

Re: SQL Server 2000 speed problems

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-03/4594.html>

From: Dan Guzman (*danguzman_at_nospam-earthlink.net*)

Date: 03/18/04

Date: Thu, 18 Mar 2004 08:19:05 -0600

I'm pretty sure that Tibor is on the money with his analysis. You can monitor the SQL Server Buffer Manager/Checkpoint pages/sec to see if it coincides with your periodic slowness. A write-intensive application like this needs a disk subsystem that is tuned for high-volume writes. SQL Server is certainly capable of providing the needed performance but is constrained by your hardware.

>From an application architecture perspective, you might consider implementing some sort of queuing mechanism (e.g. MSMQ) in order to maximize throughput.

--

Hope this helps.

Dan Guzman

SQL Server MVP

"DPM" <anonymous@microsoft.com> wrote in message
news:uH0jzfoDEHA.2920@TK2MSFTNGP09.phx.gbl...

> Hi,

>

> I was wondering if anyone could help me with this problem:

>

> I've got a database with 51 tables. The first one is a control table with
2

> columns: a timestamp column, which is written to every second, and a key
> column, which is a primary key (clustered ID).

>

> CREATE TABLE Control

> (

> TagKey int IDENTITY (1, 1) NOT NULL ,

> DataTimestamp datetime NULL ,

> ShiftName char (8) ,

> CONSTRAINT [PK_Control] PRIMARY KEY CLUSTERED (TagKey)

>)

>

> The other 50 tables have got 2 columns as well: the first one is a key
> acting as the foreign key of the PK of the control table, and the second
is

> an sql_variant field.

>

> CREATE TABLE Tag1

> (

> TagKey int NOT NULL ,

> TagValue sql_variant NULL ,

microsoft.public.sqlserver.programming: Re: SQL Server 2000 speed problems

```
> CONSTRAINT [PK_Tag1) +'] PRIMARY KEY CLUSTERED(TagKey) ,
> CONSTRAINT [FK_Tag1) +'_Control] FOREIGN KEY (TagKey)
> REFERENCES Control (TagKey) ON DELETE CASCADE ON UPDATE CASCADE
> )
>
> Every second the database gets updated with a timestamp and 50 new values.
> This is done by inserting a new row with the current time into the control
> table every time a second elapses, then getting the key for this row (with
> SELECT @@IDENTITY) and using it as a key for the sql_variant value to be
> written to each one of the 50 tag tables.
>
> The sql_variant values are written using a statement similar to the
> following :
> INSERT INTO Tag1 (Tagkey,TagValue) VALUES(34716,801)
>
> The language used is Visual Basic 6 and ADO 2.7, and the query to perform
> the writing of the 50 values is the following, where strStoreTagQuery is a
> string variable containing a batch of 50 INSERT statements.
>
> ' Perform the actual writing to the database through the batch of SQL
> statements
> ' previously built
> With m_cnnArchiveTagConnection
>
>     .Open
>     .BeginTrans
>     .Execute strStoreTagQuery, , adExecuteNoRecords
>     .CommitTrans
>     .Close
>
> End With
>
> The main requirement of the application is that every second new values
get
> written to the database, and the writing needs to be fast, in the region
of
> 100ms.
>
> So far, this has been achieved, since it usually takes about 47ms to write
> the values. The only problem is that from time to time, the time to write
> the values to the database goes up to 350ms or more for some strange
reason.
> It doesn't happen too often, but it is a problem if the writing time is
not
> uniform. And I don't quite understand it, because I thought that SQL
Server
> would be more than capable of writing 50 values to a database in less than
> 100ms.
>
> I haven't got much experience with SQL Server 2000, and I would appreciate
> any comments. I was thinking that the time to write the values might go up
> when data is actually written to disk, but I'm not sure, since this
> situation seems to occur at random times. Are there any memory settings or
> similar that can be changed on SQL Server that could help solving this
> problem? Or am I doing something wrong in inserting the values in a batch?
>
> Thanks.
>
>
```