Fastest Informix DBA Contest III

By Lester Knutsen

Performance-tuning an OLTP system

| Published April 15, 2011

Performance tuning is a continuous process for every DBA. My company has conducted three Fastest Informix DBA contests to highlight and learn what goes on in this process. Last year at the 2010 IIUG Informix Conference in Kansas City, we did something a little different that I have wanted to share with you for a while. Previous contests had focused on improving batch processing performance time; in 2010, we used transactions per minute as the benchmark. The results are very relevant to any DBA that must maintain and tune an online transaction processing (OLTP) system.

In the last contest, we used an open source OLTP benchmark, called BenchmarkSQL, that closely resembles the TPC-C standard for OLTP. It's a Java program that generates 100 sessions performing a mix of inserts, updates, and deletes against any database. The folks at AGS, creators of ServerStudio, converted the benchmark to run with Informix, and we converted the database to Informix. We challenged contestants to get the most transactions per minute during a 10-minute benchmark run.

Concurrent user performance

DBAs are accustomed to seeing multiuser OLTP environments, but this is the first year our contest incorporated that challenge—and it was a big one! When the benchmark starts, it instantly creates 100 very active user sessions that select, add, update, and delete records at the same time. With the default Informix setup, these sessions create lots of locking and concurrency errors. Tables created by Informix default to page-level locking, so when two or more users attempt to update a row on the same page, at least one will get a lock error.

The first thing every contestant needed to do was change every table to row-level locking. In any real OLTP database, implementing this change can be a problem. To determine which tables in a database have page-level locking and need to be modified to row-level locking, I use a helpful SQL script: genlockmod.sql (see Figure 1). This script will read the Informix system tables, find out which tables have page-level locking, and then generate another SQL script to change those tables to row-level locking.

Next, contestants needed to add appropriate indexes to each table, so that Informix would not scan the whole table to find the row it needed to update. This wasn't too difficult, as this benchmark database has only nine tables, and we provided the contestants a couple of hints: a list of the top SQL statements used by the benchmark, and a sample script with the primary keys for each table.

However, in the real world, adding the indexes can be difficult, and there is no magic SQL trick to do that—you must know the SQL statements being used and which fields are predicates in the where clauses for those SQL statements. In real life on large databases with thousands of tables, I use the sysmaster database table systabprof and the column seqscans to identify which tables may need an index. Figure 2 shows the code for tabscans.sql, which identifies the tables with the most sequential scans.

Sometimes a table is so small that it is faster to do a sequential scan on a table that fits in one or two pages, than to read index pages and data pages to find a row.

```
{
– Author: Lester B. Knutsen
– Advanced DataTools Corporation
– Description: Generate SQL to set row level locking for all database tables
}
output to lockmod.sql without headings — Create SQL script and don't include column headings
select "alter table " , — Text to alter table
trim(tabname) , — Table name
" lock mode (row);" — Text to change the lock mode
from systables
where tabid > 99 — Don't get the systables
and tabtype = "T" — Get real tables not views
and locklevel = "P" — Get tables with page level locking
order by tabname;
```

Figure 1: *The SQL script genlockmod.sql generates another script to alter tables to row-level locking*

```
{
– Author: Lester B. Knutsen
– Advanced DataTools Corporation
– Description: Sysmaster query to identify tables with the most scans
}

database sysmaster;
```

```
select dbsname,
tabname,
sum(seqscans) total_scans
from sysptprof
where seqscans > 0
group by 1, 2
order by 3 desc;
```

Figure 2: *The script tabscans.sql is a sysmaster query to identify tables with the most sequential scans*

Disk layout for performance

The contest machine ran Linux with four CPUs and 3 GB of memory, but only one disk drive. Databases are very disk I/O intensive, so the single drive was a significant limiting factor. It also meant that a lot of the sophisticated disk layout tuning that you can do with Informix did not help. In fact, the DBA with the fastest time did not make any changes to the baseline disk layout: four dbspaces, a rootdbs, a logdbs, a tempdbs, and a datadbs. The second fastest DBA added one dbspace to move the physical log from the rootdbs to a separate dbspace, but in my testing I have found that layout to be slightly slower since the dbspaces are all on the same physical disk. Configuring different page sizes did not help much either.

With only one disk, is there anything that will help I/O? Four of the top five contestants turned on Direct I/O, a new parameter in the ONCONFIG file that works on Linux and AIX systems to speed up disk I/O to file systems. Another strategy is to limit the number of processes writing to disk to avoid I/O contention between the processes: minimize the number of CLEANERS, LRUs, and AIOVPs.

Memory tuning

In many cases, the biggest performance increases will come from adding as many buffers as you can. Informix uses buffers to store data on the first request, so that it can be shared and reused with other users without the cost of reading it again from disk. The benchmark machine was running a 32-bit version of Informix and was limited to 2 GB of memory (the 64-bit version of Informix does not have this limit). The top two entries used as much as possible for buffers, 1.6 GB and 1.5 GB respectively. They all also examined the amount of virtual memory and set SHMVIRTSIZE accordingly. The top two also set the RESIDENT parameter to -1, which tells Linux to keep all the Informix memory segments in memory and not swap them to disk.

Another key parameter used was DS_NONPDQ_QUERY_MEM. This parameter allows you to increase the default memory for user sort space. If the sort will fit in memory, it will be very fast; otherwise, it overflows to disk and will be much slower. On OLTP systems, where user sessions are performing quick queries, tuning this parameter can be very important.

CPU tuning

Informix has an ONCONFIG parameter VPCLASS that controls how many CPUs will be used by the database server. This is also critical to get right. The default is to use only one CPU. The machine in the contest had four CPU cores, and the top configuration tuned Informix to use all four CPU cores. The only process running on this machine was the database server. Two of the contestants set the number of CPUs for Informix to be greater than the number of physical CPUs on the machine. The CPUs on this machine were fast enough to handle this setting, which helped their performance.

2010 Fastest Informix DBA contest winners

- The Fastest Informix DBA and Grand Prize Winner: Tatiana Saltykova
- Fastest IBM Developer (IBM employee): Spokey Wheeler
- Fastest DBA on Monday (first day of the contest): Tom Girsch
- Fastest International DBA (non-U.S. resident): Denis Zhuravlev
- Fastest Mid-aged DBA (30 to 50): Andrew Ford
- Fastest Youngest DBA (under 30): Pam Siebert
- Fastest Old-Timer (longest experience with Informix): Wenching Chiang
- Fastest Domestic DBA (U.S. resident): Eric Rowell

The winners and the results of the contestants are available at www.advancedatatools.com/Informix/index.html.

We are planning an exciting new contest for the 2011 IIUG Informix Conference on May 15 to 18 of this year. For more information about the conference, visit www.iiug.org/conf/2011/iiug.

Conclusion

One important note: performance tuning is very system specific. I took the top five contestants' configurations and ran them on a different machine from the one we used for the contest, and I got very different results. Know the hardware configuration of your Informix server and tune accordingly.

Conducting this contest is one of the most exciting things I do every year, and the feedback I have received from the participants is very stimulating. Since the IIUG Informix Conference, 28 people have downloaded the contest and run it on their own. Congratulations to all the DBAs who worked hard on this and especially to the winners of the contest. The results are listed in the sidebar "2010 Fastest Informix DBA contest winners."