

Re: Database performance

Source: <http://linux.derkeiler.com/Mailing-Lists/Debian/2004-05/2373.html>

From: Karsten M. Self (kmself_at_ix.netcom.com)

Date: 05/17/04

Date: Mon, 17 May 2004 00:36:24 -0700

To: debian-user@lists.debian.org

This is really old, but it's straight up my alley, so...

on Sat, Apr 03, 2004 at 07:08:39PM -0600, Christopher L. Everett (ceverett@ceverett.com) wrote:

> *I do a lot of database work. Sometimes I must do massive batch jobs on*
> *my box*
> *such as:*
>
> *-- multi-gigabyte database dumps and restores*
> *-- tests over millions of records, searching for overlooked cases*
> *-- one-off queries for sales & marketing types that join 8 or 9 tables*
>
> *The problem is that these things often take 10 to 30 minutes to run on*
> *my box. When I use the GNU time utility, I see a low PCPU number,*
> *typically between 15 and 25%. CPU utilization viewed through top*
> *remains at 35% or so, and I never go deeper than a few tens of kilobytes*
> *into swap, even though the 1 minute load average climbs to 2 and higher*
> *(I've seen peak numbers around 6).*
>
> *I'm using a single Seagate 40GB ATA-133 as my sole hard drive, and my*
> *system has an Athlon 2600 processor and 1 GB of RAM. Am I correct in*
> *thinking that the bottleneck lies in the HD subsystem?*

Various suggestions have been made:

- SCSI over IDE.
- RAID over JBOD (just a bunch of disks), for values of JBOD ≥ 1 .
- Filesystem choice.
- hdparm configurations.

There are a couple of things not mentioned to date, of which memory and partitioning are two.

Background: My some-time day job involves "large scale" data analysis. For values of large which have varied over the years, but frequently involve multi-GiB and larger datasets, with SAS, principally on Unix

Debian–User: Re: Database performance

systems, but including others (legacy MS Windows, VMS, MVS, etc.).

I've done a number of system optimizations. One was a constrained cartesian join of 100m revolving credit accounts. Runtime was reduced from 30 hours to ~4, through a set of constraints, and an improved algorithm.

Another tenfold reduction in runtime time was attained for a mainframe process which made some very poor decisions about order sequencing in performing analysis. By utilizing extant data ordering, reducing data up front (both in rows and columns), and delaying sorts until the working set was much smaller, wall time for the job was dropped tenfold.

More recently, changing the logic of a query against a remote database reduce runtime from ~20 hours to five minutes.

About a year ago, I had the opportunity to benchmark a Dell desktop, 2.4 GHz CPU, 256 MiB, 7200 RPM ATA disk, against a PIII–500 512 MiB with three SCSI drives, striped. Running identical analysis -- a straight data read, data read+write, and a complex SAS macro ("vv") which runs a wide range of SAS procs and complex SQL queries on the data, I found:

- Straight data read was 2x faster on the PIII–500 system.
- Read + write was > 2x faster on the PIII–500.
- The "vv" macro ran *10* times faster.

Fast disk matters. SCSI blows doors over ATA (including AFAIC SATA), *PARTICULARLY* when simultaneous I/O is taking place. This is what a simple file read or transfer won't tell you about, but random seeks through data makes clear.

What the discussion to date *hasn't* made explicit is this: it's not raw data throughput, but head contention, particularly head movement on read/write activity, which slows you considerably. I'm not familiar with any monitoring tools which report seek time, but you can often hear it as "head chatter" when your drive gets busy. Long, linear reads are far quieter.

Simple optimizations:

Memory: Depending on your processing needs, if a small search set can be held in memory while a larger dataset is sequentially searched, you can make significant gains. Effectively, you're cutting out one whole set of seeks, as you read the search set once at the beginning of processing.

Partitioning: Dividing up your system into OS, user, code (or script), permanent store, and scratch space, can result in significant performance improvements. Especially if these uses can be isolated to separate channels (separate controllers).

Re: Database performance

Debian–User: Re: Database performance

You can also take advantage of partitioning to apply appropriate filesystems to appropriate tasks. GNU/Linux Gazette ran a recent benchmark of common filesystems. Note that if ext2 is fastest for your processing, you can and should by all means run ext2 on your scratch partitions: the "security" provided by journaling is largely nil, as all the data are transitory and can be readily rebuilt anyway.

Other System Optimizations:

I hope it goes without saying that networked access will kill you. Gig–e _might_ match the performance of some slower drives, but in general, you're talking about a thousandfold reduction in performance for remote access. Don't do that for large data. Either bring it to the job ahead of time, or send the job to it.

SCSI beats ATA for complex processing. Yes, there's a price premium, but you're buying a lot, much of which is poorly understood.

RAID helps. Different RAID levels are appropriate to different tasks. In general, you're only concerned with 0 (stripe), 1 (mirror), and 5 (parity). Remember: RAID offers performance and redundancy options, but IS NOT A SUBSTITUTE FOR SYSTEM AND DATA BACKUPS.

RAID5 should be used for system and persistent data. While writing is slower than JBOD due to parity data storage, it's also relatively infrequently performed, *and* mitigated by having more (and in good RAID systems) independently controlled heads. Reads can be far faster, depending on data distribution over the RAID set.

For temporary and scratch data, use stripesets (RAID0). The performance gain is significant, particularly with multiple volumes, and the risk of a data loss and downtime are largely minimal (nothing you can't recover). Remember: you're trading process time saved with downtime incurred. Find a balance point.

Mirroring is best saved for highly critical, seldom–changed data. A core OS or user data mirror might be appropriate.

You may want to use LVM on your RAID volumes for additional partitioning flexibility.

Multiple controllers help. Run your persistent data store and scratchspace off separate RAID controllers, persistent data RAID5 (as discussed above) and scratch RAID0. Put your OS and user state on a third controller, possibly mirrored.

Don't forget OS and application tuning. `hdparm`, as mentioned, can give significant improvements. Some software has options for selecting blocksize, read–ahead buffering, and memory allocation. Look into your project or vendor's docs on this.

Re: Database performance

The Really Important Part – Job Specification

First: measure what you're doing and what you're getting. Blind optimization (despite some fairly generally applicable empirical recommendations above) can very often be money wasted. Performance is most often limited by a single bottleneck. Remove or mitigate this, and you'll see a quantum leap in performance until the next bottleneck is encountered. Generally, these are:

- Disk: head contention.
- Disk: throughput.
- Disk: controller capacity.
- Memory: swap.
- Network: bandwidth limitations.
- Network: protocol overhead (particularly for compression/encryption).
- CPU: contention.

In my experience, CPU is usually the **least** critical factor in data–intensive processing, though this rule can sometimes be broken in advanced statistical analysis, engineering computations, or graphics processing. The last usually benefits by data thinning as your ink density and resolution are limited anyway.

Data design can have a large impact. In particular, indexing (or lack). Indexes incur a cost to build or update, but often pay back manyfold on use. I find that if I use an index more than twice, it's generally a win (and in system development, I'll **always** use it more than once).

The most single significant factor, overall, however, is the design of the application and global data process. Remember:

- Sorting is **expensive**. Don't do it.
- Much data has an existing collate sequence, at least across local spans. Make use of this by pre–aggregating your data.
- Data you don't have to process is data you don't have to process. Get rid of rows or columns not necessary for your analysis as early as possible.
- Memory beats disk. If you (or your data engine) can make use of hash lookups, *_do it_*. You'll see performance gains of 10, 100, or 1000 times over disk–based lookup.
- Pipelining wins. If you can run data from process to process, rather than intermediating through disk, you'll save significant I/O.

Debian-User: Re: Database performance

- Be ruthless in your elimination of processing steps. Don't read or sort or write data "because that's how we've always done it".... I don't want to think of how many mainframe SAS jobs start with a wholly unnecessary initial data sort....
- Measure performance. Find out what steps are taking the most time (or other limited resource) and focus on these. Code that *looks* inefficient but doesn't use appreciable cycles isn't hurting you. Leave it alone. And *measure both ways* -- get a before and after for optimizations tried.
- Find out what your performance bottlenecks are. The standard tools are 'top', 'free', 'sysstat', and related. These *don't* give a perfect reading of what's going on on your system, and may miss details, but they're what you've got to work with.

In general:

- Load a small search/criteria set into memory, and use it to sequentially scan a larger dataset.
- Lose any data you don't need early on.
- When querying remote data sources, if possible, *run the query* remotely, and just return the result set. This was the trick with my 20 hours -> 5 minutes process. I defined a view on the remote database, populated a small (~20k rows) table on the database server, and queried the view for my result set (returning ~20k records). Querying against a 40m row table, indexed.
- Avoid disk processing by streaming / piping data between processes.
- Use hashes rather than sorts or b-trees (or get your tools to use them for you).
- Think about what you're doing.
- Do as little as possible. That's been by gag answer to "what do you do", but from an optimization standpoint, it's the goal.

It's both science and art. Treat it that way.

Peace.

--
Karsten M. Self <kmsself@ix.netcom.com> <http://kmsself.home.netcom.com/>
What Part of "Gestalt" don't you understand?
Save Bob Edwards! <http://www.savebobedwards.com/>

--
To UNSUBSCRIBE, email to debian-user-REQUEST@lists.debian.org

Re: Database performance

Debian-User: Re: Database performance

with a subject of "unsubscribe". Trouble? Contact listmaster@lists.debian.org

- application/pgp-signature attachment: Digital signature