Workflows and Data Management

Adam Brazier – *brazier@cornell.edu*
Computational Scientist
Cornell University Center for Advanced Computing (CAC)

Overview: Summary and Scope

- Workflows
  - Automation, our friend and foe
  - How should we automate a workflow?

- Data management
  - From cradle to grave: the lifecycle of data
  - How should we make a plan?

- Scope
  - The (our) university research environment
  - Process and technology
    - Not providing specific software recommendations

Workflows: what is a workflow?

- “Workflow” may mean different things to different people. Avoiding dogma, we can consider “workflow” as:

Workflows: what is a workflow?

- "Workflow" may mean different things to different people. Avoiding dogma, we can consider "workflow" as:
  - A) What it says on the tin

Workflows: what is a workflow

- "Workflow" may mean different things to different people. Avoiding dogma, we can consider "workflow" as:
  - A) What it says on the tin
  - B) A process which can be illustrated with a flow diagram

Workflows: what is a workflow?

- "Workflow" may mean different things to different people. Avoiding dogma, we can consider "workflow" as:
  - A) What it says on the tin
  - B) A process which can be illustrated with a flow diagram
  - C) "A series of tasks that produce an outcome" (Microsoft)

Workflows: what is a workflow?

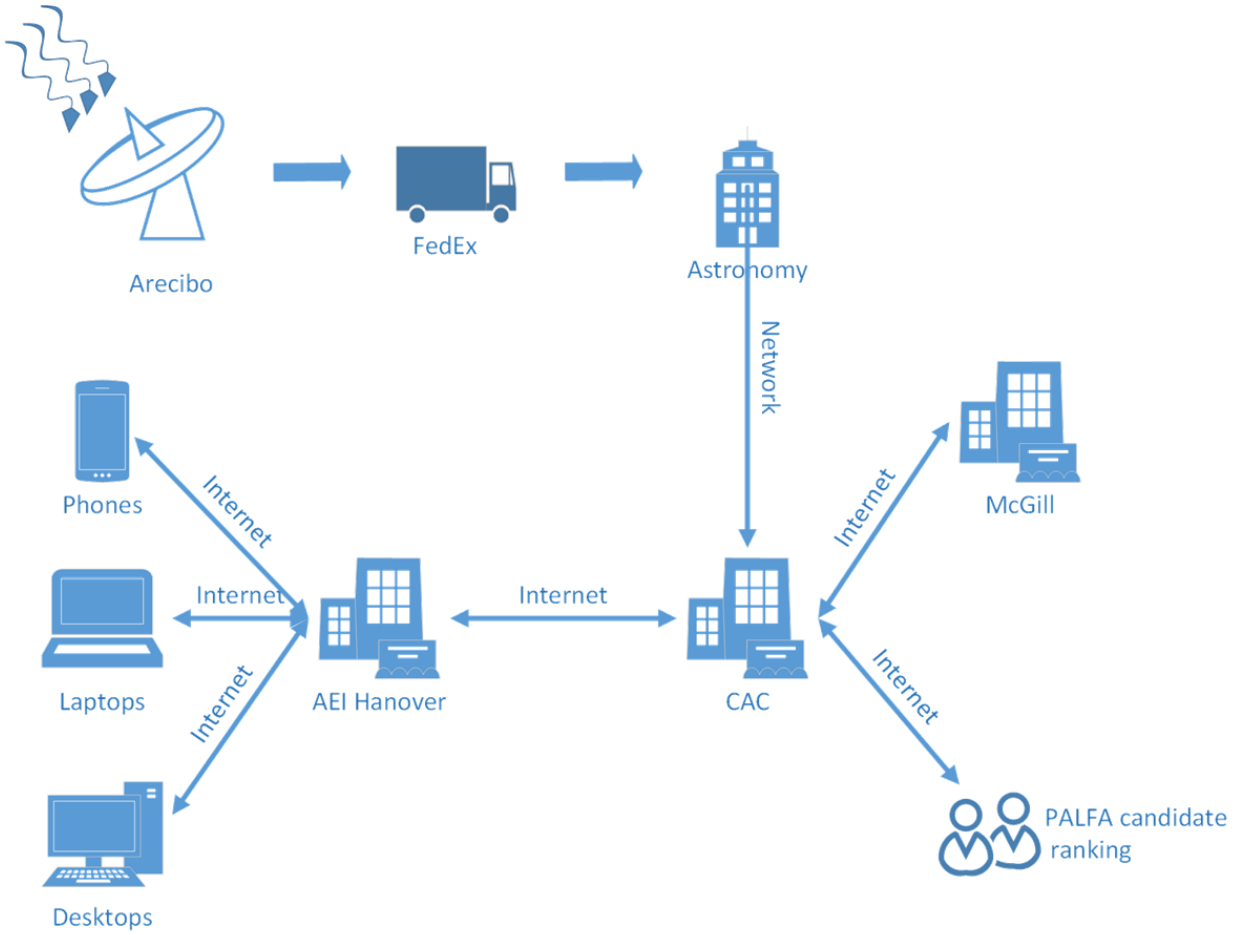
- "Workflow" may mean different things to different people. Avoiding dogma, we can consider "workflow" as:
  - A) What it says on the tin
  - B) A process which can be illustrated with a flow diagram
  - C) "A series of tasks that produce an outcome" (Microsoft)
  - D) "A workflow consists of an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information" (Wikipedia)

Workflows: what is a workflow?

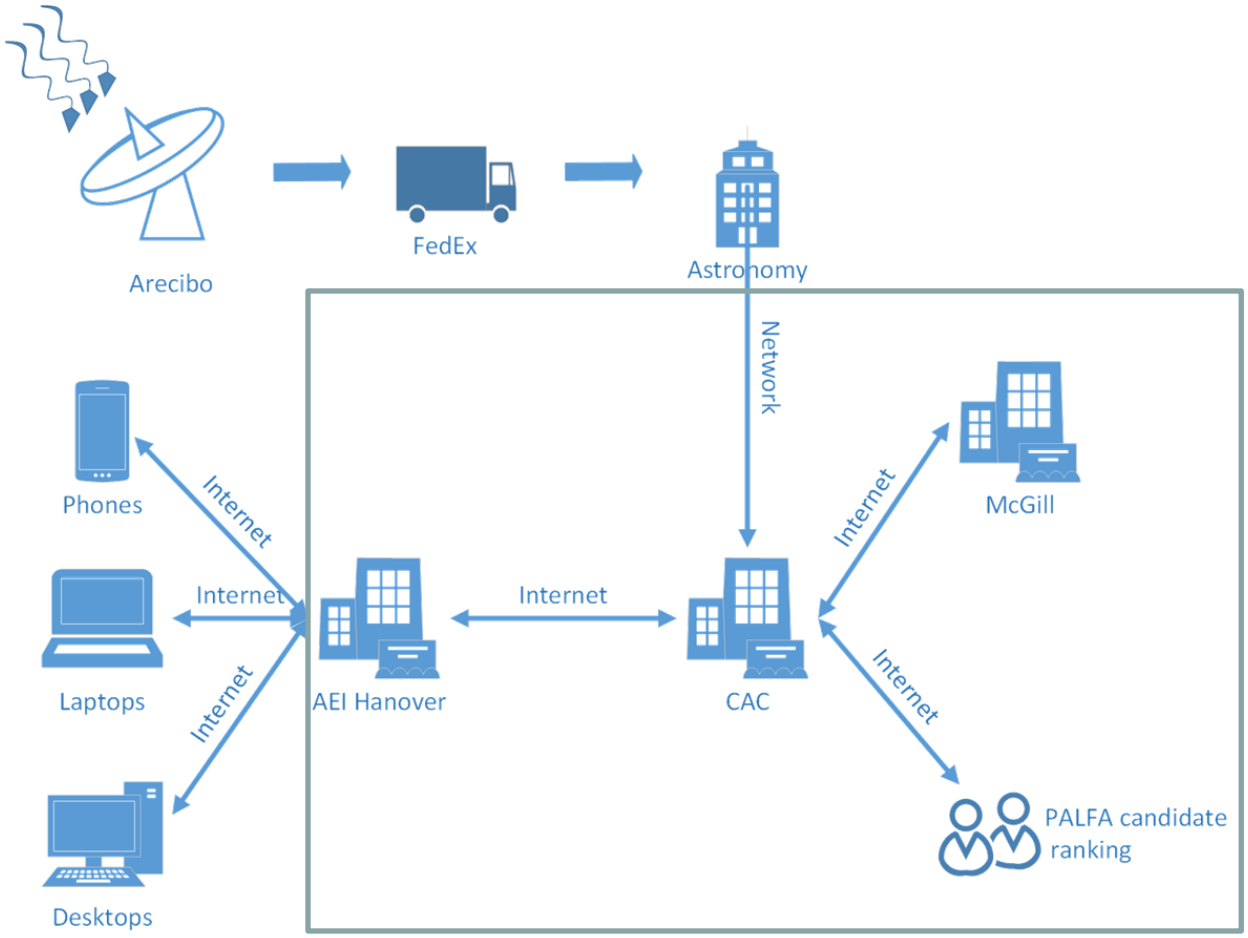
- "Workflow" may mean different things to different people. Avoiding dogma, we can consider "workflow" as:
  - A) What it says on the tin
  - B) **A process which can be illustrated with a flow diagram**
  - C) "**A series of tasks that produce an outcome** " (Microsoft)
  - D) "A workflow consists of an **orchestrated and repeatable** pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information" (Wikipedia)

Workflows: What do *our* workflows look like?

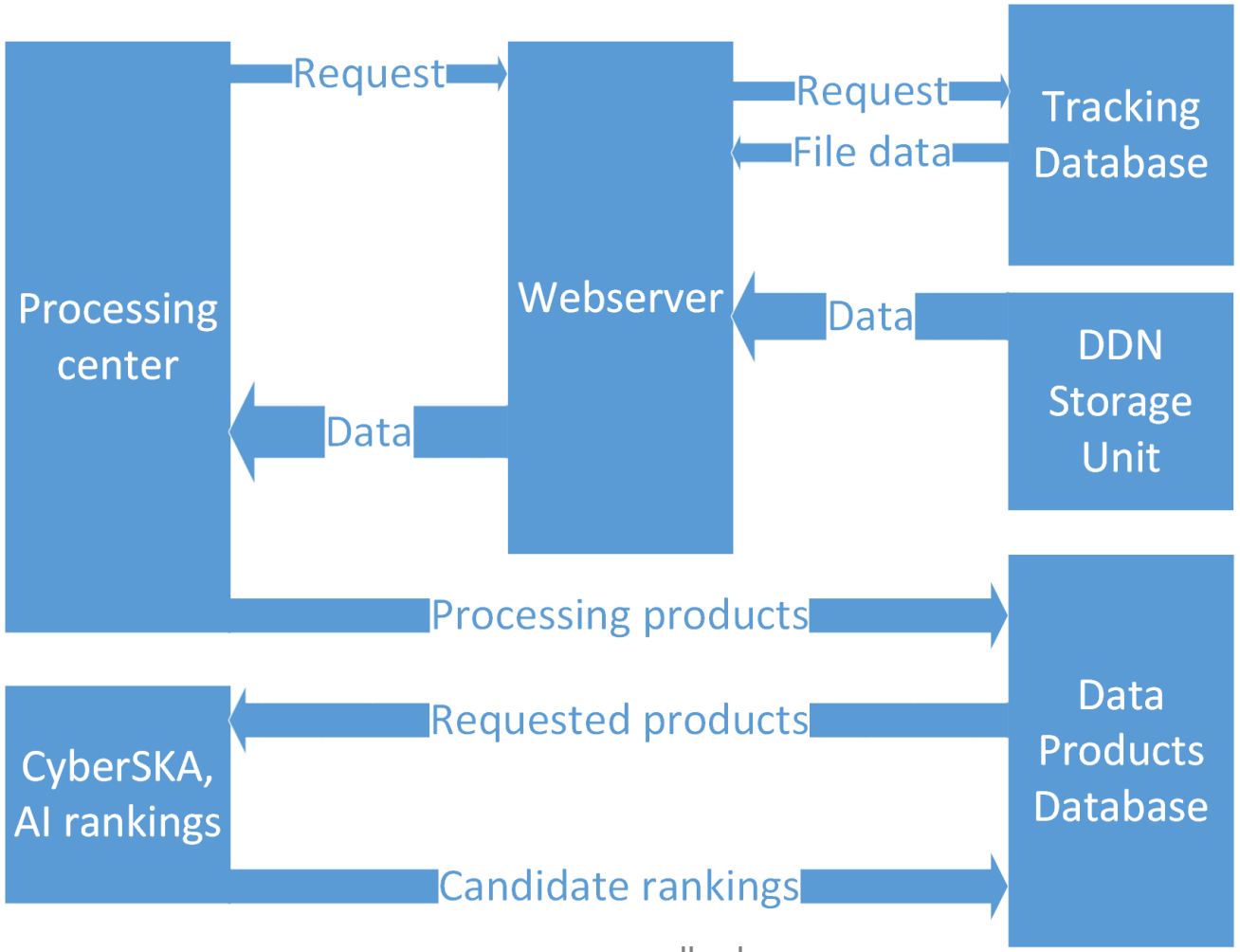
Workflows: What do *our* workflows look like?



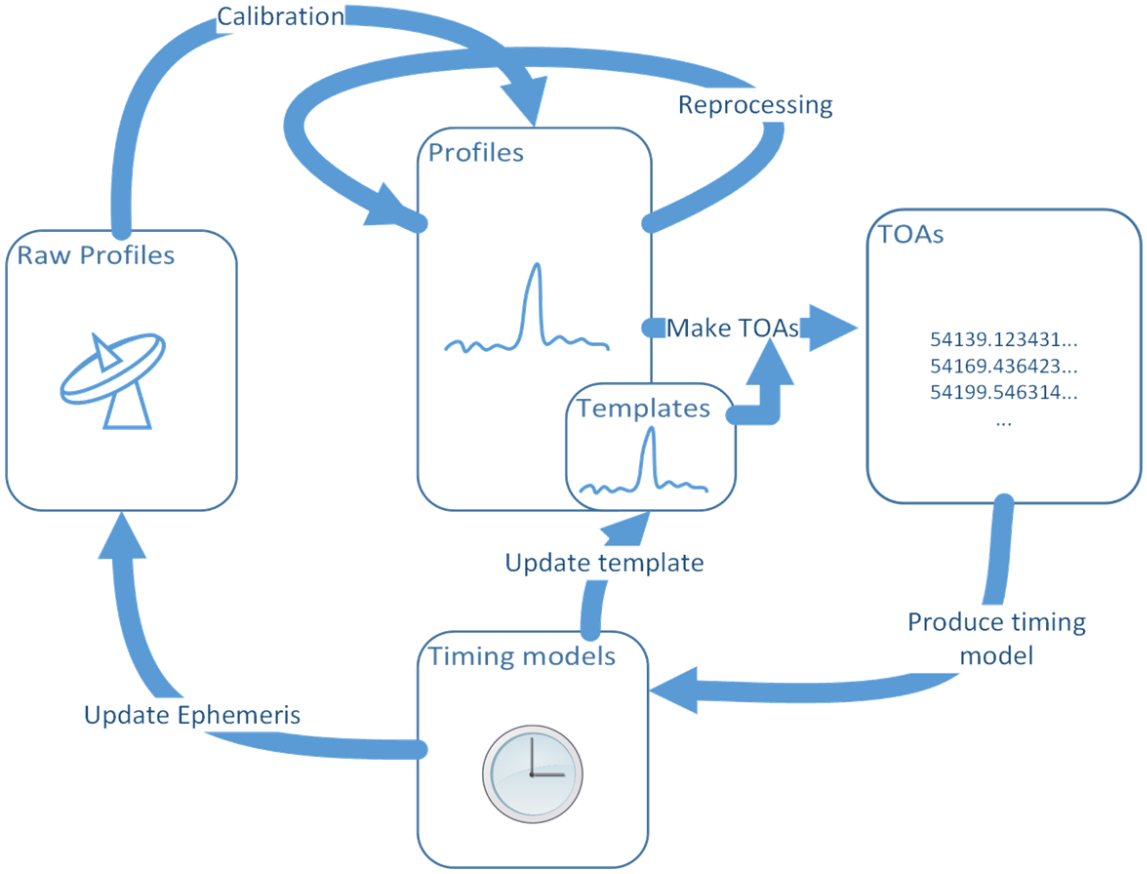
Workflows: or some part thereof



Workflows: follow the data!



Workflows: model the processes!



Workflows: why automate?

- Cheaper, in the long run
- Speed
- Reliability, Robustness
- Repeatability!
- Faster, better, stronger!

One day, you won't even have to walk down the stairs yourself

Workflows: what *can* you lose if you automate?

- Hands-on involvement, the sense of what's going on
- Grad student training ground
- Development time/cost
- Team awareness of core process

Anticipate the problems

Workflows: the hard part

- Human intervention most important when things are out of the ordinary, which includes failures
- A requirement of “no failures” is unrealistic in most cases
- Fault tolerant workflows have mechanisms for surviving failure:
  - Notifications to humans responsible for workflow
    - Tracking system records errors and their mitigation
  - Checkpointing
    - In case of interruptions of key services
  - Error-handling
    - May set aside problematic tasks for later

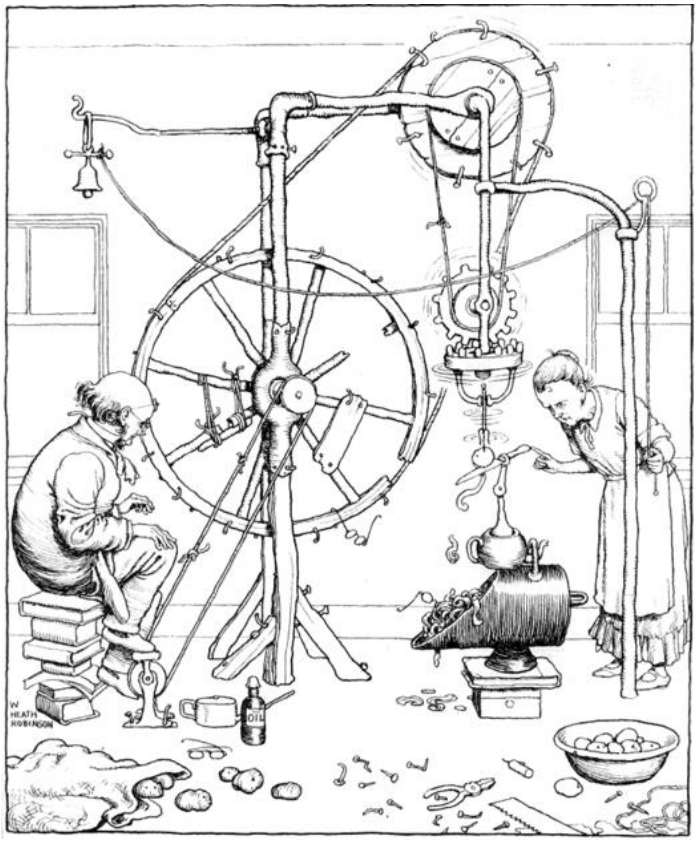
Workflows: what do we need?

- Clear requirements.
  - Avoids a solution in search of a problem
- High-level, modular/loosely-coupled design
  - Necessary to assign and estimate effort
  - Diagram it!
- Budget
  - This may really be a time estimate

Not the best design approach

Workflows: functional elements

- Data taking
- Data transfer
- Data processing
- Analysis of results
- All mediated by software!



There's more than one way to peel a potato

Workflows: available technologies

- Bulk storage at a variety of performance levels
  - Robustness also an issue to consider
- Databases and other organized storage
  - Allow sophisticated interrogation
- Networks
  - Not an issue until it's an issue
    - If it's an issue, it's often a huge issue
- Software. Sometimes lots of software
  - Bespoke or third-party
    - Most likely a mix of both

Everything will eventually seem old and outdated. Be prepared to change it all

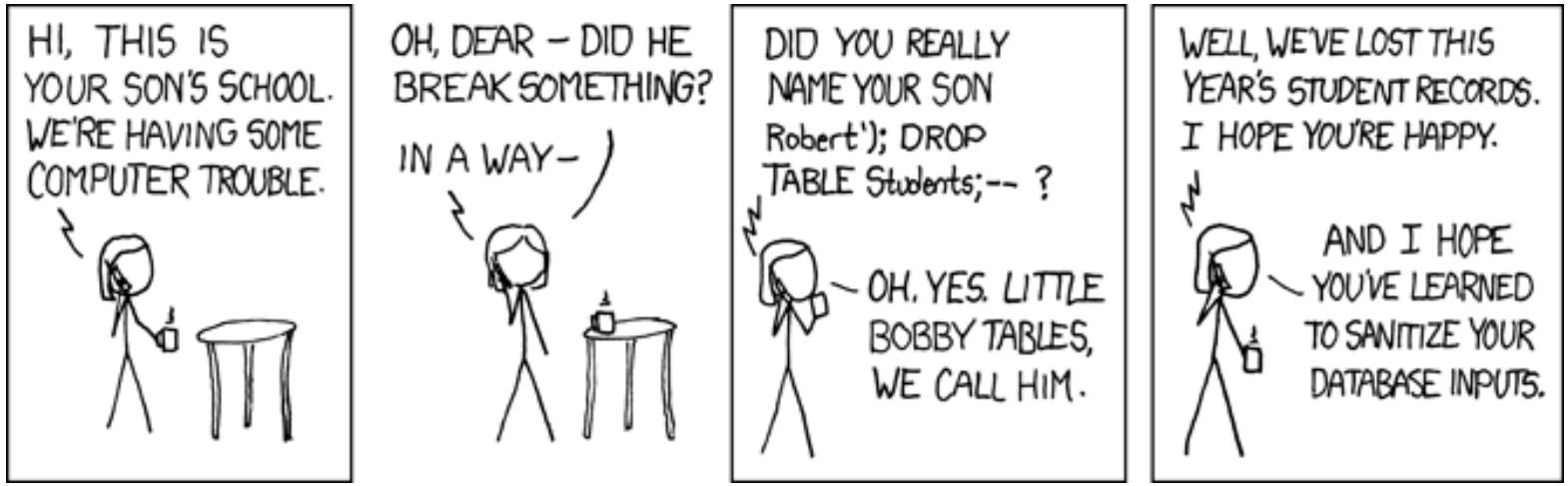
Workflows: storage

- A key HPC issue is read/write speed.
  - Basic estimate of simple – 7200 rpm – disk I/O is about 60-70 MB/s
  - Faster than this is achievable but costs more:
    - Solid State Drives are substantially faster, several 100s of MB/s
    - Many multi-disk enclosures achieve much better performance
  - Even with fast disk IO, you can't beat the interconnect/network
    - Stampede and similar HPC clusters have amazingly fast I/O

- Robustness
  - Generally achieved with redundancy in arrays of disks (RAID 6 rec.)
  - Disks for use in arrays are "NAS" or "enterprise" class
    - More expensive than commodity disks
  - Establish requirements. Monitoring can be a workflow element

Workflows: databases, etc

- Databases can be queried efficiently, often in Structured Query Language, SQL
  - Need to be properly-designed
    - Above a certain complexity, may want a DB professional to help
  - Allows remote access, but be careful!



http://xkcd.com/327/

Workflows: networks

- Moving data around is easier when “it just works”
  - See “Data Transfer” talk
  - External network traffic speed normally limited by network
  - Internal network traffic speed often limited by disk/array IO

- Network problems often best referred to professionals
  - Useful tools to eliminate software/OS as cause:
    - `ping`
    - Linux ‘`ip`’ command (cf. `ifconfig`)
    - `netstat`
    - `traceroute`
    - `tcpdump`
      - Use `-i <interface> -n -v -vv host <hostIP>`
    - `iptables –L –n`

Workflows: software

- Software of key interest in workflows
  - Control-of-flow (the glue that holds it all together)
    - Often a scripting language: perl, python, bash, etc.
  - Remote Procedure Call (RPC: allows commands to remote software)
    - "Web services" a common RPC platform
  - Database access software
  - Web applications

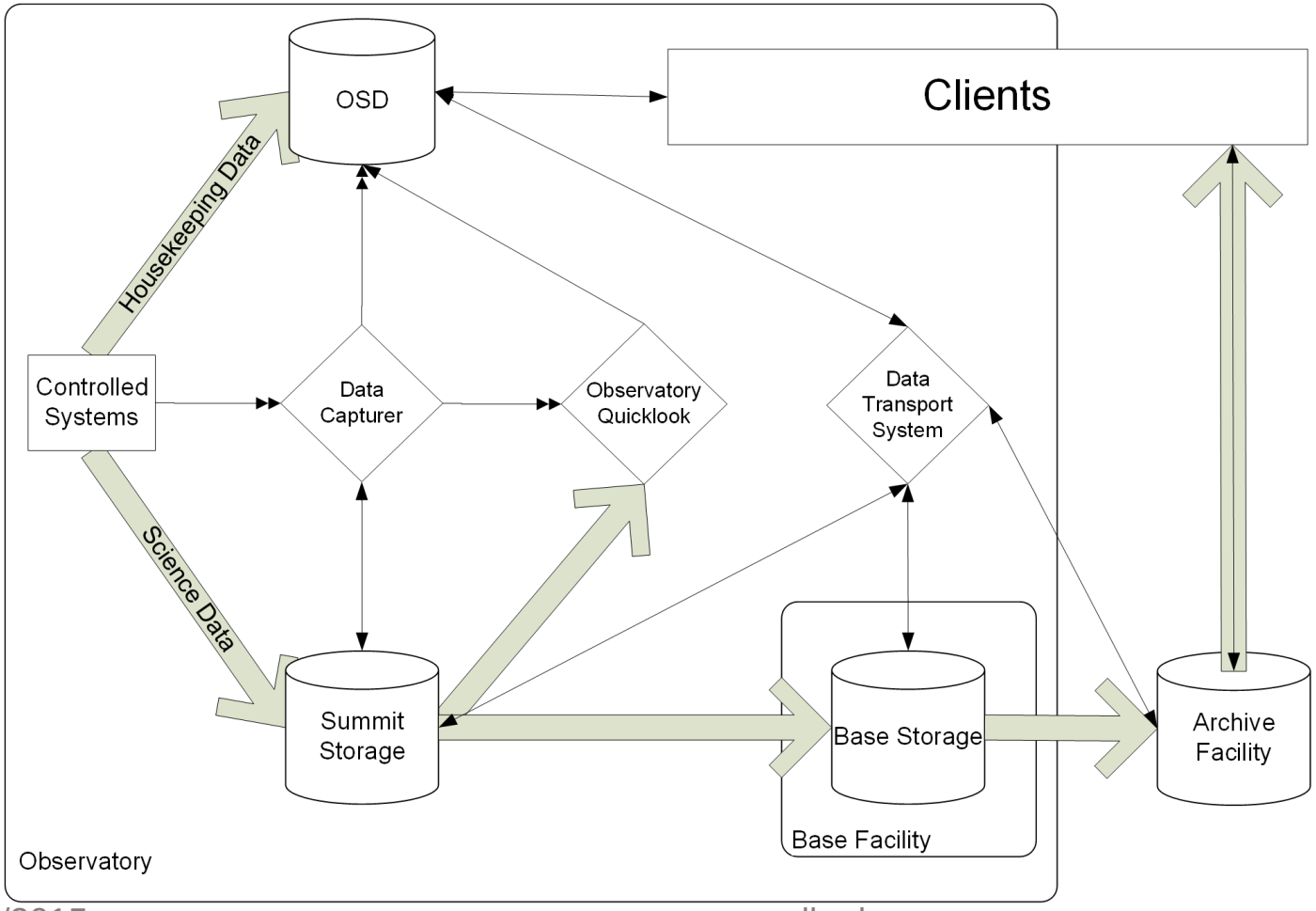
- Key functionality:
  - Commands activity
  - Routes data
  - Monitors activity
  - Records activity

Your authority is delegated

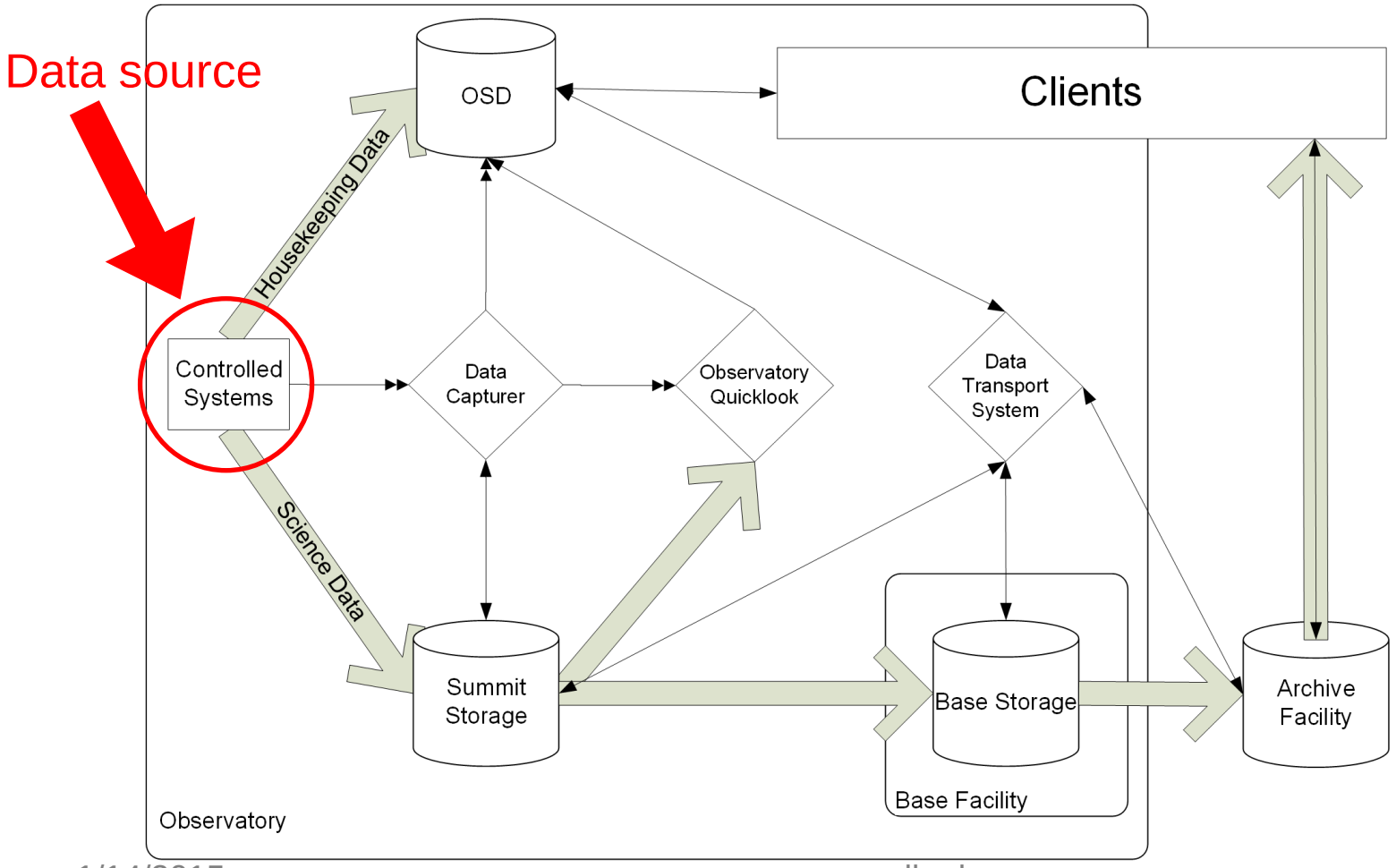
Workflows: some software decisions

- Synchronous/controlled, asynchronous and autonomous
  - Synchronous calls: send command, await response/completion
  - Asynchronous calls: send command and then do something else.
  - Autonomous process: act according to pre-set criteria without explicit command
  - Often processes have autonomous default but can also be commanded to act
- Checkpointing.
  - Workflow should recover from loss of state.
- Deployment of updated software
  - E.g., pull from repository, rebuild and automated tests

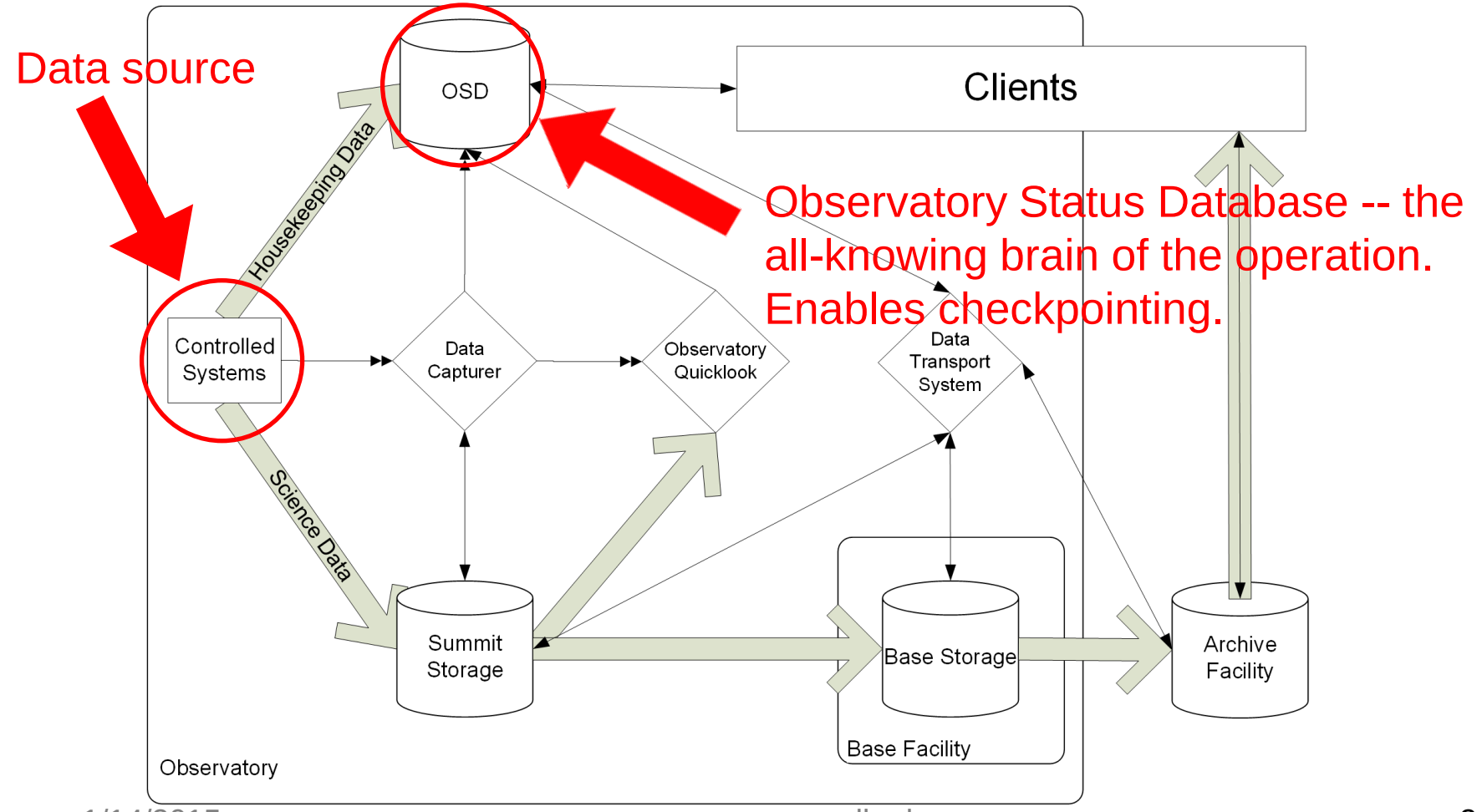
Workflows: example (CCAT Observatory)



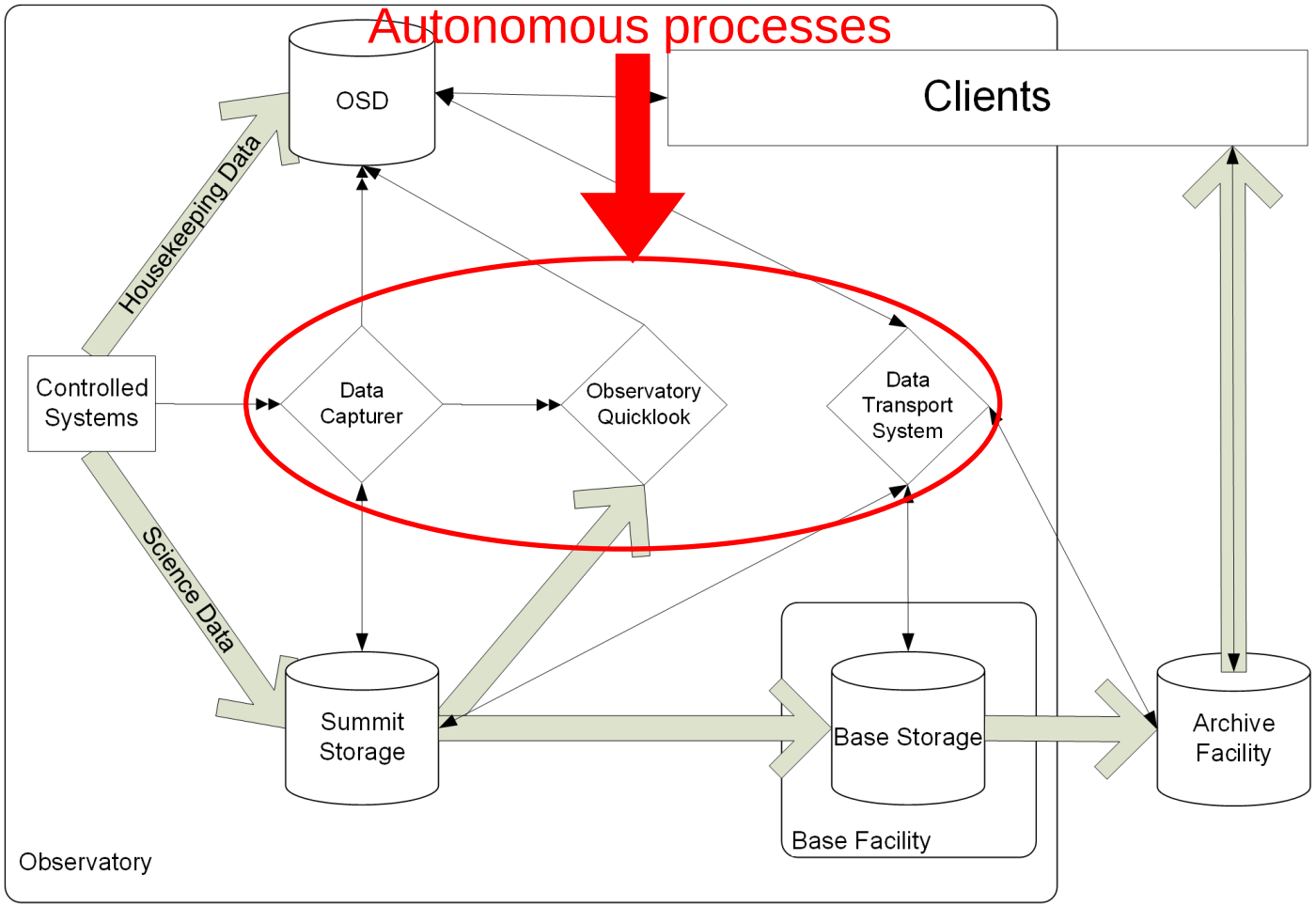
Workflows: example (CCAT Observatory)



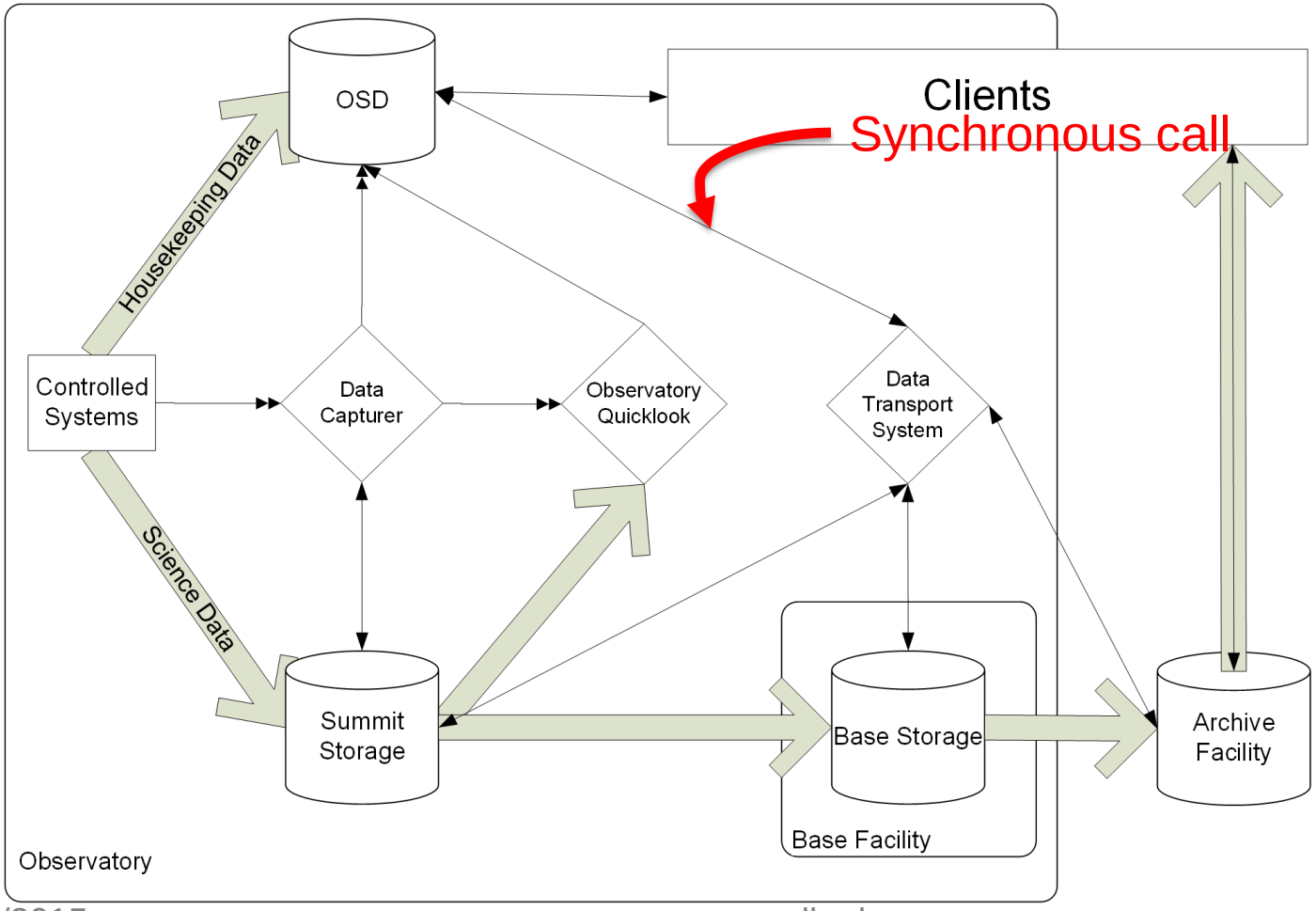
Workflows: example (CCAT Observatory)



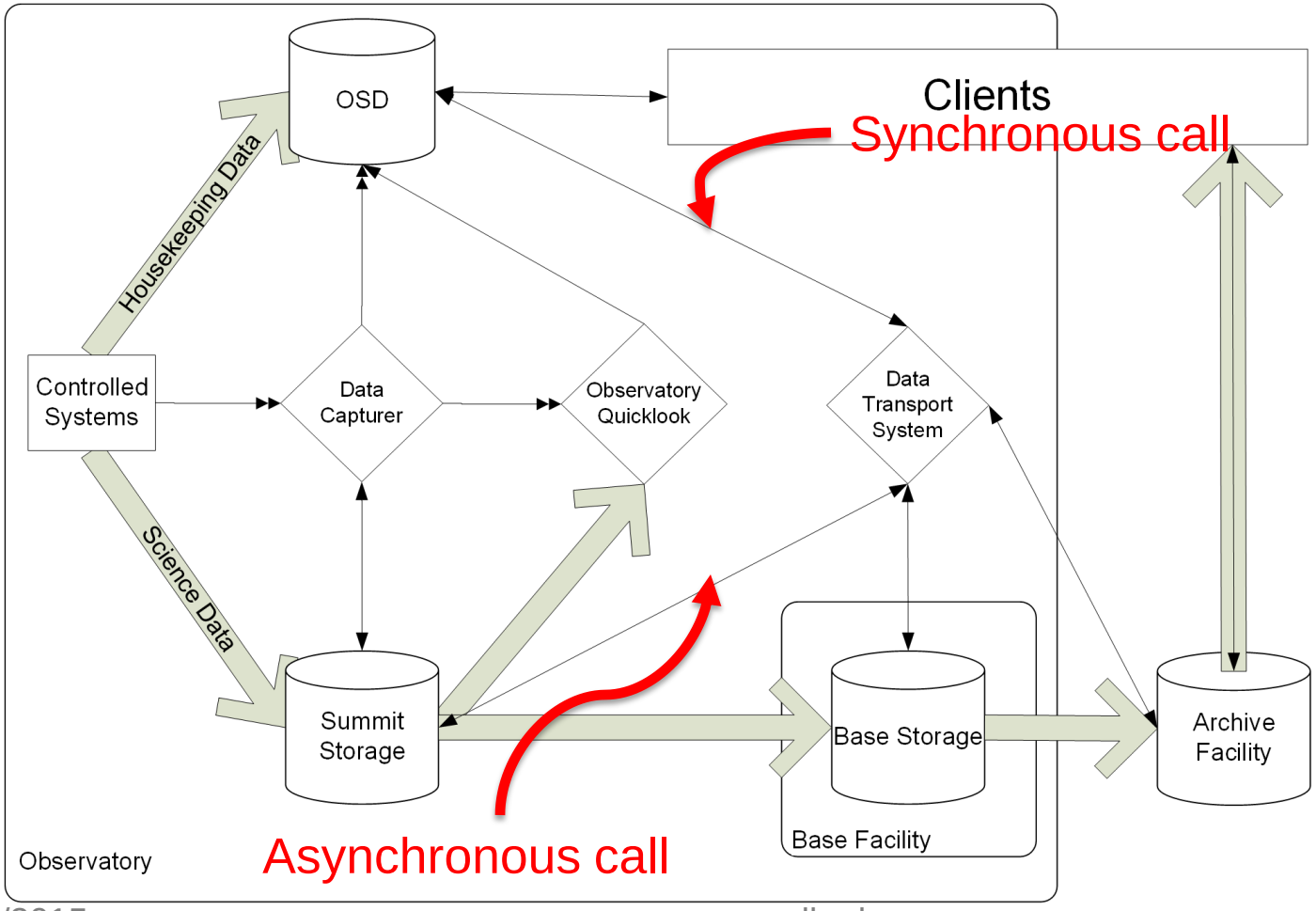
Workflows: example (CCAT Observatory)



Workflows: example (CCAT Observatory)



Workflows: example (CCAT Observatory)

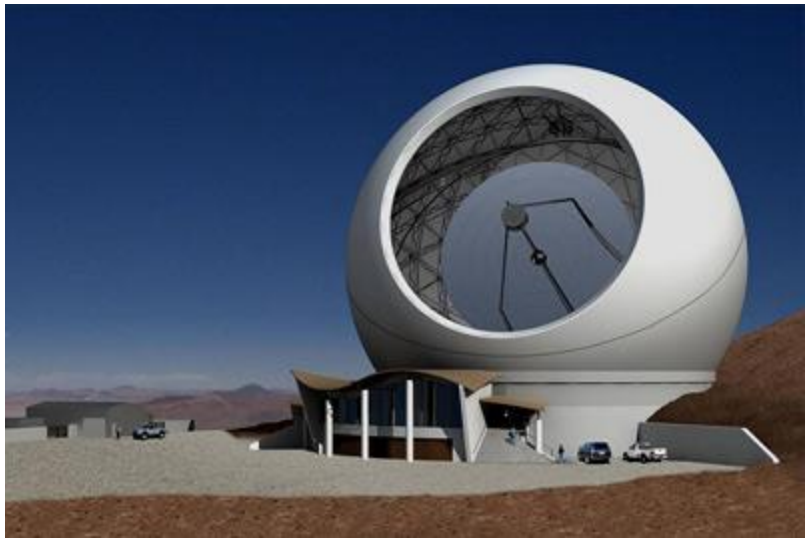


Workflows: example (CCAT Obervatory)

- Key decisions:
  - Adopted Python for control software
  - SQL database
  - Asynchronous communication via files or OSD
    - One process would write a file and write to OSD
    - File-based communication lower-latency than via OSD
    - File-based communication low-tech but reliable
  - After detailed study, picked HDF5 file format
    - Fully hierarchical
    - Strong python integration
    - Highly expansible
  - Enumerated states

Workflows: Key elements of CCAT design

- Loosely-coupled elements
  - Fault-tolerant
  - More pull than push
  - Asynchronous calls preferred
- Autonomous operations
  - Reliable and predictable
  - Planned move to fully autonomous observatory
- State-controlled
  - Observatory Status Database (OSD) stores information, serves it out
  - Autonomous processes act according to OSD information



CCAT

Workflows: Incorporating HPC (overview)

- Often you don't have root on the HPC machine
  - You may also not be able to get software installed, or policies changed

- Best use of the HPC resource is asynchronous
  - Small script to launch processing
    - Should be lightweight
  - Driven by the availability of data
  - Submitted to batch queue
    - Don't hold your breath!
    - (batch queue is asynchronous too)
  - Record activity of components
  - Monitor outputs

Is the data here yet?
Is the data here yet?
Is the data here yet?
Is the data here yet?
Is the data here yet?
Is the data here yet?
Is the data here yet?
…

Imagine a roadtrip with this autonomous process

Workflows: Incorporating HPC (methods)

- Globus can be scripted to get data in and out (cf Data Transfer talk), or `scp`, etc

- Depending on policies and permissions, workflow script can be run:
  - With `screen` command
  - As `cron` job
  - As linux service
  - On remote host
    - Access HPC resource over ssh with key, run process
    - Execute pre-defined RPC
- Batch jobs, once submitted, don't depend on your login session being live.

Workflows: when to ask for help

- Domain researchers:
  - Intimate understanding of the activities
  - Embedded into the workflow already
    - Typically involved in designing the experiment
    - Often involved in writing the proposal
- IT professionals
  - Often more current with available technologies
  - Typically more practiced
  - Outsider's view
- Provisioning effort and identifying help should be part of planning

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

- Another way of looking at it:
  - Data management *enables and underpins* the workflow
  - Your workflow will/should/can achieve NSF/other data management requirements

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

Just one more thing…

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

- CODE IS DATA, TOO!

Oh, just one more thing…

Data Management: We need a plan. It's not just about proposal hoops.

- Data Management Plans (DMPs) now required by many RFPs (including all NSF RFPs)

- Taking planning seriously makes sense:

  – It allows costing it into a budget

  – IT OFTEN *IS* THE WORKFLOW, END-TO-END

  – A proposal DMP is a higher-level description, but further planning should take place before implementation begins

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

Data Management: Description

- Enumerate your data products!
  - Include code, documentations, visualizations, online content
  - Metadata is also data!

- Decide on formats, including considerations of:
  - Format longevity
    - Does the format meet likely future demands?
  - Access to the content elements
    - Is there a common file reader?
  - Ease of use, including by others
    - Is the format commonly used in the field?

Data Management: Description

Examples of data products

Raw data: the original data, as written to disk

Intermediary products: includes calibrations, checkpointed files, etc

Final data products: the results of processing. May be several generations of products

Examples of data formats

Code: Text (ASCII, Unicode)

Graphics: PNG, JPEG, TIFF…

Documents: PDF, .docx, .xlsx, .txt

Raw Data: binary formats, csv, .txt

Video: .mp4, WMV, .mov

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

Data Management: Control

- Control includes things we *do* to our data.
  - I/O
  - Transport
  - Pipelining/processing
  - Versioning
  - Tracking
  - Quality Assurance
  - Sharing and security

- Many functional requirements arise here

The rest of us have to use software

Data Management: control (I/O, transport)

- I/O
  - I/O typically handled by operating system and hardware
- Transport
  - Physical transport of storage media
    - Wrap it up with padding!
    - Very high effective bandwidth available, but high latency
  - Internet
    - Ensure you test average speeds and evaluate data transport costs
      - Very high speeds are expensive
      - TCP/IP has overhead, window sizes reduce over bad connections
  - Local network
    - Reliable and typically fast
    - Can often use UDP for higher speeds (also allows broadcast)

Data Management: control (pipelining/processing, tracking)

- Processing pipelines are workflows themselves
  - Separate control-of-flow from algorithmic elements
  - Python, Perl are both commonly used in newer pipelines, calling compiled code for processing-intensive elements.
    - Quick to develop and debug, where performance isn't critical.
    - Interface well with compiled code, particularly C/C++

- Tracking should be done with reliable, robust storage
  - Databases allow powerful queries, preserve data integrity. Flexible
    - May drop incoming data. Sophisticated/complex.
  - Writing text files simple, well-understood
    - A bit primitive. Querying is effectively "read it all".

Data Management: control (versioning, QA)

- Versioning allows tracking of text products (cf Best Practices talk)
  - Allow easy reversion of changes
  - Can have multiple people working on same product
  - Git, Mercurial distributed version control systems
  - SVN, CVS older, centralized version control systems

- Quality Assurance (cf Best Practices talk)
  - Testing quality of output is a functional test
    - Can test against set inputs with known output
  - Can be automated
  - Should run
    - When new versions of code implemented
    - Other environmental context changed

There was only one way to be sure

Data Management: control (sharing and security)

- Enumerate groups and their access
  - Groups e.g., “project staff”, “research community”, “general public”
  - Access e.g., “write”, “delete”, “modify”, “read”, “download”

- Enumerate risks of compromise
  - Third party access to authentication information, escalation of authorization, exploitation of software vulnerabilities, “bad actor”.

- Evaluate cost of compromise
  - Permanent loss of data?
  - Consuming valuable resources (e.g., processing)
  - Improper release of results or use of resources
    - Can cost prestige, cause embarrassment, endanger ownership, etc

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

Data Management: policies

- Policies constrain and guide control, generating non-functional requirements/design constraints

- Key policy issues include:
  - Who can have our data?
  - When can they have our data?
  - Under what conditions?
    - Licensing and attribution requirements
  - For how long must we keep our data?

Decisions, decisions

- It is best to decide this early and get agreement from all involved

Data Management: Policies

- Code-sharing can be done via several licenses:
  - BSD, Apache, MIT: Permissive. Allow third parties to adapt software, redistribute and not share
  - GPL and LGPL: “Viral”, must also be applied to redistributed software which incorporates the (L)GPLed software
- Licensing, Copyright are different! Check your institutional policies
- Proprietary periods should typically include releasing results which support publications
- Retention policies should allow a public release before deletion

Data Management: what *is* data management?

- One view (congruent with NSF guidance)
  - Description
  - Control
  - Policies
  - Storage/preservation

Data Management: Storage/Preservation

- Storage: Persisting the data during the project's duration
- Preservation: Persisting the data after the project is completed
- There can be some hard decisions!
  - Paid service cost broadly scales with volume. Free services may exist
    - On-campus: CAC's Archival Storage facility, eCommons (free!), CIT's EZ-Backup and department facilities – each serves different needs
    - Github, sourceforge, etc
    - Youtube
    - Journal supplementary data resources
    - Department resources
    - TACC's Ranch has no purging policy at present, 60PB of tape storage!

Data Management: Storage/Preservation

- Tape is cheapest, but an automated tape system is expensive.
- Once a system or component is out of warranty, failure can be costly
- As disks have become larger, the risk of unrecoverable read errors (UREs) implies RAID 6 instead of RAID 5, and supplementary technologies
  - Much better performance when this is done on the controller, ie, in hardware
- Investigate appropriate compression
  - Lossy vs lossless, various compression algorithms depending on data properties, e.g, delta compression

Data Management: You are not alone!

- Research Data Management Service Group (RDMSG, http://data.research.cornell.edu/) provides DMP consulting and other services to Cornell researchers

- For those planning to use CAC services, we will provide help writing Data Management Plans and cyberinfrastructure sections of Proposals

- Many people are addressing similar questions, both inside and outside Cornell, including many other research institutions.

Workflows and Data Management: Overview

- Workflow planning and Data Management planning share considerable overlap
- Evaluate technical options and make decisions (it's OK to delay final decisions until they're relevant)
- Identify failure points
  - Unleash your inner pessimist, then confound them with fault-tolerant design