Zwicky Transient Facility
Data System Review

Jason Surace & Frank Masci
IPAC/Caltech (12/14/2015)

Why are we here?

- Review the design and functionality of the existing iPTF data system, which informs the functional requirements for ZTF.
- Examine the existing iPTF system performance.
- Discuss approach for resolving remaining design choices for ZTF.
- Show the schedule for deployment and readiness for ZTF first light.

PTF/iPTF: Evolutionary and Agile Development

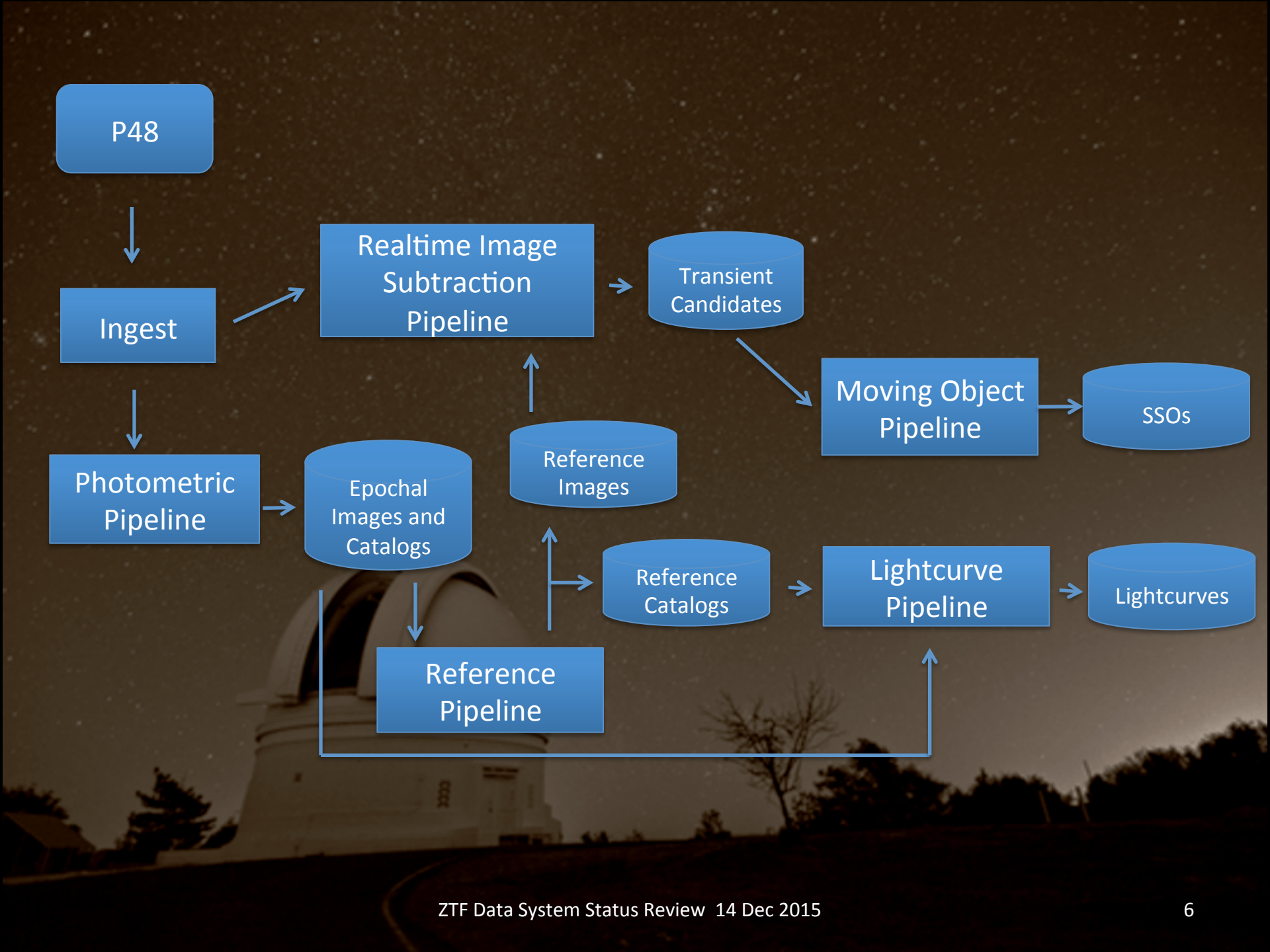
- PTF grew organically, with requirements being driven by a constantly evolving science program.
- Initial IPAC system grew as a concept to produce well-calibrated image products, to supplement the existing NERSC image subtraction and transient detection pipeline.
- System has evolved continuously over the past six years to add new capability: detection of solar system objects, light curve generation, deep coadds, etc.

Why is this relevant?

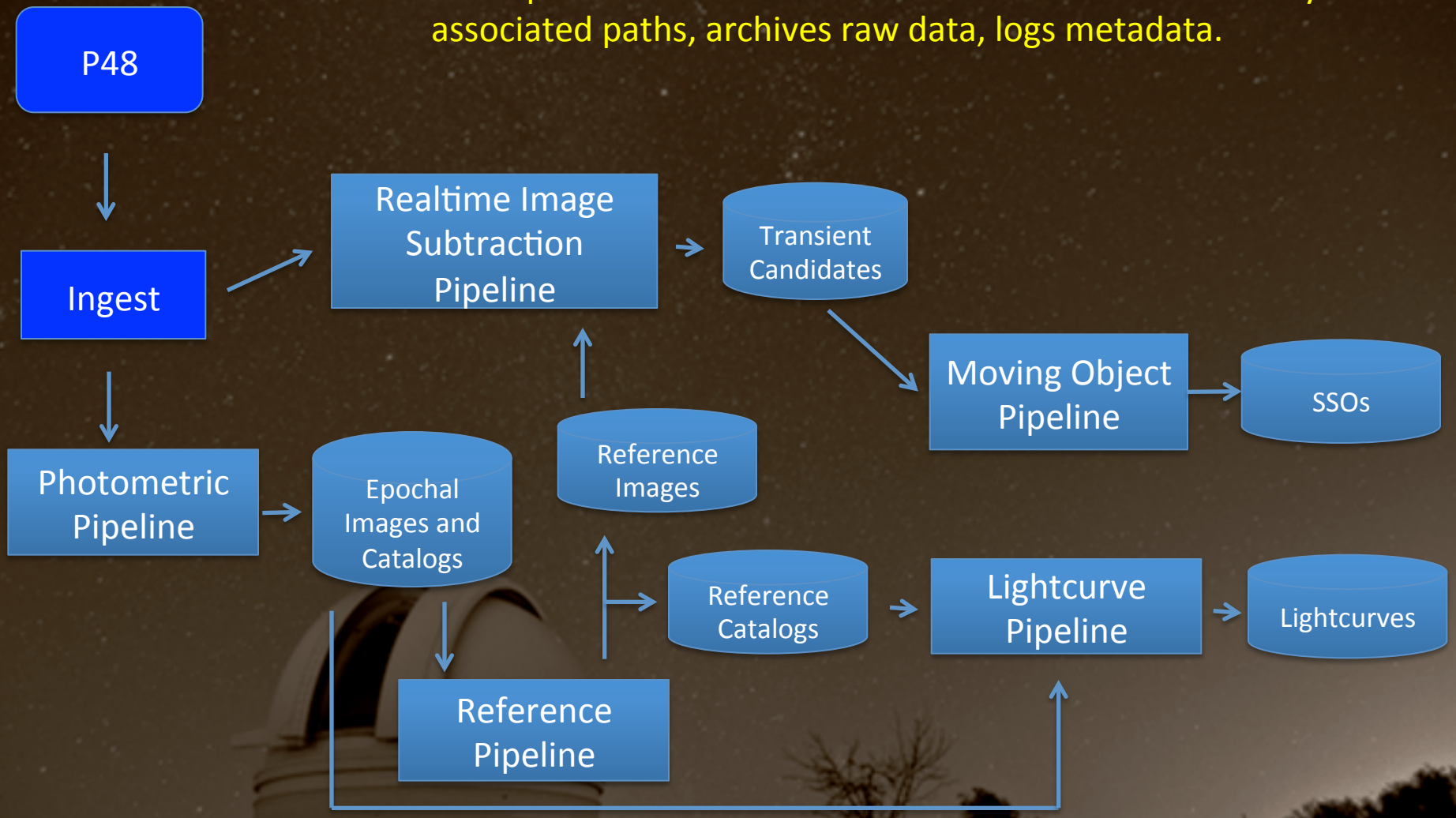
- ZTF has no formal requirements for the data system. As a result the ZTF project proposal was based on duplicating the capabilities of the existing PTF/iPTF data system, as it existed or was envisioned to exist at the time of the iPTF/ ZTF switchover.
- ZTF funding model used for the MSIP proposal was designed to be “low budget” in that it proposed to reuse the existing PTF software heritage.
- Significant new code development and new pipeline segments are outside the planned funding envelope and development horizon.
- Understanding iPTF is the model for the first year of ZTF.

Primary Pipeline Segments

- Data Ingest – receipt of data, initial logging of metadata.
- Realtime Data Processing – image subtraction, transient detection, machine vetting, and delivery of products to the treasures portal and science marshals.
- Solar System Pipeline - solar system object detection, tracklet construction.
- High Fidelity Daily Processing – nightly processing and recalibration for highest data quality images and source catalogs.
- Reference Image Pipeline – coaddition as input to other pipelines.
- Relative Photometry – periodic construction of coadded images, processing of catalogs to create high precision light curves.
- Data Archiving - storage of all raw data, processed data (images and extracted photometry), and an advanced data archive with data exploration tools enabling public and in-collaboration access.


P48
Ingest
Photometric Pipeline
Epochal Images and Catalogs
Realtime Image Subtraction Pipeline
Transient Candidates
Moving Object Pipeline
SSOs
Reference Images
Reference Catalogs
Lightcurve Pipeline
Lightcurves
Reference Pipeline

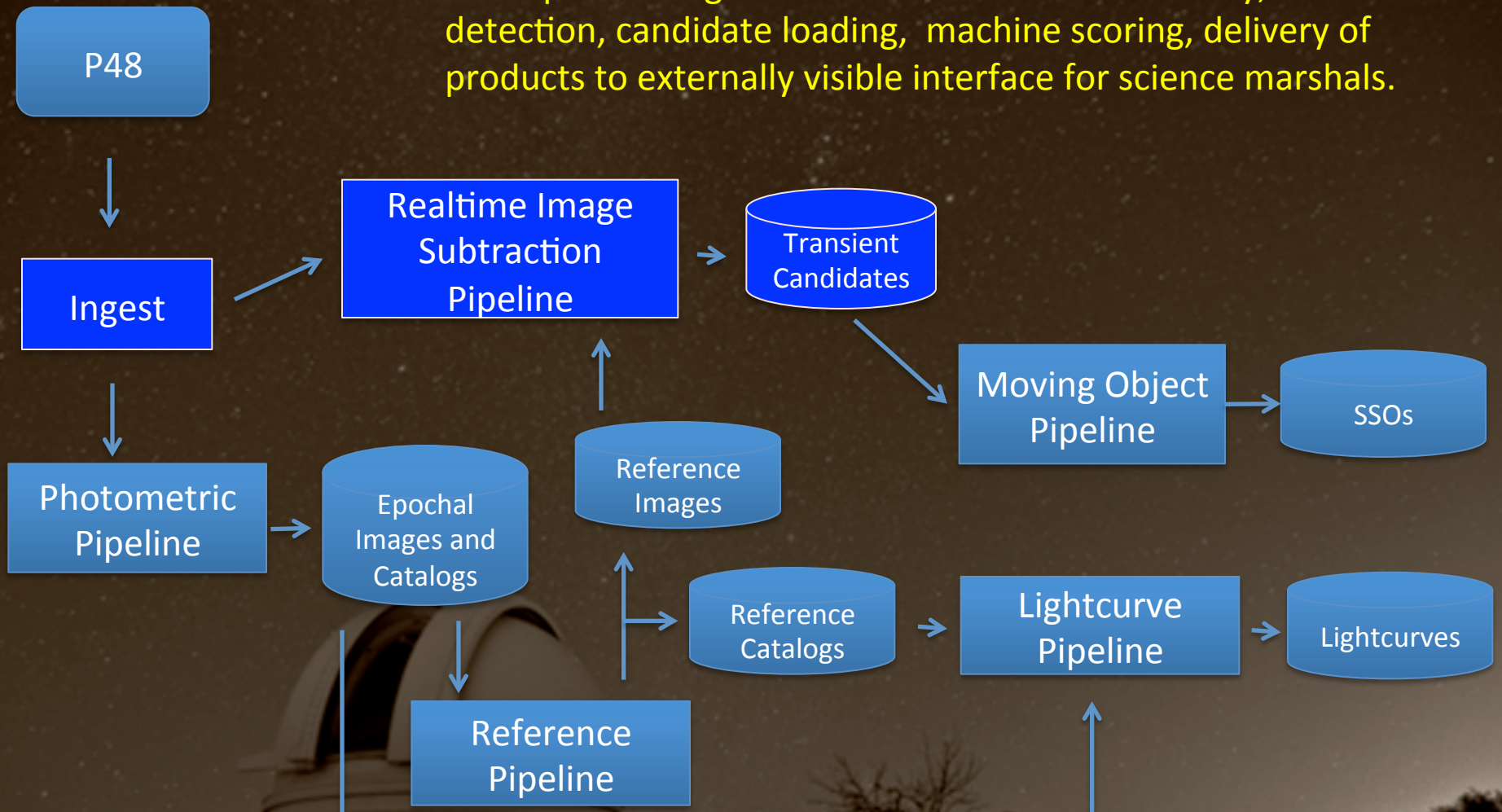
Description: receives data in realtime via microwave relay and associated paths, archives raw data, logs metadata.



Timing Requirements:
No specific req – rolled into realtime requirements.

Products: 185 MB/110 sec, 50GB/day, all image products.

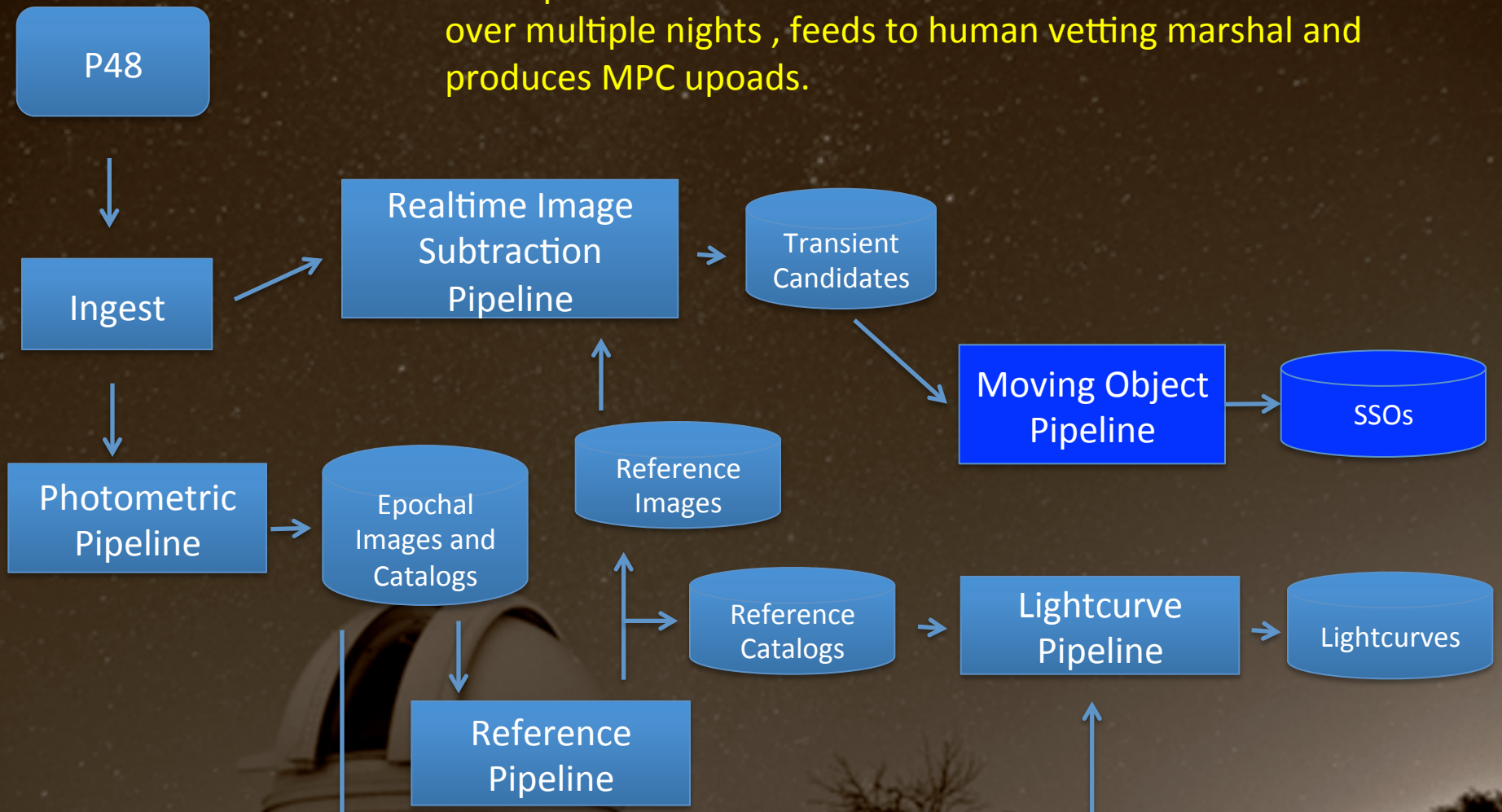
Description: image subtraction vs. reference library, transient detection, candidate loading, machine scoring, delivery of products to externally visible interface for science marshals.



Timing Requirements:
20 minutes from time of receipt at IPAC, 10 min stretch goal.

Products: 400 MB/110 sec, 100GB/day image products, 20e3 candidates/110 sec.

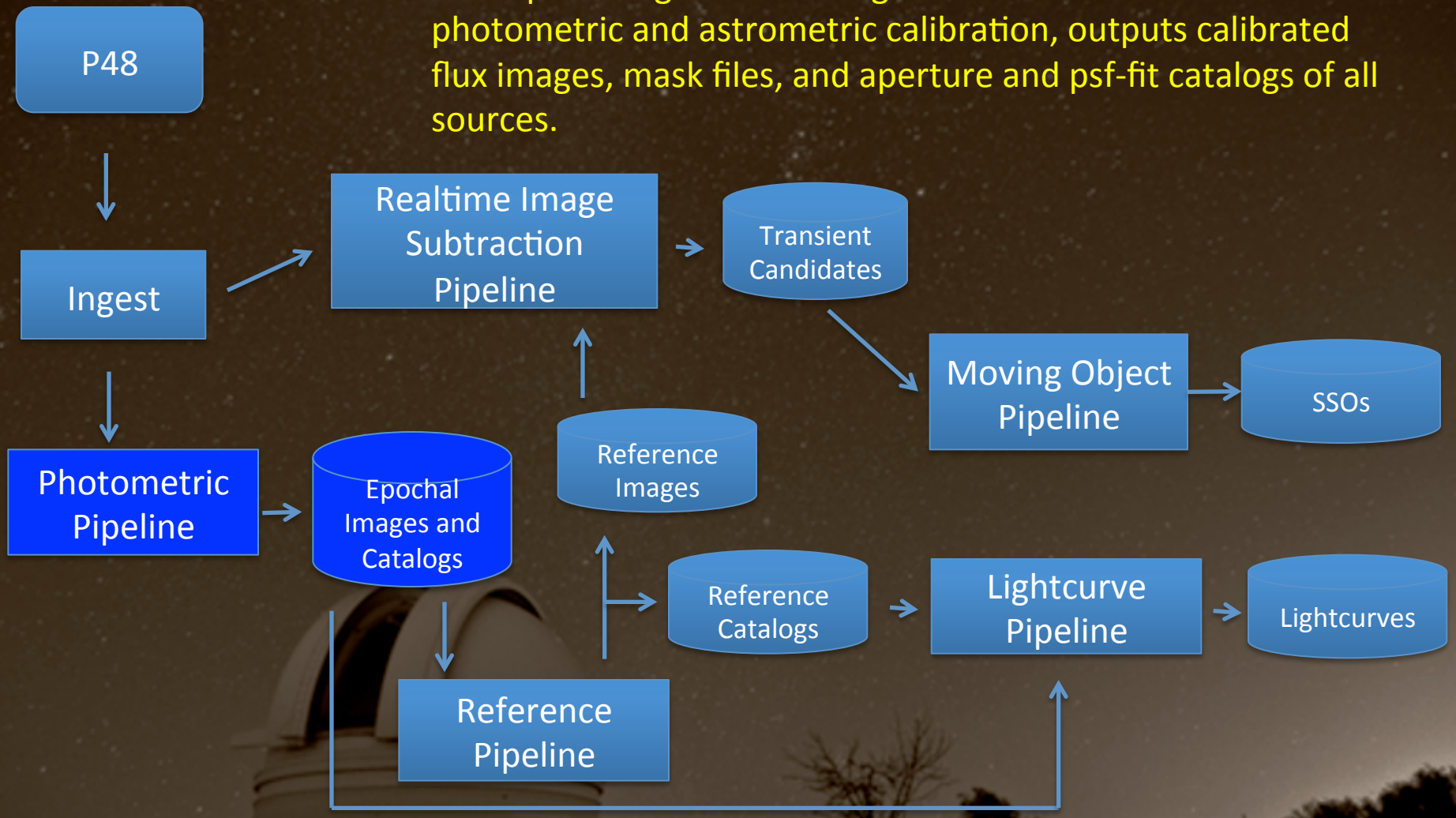
Description: assembles tracklets from transient candidates over multiple nights , feeds to human vetting marshal and produces MPC upoads.



Timing Requirements:
Runs every four hours, uses previous three nights.

Products: very small database and catalog files.

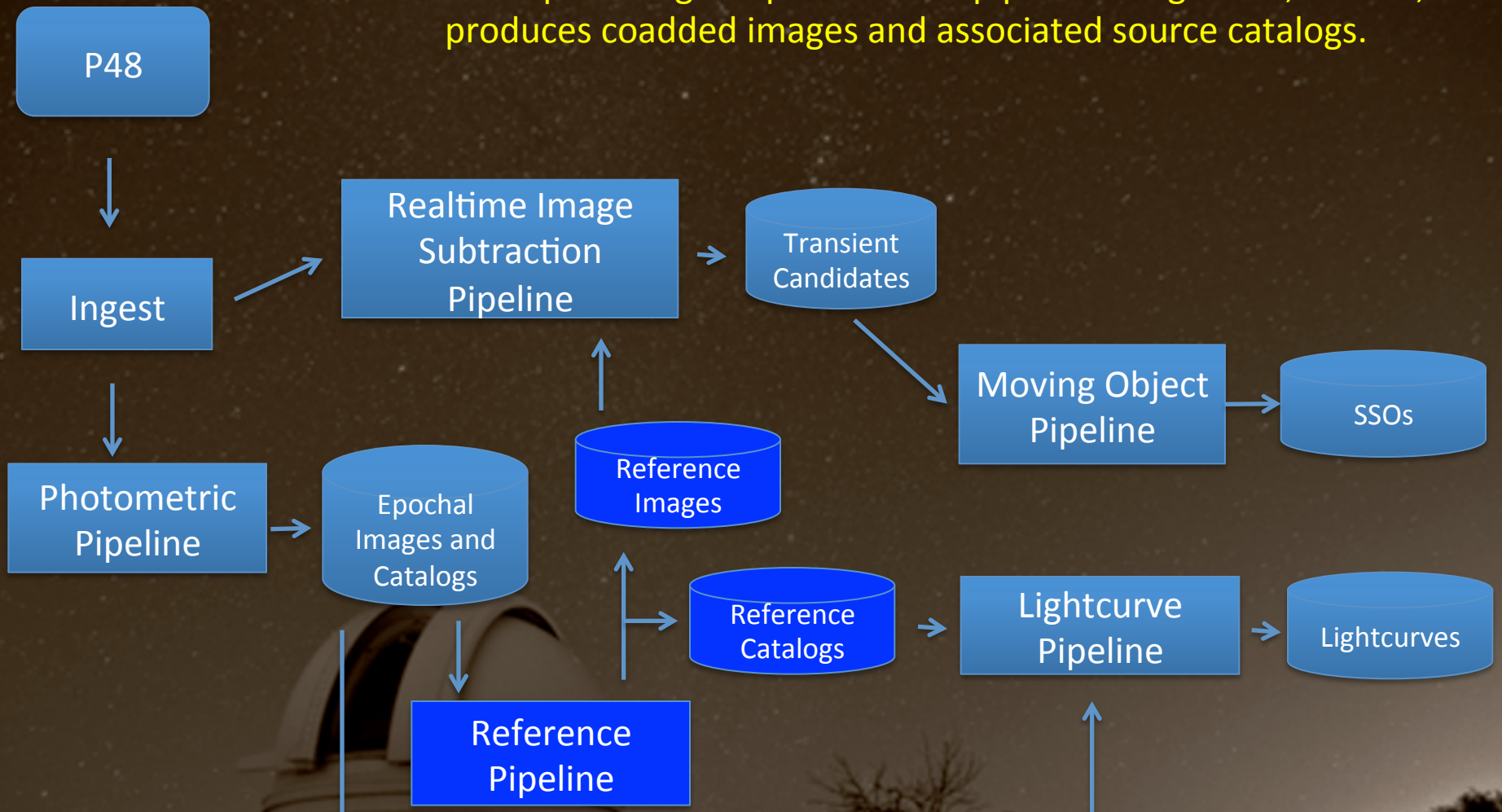
Description: ingests entire night's worth of data to derive photometric and astrometric calibration, outputs calibrated flux images, mask files, and aperture and psf-fit catalogs of all sources.



Timing Requirements:
Starts at end of night, must complete in 12 hours.

Products: 100GB/day image files, 20GB/day catalog files, no DB upload, but 10-30 million records per night to archive, 10B/yr.

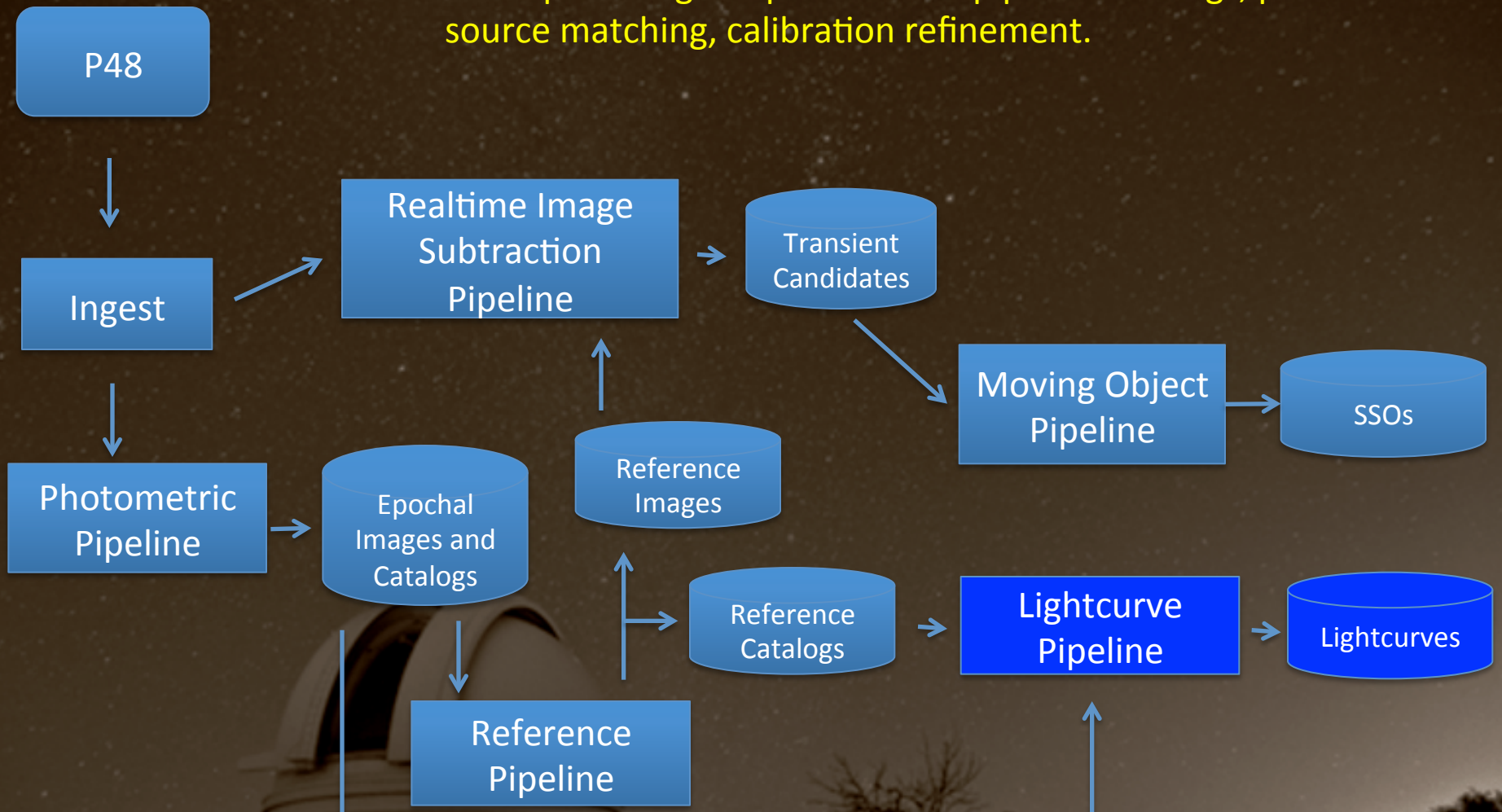
Description: ingests photometric pipeline image data, coadds, produces coadded images and associated source catalogs.



Timing Requirements: runs on-demand and infrequently, but must complete sky in less than 6 months.

Products: of order 300GBimage data for entire sky with current tiling scheme. 1B DB records.

Description: ingests photometric pipeline catalogs, performs source matching, calibration refinement.



Timing Requirements: source matching runs daily, relative photometry once a month.

Products: continuously accruing match files, 10-30M associations per night.

Archive and External Interfaces

- Primary image and catalog archive is through IRSA.
- GUI and API interfaces.
- IRSA receives nightly deliveries of image and catalog files, and sql load files for catalog data. Currently several hundred TB of file data, and 25B rows of catalog data.
- Process of transferring ownership and registering metadata typically incurs 1-3 day delay in appearance in archive.
- Realtime data is accessed via a specialized web server, containing API interfaces to query transient candidate tables, access image data.

PTF Software Architecture

- Mixture of community and in-house developed software, organized into “pipelines”, which are self-contained tasks.
- Pipelines are structured within PERL wrappers.
- Job control is managed through a central database.
- “Jobbers” running on pipeline drones poll the job control database for work.
- Heritage from previous IPAC missions.
- Pipelines work within local scratch disks, ship data upon completion to an intermediate shared disk pool called a “sandbox”.
- Archive process transfers sandboxed data to permanent archive storage.

PTF Operations Hardware

- Primary compute hardware consists of two banks of 12 drones each (24 total), running RHEL. Drones have 8/12 cores and 16/48GB of ram.
- One dedicated operations Sun/Solaris file server with Nexsan attached storage.
- One shared archive Sun/Solaris file server with Nexsan attached storage.
- Two dedicated operations DB servers. One primary and one user copy.
- One shared DB server for archive.
- Transfer computers.
- Operations web server.
- Various IPAC infrastructure (switches, firewall, etc).

Parallelization Quanta

- Massive parallelization is required for processing this sort of survey data.
- Parallelization is based on detector and spatial tiling on sky. This allows asynchronous processing.
- This is baked deep into the system architecture. Other surveys failed to use this approach, and crashed as a result.
- This improves image subtraction, light curve generation, etc, but is not conducive to absolute calibration.

PTF vs. ZTF Data Rates

- ZTF not only has more pixels, they come faster.
- ZTF focal plane has roughly 6.5x as many pixels as the functional PTF camera.
- However, the cadence has also increased from an achieved 110 second cadence with PTF to a planned 45 second cadence.
- Resulting change in data rate is a factor of 16.
- This is the single greatest challenge in adapting to ZTF.

Total Product Storage

- Note that the total storage requirement is different from nightly storage rate due to weather.
- Analysis of iPTF predicts 376 science exposures on average per night + 56 calibration exposures.
- Assume raw data is Rice compressed, processed images and masks gzipped or similar:
- Compressed Raw Data = 70TB/yr, compressed.
- Processed Image Data = 300TB/yr, compressed.

Need about 0.5PB/yr for minimum archive storage. Baseline funded operation is three years.

MSIP Proposed Architecture

- The data system portion of the MSIP proposal was based on a scaling of the existing PTF architecture.
- Existing PTF design was known to work.
- ZTF CCD readout quadrants (64 total) similar in size to a PTF CCD.
- ZTF system would replicate the PTF system into independent subunits based on detector, each of which has it's own servers and storage.
- Hardware requirements derived from direct scaling of existing system.
- This remains the fallback approach should we discover significant scaling problems – sharding by spatial sky and detector location.

Tractability Issues in Existing System

- Existing job management requires hands-on decisions regarding resource management. Non-optimal resource utilization, particularly CPU.
- Non-automated robustness to hardware failures, which are increasingly likely on a cluster this size.
- Difficulty in packaging new job types.
- Unnecessarily cumbersome deployment and configuration control for software and hardware.

Requirements & Capabilities

MSIP Proposal Requirements

- System is being designed to meet the operational requirements needed to make the NSF-mandated delivery schedule. This is commensurate with the supplied funding profile.

- Data ingest & Realtime Processing: 20 minute turnaround for scored transient candidates from receipt at IPAC.
- High-Fidelity Processing: up to 1000 science exposures completed within 12 hours
- Reference Image Processing: 40 thousand references in 6 months (full sky).
- Light Curves: new solutions on weekly timescales.

MSIP Proposal Requirements (2)

- All deliverables could be made with the iPTF system that will exist at the iPTF/ZTF handover, with the exception of the automated alerts. Additional capabilities are out of the funding and planning horizon through 2017.

- The existing PTF system is used heavily to run “ad hoc” jobs. These are primarily special or experimental processing to support in-collaboration science.

- The ZTF system will not preclude such jobs, but would require additional computing and personnel resources.

ZTF Data Systems Design

Steve Groom, Ben Rusholme,
Russ Laher, Jason Surace

outline

- Design approach
- PTF overview
- Evaluating PTF
- Major changes for ZTF
- ZTF design overview
- Status of subsystems

Design Approach for ZTF

- Overall idea has been to model ZTF after PTF
  - Extrapolate PTF system design to larger sizes and higher rates

- PTF data system is "working"
  - Data volume requirements are clear
  - Processing: pipeline algorithms implemented and working
  - Throughput is acceptable

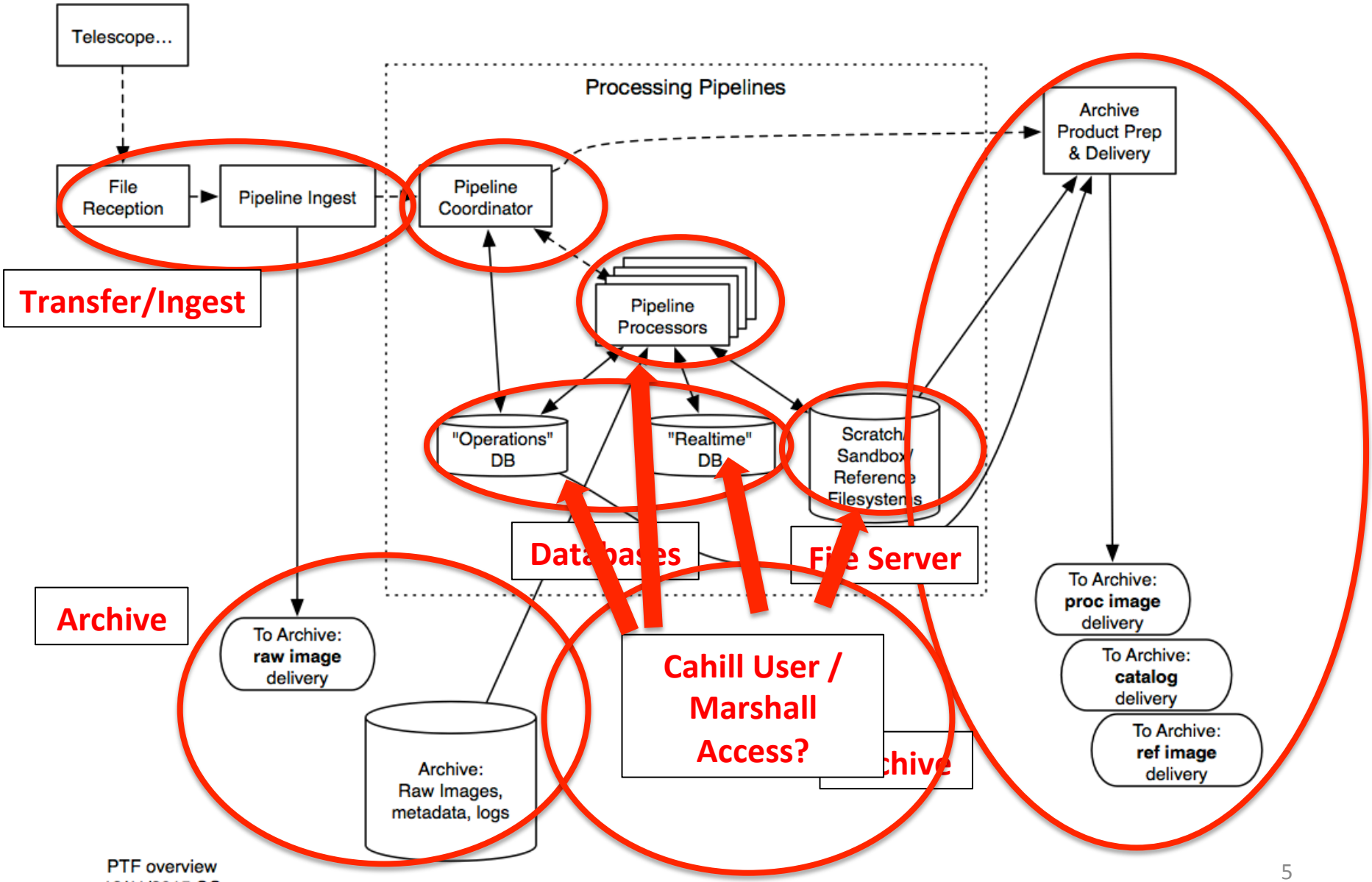
- HOWEVER: current system has not been carefully studied or tuned – overall behavior is not well understood
  - Data *movement* within the system not very clear
    - important since I/O is a major part of the picture, how to scale?
  - Database is a "black box"
    - PTF relies heavily on central DB, but we don't have a good sense of how well it is working
    - Can we scale "up" or "out"? Or ?
  - Much of PTF realtime data access (and the load that imposes on data systems) is unknown
    - Users/Marshalls tap directly into data system, with unmeasured impacts

Design Approach for ZTF (cont.)

- Challenge for design approach:
  - requirements-driven design difficult to do since we are missing important details about and won't have a complete set of requirements
  - Extrapolation difficult since we don't have a good sense of what will scale easily and what won't – don't know where stress points are in existing system

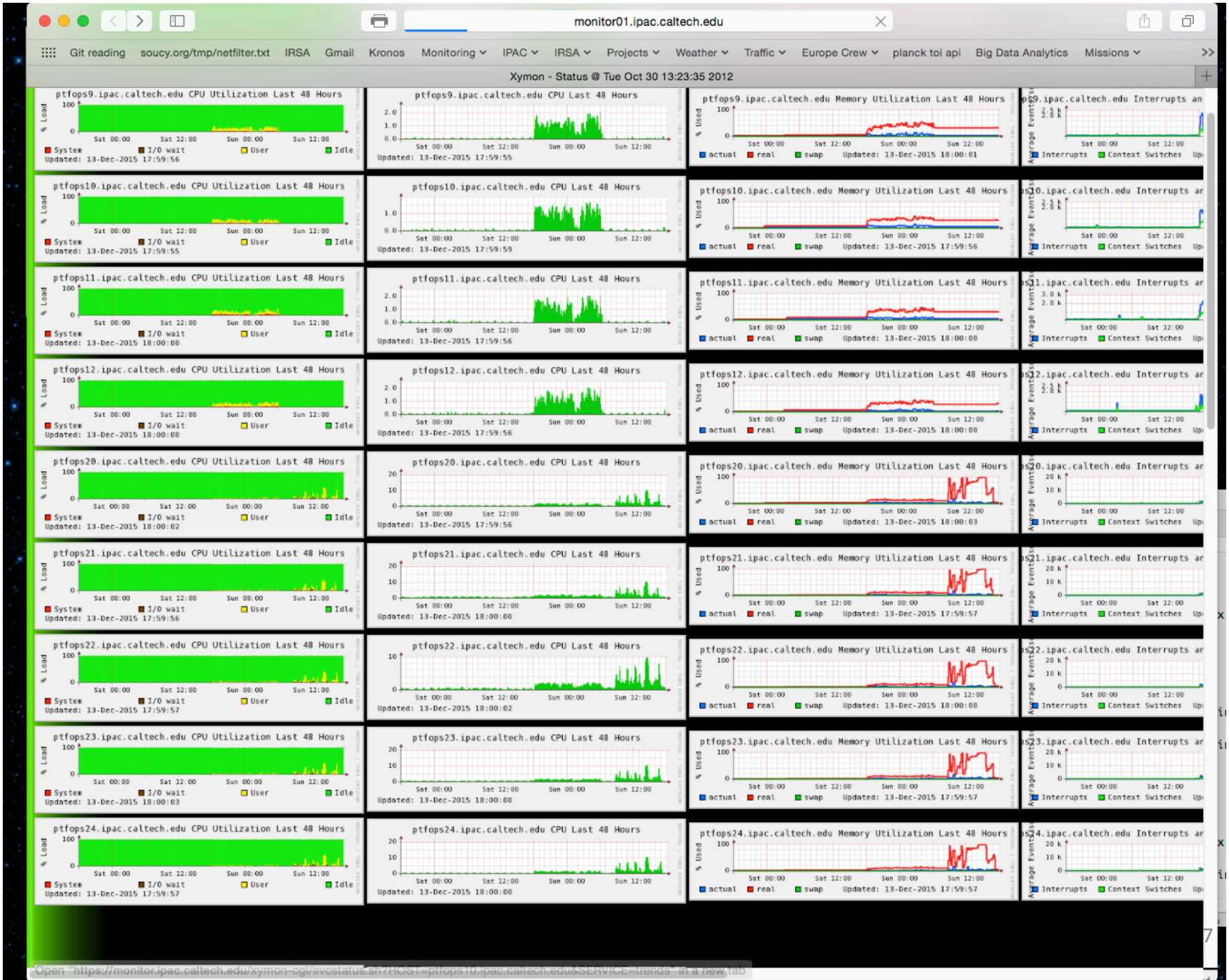
- **Result: a hybrid approach**
  - Review subsystems, assess scalability and efficiency
  - Extrapolate from PTF with some adjustments
  - Prototype and check progress against expectations

Data Flow Overview - PTF

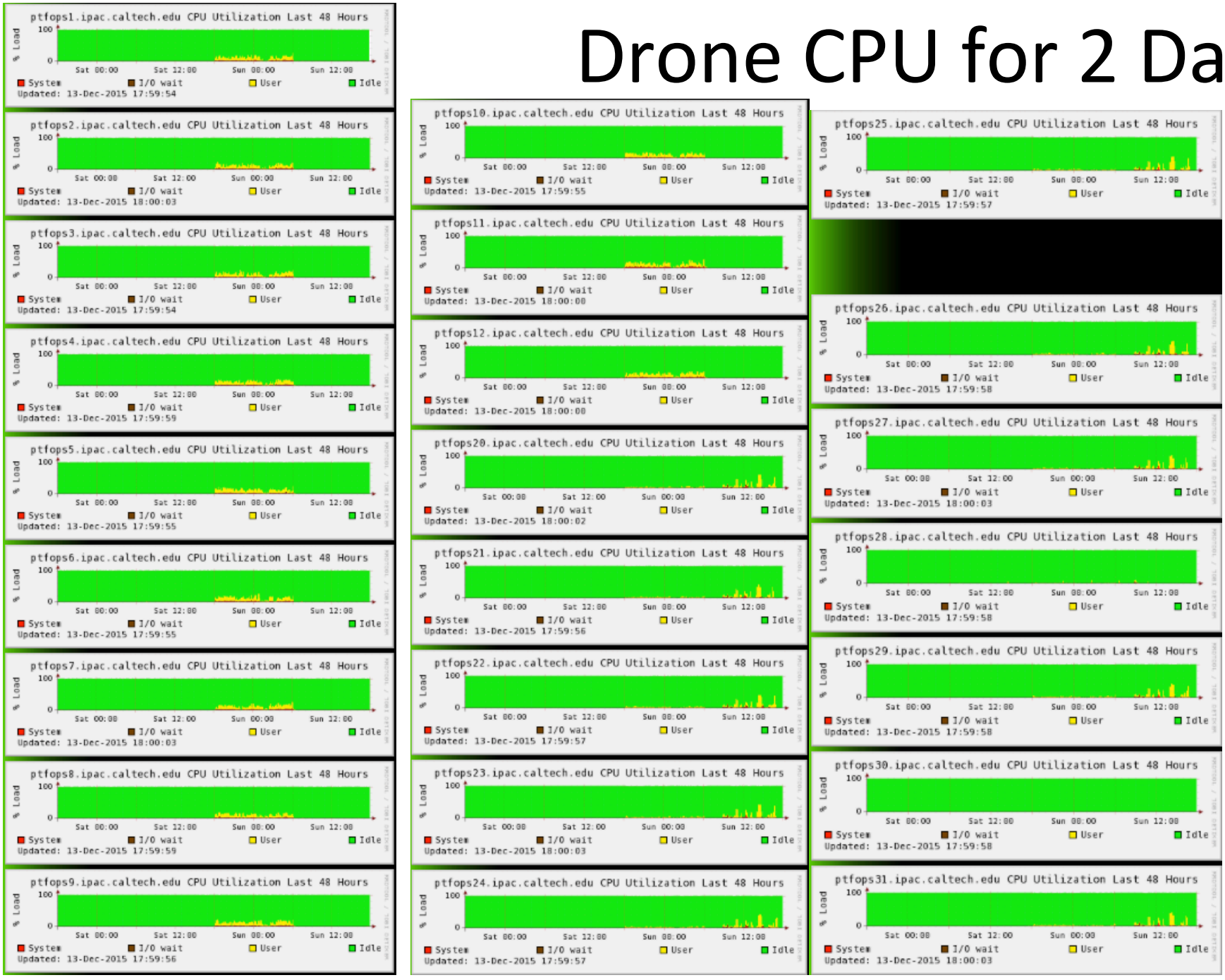


Analyzing the current system

- Instrumenting the current system to review
  - Utilization: how hard is it working?
    - vs. capacity
    - vs. time
  - Activity: what is it doing and when?
  - Bottlenecks
    - What/where are current limitations
    - Impacts on scalability
  - Efficiency: how effectively are we utilizing what we already have?
- Identifying key areas requiring attention, and insights influencing overall design and operations
- Observations influence technology choices and guiding additional work to be done

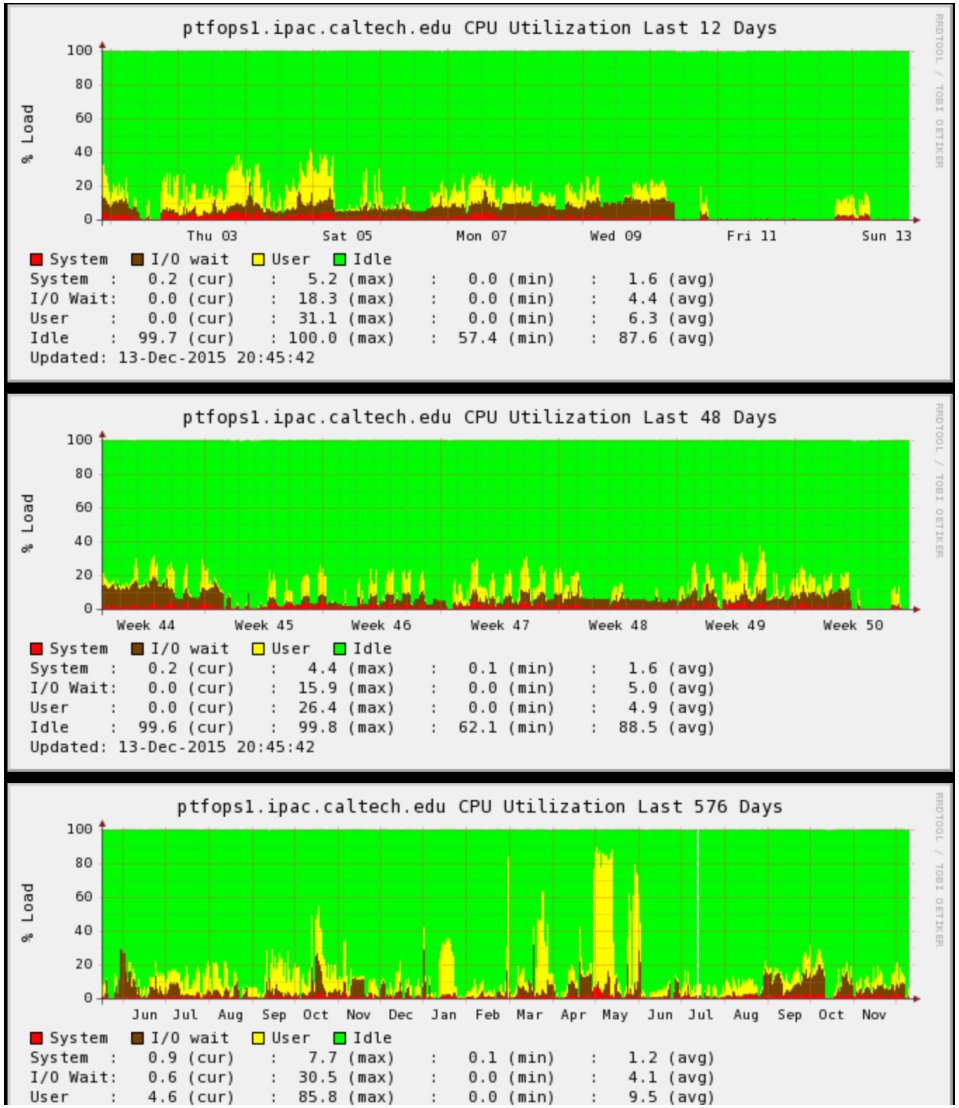

monitor01.ipac.caltech.edu
Git reading
soucy.org/tmp/netfilter.txt
IRSA
Gmail
Kronos
Monitoring
IPAC
IRSA
Projects
Weather
Traffic
Europe Crew
planck toi api
Big Data Analytics
Missions
Xymon - Status @ Tue Oct 30 13:23:35 2012
ptfops9.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops9.ipac.caltech.edu CPU Last 48 Hours
ptfops9.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops10.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops10.ipac.caltech.edu CPU Last 48 Hours
ptfops10.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops11.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops11.ipac.caltech.edu CPU Last 48 Hours
ptfops11.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops12.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops12.ipac.caltech.edu CPU Last 48 Hours
ptfops12.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops20.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops20.ipac.caltech.edu CPU Last 48 Hours
ptfops20.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops21.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops21.ipac.caltech.edu CPU Last 48 Hours
ptfops21.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops22.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops22.ipac.caltech.edu CPU Last 48 Hours
ptfops22.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops23.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops23.ipac.caltech.edu CPU Last 48 Hours
ptfops23.ipac.caltech.edu Memory Utilization Last 48 Hours
ptfops24.ipac.caltech.edu CPU Utilization Last 48 Hours
ptfops24.ipac.caltech.edu CPU Last 48 Hours
ptfops24.ipac.caltech.edu Memory Utilization Last 48 Hours
Open "https://monitor.ipac.caltech.edu/xymon-cgi/svcstatus.sh?HOST=ptfops10.ipac.caltech.edu&SERVICE=trends" in a new tab

Drone CPU for 2 Days

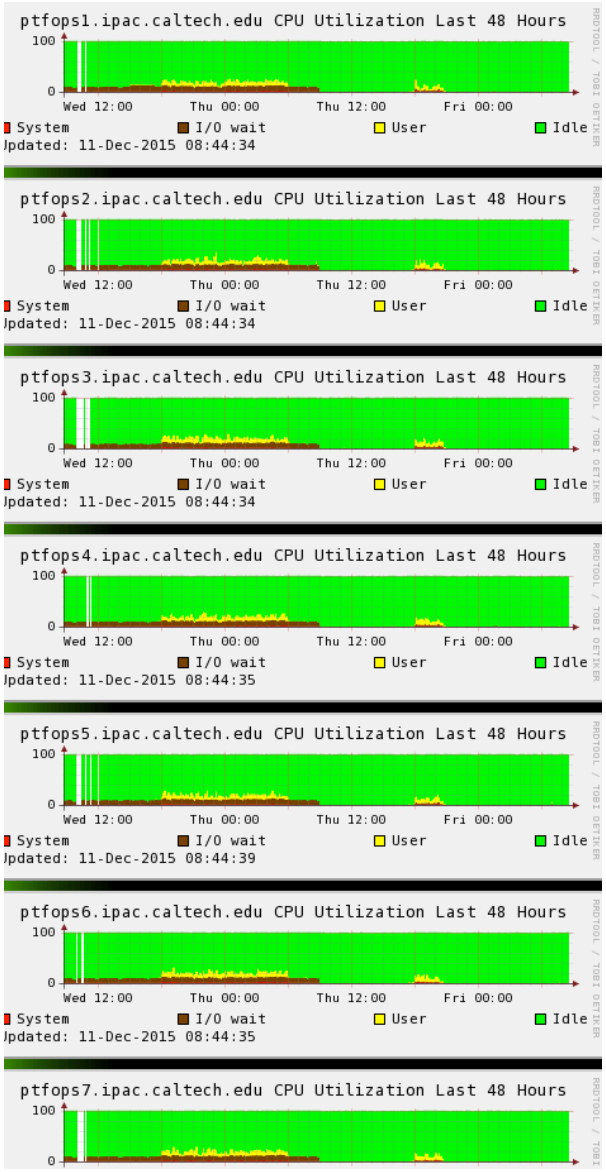


CPU history for one drone

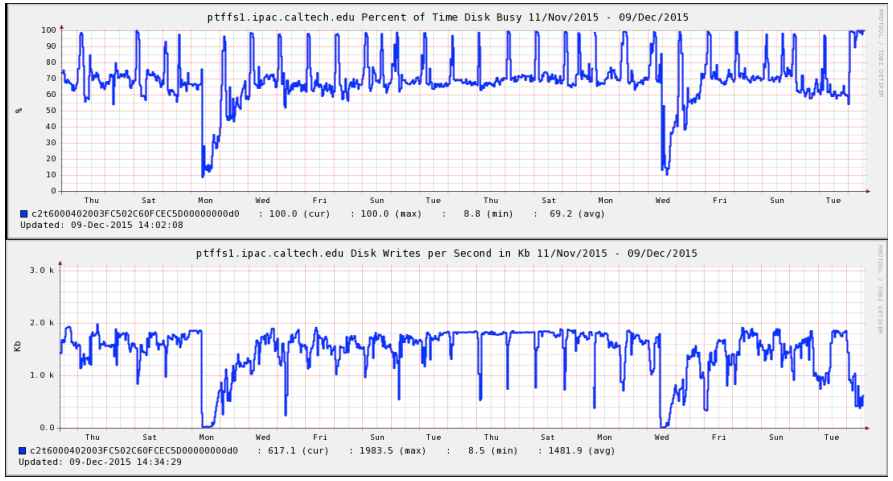
- Long term trends show high I/O overheads, low overall utilization
- Cause of overhead?
- Impact on scalability?



Case study: CPU I/O wait



- Why are drones showing so much time spent as “I/O wait”?
- What is causing certain some filesystems to be so heavily loaded?



%busy

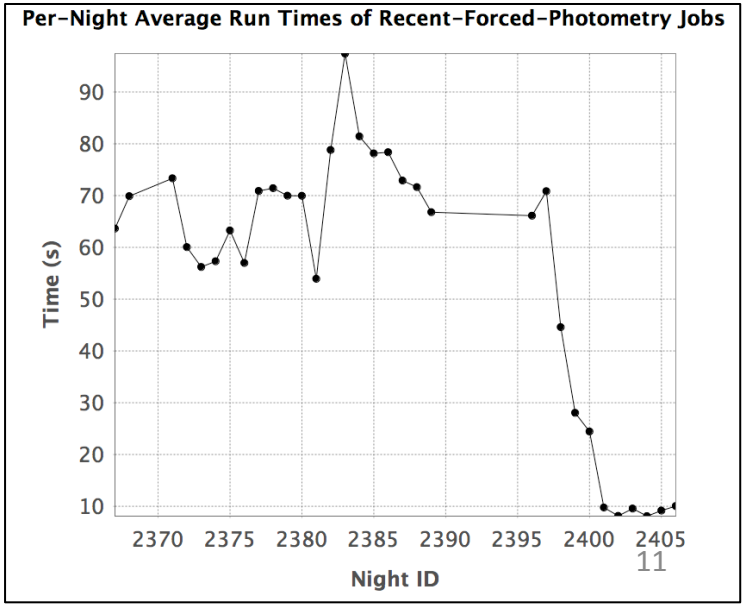
Write MB/sec

Case study: CPU I/O wait

- The find: excessive logging, combined with expensive NFS option, overloading server

- The result:
  - Python now loads 10x faster
  - RPF job runtimes
    - before: 70s
    - after: 10s

- Scalability predictions depend on understanding what the system is doing under the hood.



Observations

Resource utilization and operations practices

- Reduce I/O footprint of processing?
- localize I/O where possible
- explicit synchronization creating server spikes
- more parallelization and use of async processing to fill drone idle time, increase throughput
- software issues: logging, file locking
- background load due to server backups, RAID scrubs

High reliance on central services

- localize intermediate I/O
- single database used for everything, including job scheduling
- localize software installs

Technical performance bottlenecks

- underutilization of 10Gbit network
- excessive TCP retransmits
- server disk performance

Data Organization effects on I/O patterns

- segregate user areas from key data areas to isolate activity
- take data volatility into account, adjust caching and backup parameters
- mixing of permanent and temporary disk areas

Community Practices

- open access to key shared resources is not accounted for

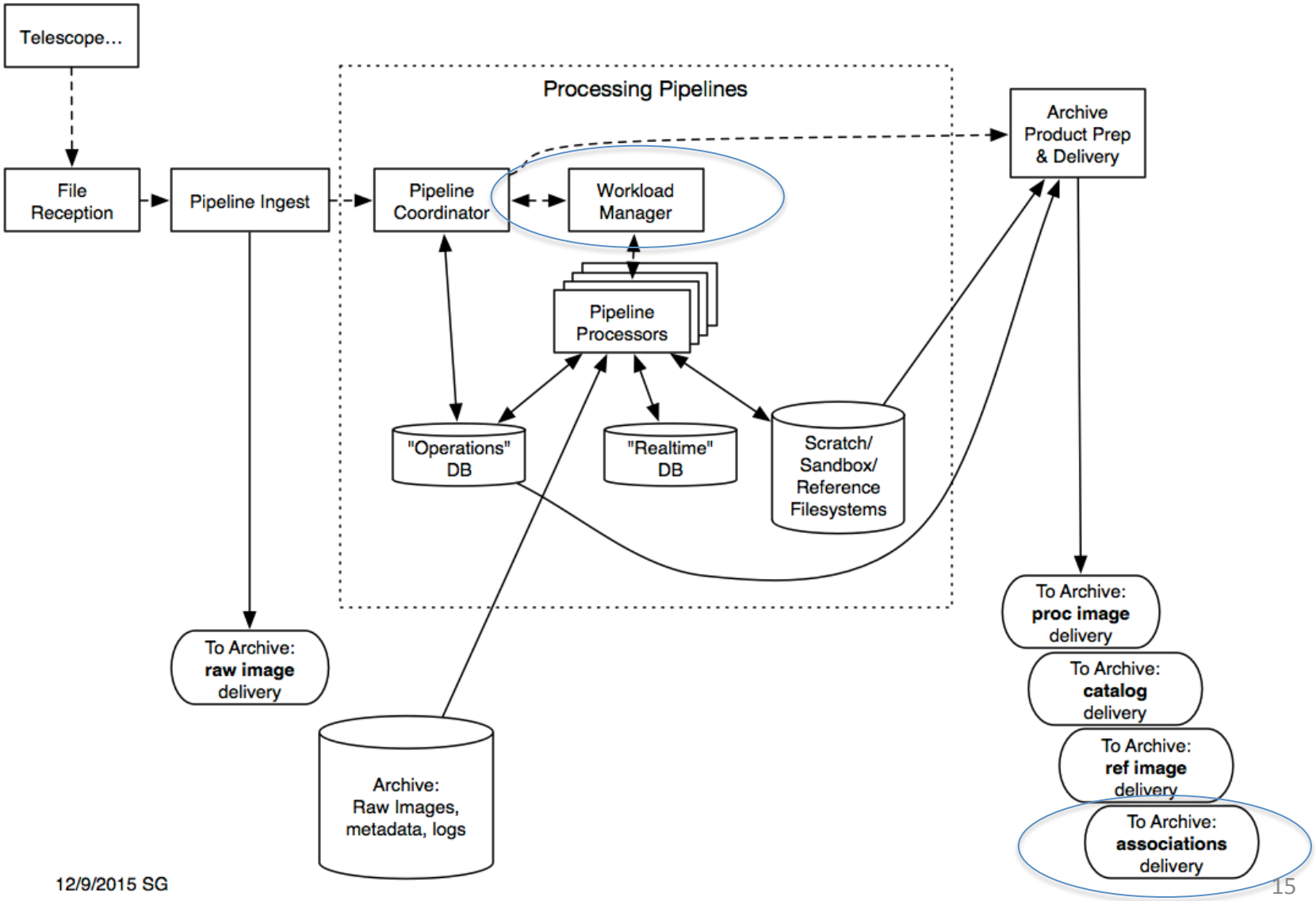
Significant changes for ZTF

- Dynamic job scheduling, use of open-source workload manger
- Careful organization of data layouts
- Tuning of software, and changes to reduce unnecessary data movement
- Basic scale-out approach for fileservers
- Database still TBD

Dynamic workload management

- Static job scheduling limits scalability
- Homegrown scheduler hard to tune
- Shift to dynamic job scheduling can improve utilization, help level resource spikes
- Open source solutions provide practical alternative
- Currently prototyping using SLURM
- More detailed illustration later

Data Flow Overview - ZTF



Subsystem status summaries

Subsystem: Transfer/Ingest

- Existing server needs simple upgrade
- Review & confirm network data rates
- Add 2nd network interface to separate internal & external traffic
- Should be OK

Subsystem: Raw Archive

- Raw data is ingested directly to archive disks
- No separate copy kept for pipeline access
- Pipeline processes read data from archive as needed
- Add interfaces based on drone scaling

Subsystem: Pipeline Coordinator

- Tracks frame completion status
- Submits processing tasks via Workload Manager

- Workload manager functionality to be split out
- Prototyping in progress

Subsystem: Workload Manager

- Manages work queue, dispatching jobs to worker nodes
- Implements priorities, enforces inter-job dependencies
- Implements tuning parameters e.g. number of running jobs per node

- Improved CPU: dynamic scheduling for better processor utilization, dynamic priority adjustment, flexible scheduling
- Improved I/O: Better load leveling helps reduce server spikes

- Prototyping in progress – discussion later

Subsystem: Processing nodes ("drones")

- Initial purchase being used for testing and prototype
- Developing prototype to test node scalability using dynamic scheduler
- Testing installation and management software

Subsystem: Databases

- Need to better understand load patterns here
- Updated server and storage hardware
- Same PostgreSQL software
- Probably need to partition data somehow across multiple servers
  - spatial? (sharding by sky region)
  - By data type? E.g. “operations” vs “realtime” db’s
- May need to restrict user access, or implement replication to support live users

Subsystem: Fileservers

- Solaris servers stable, but 10gbit enet performance has been disappointing
- Closely examining server activity
- Examining data organization, remapping content based on activity
- Should be easy to scale by spreading data across multiple servers
- Prototyping ZFS-on-Linux solution

Subsystem: networking

- Look at dividing drones into multiple networks, or other technique to utilize multiple NIC's at server
- Instrumentation of server network usage for further analysis

Archiving of processed products

- Keeping existing model
- Need to work out handoff of associations data

Access to data and processing resources by Cahill users/marshalls

- **Topic for discussion**

Summary

- Proceeding with examination of subsystems
- Applying tuning to PTF as issues identified
- Prototyping workload manager and new drone configurations
- Database organization will be key in scaling/ sizing approach
- Improving flexibility to deal with unexpected usage patterns

New ZTF Hardware/SLURM

Ben Rusholme

ZTF Data Systems Review

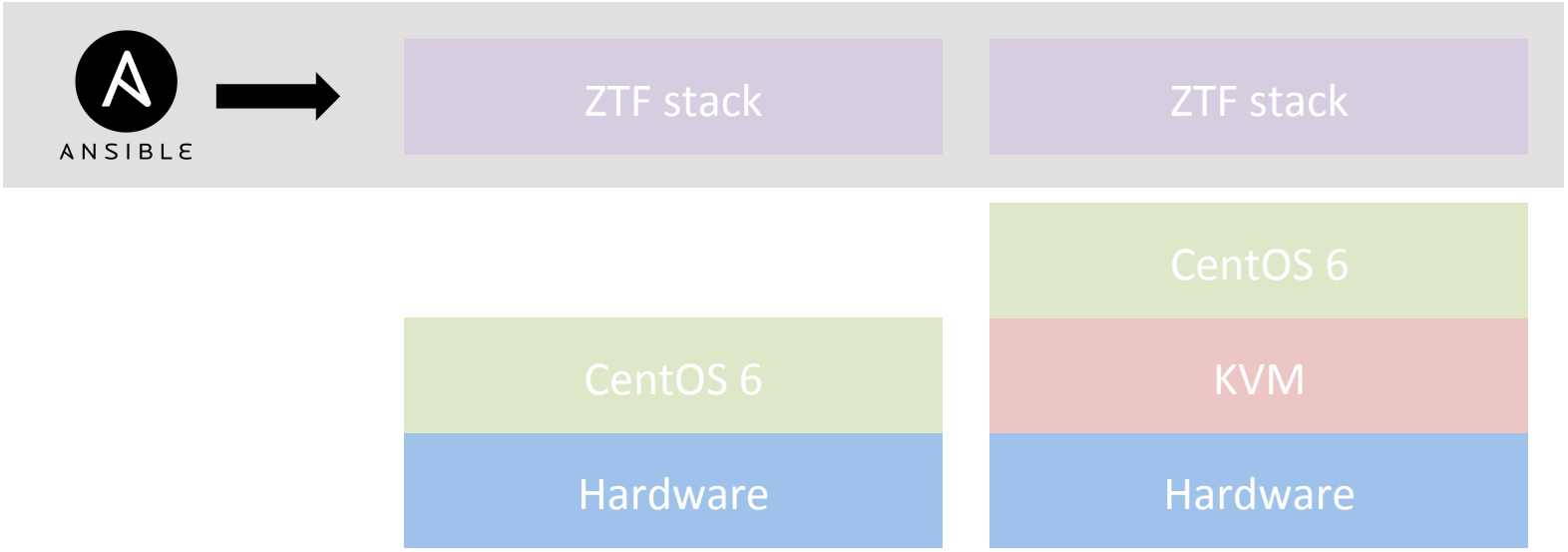
14th December 2015

Status

- 32 nodes delivered, racked, burnt-in
- Currently testing subset of 5
- CentOS 6
- Evaluating both Bare Metal & KVM
- Selected SLURM as job/resource manager
  - Throughput
  - cGroup isolation
  - Flexibility of srun intra-job
  - No MOAB scheduler license required

Project Boundary

- Provisioning/networking handled by ISG
- Project-specific configuration via Ansible
- Developed using Vagrant & Github

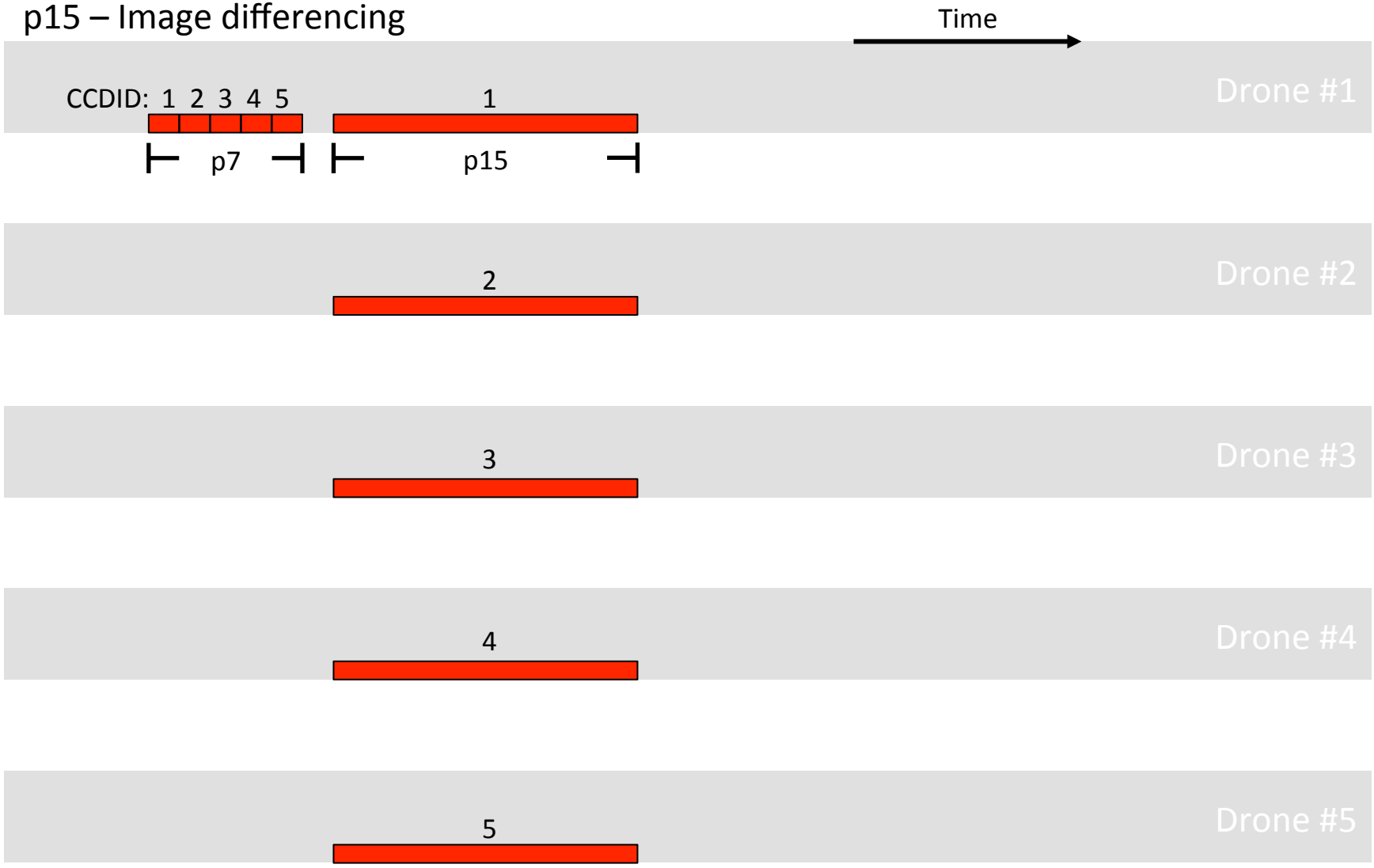


Using SLURM

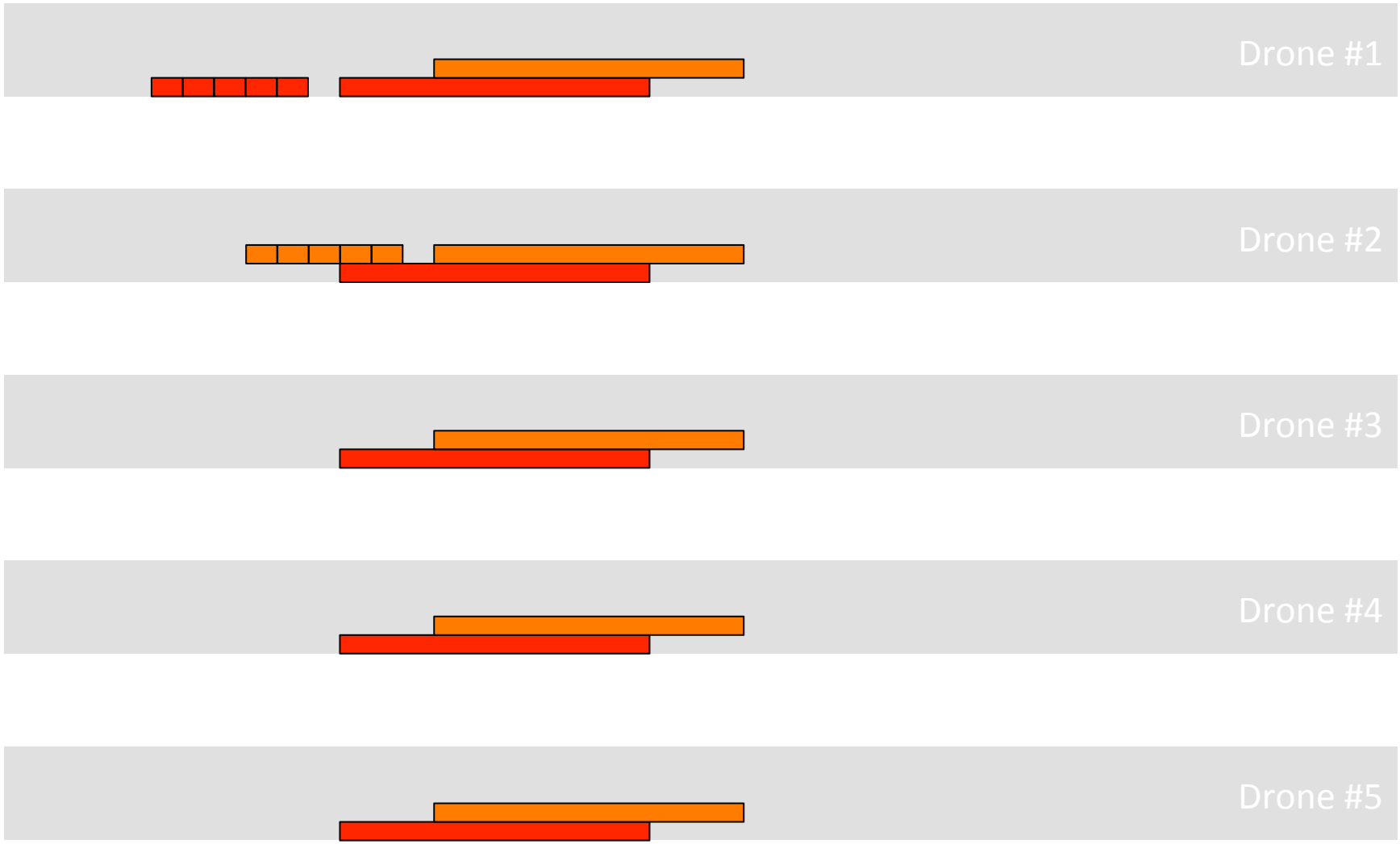
- Breaking PTF mapping of CCDID to drone#
- Advantages:
  - Location agnostic code
  - Hand-off scheduling, dependencies, node-cleanup
  - Bypass node failures
  - Flexibility in allocating resources (to achieve latency)
  - Reorganize to:
    - Maximize data locality
    - Minimize NFS transfers

PTF Nightly Pipeline

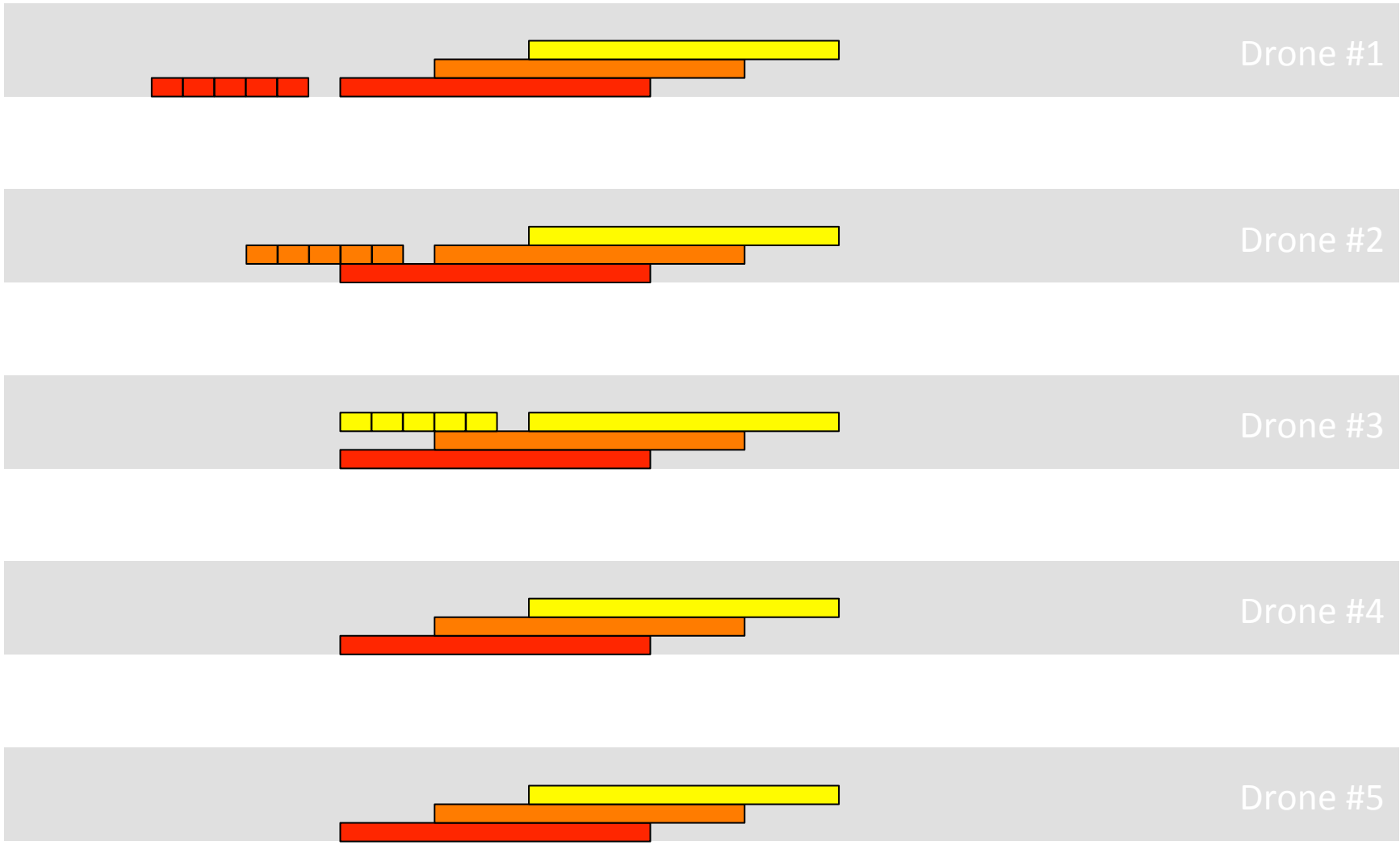
p7 – Ingest (split into CCDS, astrometric correction)
p15 – Image differencing



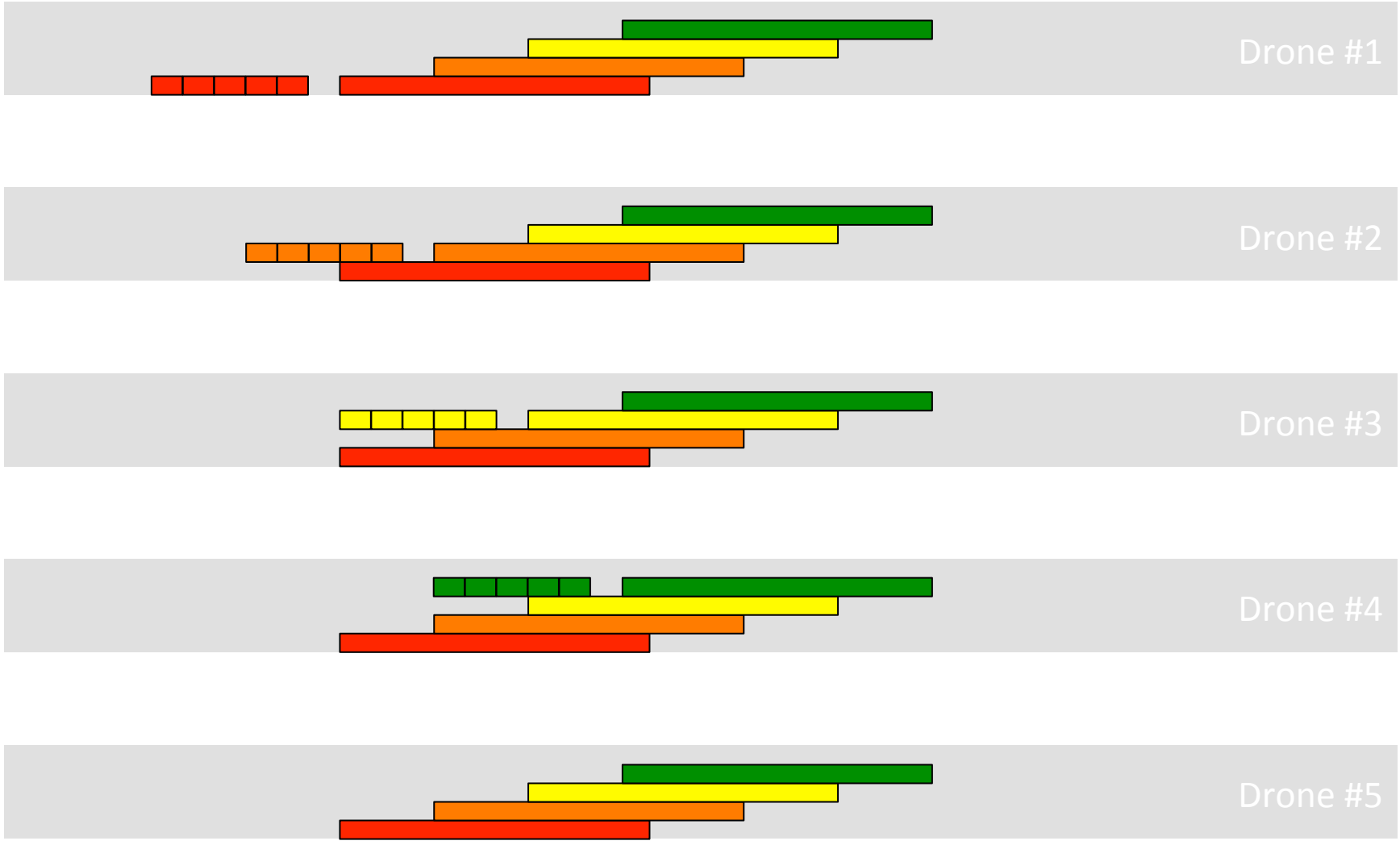
PTF Nightly Pipeline



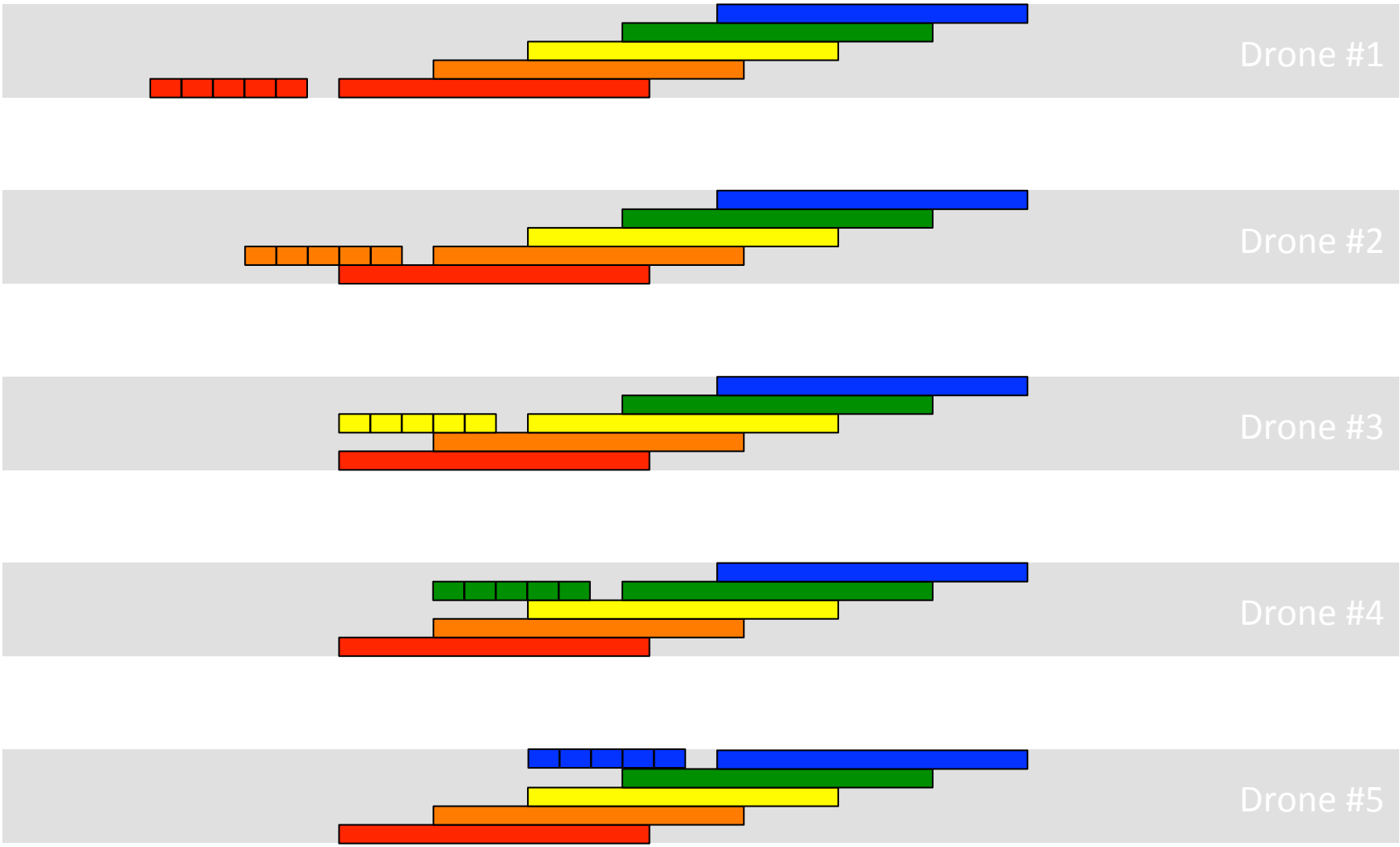
PTF Nightly Pipeline



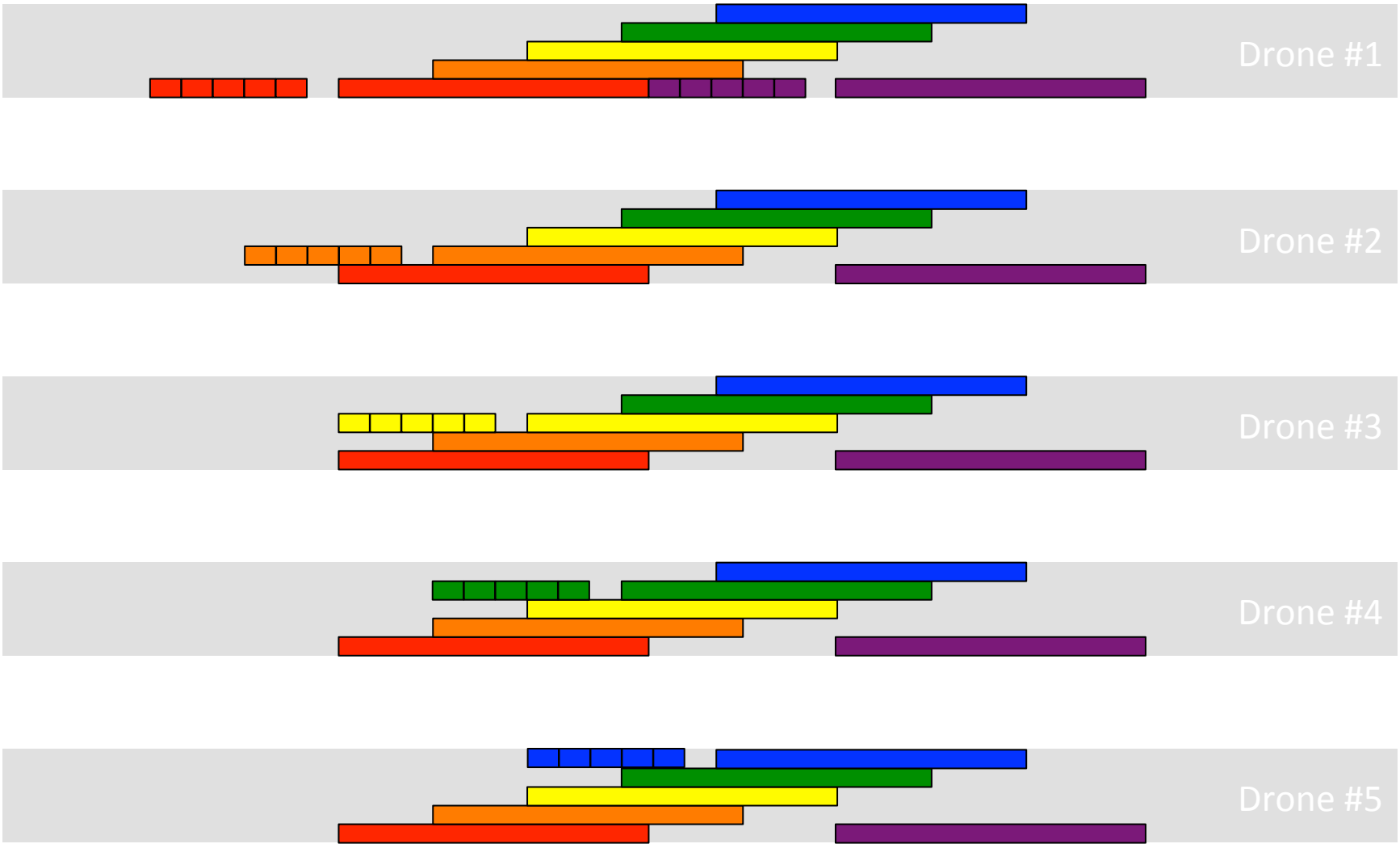
PTF Nightly Pipeline



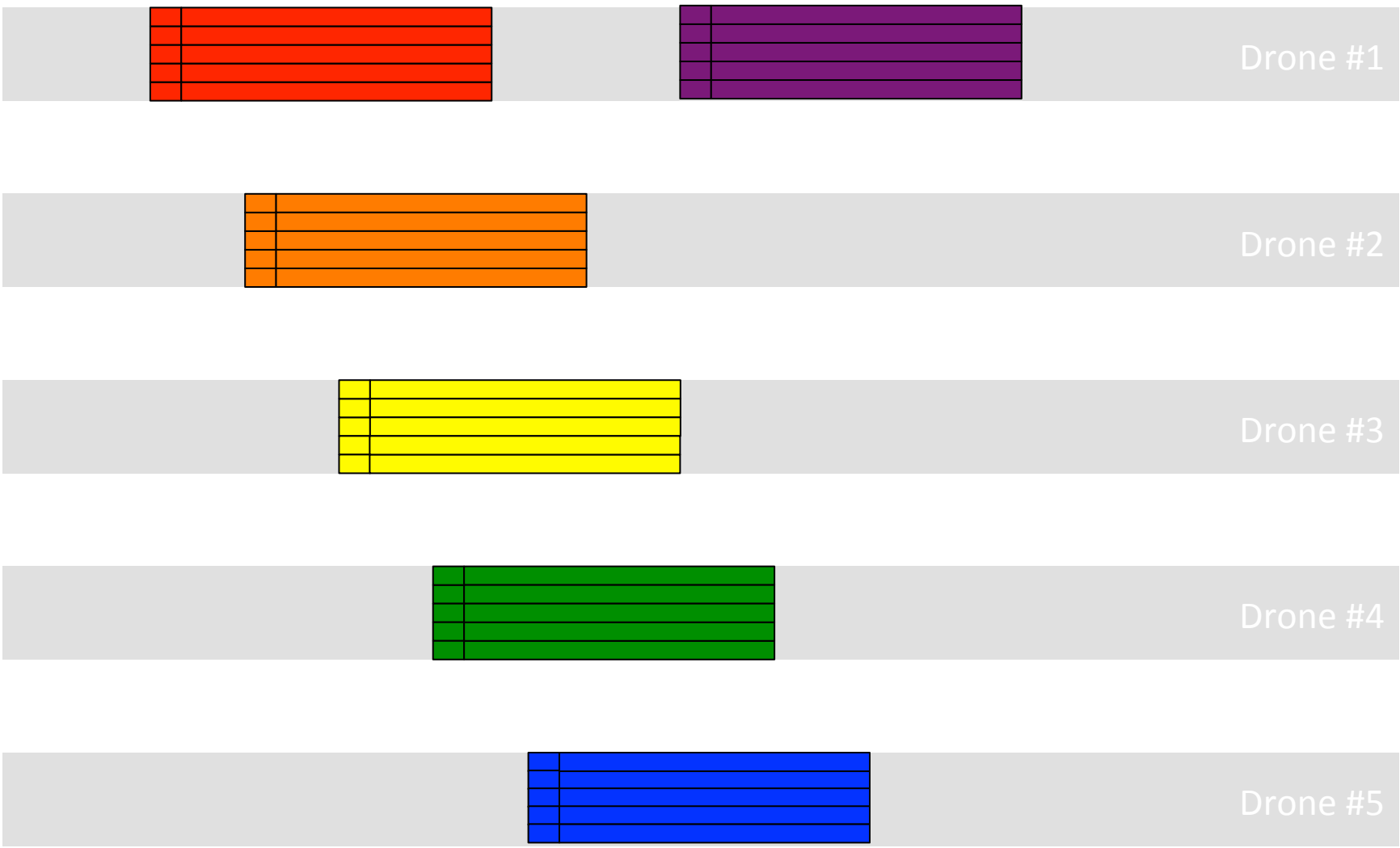
PTF Nightly Pipeline



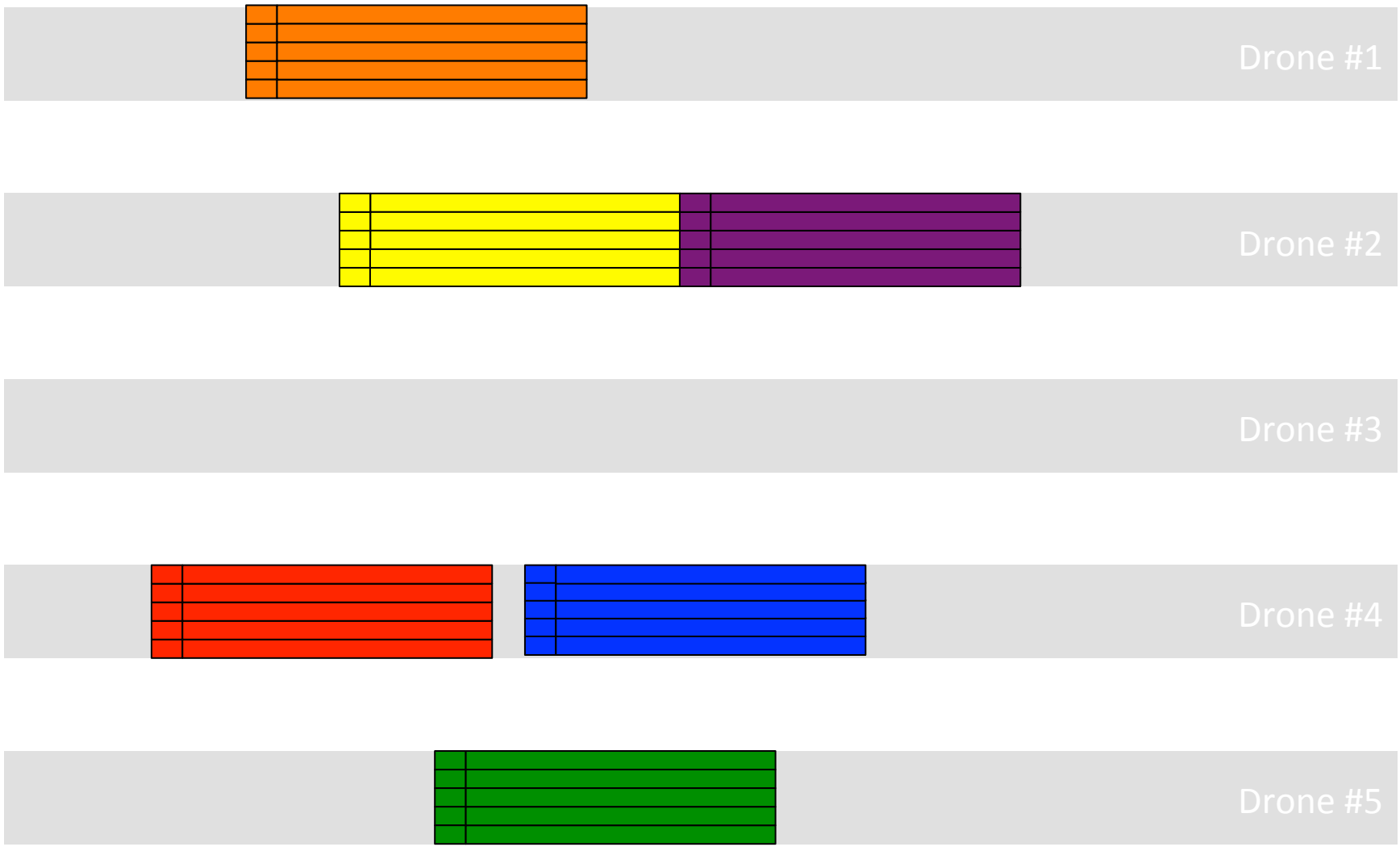
PTF Nightly Pipeline



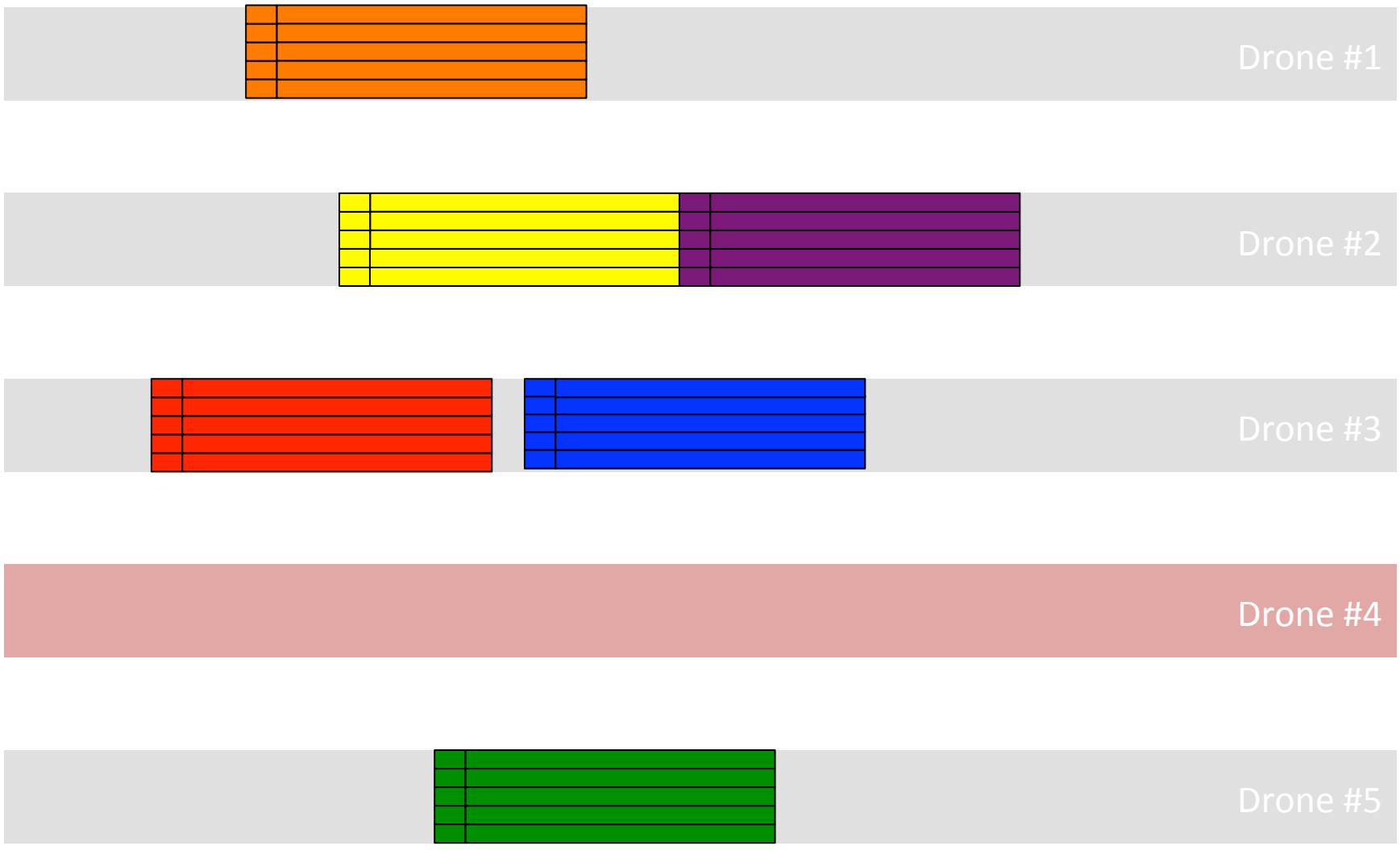
ZTF Proposed Layout



ZTF Proposed Layout



ZTF Proposed Layout



Performance Testing

p7, on PTF 2nd level hardware (8 cores), single test night

- October: 10 CCDs serial: 494s
  1 CCD: 47s
  8 CCDs parallel: 50s

- December: 10 CCDs serial: 154s
  8 CCDs parallel: 15.2s

Deployment Schedule Deliverables

Jason Surace

IPAC/Caltech

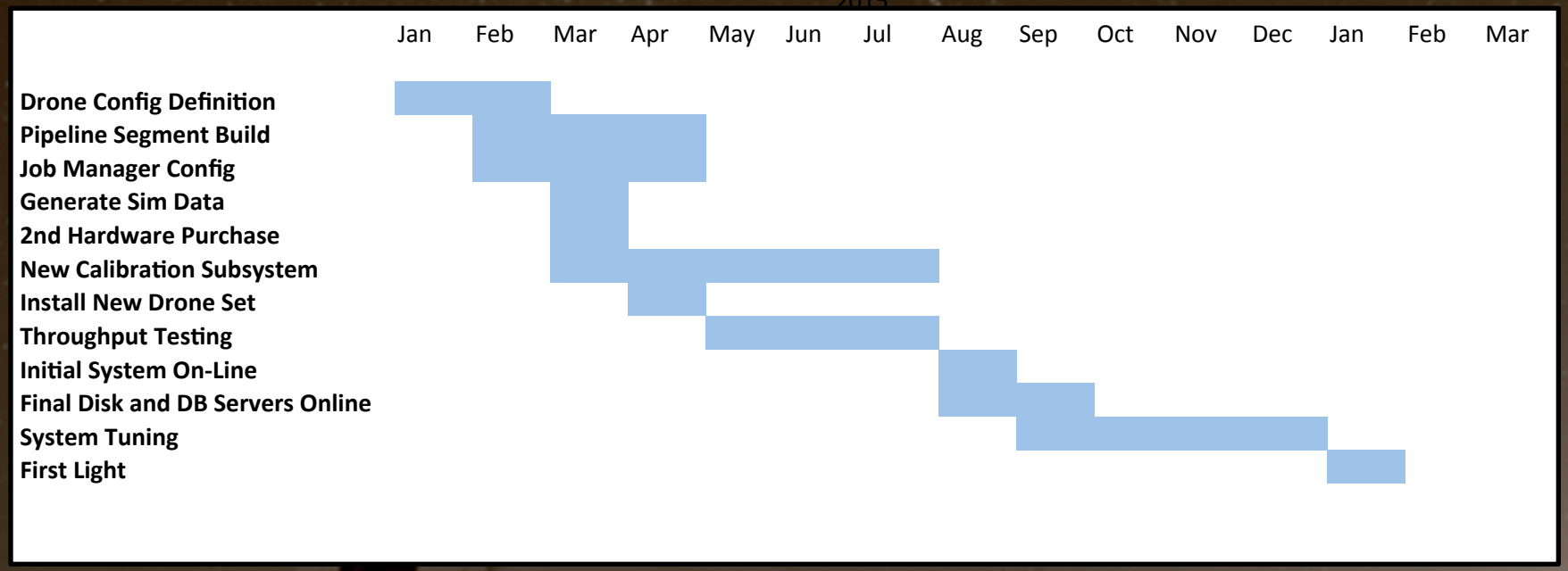
NSF-MSIP Deliverables

Year 1 (8/15) – first public data release of PTF image and catalog file data (epochal and coadded) over entire sky.
Year 2 (8/16) – release of light curve interface, as well as rolling release of iPTF image and catalog data.
Year 3 (first light + 1 yr) – release of ZTF references, catalogs, and light curves.
Year 4 – release of transient alerts and switch to semester-wise release of all other products.
Year 5 – same as year 4, but include near-realtime release of transient candidates.

Phased Hardware/Software Deployment

- Partially decoupling the software and hardware deployment to allow them to be developed in parallel.
- Initial purchase included 32 drones as well as network switches.
- Drones are currently online as a mixture of VMs and bare metal units for testing purposes.
- A temporary file and database server have been deployed to enable pipeline functionality.
- Currently testing the software deployment/build/ maintenance system, as well as the job management system.
- Under “smarter architecture” considering tradeoffs between drones and server capabilities. Will conclude this exercise in time for end of Q1 CY2016 purchases.

Deployment Schedule Overview



Deployment Schedule
Key Dates

- Hardware Procurement
  - 2/17 final decisions on content
  - 3/01 place orders
  - 3/21 – 4/22 installation and set-up

- Throughput testing – 3 phases across the summer
  - 6/6, 7/19, 8/23 (TBD)

- Full system available: 9/8
  - We do not plan to slip this schedule if first light is delayed