- We assert that we have a working system, do not need a new system, but the current system (cost and complexity) exceeds what is needed. Would the experiment agree with this statement?
- What elements are being used, and how? Which have been added and why?
- Which components failed to deliver (larger) functionality needed? Does that mean we can deprecate and obsolete these components, or was it not truly needed?
- If the middleware dropped complexity and removed functionality, does the experiment have the resources to adopt to the change?
- What is the experiment's interest in revising data placement models? What kinds of revisions have gone through?
    - What is the experiment's interest in data federations?
- What sort of assumptions does the experiment need to make for data federations work?
- Could you work directly with clustered file systems at smaller sites?
- Could you work directly with cloud file systems (i.e., GET/PUT)? Assume "cloud file systems" implies REST-like APIs, transfer via HTTP, no random access within files, no third-party transfer, and possibly no file/directory structure. See Amazon S3 for inspiration.
- How thoroughly does the experiment use space management today?
- Does your framework support HTTP-based access?
- Is the archive / disk split an agreed-upon strategy in your experiment? Can HSM be dropped?
- What role do you see for Data Federation versus Data Push? Is this useful at smaller sites?
- For smaller sites, would caching work for your experiment?

Notes from chat with Simon:

- File catalogue stuff - metadata is experiment-specific, but there seems to be lots of common ground.
    - Problem with LFC - "just" tracks file location, but not physics quantities.
    - Most comparable to ALICE, although we have two separate catalogs (one for locations, one for physics metadata).
        - Sometimes we have to do some gnarly work to synchronize major changes (site renames, file invalidation from dataset).
    - Nothing comparable to ATLAS TAG database.
    - Possibly a problem folks have been burnt badly on in the past.
    - Fairly happy with the current model. Physics database is agnostic to file location; file location DB has no knowledge of physics information.
- Space management:
    - We use all that we need. People get tied up about quotas on space, but that really gets complicated. We won't benefit much from of these - prefer to keep things simpler and "soft metric"-based. We'd rather say "about 50TB" than get into the business of micro-managing sites.
        - Micro-managing from the experiment indicates something has gone wrong somewhere.
    - Space tokens doesn't greatly benefit us. Has become a distraction from real problems.
    - Space management is delegated to the sites. If they want to use SRM space tokens to manage space, fine. If they don't, that's fine as long as they run a reasonable site.
    - CMS works just fine with clustered file systems.
        - Would like WLCG to become more of an Apache-like organization. Helps coordinate a philosophy, but not own any software.
        - There's a lot of overlap, and even possibly shared technology (i.e., all WM is python-based).
        - Would even prefer this, as we can share effort with the wider community - not just the "grid world". Would rather contribute to wider community projects, than do our own.
    - Want good site involvement - this is a good example where experiments had too much say, and went overboard.
- The SRM Question:

        - Load-balancing data access, metadata access, integrate with FTS. Storage tokens are site-specific, only done locally.
        - Would have no issues with splitting archive from disk. Convenient to have files written to disk eventually get to tape; tape recalls are now being separately managed. Scheduled access, no need fro free-for-all access. Need to verify files are on tape.
        - WAN transfer protocols: GridFTP, FDT, HTTP.
            - No requests in sight for xrootd-based bulk transfer.
            - FDT use is mostly Caltech-driven; not a huge amount of support, very much a proprietary, thing-we-grew-ourselves product - we want to get away from this, not a huge benefit over what exists.
            - Would be interested in seeing GridFTP load-balance, either via FTS or at the network level. Load-balancing GridFTP is one of the big things we use SRM for.
            - Mostly interest is based on what can be done in batch via FTS.
            - Message: We need a reliable interface for transferring files and checking metadata. Don't really care strongly about the transfer protocol.
- Non-monarc data model for the past 6 years or so, very much in the last 2 years. Quality of the transfers mostly depends on quality of sites (ie., if transfers go through firewall, you are dead). Almost never a function of the network. Needs a decent FTS admin.
- We think "cloud file systems" really just mean "web servers". We'd really like to see more use of HTTP-based access. Translation is done in-site. This is where SRM should be evolving.
        - It's "known" how to do caching and scaling horizontally with HTTP.
        - We'd possibly take a big hit with no third-party-transfers. Depends on how FTS manages it.
        - Would likely still want the file/directory structure.
        - Framework supports HTTP access (theoretically). Would take work to get it into shape.
- Agree that the system works, but is over-complicated. SEs are likely the most complicated piece. From experiment POV, FTS is very good. From site POV, FTS is very complicated.
        - PhEDEx appears to be very expensive, but is a small fraction of operational cost. Running of PhEDEx is fairly equivalent to running a good SE (i.e., making sure files are valid).
        - A lot of the burden falls on sites; want strong input from them.
- Data federation:
        - Real question: How does file access change over time? Depends heavily on the proportion of read speeds to network speed.
        - Federation is useful as a fallback, or secondary access. Individual user access is also popular. Overflow access.
        - Caching is interesting, but it's a hard one to do well. Don't see a huge benefit in doing this for any workflow-management-based job, as the WM knows a lot about the system and can push the data appropriately. Benefit is possibly in the interactive case.
            - If the whole point is to reduce the system's complexity, not clear this is a winner. Amount of knowledge needed to manage the cache well might be equivalent to space tokens.
            - Cache stuff becomes interesting at the personal level - if you think about it, you might have more TB and more reliability on your desktop than at the remote site where CMS has given you a quota. Possibly have more locality, but not for opportunistic resource.
            - Maybe useful at T3s? For opportunistic sites, probably useful. Definitely for sharing data.
            - Analysis use case is where it's interesting.

By Brian.