VCS Unified row cache performance review

Benoit Delbosc

2012-07-10

1 Context

The VCS SoftRef row cache is suspected to behave poorly on load because:

- The cache is bound to each db connection. There can be up to `nuxeo.vcs.max-pool-size` caches which is bad for hit ratio and memory footprint.
- When there is a peak of activity new VCS connection are created with an empty cache, generating lots of db hit and the overall performance falls off.
- A rollback clears the cache with the same bad effect.
- Under memory pressure cache are flushed because softref are managed by the GC, this gives a bad hit ratio and generate lots of GC activities.

NXP-9574 introduces pluggable cache to replace the default SoftRef implementation.

The most simple implementation is a unified cache shared between connection using ehcache. This should improve the hit ratio and have a smaller memory footprint.

2 Daily CI benchmark

The hit ratio can be known using the javasimon counters that are exposed via JMX (NXP-9380). The daily CI bench logs the hit ratio into the `log/misc-end.txt` file. [1]

By running the reader bench done by the CI using the different cache:

- no cache [1]
- default SoftRef cache [2]
- Unified cache [3]

Here are the numbers:

benchmark	Req/s	SQL queries	SQL time (s)	hit ratio %	cache size	GC time (s)	Full GC	blocked count	blocked (s)
no cache	20.6	1449518	425			108	33	57543	1085
softRef	23.5	236538	359	96.97	54264	121	41	64150	1226
unified	23.5	192404	319	99.89	89396	112	35	70992	1484

[1] no cache online reports: index monitoring funkload report
[2] softRef cache online reports: index monitoring funkload report
[3] unified cache online reports: index monitoring funkload report

This shows that:

- the number of request decrease by 18% with a unified cache
- There is less GC in unified than in softRef
- There is a bit more JVM contention with the unified cache because ehcache access is synchronized
- The hit ratio is very high even with softref because the cache is mostly hit by the tree rendering.
- There are no significant performance gain, because:
  - there is no network latency
  - the default CI bench is CPU bound

3 Benchmark under memory pressure

While it is hard to find the right benchmark that correlate a better hitratio with a significant throughput improvement. It is easy to compare bench under memory perssure to show performance gain

Here is a simple bench navigation on folder with 5k document with 1g JVM heap comparing softRef [4] and unified [5] cache.

benchmark	Req/s	SQL queries	SQL time (s)	hit ratio %	cache size	blocked count	blocked (s)
softRef	25.3	360454	877	87.02%	284671	38978	182
unified	31.5	238230	1167	100.00%	1190708	171081	1432

This shows that:

- unified cache handles **24% more throughput**
- unified cache avoid the fall-off after the throughput peak (Figure 1) because softref cache are much more overhelm by memory pressure (Figure 2)

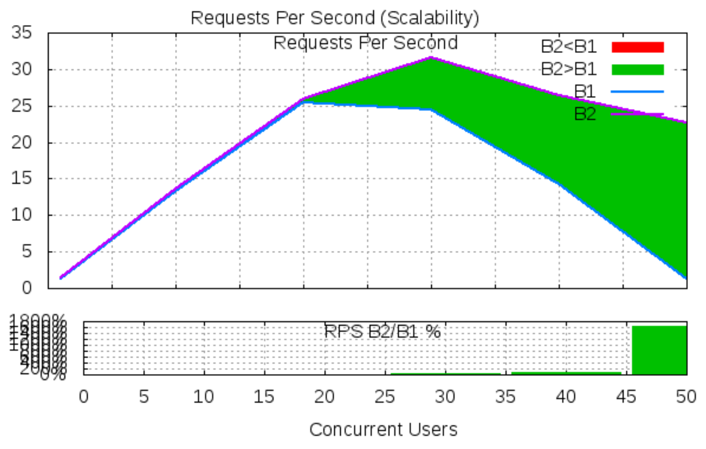

Figure 1: Under memory pressure softref fall-off, B1: softref B2: unified

[4]softRef cache online reports: monitoring funkload report
[5]unified cache online reports: monitoring funkload report

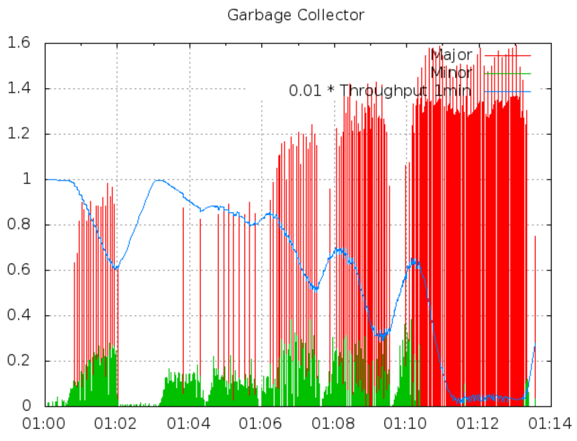

Figure 2: GC overhead with softref cache

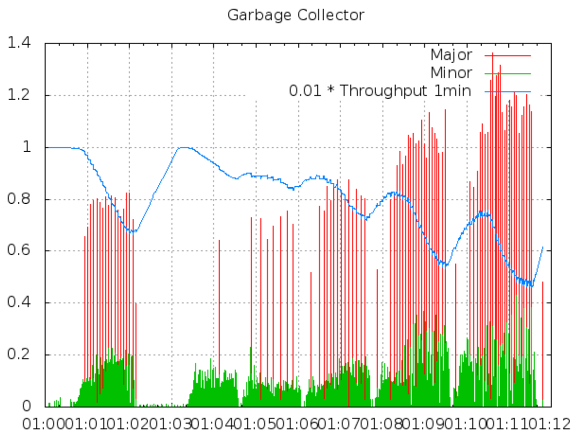

Figure 3: GC with unified cache

4 Conclusion

The unified cache works better on memory bound application, it may help also on remote database with network latency (cloud case) by reducing the number of requests.