NRC·CNRC

VOSpace v2.1

Brian Major
Canadian Astronomy Data Centre

National Research Council Canada | Conseil national de recherches Canada

Canada

VOSpace 2.1

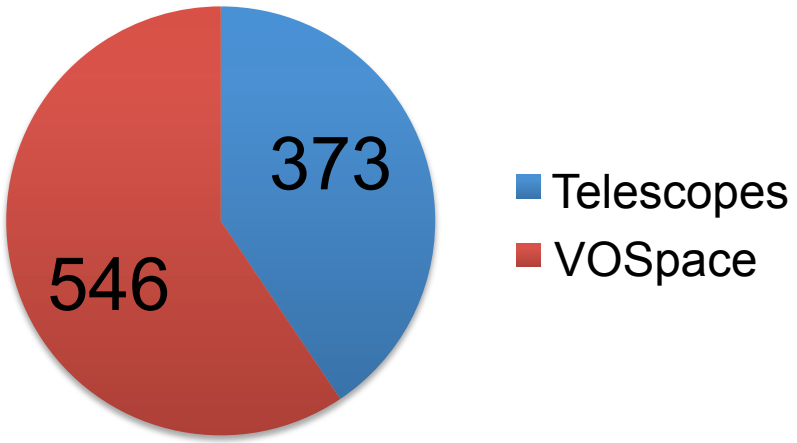
- A minor revision – any changes must be backwards compatible with v2.0
- A couple of noteworthy additions to v2.0 driven from the operational requirements of the CADC implementation of VOSpace
  - Performance
  - Access Control
- Potential VOSpace goals for the future

VOSpace at the CADC

- VOSpace is a critical component of CANFAR (Canadian Advanced Network for Astronomical Research)
- Has gone through the growing pains of improving reliability and performance
- Most of VOSpace (and the underlying data storage systems) runs on national infrastructure (non-CADC resources)
- Storage and processing are co-located whenever possible

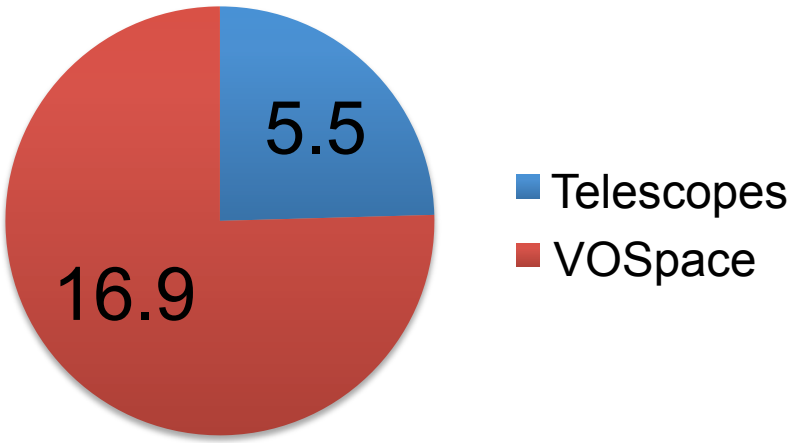
Data served from the CADC in 2013:

Bytes (TB)



60% VOSpace

Files (millions)



75% VOSpace

How is this happening?

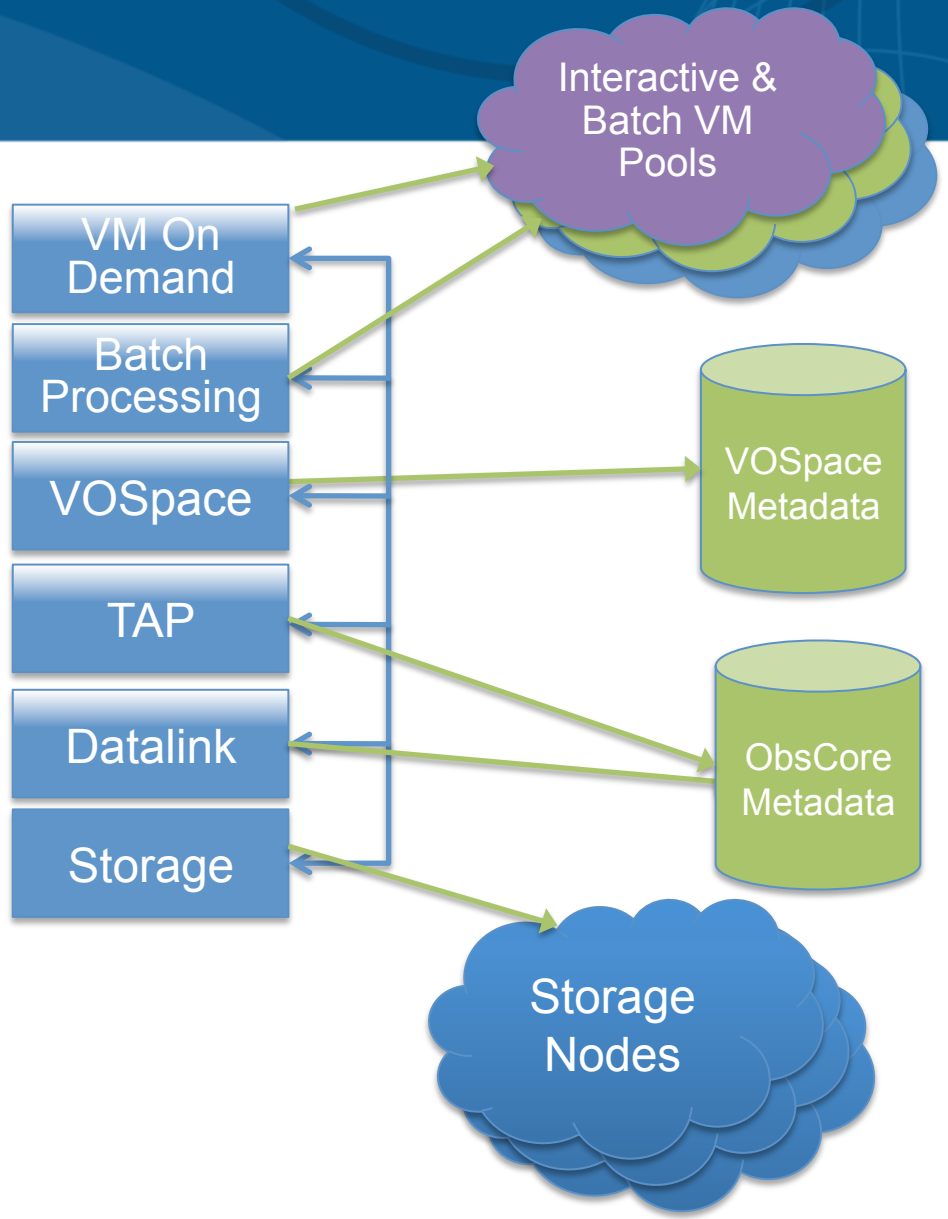
CANFAR and VOSpace

CANFAR = Batch processing

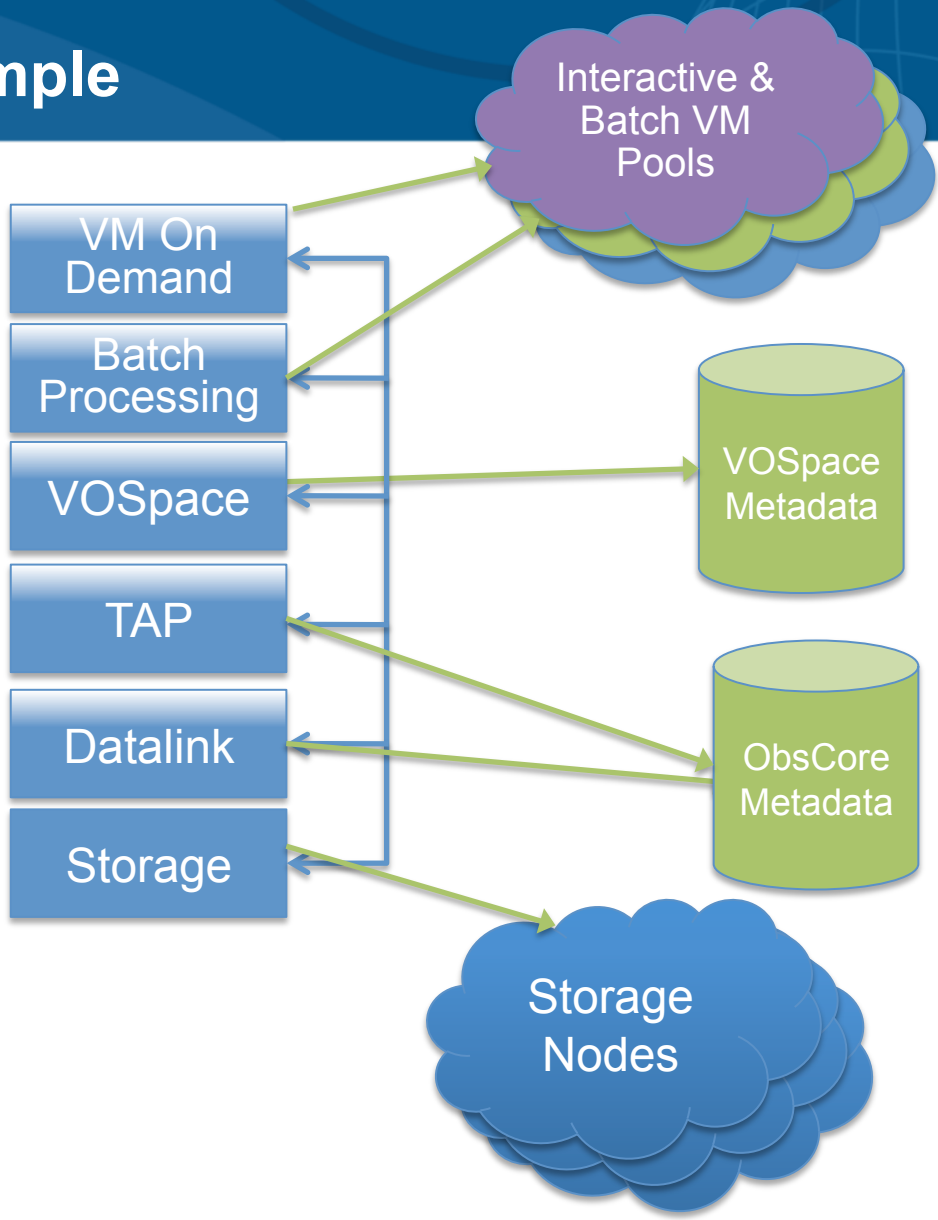
Batch processing = lots of traffic

Many instances of a pre-configured virtual machine running in parallel.

VMs process datasets (from telescope archives or VOSpace) and save the results to VOSpace.

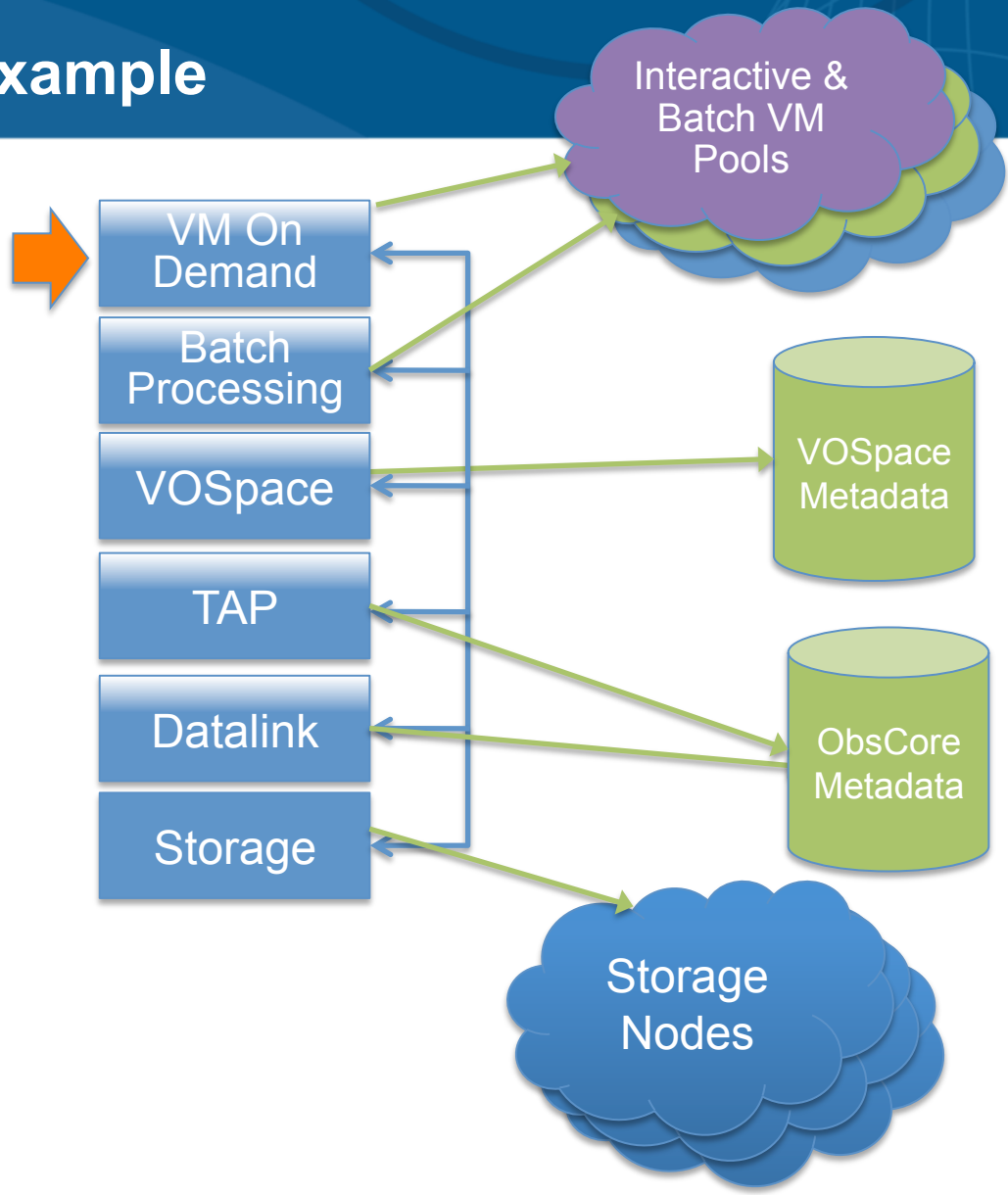


CANFAR VO Science Example



CANFAR VO Science Example

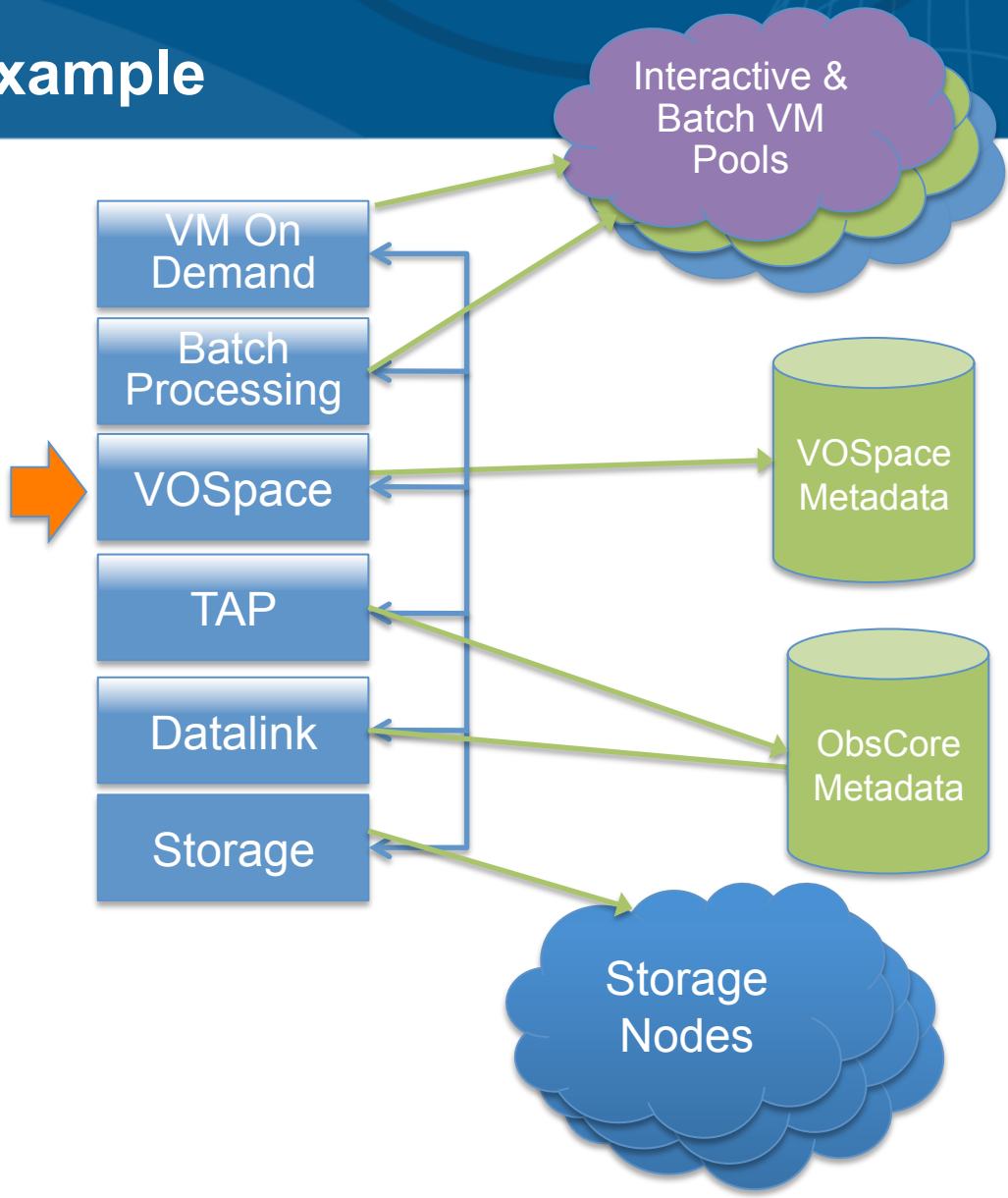
① User creates and configures VM



CANFAR VO Science Example

① User creates and configures VM

② User saves VM img in VOSpace

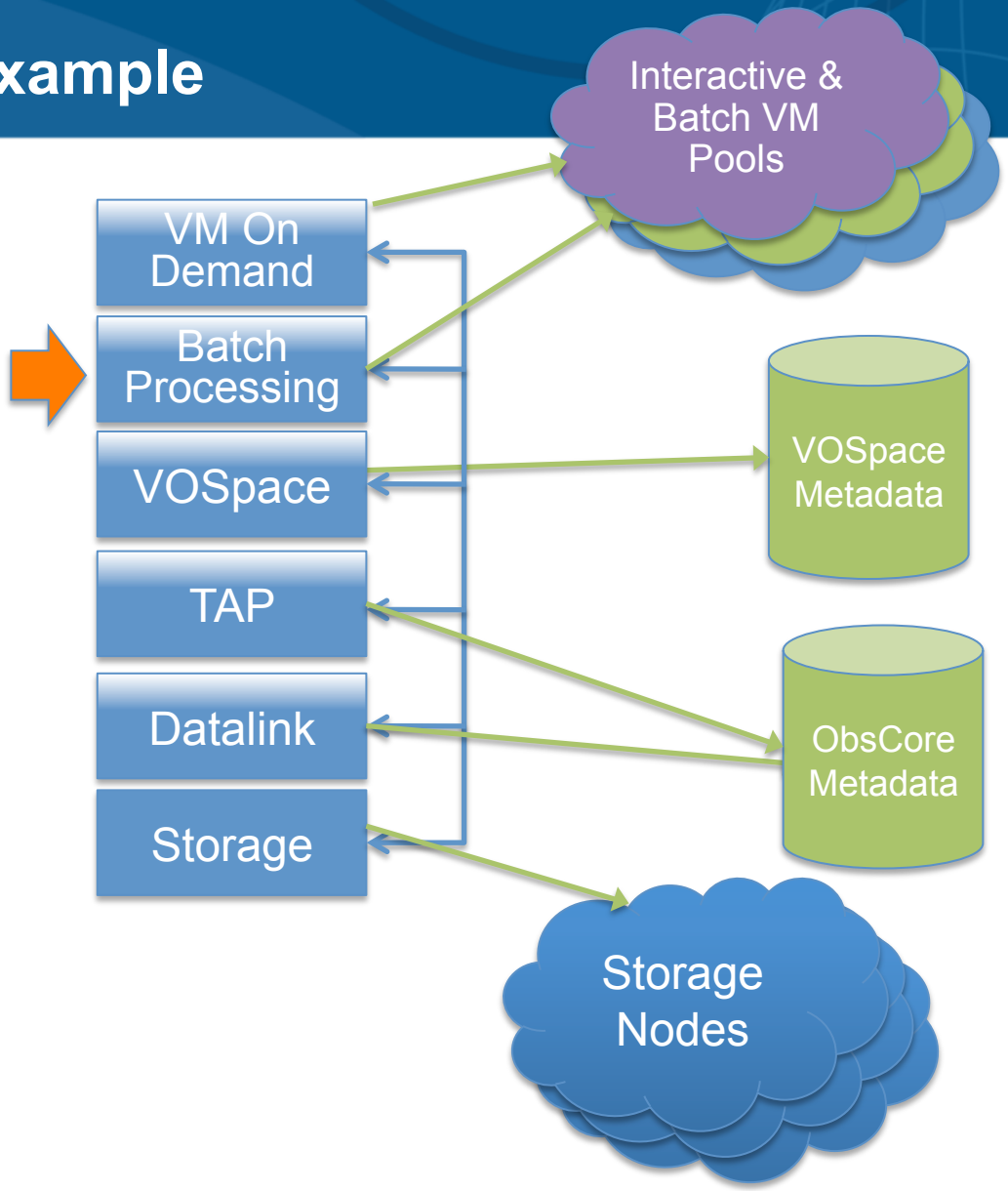


CANFAR VO Science Example

① User creates and configures VM

② User saves VM img in VOSpace

③ User launches X instances of image in batch processing



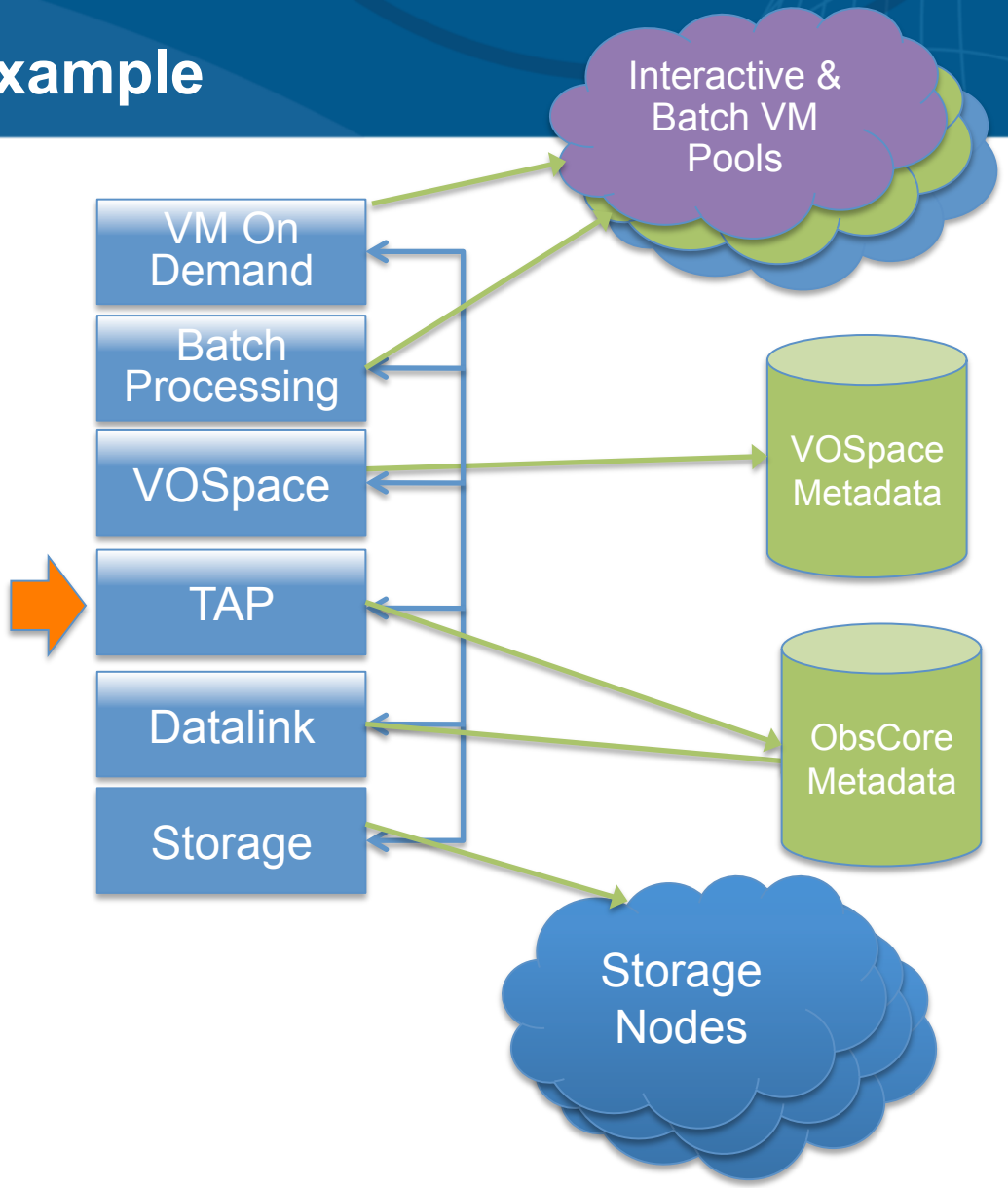
CANFAR VO Science Example

① User creates and configures VM

② User saves VM img in VOSpace

③ User launches X instances of image in batch processing

④ VMs use TAP to find data from ObsCore



CANFAR VO Science Example

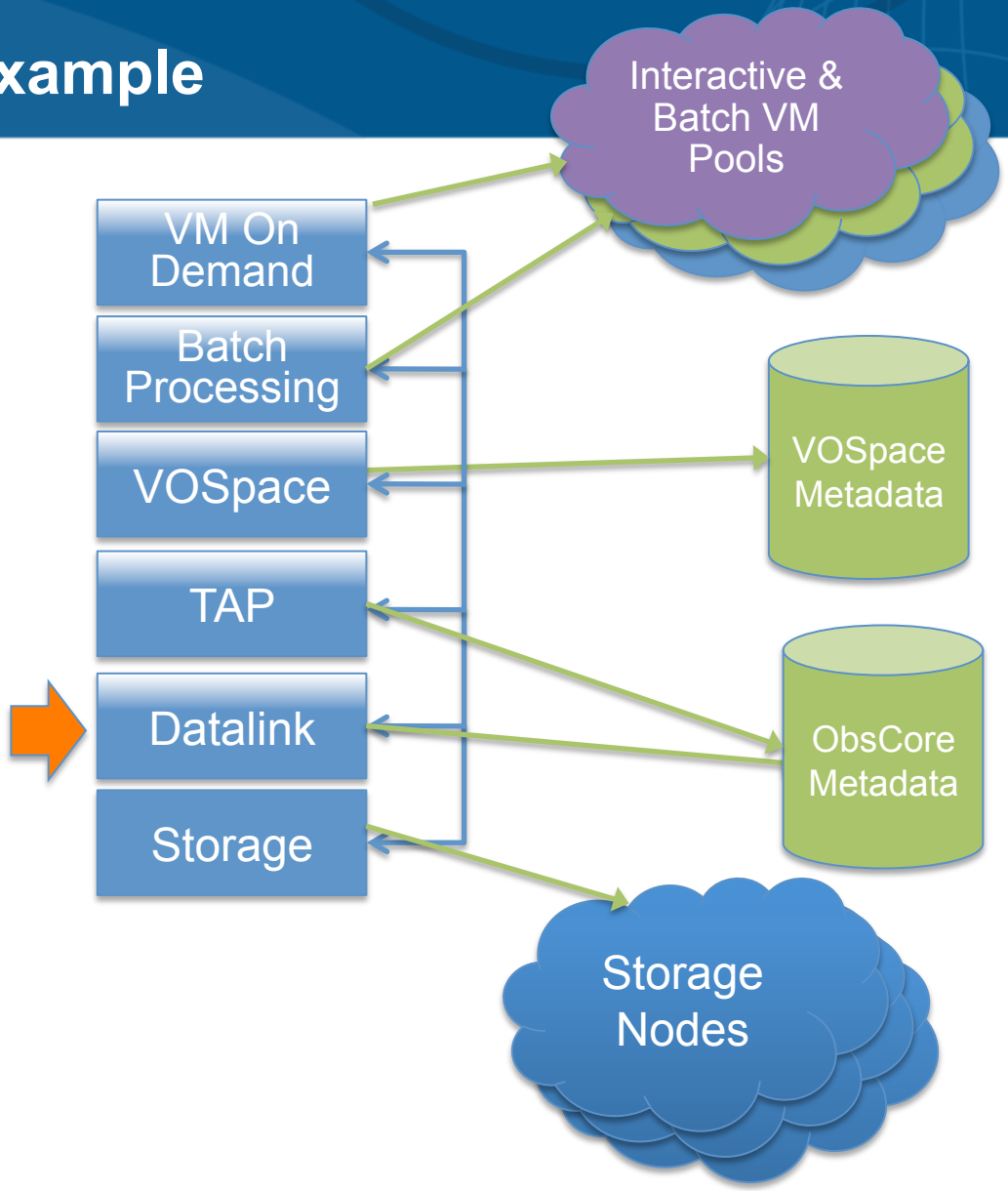
① User creates and configures VM

② User saves VM img in VOSpace

③ User launches X instances of image in batch processing

④ VMs use TAP to find data from ObsCore

⑤ VMs use Datalink to access data



CANFAR VO Science Example

① User creates and configures VM

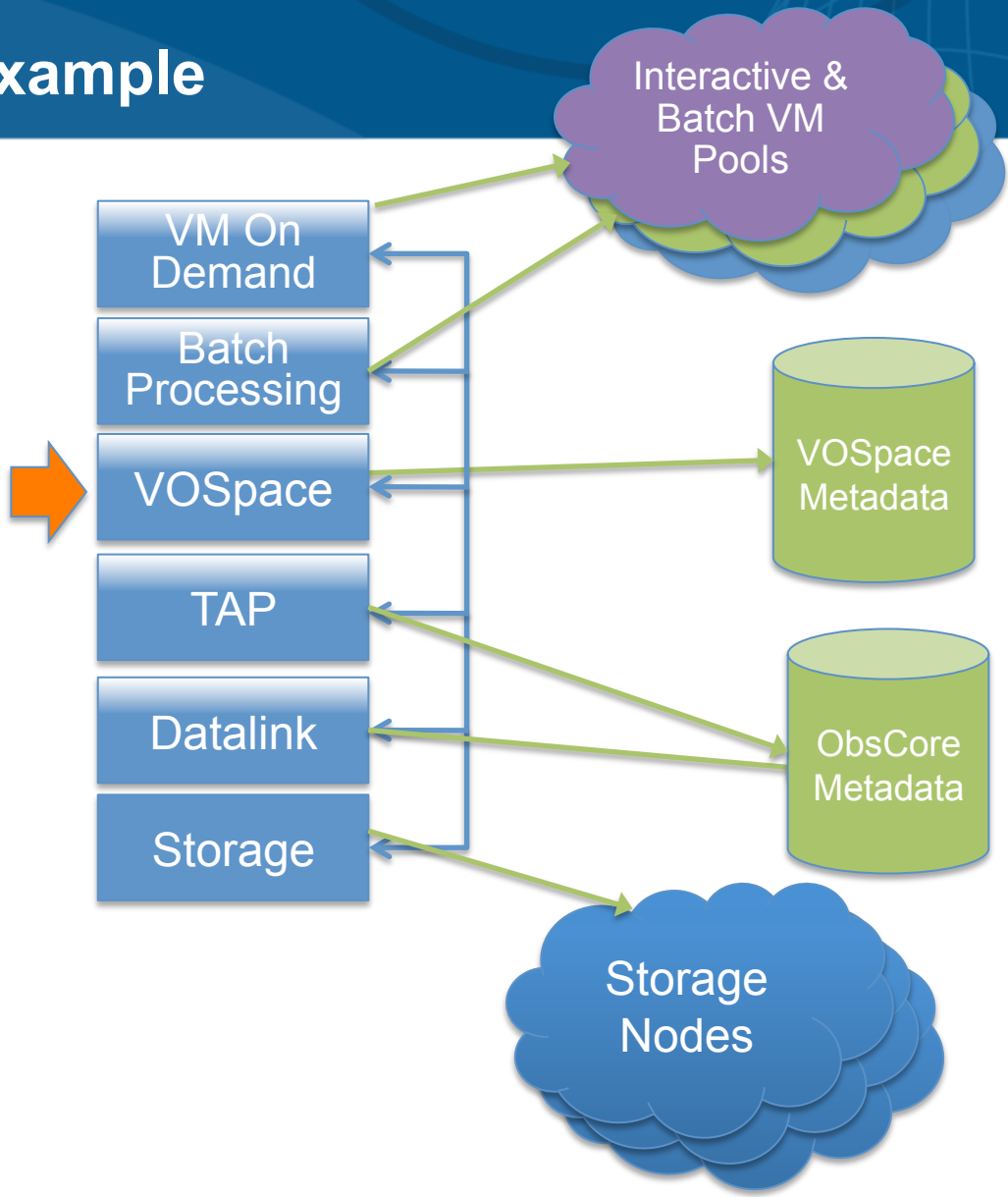
② User saves VM img in VOSpace

③ User launches X instances of image in batch processing

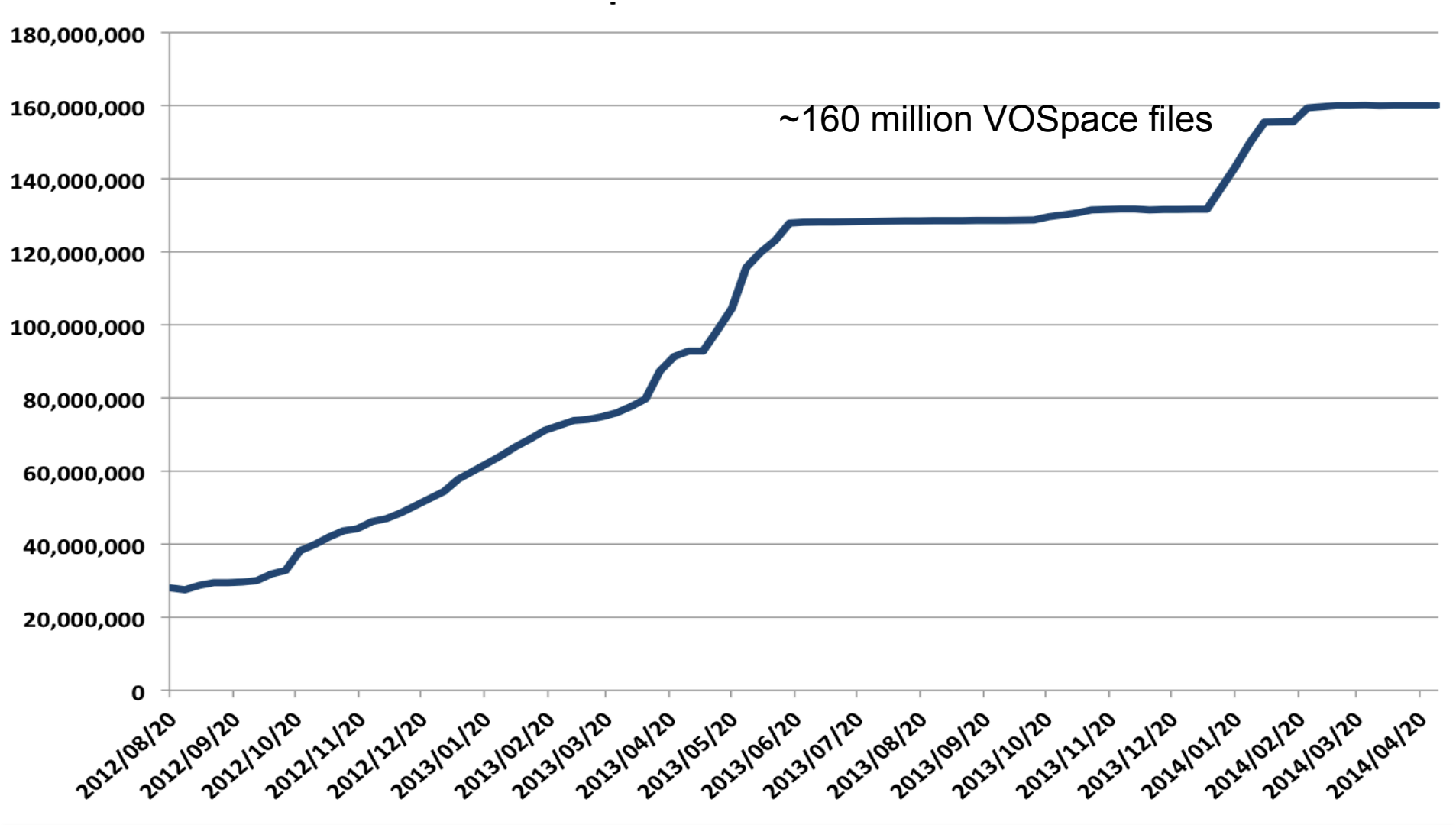
④ VMs use TAP to find data from ObsCore

⑤ VMs use Datalink to access data

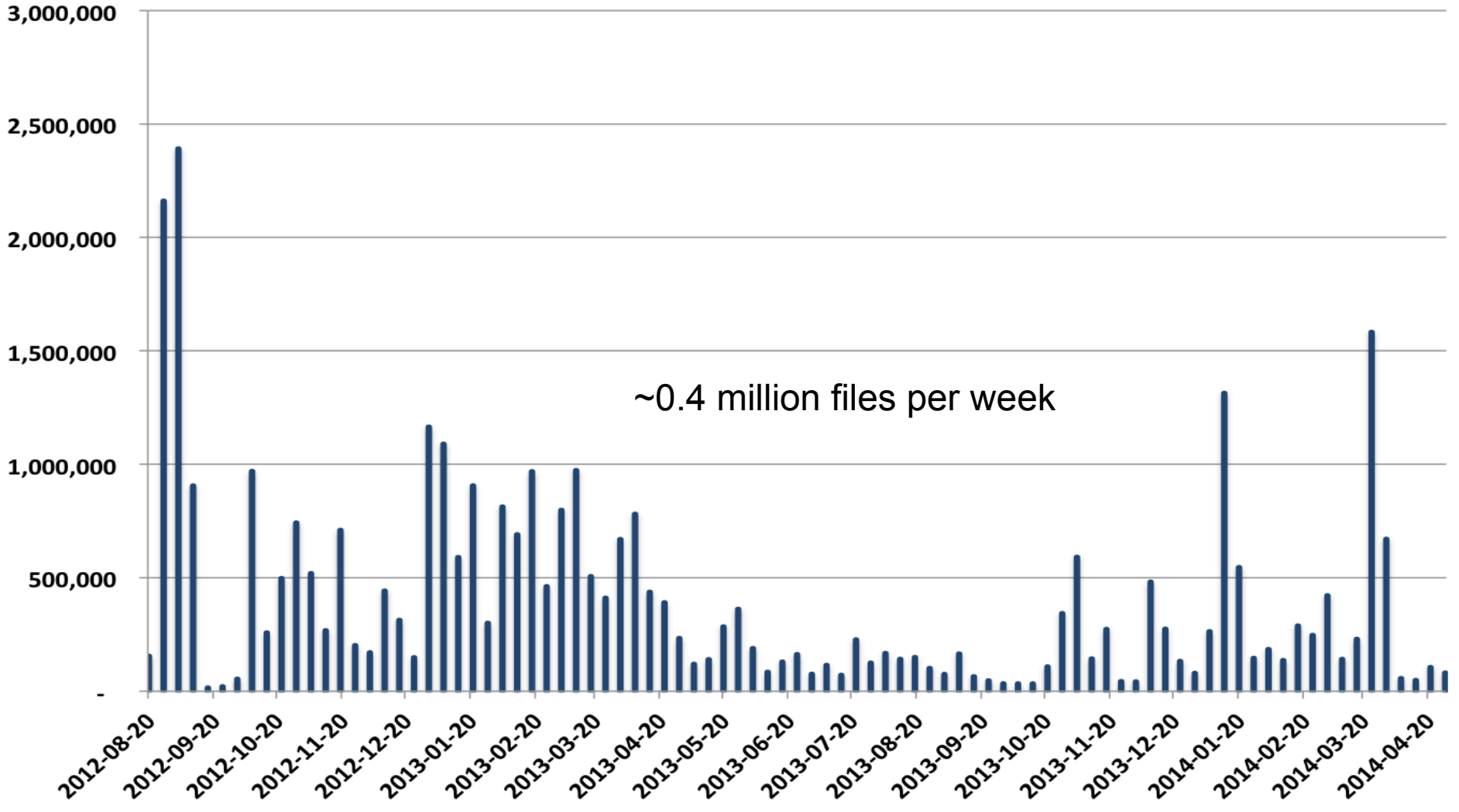
⑥ VMs save science results in VOSpace



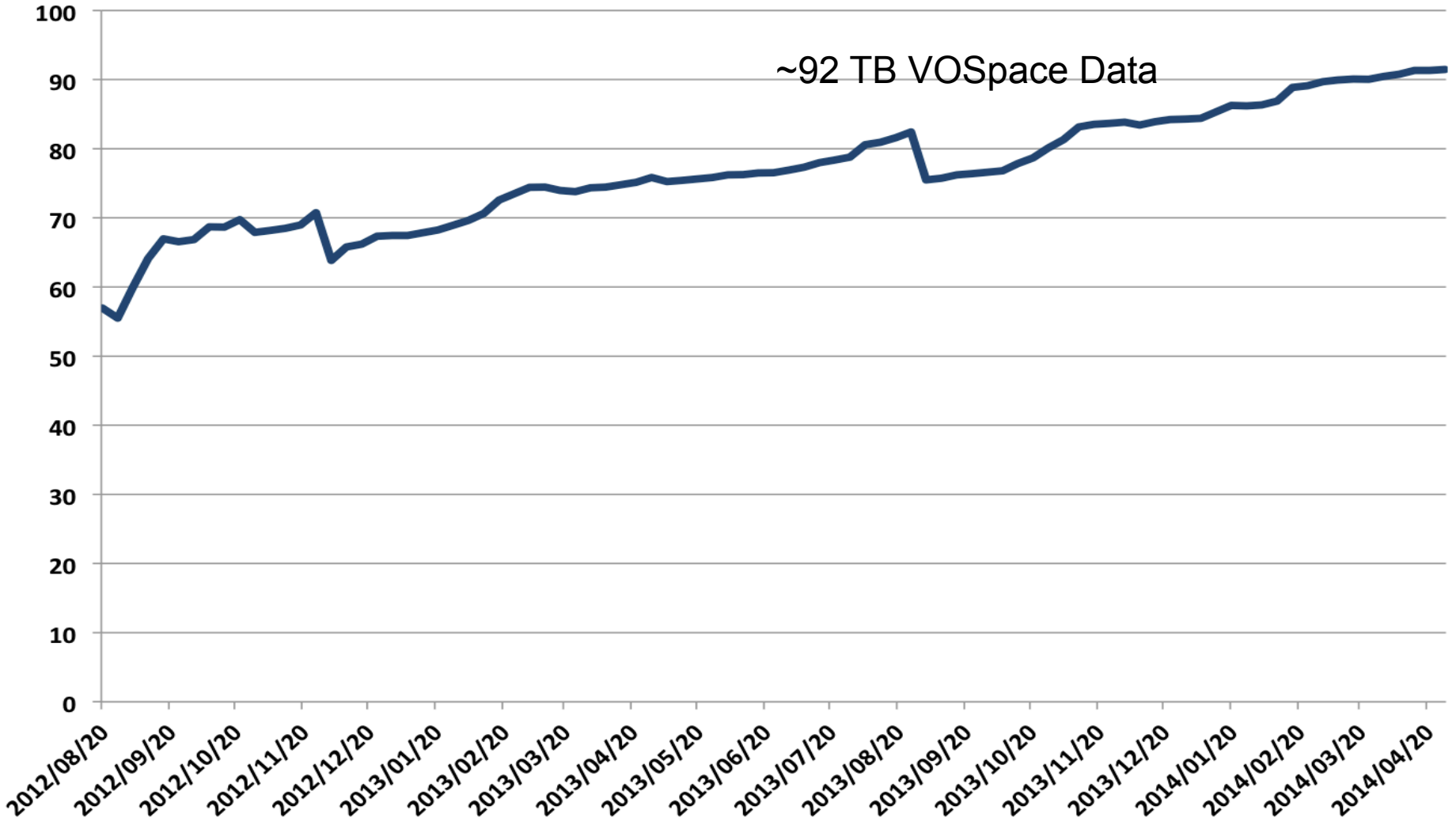
VOSpace Growth in Files (Aug 2012 – Apr 2014)



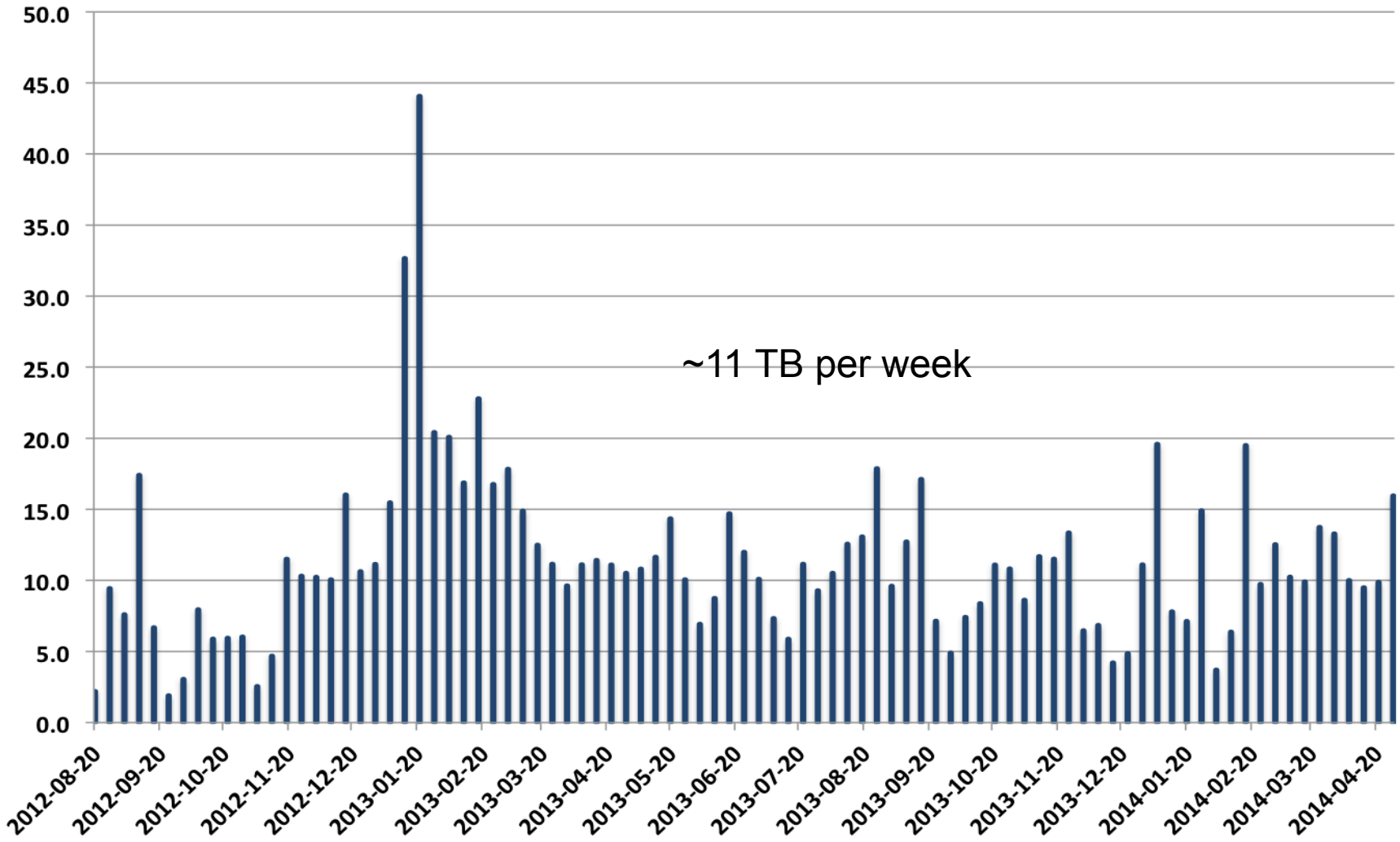
VOSpace GETs: Files per week (Aug 2012 – Apr 2014)



VOSpace growth in TB (Aug 2012 – Apr 2014)



VOSpace GETs: TB per week (Aug 2012 – Apr 2014)

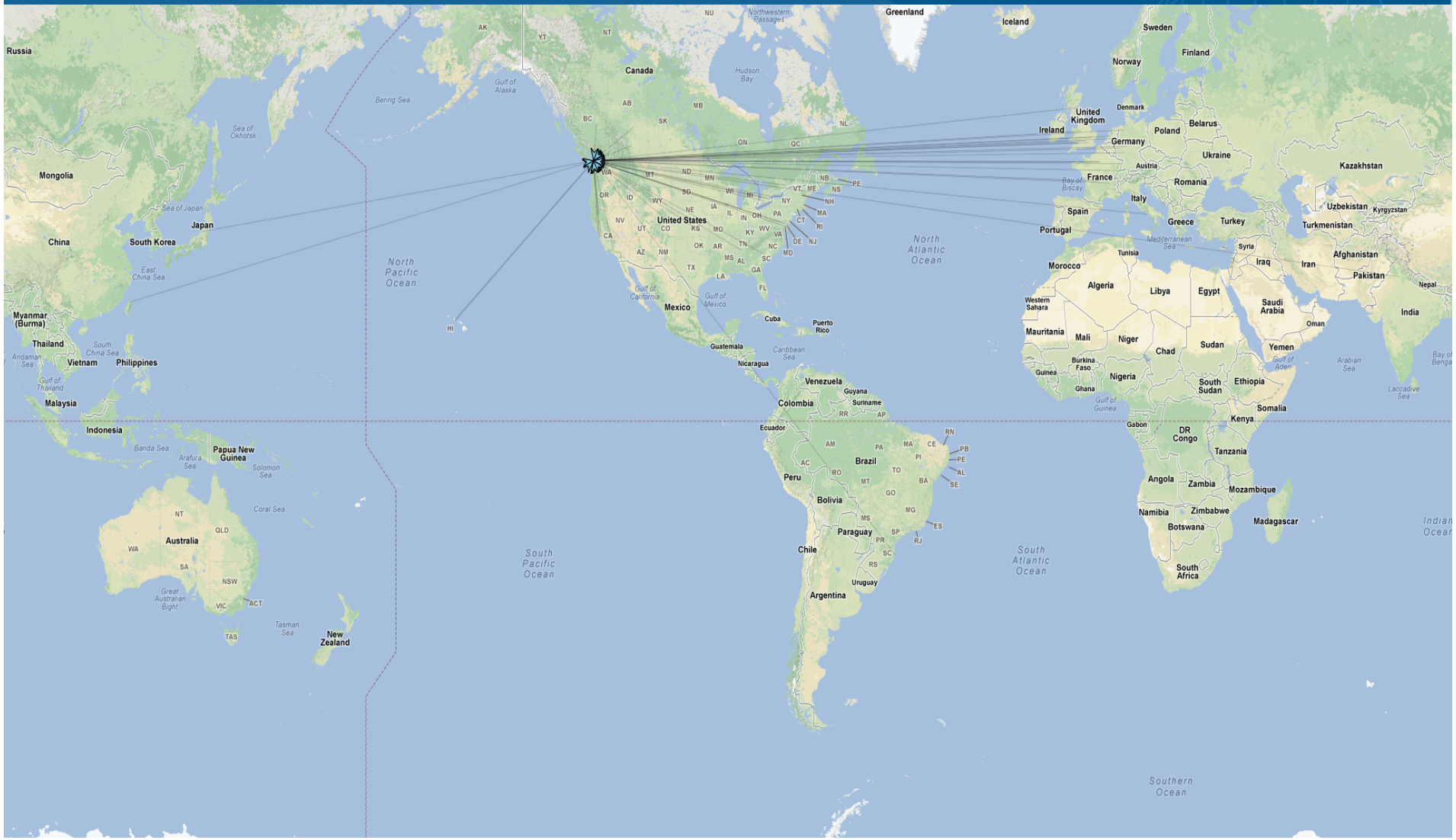


VOSpace GETs and PUTs: Aug 2012 - April 2014

	GET (TB)	PUT (TB)	GET (file count)	PUT (file count)
Total	1 107	153	35.14 million	165.58 million
Average per week	**11**	**2**	**0.39 million**	**1.86 million**
Peek per week	44	12	2.40 million	10.40 million

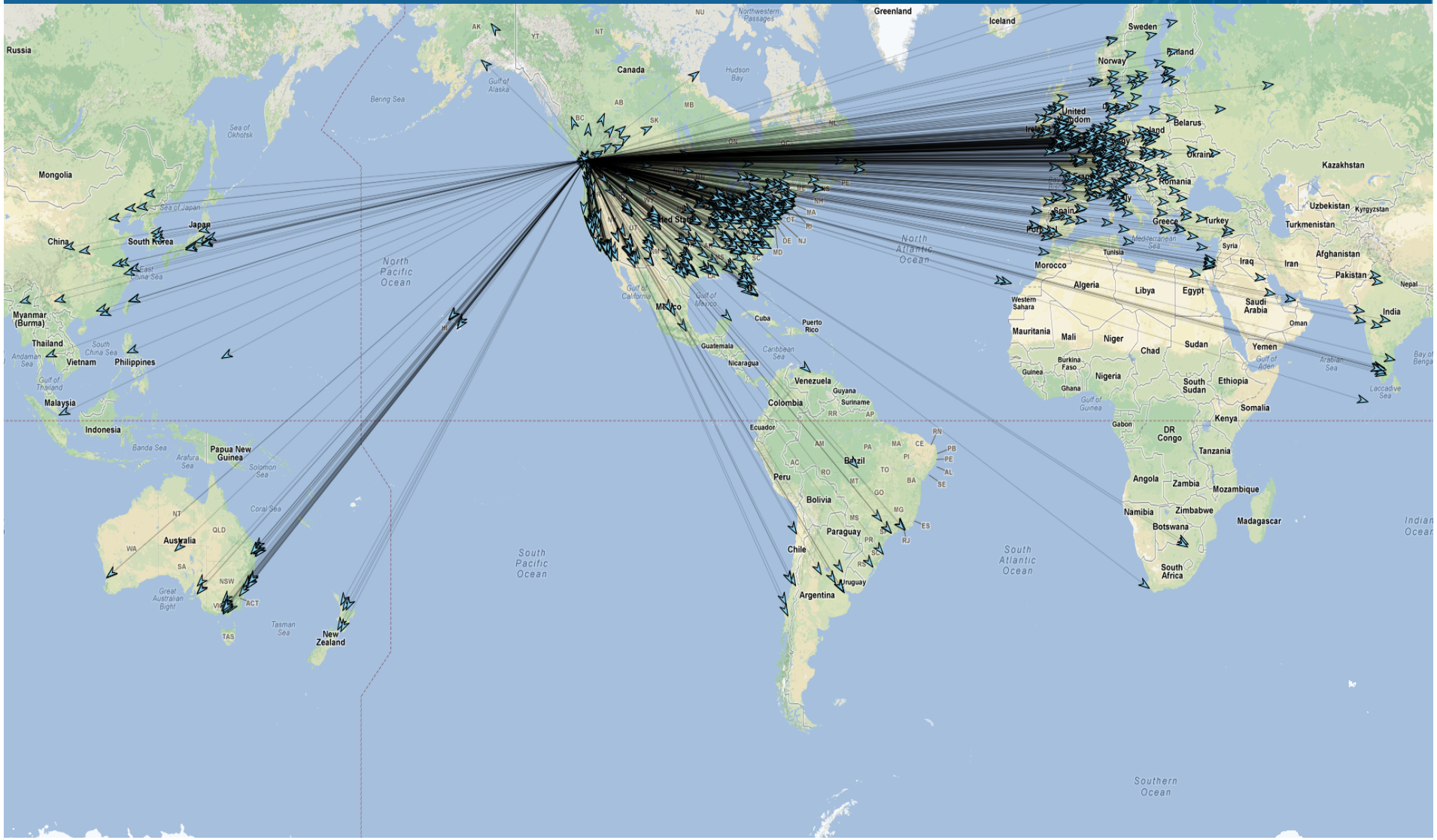
During this talk:
~ 123 GB of data transferred
~ 3500 files transferred

Geography of VOSpace PUTs



NRC·CNRC

Geography of VOSpace GETs



NRC·CNRC

Growing pains and lessons learned…

- As VOSpace usage grew, we had to adjust to meet demand
- Made and learned from mistakes along the way
- The bottleneck kept shifting: fixing one vulnerability would expose the next

Examples:

- Tuning database transactions, locking
- Authorization techniques
- Contention on root nodes in DB
- Resource pooling
- Recognizing our system limits, “try again later” rather than fail
- Building smart clients, identifying problem ones

Allowing for operational requirements…

The VOSpace specification needs to be flexible enough to allow for performance optimizations. In general, it has been.

New optimization feature in v2.1: Reduce the number of network connections and redirects clients need to follow to upload or download a file.

Standard VOSpace transfer negotiation

1. Client posts transfer request to VOSpace
2. Client asks VOSpace to run transfer (/async only)
3. VOSpace redirects to transfer details
4. Client gets next URL to follow from the transfer details
5. Client follows next URL
6. VOSpace returns list of download (or upload) endpoint URLs
7. Client starts byte transfer

Total: 5 network connections

For small files, this is a lot of overhead!

Optimized VOSpace transfer “negotiation”

1. Client posts transfer request to VOSpace with details in URL as query string parameters
2. VOSpace returns a single URL to use for data download (or upload)
3. Client starts byte transfer

Total: 2 network connections

This is the best you can do without prior knowledge of the physical location of the bytes.

Optimized VOSpace transfer negotiation

The consequences…

This is an optimistic approach, it assumes that, most of the time, there are no failures.

What is compromised? Error handling, failover, and all the benefits of using UWS for managing transfers.

Recommendation: Use the optimized/optimistic method first. If an error is encountered, use the regular/pessimistic approach to get full details on the error (if it persists) and inherit any additional failover support it may provide.

VOSpace Robustness: Techniques for failover

Examples…

The power of VOSpace Views

Another great (existing) optimization: VOSpace Views

"Move the code to the data, not the data to the code"

VOSpace views allow you to define a set of operations that usually reduce the number of bytes that need to be transferred.

Views in use at the CADC:

On FITS files (data nodes):

- Cutout view, WCS view, FITS header (fhead)

On Container nodes:

- Manifest view, RSS view

VOSpace Views in the future?

Imagine dynamic views generated by the user…

A “CustomView” that users could create and provide to do work on a data set that is run on a storage node.

- The URI for the view could be a VOSpace URI pointing to the user’s custom view.
- Target file could be streamed to the stdin of view
- stdout would be streamed to the user

Similar to CANFAR Processing, but even the short network hop is removed

Reducing I/O: Network random access

Network Random Access on VOSpace Files

- vos: A python VOSpace client API
  - FUSE binding
  - command line applications (vls, vcp, vmkdir, vrm, vchmod, vmv, vln)

Mounted FUSE VOSpace makes Random Access Calls to local cache

Cache makes HTTP Range requests to CADC storage services (endpoints from VOSpace) to seek and read

Don’t need to download whole file before working

CANFAR vos on GitHub: https://github.com/canfar

Access Control in VOSpace v2.1 and beyond

Echoing Paul on the UWS 1.1 discussion page: Should we say more about access control?

- IVOA document on Single Sign-On is good, but needs updating (2008).
- Access Control makes integration, interoperability tough. Don't underestimate it!
- Authentication: Maybe a good place start for VO standards
  - OpenID Connect: allows organizations to manage their own users and be able to authenticate against other OpenID enabled services.
- Authorization: Think that we can allow VO implementers to enforce and govern their own access control policies.
- Interoperable administration (users changing the policies and permissions): May be difficult to describe in a standard.

Knowing the desired authentication method…

Issue: When creating endpoint (download/upload) URLs, knowing the desired protocol isn't enough. You need to know the desired authentication method too.

Reason:

- You can't assume an authetication method for a protocol
  - HTTPS could use X.509 client certs, userid/password, or cookies
- Endpoint URL may change with knowledge of the auth method
  - E.g.. May have a separate, blocking URL for HTTP Basic Authentication.

Authentication in VOSpace

v2.1 addition: Add the desired ‘authType’ to the transfer object

- An optional element
- Have a standard set, allow it to be extended:
  - Anonymous Access
  - X.509 Client certificates
  - User ID / Password
  - Session based cookie Access
  - More?
- Placement in XML schema to be determined:
  - Multiple authTypes per protocol
  - Can be represented as tuples, or 1-n relationship

Summary of v2.1 Changes

- Addition of optimized HTTP GET/POST method of transfer negotiation for pullFromVoSpace and pushToVoSpace
- Addition of authType to Protocol in XML Schema
- Include a preliminary list of standard authType URIs
- Removed view=data convenience method for data download (already a ‘default’ view, overlaps with transfer functionality)

Still to do in v2.1:

- Add list of VOSpace implementations in the document
- Put the document on Volute, clean up the XML so it is valid
- Add the authType(s) to the capabilities resource

VOSpace in the Future

CADC VOSpace Goals:

- Continue to improve the reliability and performance

Suggestions for IVOA VOSpace/GWS Goals:

- Working server-to-server transfers
- Working Interoperable clients
- Tackle access control in v3.0, and perhaps apply it to other IVOA standards (UWS, DAL, etc.)
- Update Single Sign-On v1.01