Saxon - Bug #4947

Saxon 9.9 performs slower compared to Saxon 9.7 on nested loops

2021-03-24 17:55 - K L

Status:	Closed	**Start date:**	2021-03-24
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	10, trunk
Applies to branch:	10, 9.9, trunk	**Fixed in Maintenance Release:**	10.5

Description

Hi, We are currently using Saxon9.9 PE. Previously, we used Saxon9.7 PE. We have an xquery which has 4 levels of loops (3nested) with where conditions on the for expressions. It has been noted that executing in Saxon 9.9 took much longer than Saxon 9.7 the more there are nested loops.

In the sample xquery attached, with 1000rows of input,

- Saxon 9.7 took an average of 138665ms.
- Saxon 9.9 took an average of 237548ms.

If tested only with 3levels, it was much faster:

- 9.7 at 3335ms
- 9.9 at 5135ms

Based on the profiler, it took more time on the where clause in 9.9 compared to 9.7

Thanks.

History

#1 - 2021-03-28 21:04 - Michael Kay

Execution time for this query in Saxon-EE (not very carefully measured)

```
9.7 - 86.25s
9.9 - 46.26s
10.3 - 55.23s
```

For this kind of query the Saxon-EE optimiser really makes a big difference, as you can see.

There's always going to be some variation between releases: optimisation of complex queries involves a lot of guesswork, and changes will generally benefit some queries at the expense of others. We have a reference set of queries that we study very carefully looking for regression between releases, but it's a very small sample compared with what is encountered "in the wild".

#2 - 2021-04-10 21:14 - Michael Kay

My timings for Saxon-HE:

```
9.7.0.15 - 92.052s
9.9.1.8 - 164.553s
10.3 - 127.545s
```

I'm a little surprised that the join optimization in Saxon-EE isn't making a bigger difference, but I guess we should focus on the difference between releases rather than the difference between editions.

#3 - 2021-04-10 21:36 - Michael Kay

Examining the -explain output, I notice

(a) 9.7 is rewriting the expression count(X)=0 (which occurs repeatedly) to empty(X); this optimisation does not seem to be working in 10.3.

(b) at line 82, both releases are sorting the result of the expression $out_root/sa/a into document order, which is not necessary for the argument of distinct-values()

I can't see any other obvious differences in the execution plans, apart from the rewrite of count(X)=0.

#4 - 2021-04-10 21:43 - Michael Kay

To establish the significance of the count(X)=0 rewrite I modified the query to use empty(X) directly. The 10.3 HE execution time came down to 119.243s, suggesting that this is significant, but doesn't account for the entire discrepancy.

#5 - 2021-04-10 22:02 - Michael Kay

I looked at both the 9.7 and 10.3 runs in JProfiler and there weren't any blatantly obvious differences in the profiles.

In both profiles, Closure.saveContext() shows up rather more than we would usually expect: this is spending its time saving a copy of local variables to support lazy evaluation.

#6 - 2021-04-10 22:22 - Michael Kay

I ran both 9.7HE and 10.3HE with the -TP option, Oddly, under these conditions 9.7 was slower. The execution counts for the different functions were identical, and the relative time in each function was very similar in both cases. The cost is dominated by the function local:out_rowd on line 3, which is executed 98,010 times.

I note that neither release is eliminating the unused position variable $out_rowd_i, which might inhibit further optimisations.

#7 - 2021-04-11 00:16 - Michael Kay

A minor but possibly significant difference between the execution plans is that in 9.7, the evaluation mode for variable $c on line 14 is MAKE_CLOSURE, while in 10.3, it is MAKE_MEMO_CLOSURE.

I'm surprised that neither 9.7 nor 10.3 loop-lifts the let expression at line 14 out of the containing FLWOR expression, since it seems to have no dependencies on the outer clauses.

#8 - 2021-04-11 01:40 - Michael Kay

If we drop the unused variable at $out_rowd_i, the execution time in 10.3 HE comes down to 48.17s. This doesn't answer the question of why the performance regression occurred, but it does illustrate how sensitive the performance can be to tiny details of the optimization plan. Frankly, I'm more interested in identifying optimisation opportunities like this than in explaining differences between releases.

Looking at the explain output, the removal of this variable results in the declaration of $c being loop-lifted.

I'm looking now at the optimisation paths in the debugger. There has been some reorganisation of optimiser logic between releases, and some optimisations are now present in the EE optimiser only: this is why count(X)=0 is no longer rewritten as empty(X).

#9 - 2021-04-11 01:47 - Michael Kay

I'm going to test and commit the code to drop the position variable if it is unused, and then close this bug.

The change appears to reduce the elapsed time of the query to around 22s.

We sometimes get a query like this where there are so many optimisation opportunities, it's almost inevitable that different releases will handle the optimisation differently, and there's no real way of knowing which optimisations are going to be the most useful. It's a great discovery that removing an unused positional variable can make such a difference, but it's not going to affect many users, because unused variables don't arise that often.

Almost certainly the query could be speeded up significantly by adding type declarations to the parameters of functions. There are also expressions in the query that are clearly wrong, like @xsi:nil['true'] where almost certainly @xsi:nil[.='true'] was intended. So there's probably not that much benefit in further analysis of the 9.7/10.3 differences.

#10 - 2021-04-11 10:16 - Michael Kay

- Tracker changed from Support to Bug

- Subject changed from Saxon 99 performs slower compared to Saxon 97 on nested loops to Saxon 9.9 performs slower compared to Saxon 9.7 on nested loops

- Status changed from New to Resolved

- Assignee set to Michael Kay

- Applies to branch 10, trunk added

- Fix Committed on Branch 10, trunk added

#11 - 2021-04-14 17:50 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 10.5 added

Bug fix applied to Saxon 10.5 maintenance release.

Files

test4Loops.xquery	3.14 KB	2021-03-24	K L
test_xml_1k.txt	60.6 KB	2021-03-24	K L