

Tuning Preparation Lab

You should log in on the server running your DB2 server instance (from now on I assume it is running on an AWS image, but the lab applies equally well if you are running a local instance). See the course web site for details (DB2 environment).

You should not spend more than 3 hours running this lab.

Part 1 – Hardware

You should answer the following questions – note that some answers are trivial (as they might correspond to an EBS configuration parameter) others will require some simple benchmarking, or using some OS tools.

1. How many disks/ how much disk space do you have at your disposition?
2. How many different disks are you using?
3. How much RAM can you use?
4. How fast is (are) your file system: e.g., throughput for sequential read, sequential writes, random reads, random writes? Note that what is interesting is IO-bound workload, i.e., the file system relies on disks. How does that compare to a bare Barracuda drive:
<http://tiny.cc/UpYjV> ?
5. How stable is file system performance?

To answer 1 above, you can rely on the vmstat utility:

```
$ vmstat -D
```

This will give you the disk table. Note that vmstat is also useful in a slightly different context:

```
$ vmstat 1 100
```

This will display kernel statistics every 1 second, 100 times. You can use this type of long running command to monitor activity while running a benchmark or an experiment: IO (how many blocks in/ blocks out), CPU usage and swapping.

For 4. and 5., you should use the Bonnie benchmarking tool. See

<http://www.garloff.de/kurt/linux/bonnie/> for details. To install bonnie, simply run:

```
$ rpm -i http://www.garloff.de/kurt/linux/bonnie/bonnie-1.4-0.i386.rpm
```

To run the bonnie benchmark:

```
$ bonnie -s 2000 -o_direct
```

The -s option defines the size of the file that is used for the benchmark (a small file size does not guarantee an IO bound workload and artificially boosts random IO performance, i.e., seeks – you should try various file sizes and observe the differences you obtain); the -o_direct option makes sure that block operations (the ones we care about for DB performance attack directly the disk subsystem – and bypass the file system cache).

Part 2 – DB2 Configuration Parameters

Now that you have a clear picture of the HW and OS settings, let us focus on the DB2 configuration.

You should attach your DB2 instance and get the list of configuration parameters for the DB2 server:

```
$ db2 attach to db2inst1
```

```
$ db2 get dbm cfg show detail
```

More interestingly, you can access the list of configuration parameters for a given database:

```
$ db2 connect to tuning
```

```
$ db2 get db cfg show detail
```

The most interesting parameters are the following:

1. The location of the data files (dbm)
2. The location of the log files (db)
3. The size of the buffer pool (db)
4. Total log size (log file size x nb log files) (db)

What are their values on your instance?

Upload the script gentable.py and the example spec file hotelspec on the database server (both scripts are available on the class home page) and generate a file with 1 million rows:

```
$ python gentable 1000000 hotel.data hotelspec 1
```

This might take a couple of minutes. Note that the time is spent generating rows (not writing to the 62 MB file).

You should now create a hotel table whose schema corresponds to the data you have just generated, and load data into it. See <http://tiny.cc/F2mKe> for a description of create table, and <http://tiny.cc/RleVI> for a description of the load command (hint: the data is generated in a delimited ascii format, not the default one though, it is a *modified* version of del using | as a column delimiter).

Loading data should not take more than a few seconds (a 62 MB file at the bandwidth you measured in part 1). Note the time it took to load the data. Note also that if you have been waiting for a minute already, you can take a look at what is happening with vmstat then stop execution. You have a tuning problem! This can be fixed by tuning one parameter (of the db configuration). Which one? (hint: the load command documentation mentions it). Note that we will cover database writes on Tuesday 16/2.

Now run the following query:

```
$ db2
```

```
> select count(*) from hotel h1, hotel h2 where h1.hotel_id != h2.hotel_id and h1.city = h2.city and h1.zip_code = h2.zip_code
```

You should check disk activity using vmstat. What is going on? Are there a lot of reads? A lot of writes? You can stop the execution after a minute or so. Why is this query so slow?

The first step is to look at the profiler:

```
$ db2 -v update monitor switches using sort on
```

Then

```
$ db2 -v get snapshot for database on tuning
```

Look at the total number of sorts, at the number of sort overflow, and at the space allocated in the sort heap.

The second step is to look up the execution plan (using the db2expln utility):

```
$ db2expln -t -database tuning -q "select count(*) from hotel h1, hotel h2 where h1.hotel_id != h2.hotel_id and h1.city = h2.city and h1.zip_code = h2.zip_code"
```

What join method is used? You can look up the different join methods at <http://tiny.cc/y93y4> . How much space is required to avoid repeatedly accessing the disk when building the build and probe? Note that the build and probe are not constructed in the buffer pool, they are constructed in a dedicated sort heap. How large is the sort heap?

Let us try and increase it. What size should you choose (X)?

```
$ db2 update db cfg using SHEAPTHRES_SHR 2*X  
$ db2 update db cfg using SORTHEAP X
```

Check that the update has been performed with:

```
$ db2 get db cfg show detail
```

Run the query again

```
$ db2
```

```
> select count(*) from hotel h1, hotel h2 where h1.hotel_id != h2.hotel_id and h1.city = h2.city and h1.zip_code = h2.zip_code
```

Much better, isn't it! A look at the database profiler shows that there is no longer any sort overflow. Check the IO patterns with IOStat. Check the execution plan. Do you observe any difference?

You should drop the hotel table, and the hotel.data files before you terminate your AWS instance (if you have a permanent EBS).