Sparse-matrix Arithmetic Operations in Computer Clusters
A Text Feature Selection Application

Antonela Tommasel, Cristian Mateos,
Daniela Godoy and Alejandro Zunino

ISISTAN Research Institute, CONICET

June, 2014

Table of Content

Arithmetic Operations on Matrices

- They are frequent in scientific computing areas, for example in signal and image processing, document retrieval, and feature selection.
- Those operations usually become a performance bottleneck due to their high computational complexity.
- The parallel processing of matrix operations in distributed memory architectures arises as an important issue to study.
- The operations with **dense** matrices have been the subject of intensive research.

Arithmetic Operations on Matrices

Motivation

- In text analysis, as in collaborative filtering and document clustering, matrices are **sparse**.

- The performance of sparse matrix operations tends to be lower than the dense matrix equivalent.
- Algorithms that are efficient for dense representations are not suitable for sparse representations.

- This paper aims at studying the performance of several strategies for distributing sparse-matrix arithmetic operations on computer clusters.

Arithmetic Operations on Matrices

Motivation

- In text analysis, as in collaborative filtering and document clustering, matrices are **sparse**.

- The performance of sparse matrix operations tends to be lower than the dense matrix equivalent.
- Algorithms that are efficient for dense representations are not suitable for sparse representations.

- This paper aims at studying the performance of several strategies for distributing sparse-matrix arithmetic operations on computer clusters.

Arithmetic Operations on Matrices

Motivation

- In text analysis, as in collaborative filtering and document clustering, matrices are **sparse**.

- The performance of sparse matrix operations tends to be lower than the dense matrix equivalent.
- Algorithms that are efficient for dense representations are not suitable for sparse representations.

- This paper aims at studying the performance of several strategies for distributing sparse-matrix arithmetic operations on computer clusters.

Table of Content

Basic Definitions

Parallel-Factor

$$\text{Parallel-Factor} = \lfloor \#\text{physical-cores} \times (1 + \alpha + \gamma) \rfloor$$

- Focus on the intrinsic characteristics of the operations and their associated matrices.
- Used to determine the number of rows assigned to each parallel task to be created and executed.
- Inversely related to the number of rows per tasks.
- As it might be zero, an additional constraint is introduced.
- Computed for three types of operations: Addition-Subtraction, Matrix Multiplication and Laplacian.

Basic Definitions

Gamma

$$\gamma = \begin{cases} 1 - log\left(\frac{\#rows}{\#columns}\right) & \#rows \leq \#columns \\ 1 - log\left(\frac{\#columns}{\#rows}\right) & \#rows > \#columns \end{cases}$$

- Defined as the ratio of the number of rows and columns.
- Shared by all the strategies.

Strategies

Row-Sparseness

$$\alpha = 1 - \overline{\left(\frac{\text{non-zero}_i}{\#\mathit{columns}}\right)}$$

- The general sparseness of a matrix might not accurately capture the sparseness of each particular row.
- Considers the mean row sparseness of the matrix.
- Aims at establishing an inverse relation between the PF and the row sparseness.

Strategies

Row-Sparseness Standard-Deviation

$$\alpha = 1 - \left(\overline{\left(\frac{\text{non-zero}_i}{\#\mathit{columns}} \right)} - \sigma \left(\frac{\text{non-zero}_i}{\#\mathit{columns}} \right) \right)$$

- Standard deviation measures de the dispersion of data from the mean.
- Aims at adding information regarding the existence of outliers.
- Considers the lowest sparseness value in the normal distribution.

Strategies

Mode

$$\alpha = 1 - \text{most-frequent}\left\{\frac{\text{non-zero}_i}{\#columns}\right\}$$

- Favours the most common sparseness value.
- May not accurately represent the data. There may be more than one value, or not value at all.
- Could cause an unbalanced distribution of rows per tasks.

Strategies

Static

$$\text{Parallel-Factor} = \lfloor \#\text{physical-cores} \times \text{granularity-factor} \rfloor$$

- Independent of the characteristics of the matrices involved.
- Allows to directly control the extend to which the operation is divided into tasks.
- Allows the creation of an arbitrary number of tasks. [ESTA O LA ANTERIOR]
- Strategy used for the Laplacian operation.

Table of Content

Table of Content

Feature Selection

- Strategies were evaluated for a feature selection approach.
- Considered not only features and posts, but also social context of post and user relationships.
- Social interactions lead to different types of relations.
- Based on high-dimensional matrices and arithmetic operations between them.

$$B = XX^T + \beta F L_A F^T$$

$$E = Y^T X^T = (XY)^T$$

Feature Selection

- Strategies were evaluated for a feature selection approach.
- Considered not only features and posts, but also social context of post and user relationships.
- Social interactions lead to different types of relations.
- Based on high-dimensional matrices and arithmetic operations between them.

$$B = XX^T + \beta F L_A F^T$$

$$E = Y^T X^T = (XY)^T$$

Dataset

- Experimental evaluation was based on data extracted from Digg.
- Digg is a social news website that allows its users to share and comment content.

Number of Posts	42,843
Number of Features	8,546
Number of Classes	51
Number of Following Relations	56,440
Average number of Following Relations	157
Average number of Features per Post	4
Average number of Posts per Class	840

Implementation

- Java was the programming language chosen for implementing the approach.
- Matrices were implemented as sparse memory structures in order to decrease the storage and network transfer requirements.
- The distribution and execution of tasks on the computer cluster was performed by using the Java Parallel Processing Framework (JPPF) middleware.
- The baseline for comparing and evaluating the enhancements introduced was the execution of all the operations in a serial and a multi-thread manner.

Implementation

- Java was the programming language chosen for implementing the approach.
- Matrices were implemented as sparse memory structures in order to decrease the storage and network transfer requirements.
- The distribution and execution of tasks on the computer cluster was performed by using the Java Parallel Processing Framework (JPPF) middleware.
- The baseline for comparing and evaluating the enhancements introduced was the execution of all the operations in a serial and a multi-thread manner.

Implementation

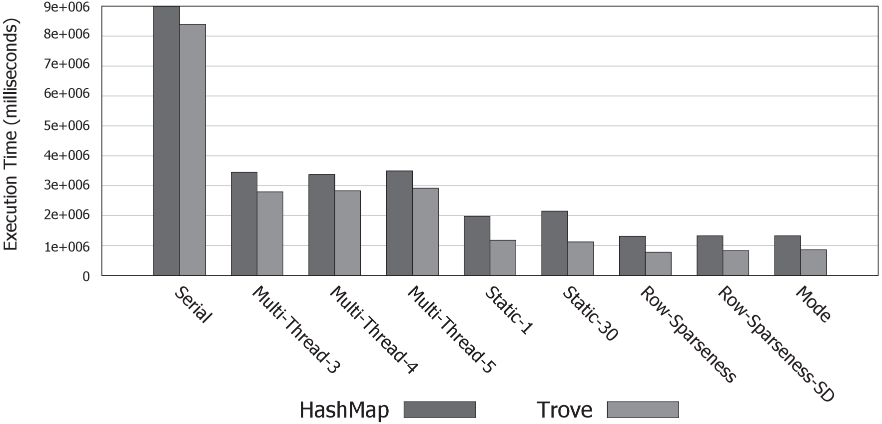
- Java was the programming language chosen for implementing the approach.
- Matrices were implemented as sparse memory structures in order to decrease the storage and network transfer requirements.
- The distribution and execution of tasks on the computer cluster was performed by using the Java Parallel Processing Framework (JPPF) middleware.
- The baseline for comparing and evaluating the enhancements introduced was the execution of all the operations in a serial and a multi-thread manner.

Table of Content

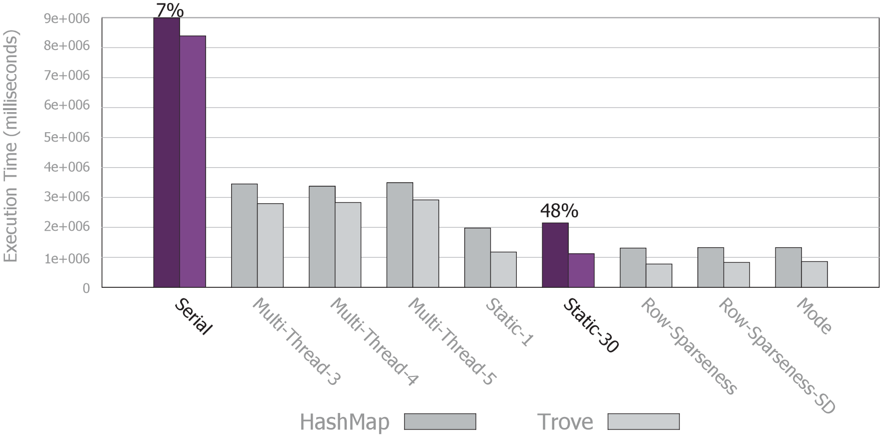
Matrices' size and sparseness per operation

	Operation	*Size*	*NonZeros*	*Sparseness*
B	Matrix Multiplication I	42,843x42,843	8,286,605	99.55%
	Addition-Subtraction II	42,843x42,843	8,285,916	99.54%
	Matrix Multiplication II	8,546x42,843	19,024,897	94.80%
	F^T	42,843x8,546	150,999	99.96%
	Matrix Multiplication III	8,546x8,546	24,142,734	66.94%
	X^T	42,843x8,546	150,999	99.96%
	Matrix Multiplication IV	8,546x8,546	386,736	99.47%
	Addition-Subtraction III	8,546x8,546	24,144,262	66.95%
E	Matrix Multiplication V	8,546x51	60,357	66.94%

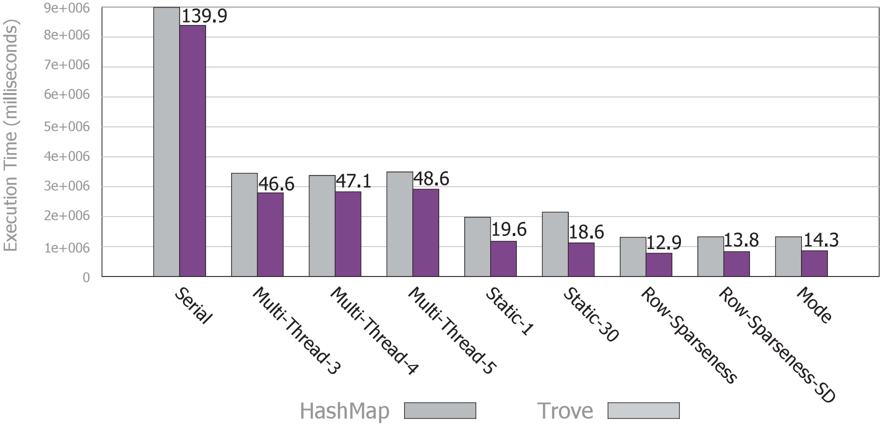
B Matrix Overall Time



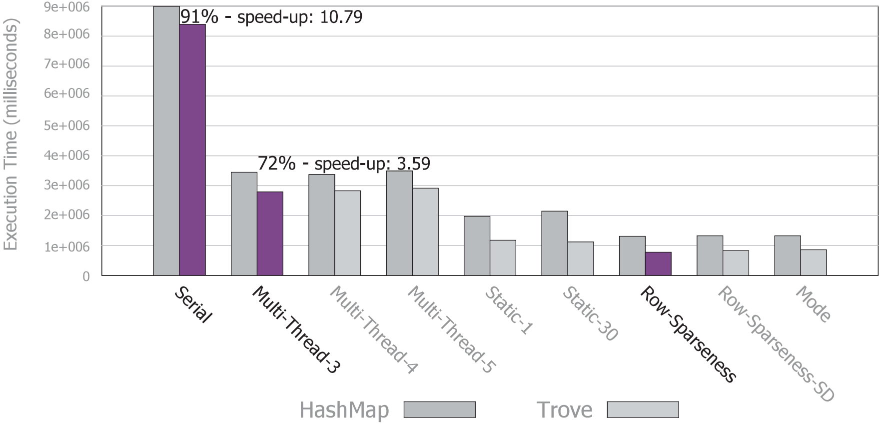
B Matrix Overall Time



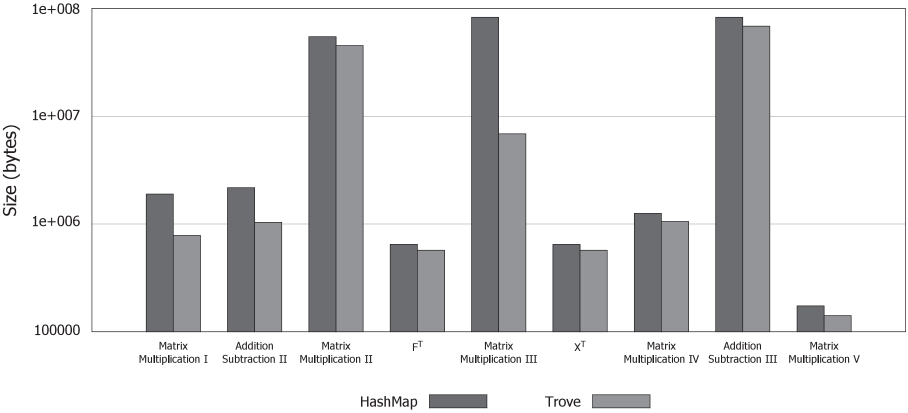
B Matrix Overall Time



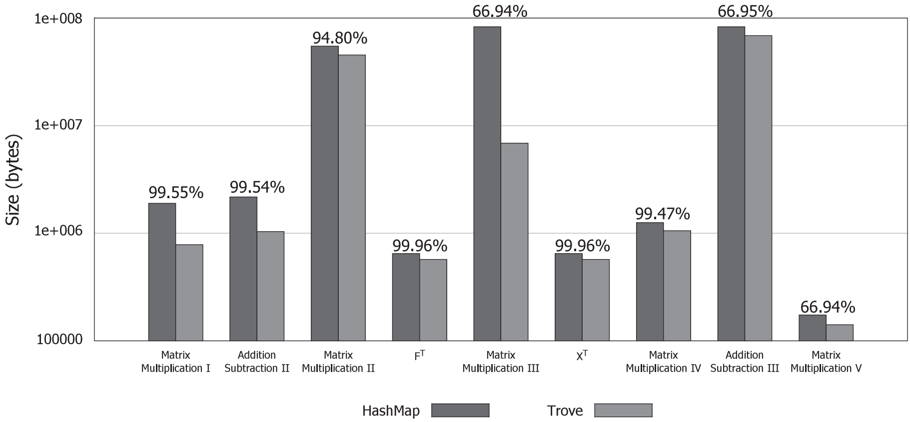
B Matrix Overall Time



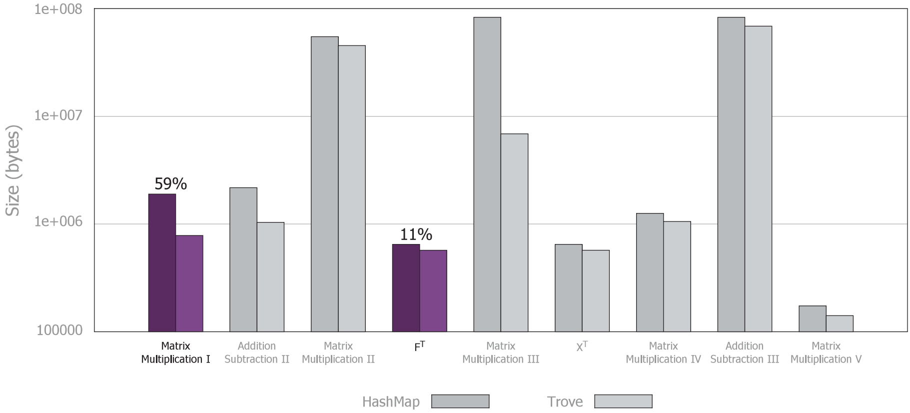
Matrix Size Comparison



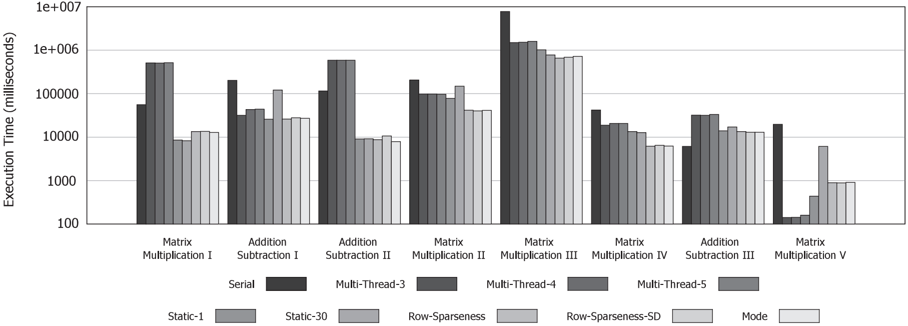
Matrix Size Comparison



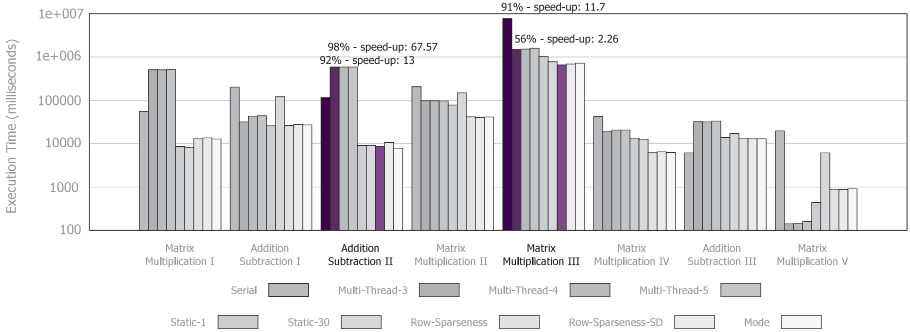
Matrix Size Comparison



Computing Time of All Individual Operations



Computing Time of All Individual Operations



Computing Time of All Individual Operations

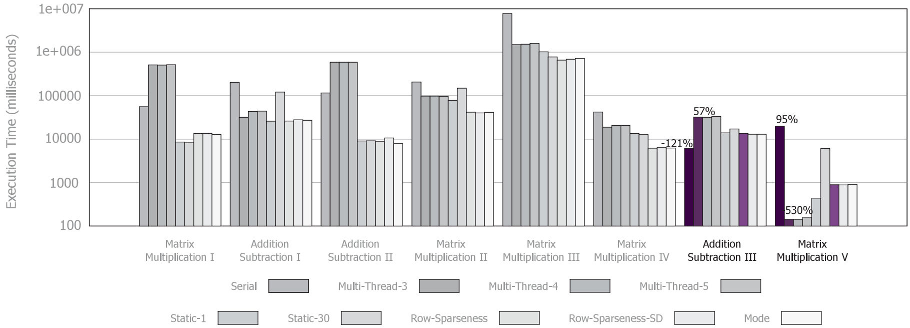


Table of Content

Summary

- This work aimed at studying the performance of several strategies for distributing sparse matrix arithmetic operations on computer clusters.
- The strategies focused on the intrinsic characteristics of the operations and their associated matrices.
- The performance of the proposed strategies was evaluated considering a high-dimensional feature selection approach.
- Two different implementation for sparse matrices were tested.

Conclusions

- The performance of the *Trove* representation of matrices was superior to the *HashMap*.
- The computer cluster executions outperformed the *Serial* and *Multi-Thread* executions when big-scale matrices were involved.
- The *Multi-Thread* executions tended to perform better than the computer cluster executions when small matrices were involved.

- Results stated the importance of considering the intrinsic characteristics of the matrices involved.
- The time spent computing the *PF* did not affect the overall performance of the strategies.

Conclusions

- The performance of the *Trove* representation of matrices was superior to the *HashMap*.

- The computer cluster executions outperformed the *Serial* and *Multi-Thread* executions when big-scale matrices were involved.
- The *Multi-Thread* executions tended to perform better than the computer cluster executions when small matrices were involved.

- Results stated the importance of considering the intrinsic characteristics of the matrices involved.
- The time spent computing the *PF* did not affect the overall performance of the strategies.

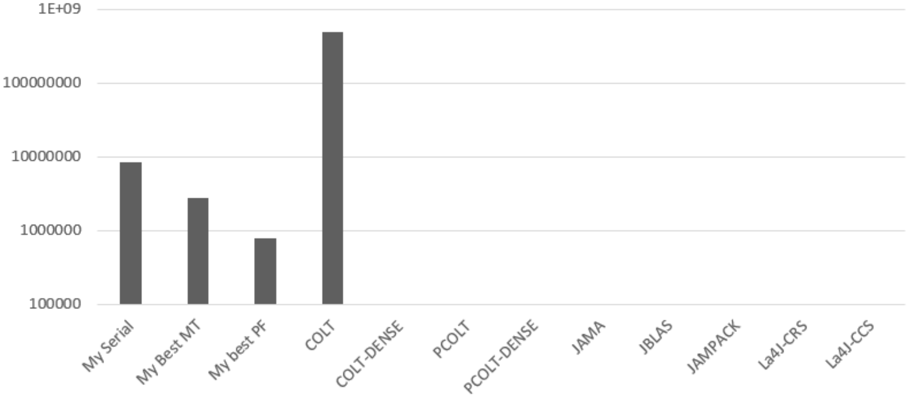
Conclusions

- The performance of the *Trove* representation of matrices was superior to the *HashMap*.
- The computer cluster executions outperformed the *Serial* and *Multi-Thread* executions when big-scale matrices were involved.
- The *Multi-Thread* executions tended to perform better than the computer cluster executions when small matrices were involved.
- Results stated the importance of considering the intrinsic characteristics of the matrices involved.
- The time spent computing the *PF* did not affect the overall performance of the strategies.

Questions

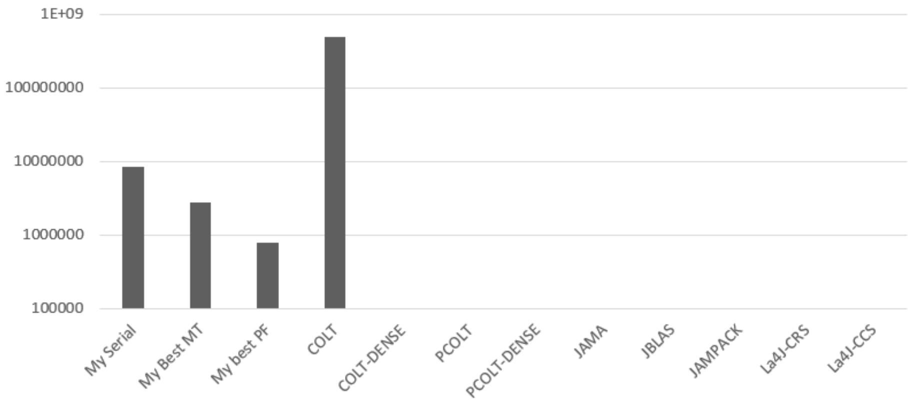
Linear Algebra Standard Libraries

Overall Computing Times



Linear Algebra Standard Libraries

Overall Computing Times [TIEMPO MINUTOS VS DIAS]



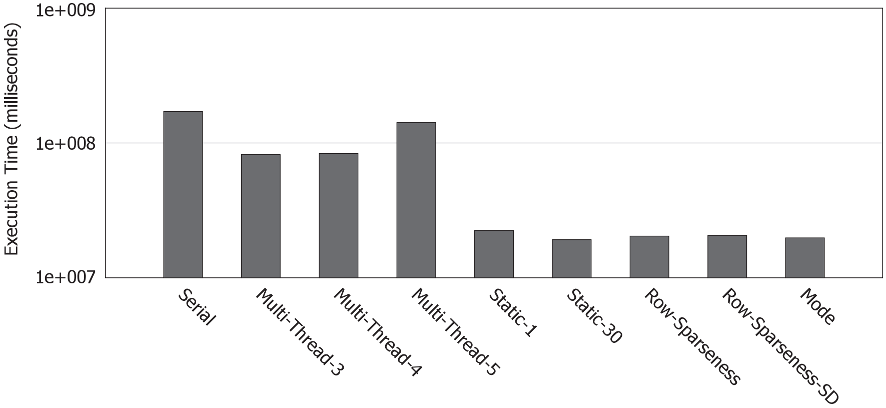
BlogCatalog Dataset

General Statistics

Number of Blogs	111,648
Number of Features	189,621
Number of Classes	11,701
Number of Following Relations	3,348,554
Average number of Following Relations	47
Average number of Features per Blog	139
Average number of Blogs per Class	10

BlogCatalog Dataset

Overall Computing Times



BlogCatalog Dataset

Overall Computing Times