

HAMMER and PostgreSQL Performance

Jan Lentfer Oct. 2009

(document still being worked upon, intermediate version)

System

The system used for this test was an Intel Atom 330, Foxconn mainboard, 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.

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 were 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

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, 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. 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.