The Synchronization engine: The case study of Jamaican health records

C. Krier & N. Évrard

B2CK

September 23, 2015
Goals

synchronize GNU Health records with central instance
only work on a subset of records
mainly append
asynchronous
Goals

- synchronize GNU Health records with central instance
Goals

- synchronize GNU Health records with central instance
- only work on a subset of records
Goals

- synchronize GNU Health records with central instance
- only work on a subset of records
- mainly append
Goals

- synchronize GNU Health records with central instance
- only work on a subset of records
- mainly append
- asynchronous
Requirements

- PostgreSQL Database
- TCP/IP connection between satellites and central instances
- Tryton XML-RPC connection between satellites and the central instance
- Tryton 3.0
Requirements

- PostreSQL Database
Requirements

- PostgreSQL Database
- TCP/IP connection between satellites and central instances
Requirements

- PostreSQL Database
- TCP/IP connection between satellites and central instances
- Tryton XML-RPC connection between satellites and the central instance
Requirements

- PostgreSQL Database
- TCP/IP connection between satellites and central instances
- Tryton XML-RPC connection between satellites and the central instance
- Tryton 3.0
Definition

The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. In this context the word unique should be taken to mean "practically unique" rather than "guaranteed unique". — Wikipedia

We are using UUID Version 4 (random). The probability of collision $\approx 1 - e^{-\frac{n^2}{2x}}$ with $x = 2^{122}$.
Probability of collision reach 50% if we generate 1 billion UUIDs per second for 100 years.
But we need good entropy sources to guarantee those results.
SyncMixin.unique_id_column

- use UUID field type
- Column name of cross-instance unique key
synchronisation_id in the configuration file

- Integer between 0 and 127
- Must be unique across the whole system
SyncMixin.last_synchronisation

- Timestamp of the last synchronisation to the central instance
- Use create / write timestamp
- Cleared if record modified on satellite

All instances must be synchronized. Use NTP!
Who knows the record

SyncMixin.synchronized_instances & SyncMixin.synchronised

- BitString (also a new type of field)
- VARBIT in PostgreSQL
- Use BAND (binary AND) search
- The index of the bit is stored in the context
3 main tasks on celery using celery_tryton working by batch of 1000 records.

- **synchronise_push_all**  push modified records since the last synchronisation
- **synchronise_pull_all**  pull changes on the central server
- **synchronise_new**    fetch new instances
What is Celery?

- **Celery** is an asynchronous task queue based on distributed message passing.
- Task queues are used to distribute work across threads or machines.
- In our case, a cron job will distribute amongst all workers.
synchronise_push_all

Loop over all subclasses of SyncMixin
Search records with an empty last_synchronisation
Push values via XML-RPC and receive success & timestamp
Set timestamp to last_synchronisation of succeeded records
synchronise_push_all

- Loop over all subclasses of SyncMixin
synchronise_push_all

- Loop over all subclasses of SyncMixin
- Search records with an empty last_synchronisation
synchronise_push_all

- Loop over all subclasses of SyncMixin
- Search records with an empty last_synchronisation
- Push values via XML-RPC and receive success & timestamp
synchronise_push_all

- Loop over all subclasses of SyncMixin
- Search records with an empty last_synchronisation
- Push values via XML-RPC and receive success & timestamp
- Set timestamp to last_synchronisation of succeeded records
synchronise_pull_all
synchronise_pull_all

- Loop over all subclasses of SyncMixin
synchronise_pull_all

- Loop over all subclasses of SyncMixin
- Send all records & ETag and receive values of changed records
synchronise_pull_all

- Loop over all subclasses of SyncMixin
- Send all records & ETag and receive values of changed records
- Write new values
synchronise_new
Loop over all subclasses of SyncMixin
- Loop over all subclasses of SyncMixin
- Get records not synchronised and receive values of new records
- Loop over all subclasses of SyncMixin
- Get records not synchronised and receive values of new records
- Create new local records