

## Re: Optimizing massive update to large table

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.server/2004-10/2081.html>

---

**From:** Sean Shanny (*shannyconsulting\_at\_earthlink.net*)

**Date:** 10/16/04

Date: Sat, 16 Oct 2004 00:11:55 -0400

Andrew,

Thank you very much, that is doing the trick.

For your info an Apple X-Raid is a low cost, \$15K for 3.5TB, IDE fibre channel based raid system. We are connected via 2 fibre channels directly to the box. Unfortunately we cannot combine the 14 drives at the hardware level hence the 2 7 drive raid 5 sets that are then striped at the OS level giving us raid 50.

Our db, which is a warehouse, is running in simple mode so I don't think, but could be wrong, that the transaction logs are a big bottle neck. If we need to we can put a fibre switch in and throw another X-Raid system in the mix to allow faster performance for log writing. We should probably move indexes to another disk subsystem as well. To be honest we are extremely pleased with the performance of SQL Server so far. Reports that took 20-25 minutes to run on the old system, exact same hardware, now take 6 minutes, only changes in the code dealt with SQL issues like date and string functions which are specific to DB vendors.

I am have a test system at home, same raid box but less drives and I am getting about 750k updates an hour which I can deal with. :-)

Thanks again.

—sean

On 10/15/04 10:59 PM, in article eLR7gwysEHA.1272@TK2MSFTNGP12.phx.gbl, "Andrew J. Kelly" <sqlmvpnoospam@shadhawk.com> wrote:

> I have no idea what an Apple X-Raid is but striping two Raid 5's at hte OS  
> level does not give me a warm fuzzy feeling<g>. Where is the log file? For  
> optimal performance for larges writes like this you really need it on it's  
> own RAID 1 array. It would help to see the actual DDL for the table  
> including indexes. Are you saying there is no clustered index not even on  
> the PK? Your batches are still probably too high for peak performance with  
> that configuration. Try batches of 10K and if that works move up and see

microsoft.public.sqlserver.server: Re: Optimizing massive update to large table

> *how it goes. Your update statement as is will always update the full amount*  
> *of rows so your batch commit size is useless. You are probably going to*  
> *have to use a subselect with TOP or maybe even temp tables. If you had a*  
> *clustered index on the PK it might be easier but here are some things to*  
> *try.*  
>  
> *Make sure that you have a clustered index on wh\_tmp\_fixed\_locations on the*  
> *ID column.*  
>  
> *CREATE TABLE #temp ([PKID] INT NOT NULL, [location\_key] INT, [content\_key]*  
> *INT, CONSTRAINT ID\_PK PRIMARY KEY ([PKID]) )*  
>  
> *WHILE 1 = 1*  
> *BEGIN*  
> *TRUNCATE TABLE #temp*  
>  
> *INSERT INTO #Temp ([PKID], location\_key, content\_key )*  
> *SELECT TOP 10000 b.ID, b.location\_key, b.content\_key*  
> *FROM wh\_tmp\_fixed\_locations*  
> *ORDER BY b.ID*  
>  
> *IF @@ROWCOUNT = 0*  
> *BREAK*  
>  
> *Update pv*  
> *Set location\_key = s.location\_key,*  
> *content\_key = s.content\_key*  
> *From f\_pageviews pv*  
> *Join #temp AS s on (pv.id = s.[PKID])*  
>  
> *DELETE FROM wh\_tmp\_fixed\_locations WHERE id IN ( SELECT [PKID] FROM*  
> *#temp)*  
> *END*  
>  
> *As always test before trying on production.*

--