Caching Strategies
Some Input to Brainstorming discussion

Dirk Duellmann, IT-DSS

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Caching - a quite open definition

- Additional data copy to speed up access via
  - storage with higher access speed
  - storage with smaller access latency
  - additional data paths
- Many people (including myself) would usually use “Transparently added data copy …”
  - which means that the cache black-box maintains the copy lifetime implicitly
- but in this discussion we wanted to include popularity based data placement, which is not really happening that transparently…

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Caching Strategies

- Are being discussed since many years and progress has been steady, but not rapid:
  - Discussion and benefit analysis is complicated by
    - missing / changing / differing access pattern knowledge
    - different cache layers affecting/hiding each other
      - in a way which is rather opaque to end-users, service providers and framework developers
- In-process cache (TTreeCache) has an enormous benefit
  - to reduce # of round-trips and repeat reads in a single process via protocol independent, pre-calculated vector-reads
  - … and hence invalidated previous assumptions/optimisations
- Second biggest change (only enabled by above!) is likely federated access
  - which needs to be coherently integrated/evaluated wrt caching
  - remote reading infrequently used data is more effective than attempting to cache

DSS

Some existing or planned cache components

CERN IT Department

Where	What	Why	Who	How	Size	Lifetime	Accessed
Disk Server	FS cache	reduce repeated disk IO	OS/VM	pull	GB RAM	hours	kHz
Site (managed)	File Placement (SE + Catalog)	push popular data to avoid transfer I/O wait	content: exp storage: site	push	10-100 TB (disk)	months	10-100Hz
Site (unmanaged)	Proxy/CDN (eg SQUID, Xroot proxy, {Event Proxy})	reduce latency for repeat reads increase bandwitdh via tree hierarchy	storage: site optionally: exp push	pull	10TB??	weeks/months	10-100Hz
	may come with file/block/{event} granule - efficiency depends on popular fraction of cache granule						
Worker Node	Async read-ahead	increase CPU/IO overlap	job	async pull	GB (RAM)	job lifetime	<Hz
	persistent version of above	reduce repeat reads between jobs (eg user laptop case)	user	pull	10 GB (disk)	weeks?	<Hz
	FS cache for file:// access or WN download	reduce repeated disk/net IO	OS/VM	pull	GB RAM	hours	100 Hz
Process	TTreeCache	reduce network/disk round-trips	root + exp framework	pull	10-100 GB (RAM)	job lifetime	<Hz
	usage currently different between experiments and partially implemented in exp frameworks						

Ideally we would look at this with an overall throughput-increase/$ perspective - but we still miss a lot of analytics to get there

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

CERN

Some papers

- CHEP 10 - The Use of Proxy Caches for File Access in a Multi-Tier Grid Environment
  - describes proxy cache, async read-ahead and persistent read-ahead cache and first tests between CERN and BNL
- CHEP 13 - CMS/AAA paper - Xrootd, disk-based caching proxy for optimisation of data-access, data-placement and data-replication
  - more detailed analysis of new caching proxy with monitoring results from CMS xroot monitoring

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it