Master thesis - Roadmap
CP based Sequence Mining on the cloud using spark

Cyril de Vogelaere : 2814-11-00

November 10, 2016

1 Objectives of this master thesis :

The objective of this master thesis, is to develop a **scalable** and **efficient** implementation of the prefix-span algorithm, using Spark, and to **release the developed algorithm** for the benefit of the Spark community.

Ideally, this objective would be concretized by developing an improvement of the CP based PPIC algorithm. **Leading to a performance improvement sufficient enough to propose, and see-through, the integration of the CP library Oscar into the conservative community of Spark.**

In case this ideal couldn't be realised due to unsufficient improvement in the results, otherwise attained improvements would still be proposed to improve the existing algorithm, and **a Spark-package would be developed and released** to propose the CP based solution to any willing user. Meaning it wouldn't be part of the official library but still available.

2 Targeted learning outcomes (LO) :

I decided to target the five first learning outcomes, the sixth one not being applicable in this case.

- LO 1 : to **demonstrate** he/she masters a body of knowledge and basic **skills in** science and/or **engineering sciences**, bound about his/her thesis.
- LO 2 : to **lead to completion** a major, in amplitude and spent time, engineering approach applied to **the development of a product**, service or facility referred to the thesis.
- LO 3 : to **lead to completion a major**, in amplitude and spent time, **research work** aiming at the understanding and the contribution to the resolution of an original scientific question of theoretical or physical type.
- LO 4 : to **organise and plan the master thesis work on the basis of allocated resources and time constraints**, of security (if applicable) and of available competencies.
- LO 5 : to **efficiently communicate both orally and in writing** (in French and/or in English) to realise the master thesis.

3 Schedule :

3.1 Realised task :

- 12 October : Through a presentation, describe the goals of this master thesis, the differences between Spark and Hadoop, the potential amelioration to the currently released algorithm, and an explanation of the inner-working of said algorithm.
- 22 October : Measure the performances differences between the existing implementation of Spark and PPIC and analyse the results.
  - NB : This step revealed that, although the execution time of PPIC is way under the execution time of the current algorithm, the preprocessing needed before the execution could be launched was so demanding that the overall performances were, on 2 dataset out of 6, lesser than existing Spark algorithm.
  - NB2 : The Spark algorithm tested at the time was searching for more solution than the PPIC algorithm. Since it took into account item permutation. Meaning running for the (1,2) 3 sequence would output 1,3; 2,3; and (1,2),3 although PPIC outputted only the two first.
- 23 October : Achieve an efficient implementation of the minPatternLength constraint and the min frequency constraint on the current Spark implementation.
- 24 October : Modify the current implementation of Spark so that it doesn't take permutation into account anymore.
- 29 October (actually achieved on November 2 due to complications) : replace the current local execution by the PPIC algorithm and measure performances. Compare those performances to those of the original algorithms and to the performances of a full spark implementation that doesn't take permutation into account. Perform those measurement for different minimum local database sizes, compare the results.
  - NB : The measured performances of the sparkNP implementation (original spark without permutations) is a net improvement to both the original PPIC algorithm and the original spark implementation. Surpassing them largely on 5 dataset out of 6 and partially on the remaining dataset (Leviathan, were PPIC provide better performance for lower supports)

    The performance of the PPIC-Spark implementation were relatively in line with the the performances of sparkNP for the tested points. For all tested local database sizes. Leading to the realisation that the performances test needed to be extended to lower support to observe larger differences.
- 9 November : Perform an extended performance measurement and compare the results.
  - NB : The results for a 32 000 000 minimum local projected database didn't put any major performance difference into light. Larger local databases size (6 400 000 000) however, showed more differences, although mixed in results.

I observed that the PPIC-spark implementation was significantly better on the Bible and Kosarak dataset, and that the sparkNP implementation showed a large improvement on the protein dataset, being roughly two time faster than the Spark-PPIC implementation at all time.

Although the performances differences are now more significatives, they may not be sufficiently so to motive the addition of the Oscar library into Spark. Further analysis must be realised on the protein dataset results to explain the difference in performances.

- 10 November : Realise a Roadmap, as asked in the vice dean's presentation on the master thesis.

3.2 To be realised tasks :

- For the end of November:
  - Modify the PPIC algorithm so that it takes permutations into account, replace the local execution of spark with it, and compare performances.
  - Analyse further the protein dataset results to explain the difference in performances.
- For the end of December :
  - Once all PPIC algorithm are completed, depending on their performances, improve the merging of their input with the middleput of Spark, as to diminish further the preprocessing need, hopefully improving the current spark algorithm in the process.

    List of identified potential amelioration to date :
    * Modify the spark implementation so that it already include a lastPosMap and firstPosMap. Measure the performances of that improvement.
    * Precalculate the item-support array during the spark execution
    * Implement efficient regex and item constraints
  - Start an outline of the Paper (in english)
  - Determine whether the attained results seem sufficient to propose the PPIC based solution improvement.
- For the end of February :
  - Test performance improvement over a varying number of threads. Determine the bottle neck of scalability and improve algorithm if possible
  - Propose chosen amelioration to the Spark community.
    * Best case scenario : start taking care of the insertion of the PPIC code inside spark library. Solve eventual bug and convince reviewer of the improvement.
    * Worst case scenario : propose a simpler modification of Spark without the addition of the PPIC based algorithm, which should be easily accepted. Start working on a spark package release to make the PPIC improvement available nontheless (Supposing they at least equivalent, which is the case so far)

  - Continue work on the paper

- For the end of Mars :
  - Every implementation related work should be finished.
  - Continue work on paper, it should be mostly finished by the end of the month. Ideally, ask for a first review by the assistant.

- For the end of April :
  - Give finishing touches to the paper and send it for review.
  - Prepare the final presentation (start and finish slides + start repetition).

- May :
  - No work scheduled for this month. In case of delay in the planning (which shouldn't happen), this month will be used to finish the scheduled work.

- June :
  - Deliver the results of the master thesis and present them orally.