Data staging

* Performance and scalability
	+ Addressed by separate data-staging service (cache-only service)
* Client tools (arcp extension, sync+async)
* Potentially hot files (job hints?)
	+ Cephs no problem
	+ Gpfs? Lustre? Independent disks?

Cache technologies:

* Shared FS
* Independent local disks (NFS)
* Squid?
* Cvmfs ?
* Objectstores - investigate the architecture/workflow/feasibility

Data Management/Rucio

* Proof of concept for Rucio+datadelivery+2ARC-CE clusters
	+ demonstrator
* Use cases:
	+ Single user (small organization) with remote data
	+ HPCs with dedicated-to-user (generic) storage, sharedfs or ceph (http)
	+ Proposal for a new project (data lake/ocean)

Cache SE:

* Publishing contents to Rucio, mostly done - need to be synced with xcache
* Discuss the job brokering and implement in panda
* Qualification taks?

SE with ceph/object store ...

Use cases:

* HPCs without SE
* Small sites
	+ T3s
	+ Diskless T2s

Cache sites:

* Without shared FS - pilot mode - xrootd cache - xcache (analogous to squid)
* ARC cache + pilots - access on WNs - cache on demand - jobs request transfer on delivery service

Suggestion:



arccp contacts ARC-CE that contacts cache to get file, so arccp will be extended to check cache, if not there, will download the file(s).fts way.

Current implementation is asynchronous

* Check packaging for arccp and arccache service only
* New command instead of arccp

SOAP is (more or less?) ready for use

REST we can think out for the future

1. Shared FS
	* ...
2. Without shared FS
	* Copy to cache ...
	* Copy to scratch ...
3. Synchronous
4. Asynchronous
	* Polling, process waiting for i/o - some check status

Service

* A-REX
* Reduced service - dont install arc-ce as arc-ce, just a service to have cache as a service. For e.g. condorCE

Outputs



Simple case without registration

ATLAS case: register as rucio