BBj DBMS Now Leaves PRO/5 Data Server in its Dust!

Nearly every application used in the business world provides multiple users with simultaneous access to the application and data. As business needs grow, it is vital that the application reliably scales with regard to user interface and data access performance. BASIS recently took a great step forward, significantly improving the performance of the BASIS DBMS, and thus, SQL data access when simultaneously accessing data files from multiple users or other programs.

The Problem

Traditionally, both the PRO/5® filesystem and the BBj® BASIS DBMS serialized reads on the same file, meaning that only one process could read from the file at a time. Since reads are extremely fast, this is not normally a performance issue. However, it can become a bottleneck if several processes perform a large number of reads on a single file simultaneously. The PRO/5 filesystem has a 'multi-reader' setopts bit to enable shared read locking on files, providing some scaling for concurrent reads. Unfortunately, the use of operating system file locks means that shared read locking is not balanced with writes and can cause writes to "starve" and not complete in a timely manner. In this case, the setopts bit cannot be used safely by most customers.

The Solution

Beginning with the BBj 13.10 preview of BBj 14.0, the BASIS DBMS scales concurrent reads on the same file so multiple readers can access the same file at the same time, limited only by the operating system and hardware. Rather than using the shared read locking at the operating system level (as with the PRO/5 filesystem), the BASIS DBMS uses internal fair read-write locking. This allows BBj to safely and transparently scale reads without starving writes. As a result, the BASIS DBMS is the ideal choice for customers who need the PRO/5 filesystem shared read or multi-reader setopts bit set because not only does it eliminate the potential for write starvation, it is always enabled and performs significantly better than the PRO/5 filesystem with the shared read setopts bit enabled, providing a fully transparent performance improvement for all customers.

The Benchmarks

Demonstrating and testing read scaling was very straightforward. We ran multiple copies of a simple program simultaneously that iterated through a large file. While this may not have necessarily reflected a real-world use case for customers, it clearly showed the benefits of read scaling during multi-user access. **Figure 1** shows the results of this simple test.

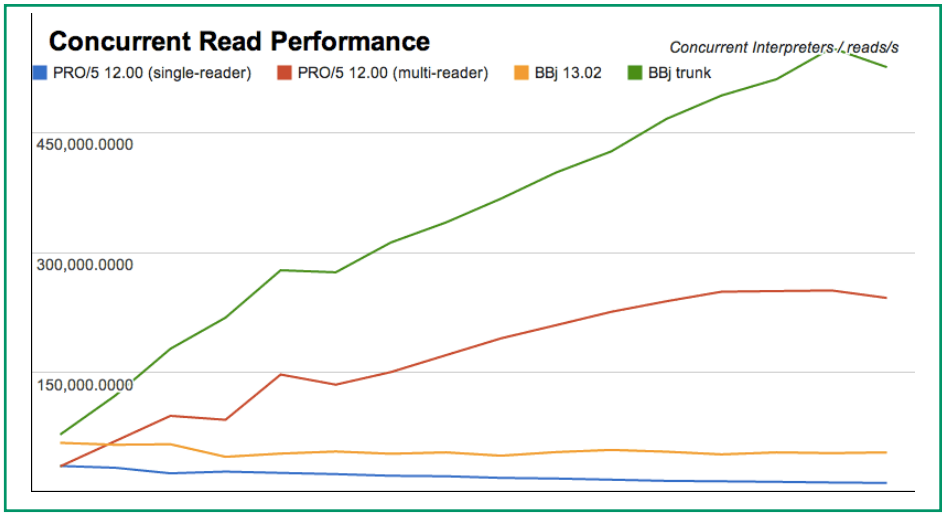

Figure 1. Concurrent read testing results, with the best performance from the new code shown in green

By Chris Hardekopf
Software Engineer

By Jeff Ash
Software Engineer

To get a good feel for the impact of the new read scaling capability, we ran our tests against both PRO/5 and BBj. Note that traditional PRO/5 and BBj concurrent read performances are rather flat as represented by the blue and orange lines. In other words, as we ran more copies of the test program, each reader performed slower in a linear fashion. For example, with two concurrent readers, each individual reader was approximately half as fast, with three it took three times as long, etc. With PRO/5 shared read locking (red line) and the new BASIS DBMS implementation (green line), the reads scale to the number of readers, up to operating system and hardware limitations. Not only does BBj eliminate the possibility of write starvation, it is also significantly faster than PRO/5 as a result of the improvements made to the filesystem. You can see this performance on your system today with BBj 13.10 which first previews this BBj 14.0 feature.

Summary

As business needs grow, it is vital that core enterprise applications and their required data access scale as well. BBj 14.0 takes another huge step forward providing this scaling in the area of concurrent read access to the data, all without any configuration or application changes. ■ links.basis.com/13code

A Case Study

As with many BBj features and improvements, BASIS' work on concurrent read performance began with a question posed by a customer.

> *"Preparing to migrate a PRO/5 deployment to BBj, our company's engineering staff performed a series of tests in a simulated production environment. One test revealed a possible problem in BBj – a bottleneck. In our side-by-side concurrent read tests, the PRO/5 Data Server outperformed BBj's BASIS DBMS and scaled with additional concurrent users and system resources. Graphing the results revealed that PRO/5 trended up with additional users while BBj's line remained flat. Why isn't BBj performing faster?"*

This question left many BASIS engineers scratching their heads. In BASIS' own benchmark testing, BBj always outperformed PRO/5 in concurrent reads. With the customer's test in hand, BASIS sought to reproduce those results. Bruce Gardner, Technical Support Supervisor at BASIS reports, "*For nearly a week, we ran their test on various Amazon EC2 instances – varying the operating systems, varying the memory, varying the number of CPUs. Our results were always the same: While there was a lot to be desired in the area of scaling – for BBj and for PRO/5 – BBj always outperformed PRO/5 in concurrent read performance.*"

Gardner continues, "*A closer look at the customer's PRO/5 setopts vector revealed what we had been missing. The customer had enabled 'Multiple Reads'. This setopts bit blocks WRITE processes in favor of READS; for this reason the setting is practically unusable in even small deployments. However, with the vast majority of their data operations being READs, this customer happily worked with this setting for years with hundreds of users.*"

Now that BASIS could explain the test result discrepancies, they turned their attention to improving BBj's concurrent reads. As discussed above, the research was very fruitful. Not only was the BBj concurrent read bottleneck removed, but now BBj's BASIS DBMS far outperforms and scales better even than the PRO/5 Data Server with the multiple read setopt bit enabled.

Bruce Gardner
Technical Support Supervisor