Some notes on Computing Model evolution/tests/ expectations for HL-LHC

Infrastructure

Even if the time span from now to HL-LHC (~ 10 years) can seem a lot, in practice any drastic change to the distributed computing infrastructure should be at least known in the coming -4 years, since there are large implications with national infrastructure, personnel acquisition, and such.
That said, the expectation is that of further diminished importance of “old” Monarc-style definitions: leaving aside T0, which will still be “close” to the DAQ, T1 and T2 differences should fade away, and become of second order importance with respect to real resource deployments at sites:

- Are there CPU available ? → Compute site
    - If so, compute loads can be run, to a certain extent irrespectively of other site characteristics
- Is there Disk available ? → depends on the kind of storage between
    - Is it “long-lasting”, such that experiments can use it for managed / logged transfers?
    - Is it only a self-regulating cache?
- Is there tape / any other form of long term storage available? → Storage site
- Which is the network capability of the site (towards its NREN / as a full matrix with other peering centers)?

In particular, what can stop being a problem is the old rule-of-thumb for sites for N TB/M kHS06, in order to maintain disk and CPU fully utilized at a site.

Clearly, from the CPU utilization point of view, but not only (see later), a complete decoupling from the storage and compute aspects would be beneficial.
This needs to be based on a very solid, dedicated, multi Tbit network connection between a subset of the centres, especially those with lost-term storage available.
This would form the backbone of a data serving system (which incidentally has also computing power, but that is ~ irrelevant at this point), where data to be accessed by compute nodes is stored.
It would be wise to treat this a single entity (a backbone, a virtual data center, a logical unique storage, …) where data is injected, even without direct knowledge of the physical sites, given a series of QoS definitions for the data upon injection.

Computing (i.e., CPUs) should be pluggable to the backbone, and removed on short notice, in order to accommodate as much as possible windows of opportunities for ephemeral resources.
Data should not exit the backbone in a managed way , but be accessed in place via remote protocols or eventually via caches. This requires a very clear manifest of added computing resources, which should present themselves with an enlarged list of capabilities. Today this is mostly limited to # of cores, OS instances, and such. In a detached CPU/data model, the manifest should include network latencies / bandwidth towards the data center (or a few main nodes), and the description of the cache, if any, in front of the CPUs.

Regarding caches, in WLCG there is some experience about site-deployed caches (mostly XrootD-operated). It would be important to test setups in which caches are instead operated by NREN itself, using hep-agnostic protocols like Https. Caches @ NREN would be completely transparent to the experiments, and would work very closely to what, for example, Netflix does with ISPs.
In this way, caching could become a network feature instead of a site tool, and serve potentially more users. Being deployed at the NREN, moreover, would imply additional network bandwidth wrt what is assigned to the site. Another line of development regarding caches should be distributed caches, at multiple NREN PoPs or Compute sites. Assuming national (regional?) bandwidth and latencies will remain more performing than transatlantic ones, a distributed cache can be as effective as a site one, and reduce cache duplications and thus sizes.

A central aspect is about connection of commercial clouds / provider to the backbone. A central (Geant/ESNet driven?) effort should be done, in order not to have all FAs and NREN needing a separate discussion.
It is crucial to set channels, centrally, via which ingress/egress are not part of the monthly bill for LHC computations. Disk would be limited to the local disk of the worked nodes, with outputs safely written from scratch on the backbone.

Processing model

Initial, primary data (from the accelerator, for example), would be injected to the backbone, seen as a logical entity, together with a set of QoS requirements (number of copies, custodiality requirements, access speed requirements, …).
From that point on, the backbone would serve as storage for input / output of all the organized processing activities, which would run on any compatible compute node (see the manifest description).
Input can be mediated by caches, output would not (or at least, not in the simplest model).

Analysis would be different in the sense that potentially the outputs can be put outside the backbone, particularly true in case of very small and specific studies.

In that case, options would be offered for saving to backbone OR university level clusters OR to common services like DropbBox, or shipped back on demand (think of a web server retaining output for a few days - does not even need to be with AAI, uuid in the URL names would be enough).
In all the cases where the data is not ingested in the backbone, no storage bookkeeping would be available, just bare files + eventually a report by the WMS engine.

Analysis model

Analysis models in use today are mostly

- Event loop driven (process this event and then the next, maybe in a different job)
- Either “chaotic” (user submitted) or “scheduled” (production submitted - trains)

Modern techniques like DB-style queries are certainly interesting, but one has to remember that our analysis jobs are maybe not 100% CPU efficient, but nearly so (there is no factor 10 to gain, probably not even a factor 2).
So these methods would guarantee a better throughput only if

1. They have to run less code to perform the same task (difficult if timeis spent mostly on algorithmic code);
2. They are specialized to use new hardware features (like GPU etc);
3. They are faster in (for example) IO CPU (uncompress/compress) events.

The first and the third are probably the best places where to get efficiency squeezed from.

Introducing new hardware for analysis is difficult, since it would require:

- Specialized sites (only for analysis?);
- Specialized code, different from what is run elsewhere.

So, while the idea of specialized analysis “farms” is appealing, we would need to demonstrate its increased throughput with respect to more standard solutions. A real breakthrough would be the demonstration that with a DB like approach ROOT would be faster in streaming / unstreaming data.

Data distribution and retention

The easiest place where to squeeze more efficiency from is by removing data duplication and (hence) data movement.
Today’s picture is

- RAW data stays at CERN + gets moved and saved at Tier-1s (2 copies)
- Analysis level formats are moved / cached to all level of tiers. Number of copies varies as decided by popularity, and is higher for smaller data formats

This still reflects (at least at order 0) a datagrid paradigm, where jobs un where data is local, and hence a correct data distribution is crucial to maintain a good CPU utilization.
In recent times, this is relaxed via remote reads, which still account for a quite low fraction of the total analysis jobs.
As detailed in the first part, a transition to a

- Solid, logically transparent data federation, with WAN-as-LAN connectivity;
- Pluggable storage less / caching compute centers;
- Self describing computing resources, including their connectivity to the virtual data center / to some big sites

Goes in the direction of a complete decoupling Compute vs Storage sites.
Caches can be standard, or distributed, and either a sites or in the infrastructure. In general, prefetching is not a necessity but could be important if some level of ML-based popularity prediction can be inferred from the past. In a sense, a system which:

- Has caches @ NRENs
- Caches are distributed, with many regionally well connected caches serving sites. Cache coherency has to be at distributed level (no duplication in close-by caches).
- Caches work at infrastructure level, are transparent to sites (so sites access the original file, with squid like caching in the middle).
- Caches can ebentually be prefilled with data, following the output of a ML-enabled popularity system

Would be able to limit the bandwidth to the central virtual data center nodes, and insure vicinity between data and compute nodes. It would work in a way very similar to netflix caches at ISP Nodes.

Proposed experimentations

1. Test a Distributed (> 3 sites >) cache with XrootdD / Https
    a. At sites
    b. At NREN
2. Prepare a cache-in-a-box system, which can be deployed anywhere and be instantly part of a distributed cache. Prefetch is not essential but could be a solution if matched to a ML-like system for popularity prediction.