Scalable and Cloud programming - Co-Purchase-Analysis Technical Report

1st Michele Dinelli
dept. of Computer Science and Engineering, University of Bologna

***Abstract*—This document is the technical report for the scalable and cloud programming project developed for the university course held in Bologna (a.y 24/25). Source code is available online as a GitHub repository.**

***Index Terms*—cloud programming, scala, apache spark, google cloud platform, dataproc**

I. Introduction

This document is a technical report that describes the implementation and scaling evaluation of a co-purchase-analysis script written in Scala. The system is required to read purchases data from a dataset [1] and it must identify co-purchased products by calculating, for each pair of products, the number of orders in which they were purchased together.

II. Implementation

The solution used Apache Spark [2] and runs on distributed nodes using DataProc [3]. Versions used are Scala 2.3.15 and Spark 3.5.5.

III. Analytical Metrics

A. Speedup

We define $T(n)$ as the execution time of a parallel program with n nodes. The speedup $S(n)$ is defined by the formula

$$S(n) = \frac{T(1)}{T(n)} \tag{1}$$

Ideally, the program with n nodes requires $1/n$ the time of the program with 1 node. $S(n) = n$ is a linear speedup but in practice it's sublinear $S(n) \leq n$. This limitation is captured by Amdahl's Law [4], which states that if a task consists of a fraction f hat is inherently sequential (i.e., cannot be parallelized), and the remaining fraction $1 - f$ can be sped up by a factor of P then the maximum achievable speedup is

$$\frac{1}{f + \frac{1-f}{P}} < \frac{1}{f} \tag{2}$$

Even if we could infinitely speed up the parallelizable part (i.e., $P \to \infty$), the overall speedup would still be limited by the sequential portion f.

B. Scaling Efficiency

To evaluate the impact of Amdahl's law (2) on a distributed system we introduce Strong Scaling Efficiency (SSE) and Weak Scaling Efficiency (WSE).

1) SSE: measures how increasing the number of nodes impacts the speedup while keeping the total amount of work fixed.

$$SSE(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)} \tag{3}$$

The total amount of work remains constant, while the amount of work for each processor decreases as n increases. SSE is limited by the constant $\frac{1}{f}$ so it tends to zero.

$$\lim_{n\to\infty} SSE(n) = \lim_{n\to\infty} \frac{T(1)}{nT(n)} = \lim_{n\to\infty} \frac{1}{fn} = 0 \tag{4}$$

2) WSE: measures how productively a program uses added resources and is defined by

$$WSE(n) = \frac{T_1}{T_n} \tag{5}$$

Where T_i is time required to complete i work unit/s with i node/s. WSE increases the number of nodes n keeping the per-node work fixed as the total amount of work grows as n increases.

IV. Results

A. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
Table Type Styles

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of

Fig. 1. Example of a figure caption.

quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

REFERENCES

[1] Kaggle. [Online]. Available: https://www.kaggle.com/datasets/psparks/instacart-market-basket-analysis

[2] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, “Apache spark: a unified engine for big data processing,” *Commun. ACM*, vol. 59, no. 11, p. 56–65, Oct. 2016. [Online]. Available: https://doi.org/10.1145/2934664

[3] Google. [Online]. Available: https://cloud.google.com/dataproc

[4] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS ’67 (Spring). New York, NY, USA: Association for Computing Machinery, 1967, p. 483–485. [Online]. Available: https://doi.org/10.1145/1465482.1465560