BRENT OZAR
UNLIMITED®

Lab 2: Finding the Right (Wrong) Queries to Tune

Restore your Stack Overflow database now to start from scratch. There's a SQL Agent job to do the restore. Right-click on that and start it. Takes about 15 minutes.

1.6 p1

Earlier, I spoon-fed you.

I told you what to tune,
and problems to expect.

Feeling overwhelmed?
No problem! Let's take
it down a notch. Tune:
mqt_Lab2_Level1

1.6 p2

Afternoon lab setup

In the afternoon lab, you'll have 3 choices:

1. Tune a single query, or
2. Run a load test against your lab SQL Server, then use sp_BlitzCache to figure out which queries to tune, then tune 1, or
3. Run sp_BlitzCache against your live production server to figure out which queries to tune

If you want to do #1 or #2:

1. Restart your SQL Server service (clears all stats)
2. Restore your StackOverflow database (Agent job)
3. Copy & run the setup script for Lab 2

And if you want to do #2, the load test,
start SQLQueryStress
with QueryLab2.json

1.6 p5

How I'd budget this hour

Stop SQLQueryStress (so your VM goes faster.) You can restart it later if you want to rerun the loads.

20-30 minutes – sp_BlitzCache & query review: run sp_BlitzCache, poke around in the top resource-using queries, look for queries you can tune and make a difference. Tell me which ones you want to tune, and why.

20-30 minutes – tune 1 query: based on the estimated plan from sp_BlitzCache, change the query/indexes, get the new actual plan. Show before & after in Slack like Lab 1.

How to turn in your homework

Use Imgur.com to show your sp_BlitzCache results, and what you think you want to do with the queries:

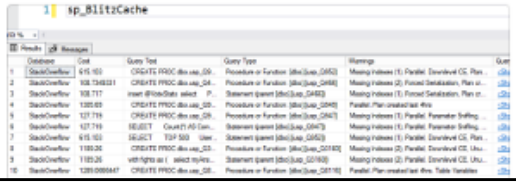
Brent Ozar 7:28 AM

Here are my sp_BlitzCache results: https://imgur.com/a/OfT4Lnb I think usp_Q952 needs to be broken up into a few different queries instead of one big one, but it would take me quite a while to rewrite, and I'm not confident in it. So instead, I'm going to work on usp_Q466 because I think I can make a huge difference with 1 index, plus replace the cursor with a CTE.

imgur.com

Imgur

Post with 0 views. (60 kB)



sp_BlitzCache warnings to ignore

In this lab (but not in real life), you can ignore these:

Downlevel CE: if you're on SQL Server 2019, we're not using that cardinality estimator (CE) yet. We're still using compat level 2017 for this class.

Plan created in the last 4 hours: because this is a short-running load test.

If you do start changing SQL...

This lab doesn't have a clear finish line.

My goal isn't to get you to fix T-SQL.

A lot of the modules in Mastering Query Tuning are fun to revisit: I give you a lot of bad T-SQL.

How the load test code works

1.6 p10

My goals

I want this to be as simple as possible.
I want it all to work on one VM, using off-the-shelf tools that you can use again at home if you want.

I want you to see the moving parts.
I don't want to give you some sealed C# app that you need to recompile or tweak. You're here to performance tune SQL Server, not develop code.

1.6 p11

How I run a workload

SQLQueryStress:

- Open source .NET app that will run 1 query a lot
- https://github.com/ErikEJ/SqlQueryStress

usp_QueryLab2:

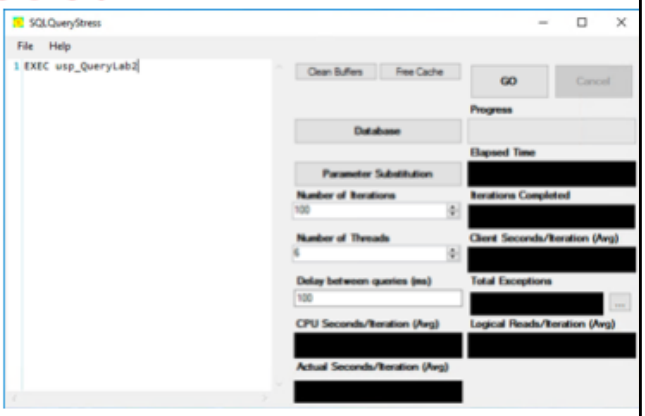
- Stored procedure that generates a random number, then based on that number, will call different proc.
- Don't bother tuning this: it only exists to run procs.
- https://BrentOzar.com/go/stresstest

Start your workload.

Open SQLQueryStress

File, Load Settings, Labs\QueryLab2.json

Click Go



1.6 p13

This is a random load test.

It's okay to see SOME errors, but if your test finishes within seconds and "Iterations Completed" = "Total Exceptions", there's a setup problem (like the database is still restoring, or you forgot to run a setup script.)

The test doesn't need to finish: our goal here is just to put a bunch of query plans in your cache over the span of a few minutes.

Pseudocode

```
DECLARE @Id INT;
SET @Id = (code to build a random #);

IF @Id % 20 = 0 --if it's divisible by 20
   EXEC usp_QueryEvenNumbers;
ELSE IF @Id % 20 = 19  -- remainder is 19
   EXEC usp_QueryOddNumbers;
```

1.6 p15

Examples of modulo (%)

```
DECLARE @Id INT;
SET @Id = 2;
SELECT @Id % 2;
```

Results | Messages

	(No column name)
1	0

```
DECLARE @Id INT;
SET @Id = 100;
SELECT @Id % 2;
```

Results | Messages

	(No column name)
1	0

```
DECLARE @Id INT;
SET @Id = 99;
SELECT @Id % 2;
```

Results | Messages

	(No column name)
1	1

```
DECLARE @Id INT;
SET @Id = 99;
SELECT @Id % 10;
```

Results | Messages

	(No column name)
1	9

And we reuse the random ID

```
DECLARE @Id INT;
SET @Id = (code to build a random #);

IF @Id % 20 = 0 --if it's divisible by 20
   EXEC usp_QueryEvenNumbers @Id;
ELSE IF @Id % 20 = 19  -- remainder is 19
   EXEC usp_QueryOddNumbers @Id;
```

Because code usually needs inputs

```
CREATE PROC usp_QueryEvenNumbers @Id INT
AS
SELECT *
FROM dbo.Users u
WHERE OwnerUserId = @Id;
```

Using the random @Id means we search for different records, and we don't keep caching the same one.

```sql
ALTER PROC dbo.usp_Lab12 WITH RECOMPILE AS
BEGIN
/* Hi! You can ignore this stored procedure.
   This is used to run different random stored procs as part of your class.
   Don't change this in order to "tune" things.
*/
SET NOCOUNT ON

DECLARE @Id1 INT = CAST(RAND() * 10000000 AS INT) + 1;
DECLARE @Id2 INT = CAST(RAND() * 10000000 AS INT) + 1;
DECLARE @Id3 INT = CAST(RAND() * 10000000 AS INT) + 1;

IF @Id1 % 20 = 0
    EXEC dbo.usp_Q3160 @Id1
ELSE IF @Id1 % 20 = 19
    EXEC dbo.usp_Q36660 @Id1
ELSE IF @Id1 % 20 = 18
    EXEC dbo.usp_Q6772 @Id1
ELSE IF @Id1 % 20 = 17
    EXEC dbo.usp_Q6856 @Id1
ELSE IF @Id1 % 20 = 16
```

1.6 p19

```
1  ALTER PROC dbo.usp_Lab12 WITH RECOMPILE AS
2  BEGIN
3  /* Hi! You can ignore this store      cedure.
4     This is used to run diffe                    as part of your class.
5     Don't change this in orde
6  */
7  SET NOCOUNT ON
8
9  DECLARE @Id1 INT = CAST(RAND                                  ;
10 DECLARE @Id2 INT = CAST(RAND                                  1;
11 DECLARE @Id3 INT = CAST(RAND() * 10000000 AS INT) + 1;
12
13 IF @Id1 % 20 = 0
14     EXEC dbo.usp_Q3160 @Id1
15 ELSE IF @Id1 % 20 = 19
16     EXEC dbo.usp_Q36660 @Id1
17 ELSE IF @Id1 % 20 = 18
18     EXEC dbo.usp_Q6772 @Id1
19 ELSE IF @Id1 % 20 = 17
20     EXEC dbo.usp_Q6856 @Id1
21 ELSE IF @Id1 % 20 = 16
```

RECOMPILE keeps this proc out of your plan cache queries.

1.6 p20

```
1  ALTER PROC d                                  ILE AS
2  BEGIN
3  /* Hi! You                                   cedure.
4     This is                                    om stored procs as part of your class.
5     Don't ch                                   e" things.
6  */
7  SET NOCOUNT
8
9  DECLARE @Id1 IN      AST(RAND() * 10000000 AS INT) + 1;
10 DECLARE @Id2 INT = CAST(RAND() * 10000000 AS INT) + 1;
11 DECLARE @Id3 INT = CAST(RAND() * 10000000 AS INT) + 1;
12
13 IF @Id1 % 20 = 0
14     EXEC dbo.usp_Q3160 @Id1
15 ELSE IF @Id1 % 20 = 19
16     EXEC dbo.usp_Q36660 @Id1
17 ELSE IF @Id1 % 20 = 18
18     EXEC dbo.usp_Q6772 @Id1
19 ELSE IF @Id1 % 20 = 17
20     EXEC dbo.usp_Q6856 @Id1
21 ELSE IF @Id1 % 20 = 16
```

I use a few different random numbers because some procs join to multiple tables.

1.6 p21

```
1  ALTER PROC dbo.usp_Lab12 WITH RECOMPILE AS
2  BEGIN
3  /* Hi! You can ignore this stored procedure.
4     This is used to run different random stored procs as part of your class.
5     Don't change this in order to "tune" things.
6  */
7  SET NOCOUNT ON
8
9  DECLARE                                  10000000 AS INT) + 1;
10 DECLARE                                  10000000 AS INT) + 1;
11 DECLARE                                  10000000 AS INT) + 1;
12
13 IF @Id1 % 20 = 0
14     EXEC dbo.usp_Q3160 @Id1
15 ELSE IF @Id1 % 20 = 19
16     EXEC dbo.usp
17 ELSE IF @Id1 %
18     EXEC dbo.us
19 ELSE IF @Id1 %
20     EXEC dbo.usp_Q6856 @Id1
21 ELSE IF @Id1 % 20 = 16
```

If @Id1 is evenly divisible by 20...

Then go run this first stored proc.

1.6 p22

```
1  ALTER PROC dbo.usp_Lab12 WITH RECOMPILE AS
2  BEGIN
3  /* Hi! You can ignore this stored procedure.
4     This is used to run different random stored procs as part of your class.
5     Don't change this in order to "tune" things.
6  */
7  SET NOCOUNT ON
8
9  DECLARE @Id1 INT = CAST(RAND() * 10000000 AS INT) + 1;
10 DECLARE @Id2 INT [illegible] AS INT) + 1;
11 DECLARE @Id3 INT [illegible] AS INT) + 1;
12
13 IF @Id1 % 20 = 0
14     EXEC dbo.usp_Q31[illegible] @Id1
15 ELSE IF @Id1 % 20 = 19
16     EXEC dbo.usp_Q36660 @Id1
17 ELSE IF @Id1 % 20 = 18
18     EXEC dbo.usp_Q6[illegible]
19 ELSE IF @Id1 % 20 = [illegible]
20     EXEC dbo.usp_Q6[illegible]
21 ELSE IF @Id1 % 20 = [illegible]
```

If not, is the remainder 19?

Then go run this stored proc.

1.6 p23

```
1  ALTER PROC dbo.usp_Lab12 WITH RECOMPILE AS
2  BEGIN
3  /* Hi! You ca
4     This is us                                  rocs as part of your class.
5     Don't char
6  */
7  SET NOCOUNT C
8
9  DECLARE @Id1                                  ) + 1;
10 DECLARE @Id2 INT = CAST(RAND() * 10000000 AS INT) + 1;
11 DECLARE @Id3 INT = CAST(RAND() * 10000000 AS INT) + 1;
12
13 IF @Id1 % 20 = 0
14     EXEC dbo.usp_Q3160 @Id1
15 ELSE IF @Id1 % 20 = 19
16     EXEC dbo.usp_Q36660 @Id1
17 ELSE IF @Id1 % 20 = 18
18     EXEC dbo.usp_Q6772 @Id1
19 ELSE IF @Id1 % 20 = 17
20     EXEC dbo.usp_Q6856 @Id1
21 ELSE IF @Id1 % 20 = 16
```

You can ignore this proc. (And if you want the class to be more fun, don't look at the proc until you've done your work.)

You'll see a lot of other procs and ad-hoc SQL, though. Those are going to be where the performance gains are.

1.6 p24