HAMMER and PostgreSQL Performance

Jan Lentfer April. 2010

(document still being worked upon, intermediate version)

System

The system used for this test was an Intel Atom 330, Foxconn mainboard with ICH7 SATA Controller, 2 GB RAM and 2 Seagate Barracuda ES.2 250GB SATA II disks. The operating system was installed on the first disk, while all the PostgreSQL files (data) were installed on a 150GB partition on the 2nd disk. Both disks operated on their own SATA channel.

Later on the Data Disk was moved onto a Dawicontrol DC-3410 SATA PCI controller which is using a Silicon Image 3124-2 chip. This adapter was kindly donated by Matthew Dillon to support further testing.

PostgreSQL

PostgreSQL 8.3 was installed from binaries using the respective packaging system. In postgresql.conf Shared buffers were set to 512MB, Effective Cache Size to 1024MB. Otherwise default values from the Dragonfly BSD installation were used. The same postgresql.conf file was used on all Operating Systems.

Dragonfly BSD

The version used was 2.4.1. The Kernel was build using “make nativekernel”. The KERNCONF was a copy of GENERIC with these changes:

```
--- GENERIC     2009-09-27 22:01:49 +0200
+++ ATOM_SMP    2009-10-11 00:42:24 +0200
@@ -9,10 +9,10 @@
 platform       pc32
 machine        i386
 machine_arch   i386
-cpu            I486_CPU
-cpu            I586_CPU
+#cpu           I486_CPU
+#cpu           I586_CPU
 cpu            I686_CPU
-ident          GENERIC
+ident          ATOM_SMP
 maxusers       0

 makeoptions    DEBUG=-g                #Build kernel with gdb(1) debug symbols
@@ -58,7 +58,7 @@
 # boot fine for non-SMP builds *might* work in SMP mode
 # if you define SMP and leave APIC_IO turned off.
 #
-#options       SMP                     # Symmetric MultiProcessor Kernel
+options        SMP                     # Symmetric MultiProcessor Kernel
 #options       APIC_IO                 # Symmetric (APIC) I/O

 # Debugging for Development
```

The acpi kernel module was not loaded.

Test Scenario

pgbench was run from an Ubuntu Linux system on the same 100MBit switched network. At first a database of approx. 5GB was created (-s 400) onto a freshly (newfs'd) created file system to actually involve the file system in the test. At first "SELECT Only" runs where started against the freshly created database three times in a row to fill up database and file system caches. The result of the fourth "SELECT Only" run is presented as "initial random seek performance".

Thereafter three TPC-B tests were run and the average result is presented as "TPC-B" in the chart. Afterwards another "SELECT Only" test was run, which is shown as "final rand/seek".

Only for the test on HAMMER with mount options nohistory,noatime "hammer cleanup" was run also and the final "SELECT Only test" was redone three times (line 3 in the chart).

Options used for pgbench was run with –c 10 and –t 1000, so 10 concurrent sessions with 1000 transactions each, ending up at 10.000 transactions per run.

	OS	Filesystem	Options	initial rand/seek	TPC-B	final rand/seek
1	Dragonfly BSD	HAMMER		100 tps	14,8 tps	71 tps
2	Dragonfly BSD	HAMMER	nohistory,noatime	102 tps	16,3 tps	68 tps
3	Dragonfly BSD	HAMMER	hammer cleanup			57 / 69 / 69 tps
4	Dragonfly BSD	UFS	newfs –b 8192	77 tps	36,6 tps	67 tps
5	Dragonfly BSD	UFS	newfs –b 8192, noatime	87 tps	44,3 tps	78 tps
6	Debian 5.0	XFS	-b size=4096 –s size=4096 noatime,logbsize=128k,logbufs=8	75 tps	27,1 tps	85 tps
7	Debian 5.0	ext3	-b 4096 –f 4096, noatime,data=writeback	127 tps	64,6 tps	90 tps

When running the tests on Debian I noticed that results of single runs where not consistent so I over thought my test setup and decided to redo all the tests with the following changes:

- Turn off autovacuum in PostgreSQL
- One run with 10.000 transactions with a concurrency of 10, so 100.000 transactions total (before that was 30.000 total) for initial random seek and TPC-B test. Final random/seek is still run at 10/10.000.

	OS	Filesystem	Options	initial rand/seek	TPC-B	final rand/seek	Decrease ratio
1	Dragonfly BSD	HAMMER	nohistory, noatime	95 tps	16 tps	45 tps	53%
2	Dragonfly BSD	UFS	newfs -b 8192 –f 8192, noatime	70 tps	36 tps	55 tps	21%
3	Dragonfly BSD	HAMMER	nohistory, noatime, sysctl adoptions *)	94 tps	38 tps	46 tps	51%
4	Debian 5.0	JFS	Noatime	79 tps	47 tps	61 tps	23%
5	Debian 5.0	ext3	mkfs.ext3 –b 4096, noatime	75 tps	61 tps	93 tps	-24%
6	Debian 5.0	XFS	mkfs.xfs –b size=4096	114 tps	29 tps	68 tps	40%

*) vfs.getattr_mpsafe=1, vfs.read_mpsafe=1, vfs.hammer.fsync_mode=2

Tracking HAMMER Development

Tests were redone with the master branch of DragonFly.
Starting at line 7 the PostgreSQL-Data-Disk was moved to a sili based adapter (Dawicontrol DC3410) which supports NCQ. So numbers are only congruent from line 7 onwards.

	OS	Filesystem	Options	initial rand/seek	TPC-B	final rand/seek	Decrease ratio
1	DF 2.5.1-91ed38	HAMMER v4	nohistory, noatime, fsync_mode=3	98 tps	25 tps	48 tps	49%
2	DF 2.5.1-91ed38	HAMMER v4	nohistory, noatime, fsync_mode=2	88 tps	25 tps	60 tps	
3	DF 2.5.1-91ed38	HAMMER v4	nohistory, noatime, fsync_mode=4	82 tps	35 tps		
5	DF2.5.1-3eb2d2	HAMMER v4	nohistory, noatime, fsync_mode=2	87 tps	26 tps	63 tps	28%
6	DF2.5.1-3eb2d2	HAMMER v4	nohistory, noatime, fsync_mode=3	101 tps	31 tps	75 tps	26%
7	DF2.7.0- 5d6a9	HAMMER v4	nohistory, noatime, fsync_mode=2				

Discussion

My knowledge about the internals of the HAMMER file system is by far too little to make any assumption to what is happening on the file system. So I will stick to describe what the numbers show:
Random Seek performance for a fresh database is really good compared to UFS and also better than most Linux filesystems, but as soon as you have a mix of random seeks, random writes and sequential writes (WAL Files) the performance of HAMMER collapses.
What is also remarkable: After the file system has been exposed to some write activity the former fast random seek performance drops by around 30%. My thought was that this might be due to hefty changes in the btree of HAMMER. That is why “hammer cleanup” was run on the file system (which rebalances, reblocks and recopies) and “SELECT Only” test was redone three times for the first chart. As one can see from the chart, the performance did not go back to where it was.
Apparently also the “noatime” mount option does not help HAMMER as much as it does help UFS. Overall you have to say that for real OLTP work loads UFS operates almost 3 times faster as HAMMER if you use it “out of the box”.

Thanks to input from Matthew Dillon I also redid the tests for HAMMER with vfs.hammer.fsync_mode=2 (synchronous fsync on close if fsync called prior to close, line 3 in 2nd chart.) to see how much fsync impacts the performance of HAMMER, as fsync is known to be very expensive on it. By this the TPC-B performance more than doubled to a level comparable to that of UFS, but still the drop in seek performance after intensive writing is remarkable compared with other file systems.

Conclusion

With the sysctl tuning done as suggested by Matthew Dillon HAMMER could be an alternative for PostgreSQL databases on Dragonfly assuming that these tuning measures do not interfere with PostgreSQL data integrity (One has too keep in mind that PostgreSQL heavily relies on fsync for data integrity and recovery. This is a field I still need to investigate.), if there wasn’t this immense performance degradation. If “hammer cleanup” would come into help you could still think about using it for databases where you have time windows of low usage but my tests did not show it to do so.

Matthew Dillon suggested to do the tests also with the latest versions of HAMMER and Dragonfly, so be ready for another update of this document!