The art of the cache

Axel Faust, Acosix GmbH

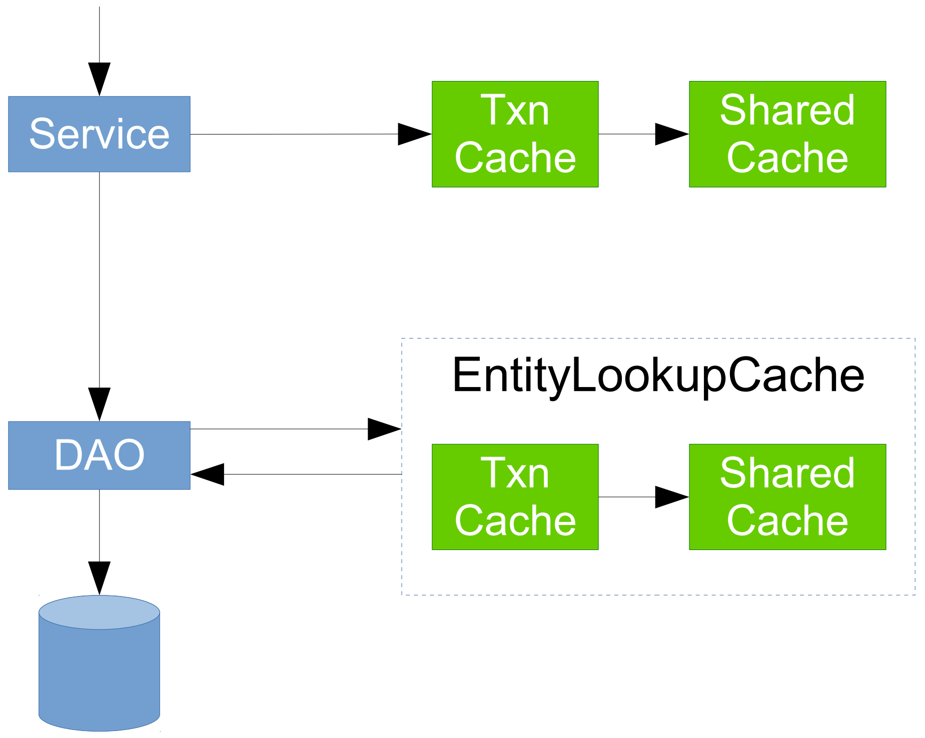
Why talk about caches?

- Everyone wants “performance”

- Little documentation around
  - Configuration options
  - Tuning guides without explanation

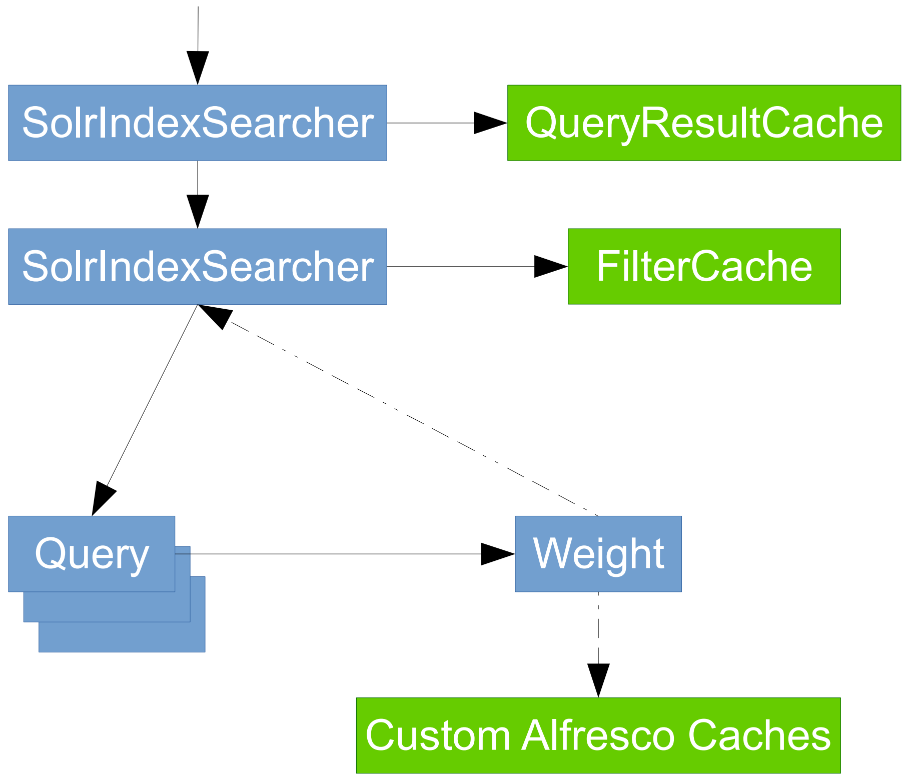
- “Art” is proper understanding

Focus: Repository caching



- Multi-layered caching
  - Transaction vs. global
  - Default vs. custom caches
- Technically "pluggable"
  - Declared via config
  - Generic API
- EE-only: distributed data

Focus: SOLR caching



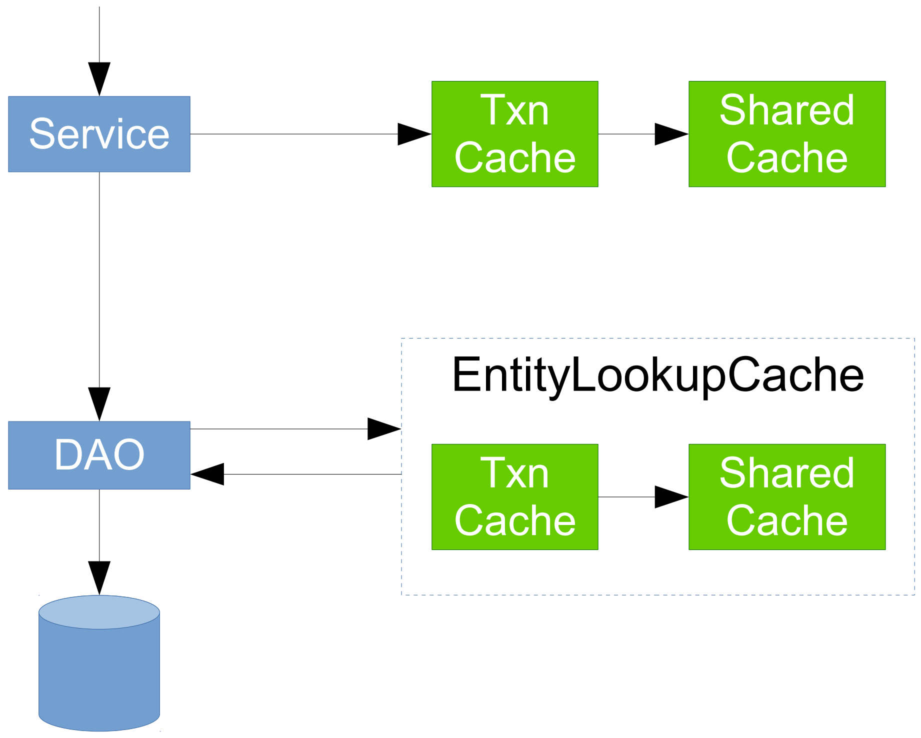
- Multi-layered caching
  - Sub-query recursion
  - + low-level caches
- Easy to understand?
  - Conditional use
  - Indirect relevance
- Technically “pluggable”

Out of scope

- HTTP caching
  - Client (browser) or HTTP proxy (Apache / nginx)
  - Custom web scripts: use “cache” root object
- Surf caching
  - Model objects (pages, components…)
  - Read-only requests (per Surf request)
- Content caching
  - E.g. CachingContentStore

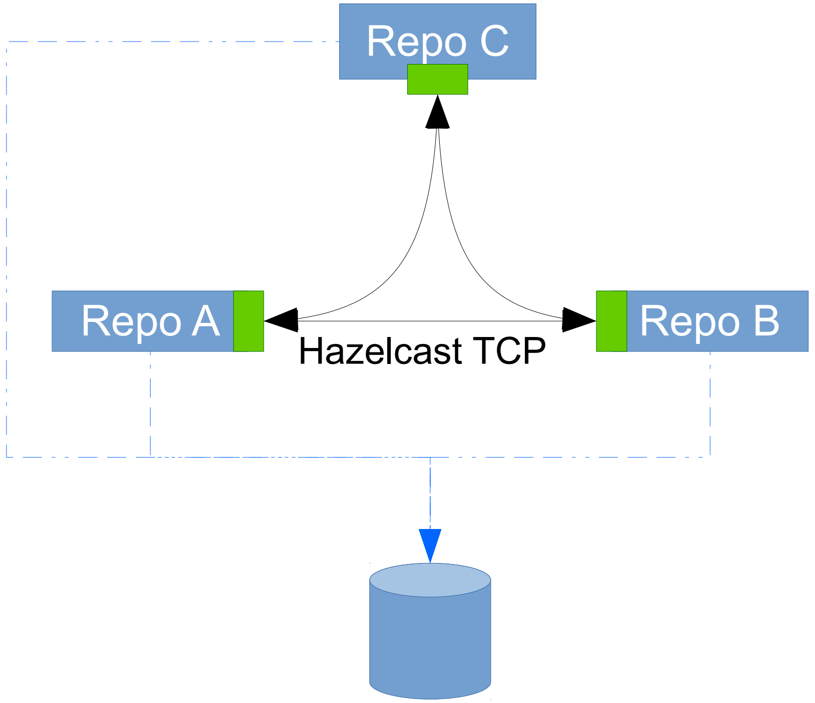
Repository Caches

Components



- SimpleCache (base API)
  - 3 base impl. + Hazelcast (EE)
- TransactionalCache
  - Txn-aware facade
  - Thread-local data until commit
- EntityLookupCache
  - Common DAO logic
  - Callback for specifics

Caching in a cluster (Enterprise)



- Better: “cluster due to caching”
- Mode 1: Fully-distributed
  - 100% Hazelcast
  - Distributed read / write (incl. backups)
- Mode 2: Invalidating
  - Local cache + asynch invalidation
  - Best-effort consistency

Cache definition (Spring beans)

```
<bean name="myAcmeSharedCache" factory-bean="cacheFactory" factory-method="createCache">
		<constructor-arg value="cache.myAcmeCache"/>
</bean>

<bean name="myAcmeCache" class="org.alfresco.repo.cache.TransactionalCache">
		<property name="sharedCache" ref="myAcmeSharedCache" />
		<property name="name" value="com.acme.myCache" />
		<property name="maxCacheSize" value="${cache.myAcmeCache.tx.maxItems}" />
		<property name="mutable" value="true" />
		<property name="disableSharedCache" value="${system.cache.disableMutableSharedCaches}" />
		<property name="tenantAware" value="false" />
		<property name="cacheStats" ref="cacheStatistics"/>
		<property name="cacheStatsEnabled" value="${cache.myAcmeCache.tx.statsEnabled}"/>
</bean>
```

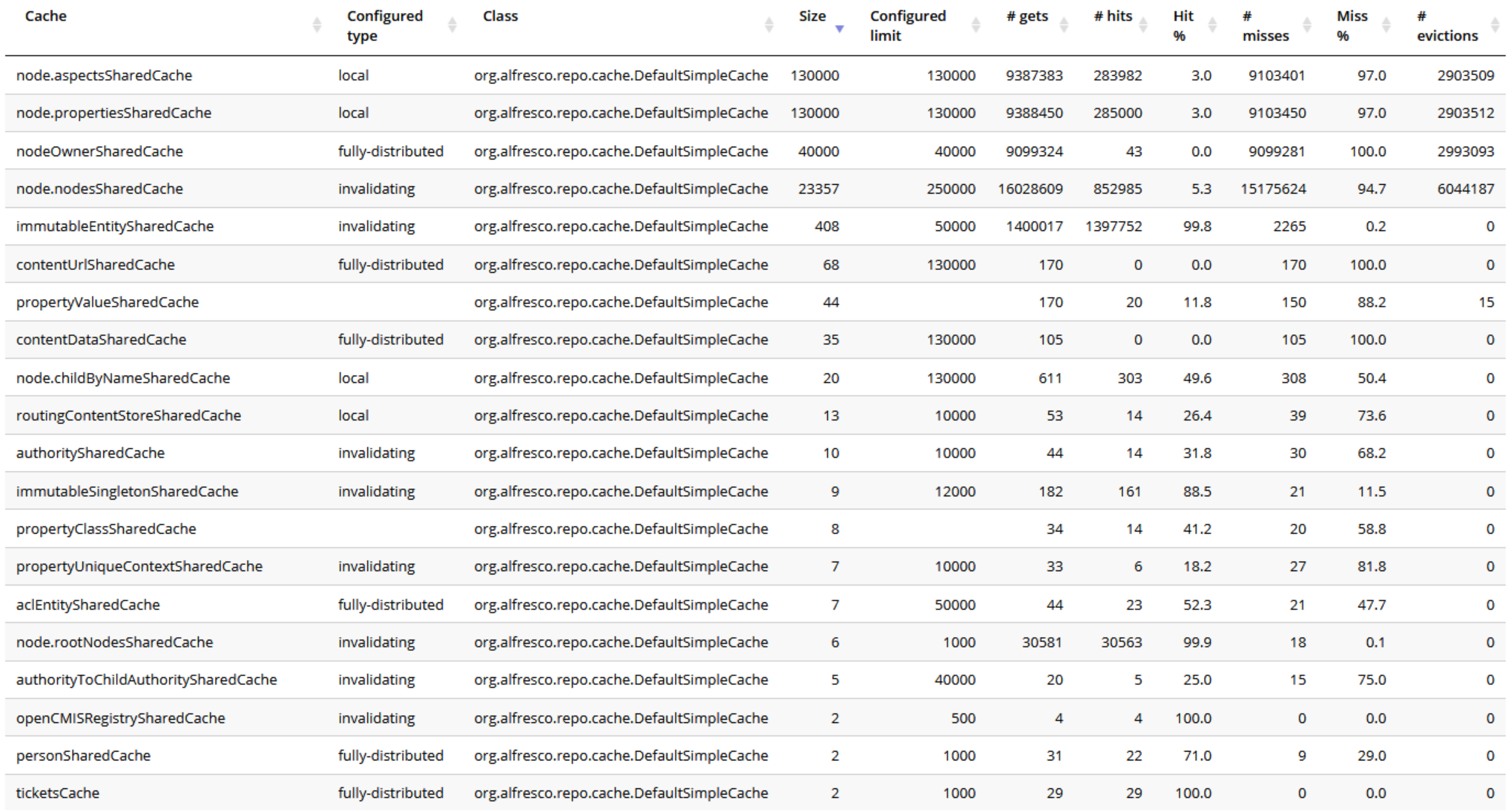
- Generic pattern – simple to abstract, e.g. via bean factory / emitter

Cache definition (properties)

```
cache.myAcmeCache.tx.maxItems=1000
cache.myAcmeCache.tx.statsEnabled=${caches.tx.statsEnabled}
cache.myAcmeCache.maxItems=10000
cache.myAcmeCache.timeToLiveSeconds=300
cache.myAcmeCache.maxIdleSeconds=0
cache.myAcmeCache.eviction-policy=LRU
cache.myAcmeCache.cluster.type=invalidating
cache.myAcmeCache.backup-count=1
cache.myAcmeCache.eviction-percentage=25
cache.myAcmeCache.merge-policy=hz.ADD_NEW_ENTRY
cache.myAcmeCache.readBackupData=false
```

- Enterprise: “cluster.type” + below
- Community: “eviction-policy” != NONE => maxItems respected

What caches exist? (default: 52 in 5.2)



Cache	Configured type	Class	Size	Configured limit	# gets	# hits	Hit %	# misses	Miss %	# evictions
node.aspectsSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	130000	130000	9387383	283982	3.0	9103401	97.0	2903509
node.propertiesSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	130000	130000	9388450	285000	3.0	9103450	97.0	2903512
nodeOwnerSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	40000	40000	9099324	43	0.0	9099281	100.0	2993093
node.nodesSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	23357	250000	16028609	852985	5.3	15175624	94.7	6044187
immutableEntitySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	408	50000	1400017	1397752	99.8	2265	0.2	0
contentUrlSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	68	130000	170	0	0.0	170	100.0	0
propertyValueSharedCache		org.alfresco.repo.cache.DefaultSimpleCache	44		170	20	11.8	150	88.2	15
contentDataSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	35	130000	105	0	0.0	105	100.0	0
node.childByNameSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	20	130000	611	303	49.6	308	50.4	0
routingContentStoreSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	13	10000	53	14	26.4	39	73.6	0
authoritySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	10	10000	44	14	31.8	30	68.2	0
immutableSingletonSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	9	12000	182	161	88.5	21	11.5	0
propertyClassSharedCache		org.alfresco.repo.cache.DefaultSimpleCache	8		34	14	41.2	20	58.8	0
propertyUniqueContextSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	7	10000	33	6	18.2	27	81.8	0
aclEntitySharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	7	50000	44	23	52.3	21	47.7	0
node.rootNodesSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	6	1000	30581	30563	99.9	18	0.1	0
authorityToChildAuthoritySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	5	40000	20	5	25.0	15	75.0	0
openCMISRegistrySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	2	500	4	4	100.0	0	0.0	0
personSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	2	1000	31	22	71.0	9	29.0	0
ticketsCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	2	1000	29	29	100.0	0	0.0	0

Impact of caches

- Searches: node.*Cache(s)
  - Example SOLR benchmark: 312 ms (1st run) vs. 135 ms (2nd run)
- Memory footprint: immutableEntitySharedCache
  - Namespace, QName, mimetypes et al
  - Saves e.g. <~ 350 byte per QName re-use
- User-group lookups: userToAuthoritySharedCache
  - Avoid recursive DB lookup over parent group tree

Using custom caches

- DAO: stick to EntityLookupCache

- Checklist questionnaire
  - Is caching really a necessity?
    - Miss cost vs. memory cost
  - Are types of keys / values suitable?
  - How to detect / avoid stale data?
    - Versioned key vs. invalidation by behaviour / scheduled action

Repository Caches Tuning

Understanding the issue

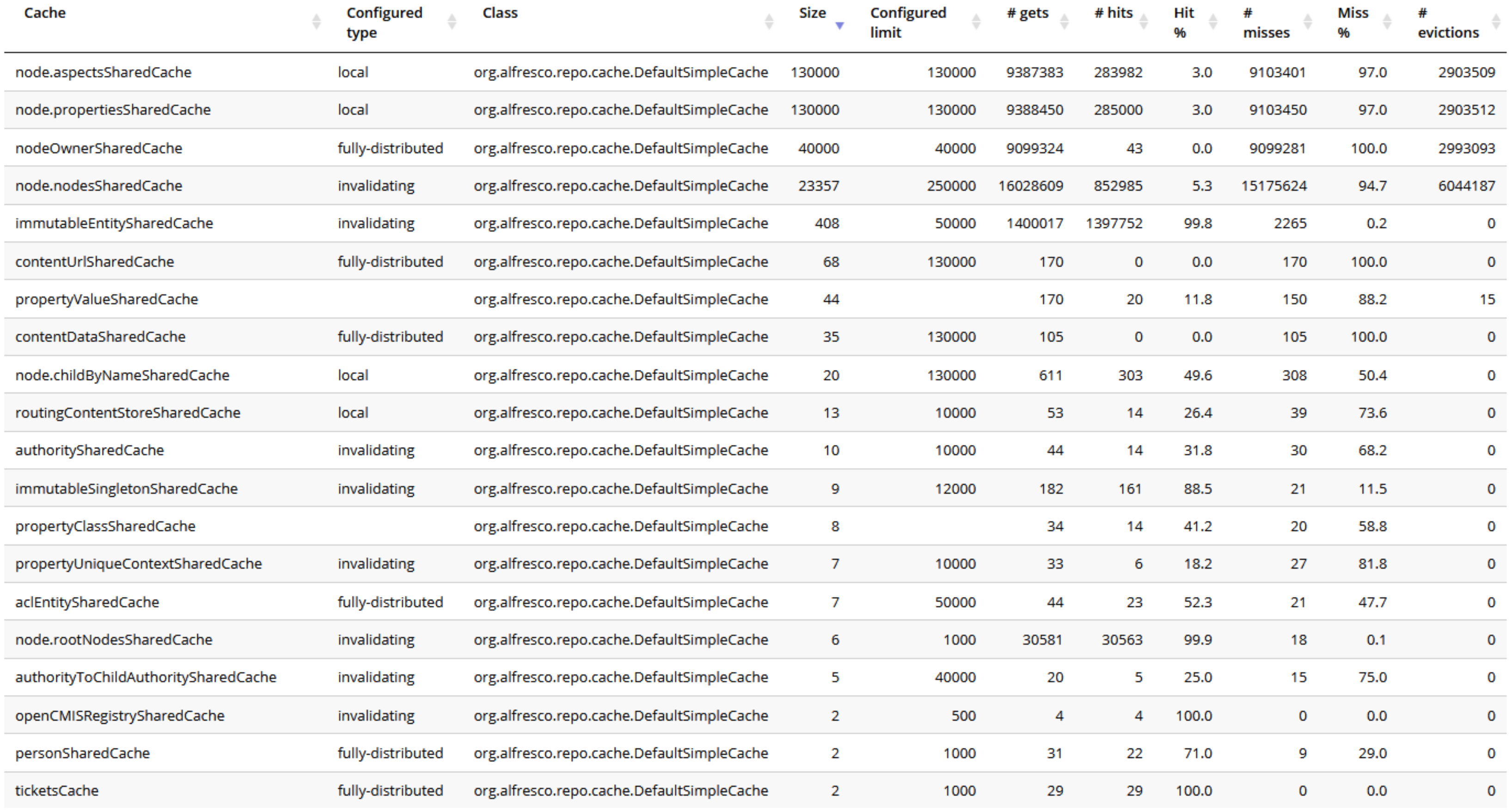
“Tuning the caches in a wrong way may lead to out of memory issues.”

- Sources of information
  - Logs: “Transactional update cache '[name]' is full ([limit])”
  - Monitoring: JVM heap / garbage collection
  - Code: cache usage patterns
  - Tuning guides: be wary of source / age

- No cache insights in default Alfresco

  => OOTBee Support Tools

Understanding the issue



Cache	Configured type	Class	Size	Configured limit	# gets	# hits	Hit %	# misses	Miss %	# evictions
node.aspectsSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	130000	130000	9387383	283982	3.0	9103401	97.0	2903509
node.propertiesSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	130000	130000	9388450	285000	3.0	9103450	97.0	2903512
nodeOwnerSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	40000	40000	9099324	43	0.0	9099281	100.0	2993093
node.nodesSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	23357	250000	16028609	852985	5.3	15175624	94.7	6044187
immutableEntitySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	408	50000	1400017	1397752	99.8	2265	0.2	0
contentUrlSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	68	130000	170	0	0.0	170	100.0	0
propertyValueSharedCache		org.alfresco.repo.cache.DefaultSimpleCache	44		170	20	11.8	150	88.2	15
contentDataSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	35	130000	105	0	0.0	105	100.0	0
node.childByNameSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	20	130000	611	303	49.6	308	50.4	0
routingContentStoreSharedCache	local	org.alfresco.repo.cache.DefaultSimpleCache	13	10000	53	14	26.4	39	73.6	0
authoritySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	10	10000	44	14	31.8	30	68.2	0
immutableSingletonSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	9	12000	182	161	88.5	21	11.5	0
propertyClassSharedCache		org.alfresco.repo.cache.DefaultSimpleCache	8		34	14	41.2	20	58.8	0
propertyUniqueContextSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	7	10000	33	6	18.2	27	81.8	0
aclEntitySharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	7	50000	44	23	52.3	21	47.7	0
node.rootNodesSharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	6	1000	30581	30563	99.9	18	0.1	0
authorityToChildAuthoritySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	5	40000	20	5	25.0	15	75.0	0
openCMISRegistrySharedCache	invalidating	org.alfresco.repo.cache.DefaultSimpleCache	2	500	4	4	100.0	0	0.0	0
personSharedCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	2	1000	31	22	71.0	9	29.0	0
ticketsCache	fully-distributed	org.alfresco.repo.cache.DefaultSimpleCache	2	1000	29	29	100.0	0	0.0	0

“Transactional update cache is full”

- Cause: too many read/updated or removed entries
  - put() trigger: #uniqueKeys(read + update) >= #max
  - remove() trigger: #keys(remove) >= #max

- Quick & dirty: cache.XXX.tx.maxItems
- Ideal: analyse + fix use case
  - Smaller batch sizes / scope of work
  - Single txn recursion => multi-txn batch processing

Shared cache optimisations

- Sizing considerations
  - Fit in JVM heap / avoid GC pressure (use adaptive GC!)
  - Support X% of total DB entries
  - Support Y minutes worth of K-times user access repetition
    - Indicator 1: evictions per Y minutes <= cache size
    - Indicator 2: hit count per Y minutes >= K * cache size

- Option: pre-load expected data sets
  - Asynch txn post-processing / scheduled process
  - Based on business metadata, access or query logs

Estimating memory consumption

- Recommendation: load realistic data sets + analyse heap dump
  - Too many factors for generic formulas
  - Highly dependent on model / user data

- Personal experience

node.nodesSharedCache:	#entries * ~700 byte
node.popertiesSharedCache:	#entries * avg(#propsPerNode) * ~100 byte
node.aspectsSharedCache:	#entries * (~150 byte + avg(#aspectsPerNode) * 32 byte)
content*SharedCache:	#entries * ~150 byte
General overhead:	#entries * ~125 byte

Extreme: custom implementation

- Example: apache Ignite-backed module
  - Off-heap data storage = less GC pressure
  - Distributed cache, compute and messaging

- Approach
  - Implement CacheFactory / SimpleCache
  - Deal with “weird” Alfresco cache use-cases
    - E.g. immutableSingletonSharedCache, globalConfigSharedCache...
  - Optional: use original cache properties as default

Annoying cache bits

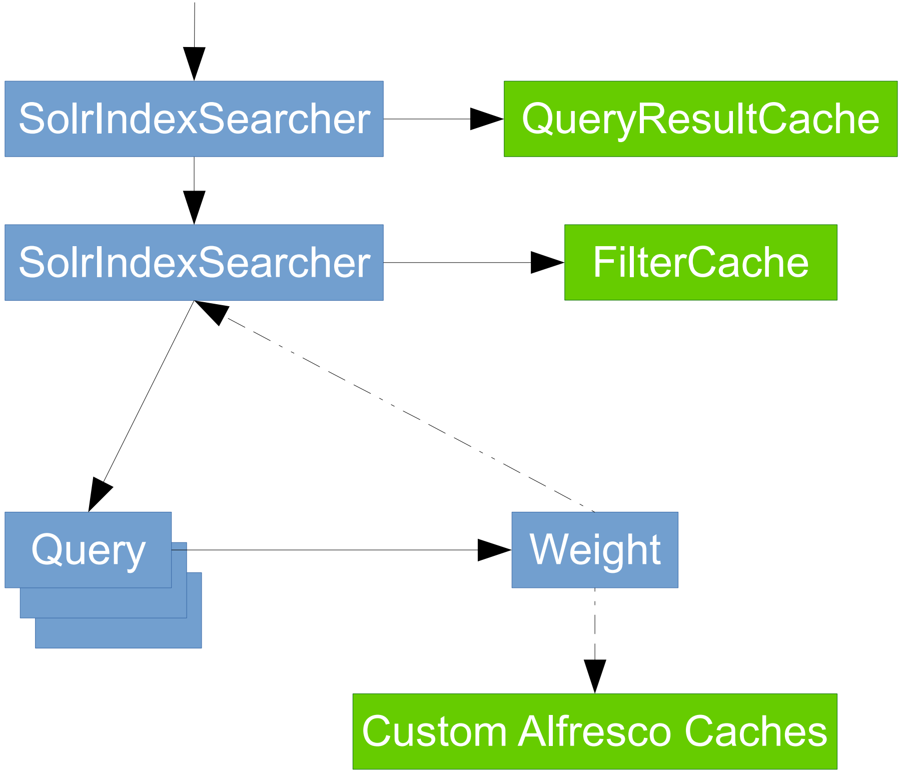
- TTL on node.nodesSharedCache unnecessary
  - Default 300s quite aggressive
  - Significant for bulk-fetch, e.g. during search

- Obsolete Hazelcast version (2.4 vs. 3.8)

- Non-standard cache: parentAssocsCache
  - Not measurable
  - Limited configurability – only maxSize + “limitFactor”

SOLR Caches

Caches in use



- SOLR default caches
  - queryResultCache
  - filterCache
  - documentCache (hightlight/tracking)
  - fieldValueCache (facets)
- Alfresco default caches
  - denied/authorityCache (ACL checks)
  - reader/ownerCache (ACL checks - indirect)
  - pathCache

Cache relevance

- Unscientific comparisons (4m docs, SSD)

– PROPERTIES:”<qname>”:	~35ms vs. 1ms
– PATH:”<path>”:	~1s vs. 1ms
– Facets of unique-valued field:	~750ms vs ~150ms

- Sort by score prevents some cache use

- Typical cache entry bytes: ~ #index docs / 8 (+key)
  - QueryResultCache: ~ #resultsInSlice * 4

Understanding “auto-warm”

- Cache data “refresh”

- Default limits quite low
  - 4 authority and 32 path sub-queries, 32 filter queries
  - 4 complete result sets

- Option: reject or off-load complex / costly entries
  - Interface for custom code: CacheRegenerator

Cache tuning

- Typical focus: high hit ratio – low warmupTime
  - Problem: layering + arbitrary queries skew metrics

- General priority: I/O caches + throughput

- Different system = different approach
  - No warmup if update frequency high
  - Increased cache size if used queries granular / re-usable
  - Custom regenerator for expensive queries – or no warmup

“The art of the cache”

Levels

- Basic
  - Understand default behavior
  - Able to judge metrics + tune accordingly
    (filter out the noise / bad advice)

- Advanced
  - Build custom solutions utilising custom caches
  - Implement tweaks / hooks

- ...go “wild” beyond that

@ReluctantBird83

axel.faust@acosix.org