点から線分に変換することで O(n) ~ $n^{1.5}$ の高速化を実現
- Tomo-e Gozen のサーベイ観測に適用 ⇨ 35 天体の地球接近小惑星を発見



Ohsawa (2021), JSSIJ, 11, 1-10

Struggles with Automated Data Reduction

迅速な追観測のためには (ほぼ) リアルタイムのデータ解析が必要

- できる限り中間ファイルは出力しない (ファイル I/O を減らす)
- 避けられない場合は高速なストレージに置く (calibration = RAM disk, astrometry = PCIe SSD)
- 高速で動作するアルゴリズム・ソフトウエアを採用する & 妥協する
- 時刻付きでログを残してパフォーマンスのボトルネックを把握する

大量のデータ解析 ≒ 常にエラーメッセージが出続けているシステム

- 例: 曇ってくる ⇨ アストロメトリの失敗 ⇨ 84 件のエラーがほぼ同時に発生
- 把握している例外にはすべて名前を点けておく (見慣れないメッセージ ≒ 想定外の動作)

Safe & Stable Operations

じこはおこるさ (Accidents will happen)

- コマンド･パラメタを書き間違える
- 複数人で同時に操作して干渉する / 連絡･報告を忘れる･行き違う
- ネットワークエラーによるシステム障害
- ハードウエアの想定外の挙動 / 計算機の故障

リモート切り替えスイッチ

障害･事故が起こりにくい環境を整備する

- ドーム作業時には望遠鏡コントローラをネットワークから隔絶させる
- 情報の流れを整理して競合が発生しないデザインにする

「正常であること」をきちんと定義して異常を見逃さない

- 「正常ではない」ことを検出したらすぐにチーム Slack に投稿 (自動化)
  - 例: ドームが閉まっているのに天体を観測しようとする
  - 例: 観測開始時にミラーカバーが開いた状態になっている

Safe & Stable Operations

システムの状態を常に把握する

サーバなどの状態を持った常住プロセスを組み合わせてシステムを構築
cron はトラブルのハンドリングが煩雑になるので避けた
プロセスとログをまとめて管理できるソフトウエア (supervisor) を採用

ウェブベースで可視化 & Slack との連携

観測の進捗や計算機の状態を一覧できるウェブページを用意
ブラウザさえあればどこでも確認できる (not リモートデスクトップ環境)
観測システムからチーム Slack に逐次通知を送る ⇨ 問題の即時共有

Software Development

Tomo-e Gozen は10 年超の運用 (> 僕の任期)

コードの共有や長期間のメンテナンスを見据えた開発環境の整備と布教

バージョン管理ソフト/プラットフォームの採用

バージョン管理ソフトウエア: git, 開発プラットフォーム: bitbucket.org
複数人でのコード開発とレビュー文化を布教中

- ×ファイル名に日付を付けて複製
- ×更新履歴をコードのコメントに残す　⇨ だいたい他人にとってはただのノイズ
- ×開発途中に生成したデータを残す

docker 等のコンテナ技術の有効活用

テストの作成･テスト環境 (ダミーシステム) の整備

開発をふりかえって

Key Factors behind Tomo-e Gozen

なぜ Tomo-e Gozen の開発は頓挫しなかったのか？

段階的な開発サイクル (Tomo-e PM → Q0 → Q1 → FM)

小さいスケールで先に開発をすることで問題の洗い出しができた

カメラの開発も並行 ⇨ 観測システムの一部を先行開発 ⇨ 疎結合なシステムの実現

Tomo-e Gozen のポリシーが明確に決まっていた

要求がシンプルかつ明確 (撮像観測のみ/装置・フィルタの交換はなし).

既に自動サーベイ観測の経験があり, ユースケースが具体的に想定できた.

必要なこととやらなくてもいいことが明確だったので割り切った設計ができた.

チーム内で合意がとれていたので開発中に方針がブレなかった.

Key Factors behind Tomo-e Gozen

なぜ Tomo-e Gozen の開発は頓挫しなかったのか？

木曽観測所のメンテナンス・サポート体制

望遠鏡の動作とと天候モニタの判断を信頼することができた.
「信頼できる」要素はモジュールとして扱えるのでシステムの見通しが良くなる.
ネットワークや電力などのインフラに先行投資 ⇨ 滞りなく開発が進められた.

高機能なソフトウエア・ライブラリの活用

astropy, astrometry.net, sep, photutils などの天文解析用ソフトウエア
flask, requests, supervisor, docker などのシステムを管理するためのソフトウエア
低レベルの I/O や通信に苦労することなくリソースを必要なところに集中できた

適切なツールの選択 ⇨ 目的が明確に定義できた (言語化できた) ことが重要

Lessens Learned in Tomo-e Gozen

Tomo-e Gozen の開発で感じた反省

テスト環境の用意やテストに基づいた開発をするべきだった

計算機の故障にそなえて docker 等のコンテナをもっと活用するべきだった

望遠鏡コントローラなどを含めたテスト環境を用意することはむずかしいが,
長期間の運用とメンテナンスを考えるとリソースを割く価値はあった.

計算機のリプレイス (環境の再構築) が必要な事態に備えてコンテナ環境で運用したい

ドキュメントの整備がいたらない部分が多かった

開発がノッているときにでも立ち止まってドキュメントを整備する余裕 (がない)

数が多いとストレージは頻繁に壊れる

年間で 3-4 件程度ハードディスク障害で交換している (幸いデータのロストはまだない)
大容量のハードディスクが利用可能になる ⇨ リプレースに苦戦

Summary

木曽観測所 Tomo-e Gozen のバックエンドの開発をふりかえった

観測システムとしての Tomo-e Gozen

- 大量 (~ 400 件) の観測をスケジュールに沿って自律的に実施する
- 大量 (~ 10 TB/night) のデータを滞りなくハンドリングする
- データを自動で解析してサイエンスに使用可能な状態にする
- >10 年間継続的に運用可能な体制を整える

開発を支えた要素

- 段階的な開発サイクル (問題発見・解決の繰り返し/粗結合なシステム設計)
- 明確な装置ポリシー (優先課題の特定とわりきったシステムデザインの採用)
- 木曽観測所の経験と体制 (インフラ整備/メンテナンス/天候モニタ)