Executive Summary

Isai Mercado
Jeffrey Young
April 12, 2016

1 - Problem

Determine what percentage of an area's population at any given period of time does FamilySearch have record of.

2 - Solution

Count records per country and year by running mapreduce jobs in Spark. Export Spark outputs to MySQL database so that visualization app may query database. Make Visualization app with Javascript so that users do not have to install any special software.

A) Difficulties

Dates come in different formats. For example: About 1887, June 1998, 01 January 1943, 09 - 12 - 1967, 01 - 02 - 90.

Places come in different formats.

- Places may only contain a country, a state, a city, or an abbreviation. For instance, USA, United States, United States of America, US, Brooklyn, New York, NY
- Places may have spelling differences because of accent marks, For example, México vs Mexico
- Places may have spelling differences because of transliteration. For example, Seoul vs Seul
- There may exist several names for the same place. This is the case when countries have had several conquerors, or talk several languages. For instance, All Belgium states have Dutch names and French names
- Records entered in another country may use the translated version of the country name. For example, if temple work is done by American members in behalf of Spanish people, they may enter Spain instead of Espana.
- Old country names. For example, "England", and "Scotia"

B) Resolution

To obtain years

- A regular expression can look for a four digits string. This will lose 2 digits years, but two digits years need very special parsing techniques that may not be worth trying.

To obtain countries

- Look for country's short name substring. For example, looking for "united states" would find "united states of america", or looking for "france" would find "the kingdom of france"
- Look for states' names substrings because there are records that only contain states names
- Look for 2 Alpha and 3 Alpha Abbreviations substrings. There are some records like "cuernavaca, mex"
- Map Unicode to Ascii to decrease spelling variation. For example México is mapped to Mexico
- Create a list of more common country or states names variations. For example, look for both "Seoul", and "Seul"
- Create a list of common translated countries. For instance, look for both "Spain", and "Espana", or both "Brasil", and "Brazil"

- Create a list of countries whose states have several names such as Belgium which has Dutch Names, and French Names
- Create a list of old country names and map them to current country names. For example, “england” and “scotia” to “united kingdom”

C) Our Resolution

We had several problems running Spark in the BYU Super Computer. After many many trials, we were able to use 1 node with 16 CPU. However, we were not able to able to completely implement all solutions mentioned above because a lot of it needs to be done manually, or the code slowed down the solution too much.

Our solution only looked for country names, states names, 2 alpha, and 3 alpha country name abbreviations. We did not implement transforming Unicode strings to ASCII strings. We did not create a list of variations names like “Seoul” and “Seul”. We did not implemented a list of translated country names or states names that looked for “Brasil” and “Brazil”, “Spain” and “Espana”, or Belgium states. However, we think that some of those records were still counted because of the states lists, but there might be many records that we were not able to count because our difficulties on running Spark.

We tried mapping Unicode to ASCII, but it slowed down the code by a magnitude of 20 in our local experiments. As mentioned before, we did not try that in the super computer because we had a limited time before the job was canceled, and spark only had one node. We think that there is a better way to map Unicode to ASCII but the python library that we used, was very, very slow.

3 - Recommendations

We think that by completely looking for all the substrings mentioned above the count accuracy will increase significantly. Although it might seem that looking for substrings is very inefficiency, we think that there is no other way to completely count all records in the family tree whose information is scarce and without format. Since many of those lists have to be done somewhat manually, they may take some time to be made. However, we estimate that in 5 nodes with 16 CPUs this job can be completed in less than 10 hours, and then Family Search can have a very accurate count of records per country and year.

4 - Conclusion