Performance of Freereg

Observations on the performance of certain aspects of Freereg. **These observation focus mainly on the creation aspects at this time.**

They are based on a 64 bit Window 7 platform with 8GB of memory. A quad core processor with each core rated at 2.67GHz. Operating system and data are on separate channels and using 2 Samung HD501LJ drives (interface : SATA 3.0Gb/s, Capacity : 500GB, RPM: 7200 RPM, Cache: 16MB, Average Seek Time: 8.9ms, Average Latency: 4.17ms)

Mongodb (2.4.4) for the database with Mongoid as its interface

This version of Ruby (1.9.3) on Rails (3.2.16) is multi threaded but restricted to a single processor. Hence each rails process runs on one core.

1. **The csv file processor** runs at 95% cpu utilization on a single core occupying no more than 300MB of memory. Mongodb runs in another single core at about 25% cpu utilization. Its use of memory has about 1GB of committed and 4GB of sharable. Disk traffic is about 500KB/s with 75% of this associated with the latest Mongodb working file and split 50/50 read/write; another 20 % is writing to the Journal.
2. Based on this running more than 1 copies of the **csv file processor on a single core cpu are likely unproductive. With a quad core it appears useful to run 3 copies simultaneously.** This achieves close to 100% overall cpu utilization and allows Mongodb maximum use of memory. Disk conflicts and memory conflicts appear to be minimal.
3. A single copy of the csv file processor creates about 6,00 data record entries/sec if it stops there and 3,300 search record and data record entries/sec if we complete the creation of the search records.
4. The current database when complete holds information on 6,393 distinct places; 8,956 churches, 10,215 registers, 35,272 files and 26,243,918 each of original data entries and search records. It occupies 17.7GB of storage. Indexes are a further 10GB.
5. So recreating the database takes 5 processor days with a single 2.67Ghz cpu. A quad core running 3 copies can reduce this to about 2 days. Index creation takes a further day.
6. Updating the database with just 10% changed/added files is likely to take about 10hrs on a single cpu and 4 hrs on a quad. (This has to be confirmed)
7. **Conclusions: a)** We need to consider adopting an update processing model earlier rather than later. **b**) We need to make sure that we are planning for an adequate level of server processing power for production.

## Detailed performance investigation of the FreeREG1 CSV File processor

A detailed timing investigation the Freereg1 Csv Processor in the creation of the database shows the following; Times are in **micro seconds per record** and writing does not include any significant index creation. Just those needed for efficient csv processor functioning. These numbers are consistent with the bulk number experience.

Setup: 40 (Array/hash and variable setup together with creation of a file digest)

Checking prior existence: 15

Slurping Csv File: 75

Data parsing, including all error checking and formatting : 250

Writing entries and creating Place/Church/Register/File: 2750

Deleting entries and search records when replacement file being loaded: 5000

Creating and writing search records: 20000

**A couple of conclusions**:

1. We can run a stripped down processor to do error detection on the upload of a csv file; a 10,000 uploaded file could be processed in less than a minute. 95% of all files are less than this. There are some at 40,000 records so we likely should give a time estimate while processing as part of the upload.
2. Priority should be given to looking at the search record creation code for efficiencies followed by a look at our entry/record deletion and then entry/record writing to see if we can gain performance improvements.