Performance Analysis for Table Caching (PA4TC)

A pre-POC guide to estimating QuickSelect caching benefits

The best way estimate QuickSelect CPU and IO savings is to run QuickSelect in SURVEY mode in your production environment. There may, however, be situations where you need to an estimate of potential QuickSelect savings before you bring the QuickSelect in-house. This is best accomplished by analyzing performance data generated by commonly used Db2 performance monitors including CA Detector® for Db2, BMC AMI Apptune®, IBM Db2 Query Monitor® and Syncsort® Optimize Db2.

A Db2 performance monitor can be used not just to collect, but also to aggregate performance data by SQL – the starting point for this analysis. This paper sets forth a simple approach to using the data these monitors collect to estimate potential QuickSelect savings.

If possible, collect performance data during peak production hours. This is typically the busiest time for Db2 applications as well as the period that drives your peak four-hour rolling average setting the monthly license charges for your mainframe software. If available, SCRT reports can help pinpoint the prior month's peak giving you a likely target for your collection period. Use your monitor to collect and aggregate the following data to intervals of between 1 to 3 hours:

- CollectionID
- PackageName
- SQL Statement
- SQL Section
- SQL Type (Select, Delete, Insert etc)
- Number of GETPAGE
- Number of executed SQL
- Number of SYNCRD
- CPU percentage spent on the specific SQL
- Elapsed time percentage on the specific SQL

Once you have collected your data and generated your reports, look for FETCH and SELECT SQL statements whose aggregate data has all four of the following characteristics:

1) Consumes significant percentage of CPU during the monitored period
2) Aggregate GETPAGE per SQL Section that is **high**
3) Aggregate number of SQL Calls is **high**
4) Aggregate SYNCRD is **low**

Statements meeting these criteria are **good** candidates for QuickSelect caching for the following reasons:

1) A high CPU percentage indicates good potential ROI
2) A high number of SELECTS or FTECHES means the SQL it is used a lot by the package and that data use may possibly be repetitive
3) A high GETPAGE number can indicate that the data is frequently looked up by DB2 via DB2 Buffers – an indication that the data may be relatively static
4) A low number of SYNCRD means the data was not read from the disk again – and indication that it changes infrequently

Once you have identified good candidates using above criteria, next consider the characteristics of the tables the SQL are accessing to help you narrow the list:

1) If the table (or tables) a candidate SQL is accessing has a very high number of rows (many millions or more), the SQL will remain a good candidate only if the table remains stable for relatively long intervals of time. A loose rule of thumb for QuickSelect: the greater the number of rows in a table, the more stable the data should be.
2) If the average number of accesses per row in a table is high for a candidate SQL, that SQL remains a good candidate. A high number will point to repetitive access to the data – ideal candidate for caching. To determine the average number of accesses per row, divide the number of executions of a candidate SQL by the number of rows in the table. (In practice this number can be misleading as not every row is read the same number of times. There will be certain values that are read many times over again and these are perfect candidates for caching even if the average number of accesses per row is low.)

Once you have narrowed down your list of candidates, a rough estimate of potential CPU savings is 90% of the CPU spent on the above statements. In some cases, the amount saved may be less. In others it may be very close to 100%.

Please refer to the small sample report in Appendix A containing a few SQL statement examples that fit the above criteria. Statements such as these are good candidates for analysis using QuickSelect's SURVEY mode. If your preliminary analysis yields promising results, you may feel justified in installing QuickSelect for a Proof of Concept (POC). Running QuickSelect in SURVEY mode in production will, with minimal overhead during peak times, take the analysis one step further by pointing to repetitive access to the same tables by the same SQL STATEMENT with the same host variables.

APPENDIX A

Sample from a common DB2 Analyzing tool

The lines marked yellow are good candidates for QuickSelect caching

Type	Prog-Collid/Stmt	TimePct	CPUPct	Getpage	SQL Calls	SyncRd
PKGE	QA7CSQ1-ALQAP0CO	1.33%	3.21%		1359386	827
SQL	0001857-SELECT	.72%	1.51%	1317813	441247	304
SQL	0001996-SELECT	.35%	.98%	1318598	443497	141
SQL	0001778-SELECT	.13%	.37%	666768	224673	0
SQL	0001958-SELECT	.11%	.33%	678031	246548	0
SQL	0002072-SELECT	.00%	.00%	9237	3079	376
SQL	0001930-UPDATE	.00%	.00%	822	274	0
SQL	0002154-SELECT	.00%	.00%	204	68	6
PKGE	AS7CEXEN-ALASP0CO	.91%	1.98%		2018707	0
SQL	0000616-FETCH	.73%	1.63%	436712	1489139	0
SQL	0000594-OPEN	.16%	.31%	0	264784	0
SQL	0000892-CLOSE	.00%	.02%	0	264784	0
PKGE	TC9C1500-ALTCP0CO	.42%	1.22%		727635	28
SQL	0001233-SELECT	.42%	1.22%	897432	727635	28
PKGE	PE9C1640-ALPEP0CB	.40%	.05%		39930	32870
SQL	0001372-FETCH	.19%	.02%	26546	13310	14551
SQL	0001426-SELECT	.15%	.01%	23004	6655	11838
SQL	0001087-SELECT	.06%	.01%	19947	6655	6478
SQL	0001341-OPEN	.00%	.00%	538	6655	3
SQL	0001606-CLOSE	.00%	.00%	0	6655	0
PKGE	BG9C7710-ALACP0CO	.32%	.90%		647478	0
SQL	0000279-SELECT	.32%	.90%	530123	647478	0
PKGE	AS9CMVRE-ALASP0CO	.31%	.89%		302723	0
SQL	0000346-SELECT	.20%	.51%	302792	151396	0
SQL	0000369-SELECT	.10%	.38%	302654	151327	0
PKGE	QX2C1DI5-ALAXP0CO	.26%	.54%		224230	0
SQL	0000372-SELECT	.26%	.54%	444508	224230	0
PKGE	AS9CMVED-ALASP0CO	.23%	.63%		679381	0
SQL	0000408-SELECT	.22%	.61%	543308	658193	0
SQL	0000509-SELECT	.00%	.01%	22162	21188	0