Re: What's the deal with PAGEIOLATCH_SH?

Source:
http://www.tech−archive.net/Archive/SQL−Server/microsoft.public.sqlserver.programming/2004−04/4472.html

---

From: Andrew J. Kelly (*sqlmvpnooooospam_at_shadhawk.com*)
Date: 04/26/04

Date: Sun, 25 Apr 2004 21:53:03 −0500

In addition to what Greg suggests are you monitoring the Checkpoint Pages Per Sec counter? If not you may want to and see if your getting hit with a checkpoint at that time. Checkpoints can really hose a disk with low bandwidth.

```
−−
Andrew J. Kelly  SQL MVP
"James Bradley" <jbradley@isa−og.com> wrote in message
news:o03o801ne924b1kr7im1rkrv61sj3gn4i3@4ax.com...
> I took a look at the disk queue length (all of this work is on a
> single temp drive, not on the RAID), and it is happily running about
> 0.7 for most of the query, with occasional spike to about 13 for a few
> seconds.  About 20 minutes into the query everything goes bad.  The
> disk I/O *drops* to 1MB/second in and out, and the disk queue length
> jumps to 60−70 and stays there!  CPU utilization stays low −− less
> than  20% of one CPU.
>
> That sounds like a disk I/O problem to me, but basic test of reading
> and writing the disk seem to work just fine (good queue lengths and
> data rates).  One thing I have noticed in PerfMon is when disk writes
> go over some threshold, which seems to vary, the disk queue length
> skyrockets.  Sometimes I see this occur around 10MB/s write, just now
> I the queue length going > 100 when the writes go over 4MB/s.  Bizarre
> behavior :−)
>
> I'll work on getting the real execution plan, instead of the estimated
> one out of Query Analyzer, but I have never let the command finish (I
> gave up after two hours).  When I try a smaller table size (say 500K
> rows instead of 4.6M), the query executes very quickly, and I don't
> see these problems.  This makes me think that maybe it is an in−memory
> versus out−of−memory execution issue, except that the CPU utilization
> is much too low.
>
> Here is the current statement I'm using:
> UPDATE Table1
> SET hasChildren=1
> FROM
> (
>    SELECT DISTINCT parentID FROM Table2
> ) T
> WHERE Table1.ID = Table2.parentID
>
> The tables are indexed (NC) on ID and parentID.  There are no triggers
> on Table1 and hasChildren is not reference by any index, clustered or
> otherwise.
```

>
> I don't know how to post the estimated execution plan, but I can
> describe it pretty easily.
>
> 1. Index scan on Table1 (31% of time − this is the big table)
> 2. Index scan on Table2 (0% of time − this is a small lookup table)
> 3. Hash Match/Right Semi Join (67% of time).
>
> #3 above shows an estimated row count of 888K, and a row size of 24.
>
> Total cost of the entire execution is 48.0 for 888K rows.
>
> Thanks for any tips!