Secure Strassen-Winograd Matrix Multiplication with MapReduce

Radu Ciucanu[1] Matthieu Giraud[2] Pascal Lafourcade[2]
Lihua Ye[3]

[1]LIFO, INSA Centre Val de Loire
Université d'Orléans

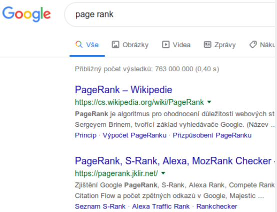
[2]LIMOS
University Clermont Auvergne, France

[3]School of Computer Science and Technology
Harbin Institute of Technology, China

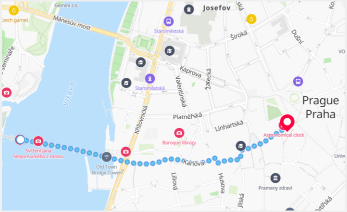
26 July 2019 @ SECRYPT, Prague

Matrix Multiplication

Applications



PageRank algorithm



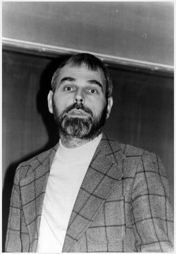
Shortest path problem

Genetic

Matrix Multiplication

Strassen-Winograd (SW) Algorithm[1]

Volker Strassen (1936–)



Shmuel Winograd (1936–)

- *Recursive* algorithm
- *Faster* than the standard algorithm: $\mathcal{O}(n^{2.81})$ vs $\mathcal{O}(n^3)$

[1] V. Strassen. *Gaussian Elimination is not Optimal*. In Numerische Mathematik, Vol. 13, 1969.

Contributions

1. SW matrix multiplication with MapReduce
2. Secure SW matrix multiplication with MapReduce
3. Comparison with CRSP MapReduce protocols[2]

[2] X. Bultel, R. Ciucanu, M. Giraud, and P. Lafourcade. *Secure Matrix Multiplication with MapReduce*. In the proceedings of ARES 2017.

Outline

MapReduce[3]

MapReduce Environment

Take care of:

- Partitioning input data
- Scheduling program execution on a set of machines
- Handling machine failures

Programmer

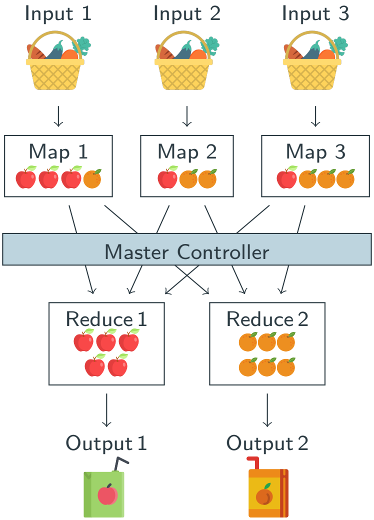
Specify:

- Map and Reduce functions
- Input files

[3] J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. In the proceedings of OSDI 2004.

MapReduce Example

Application: Make pure fruit juice



MapReduce in 3 Steps

1. Map tasks

Input: ID of chunk
Output: *key-value* pairs

2. Master Controller

- Key-value pairs aggregated and sorted by key
- Pairs with same key sent to the same Reduce task

3. Reduce tasks

Input: One key
Output: Combine values associated to the key

Standard Algorithm

Multiply square matrices: $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

1. 8 multiplications

$$S_1 = A_{11} \times B_{11} \qquad S_2 = A_{12} \times B_{21} \qquad S_3 = A_{21} \times B_{11} \qquad S_4 = A_{22} \times B_{21}$$
$$T_1 = A_{11} \times B_{12} \qquad T_2 = A_{12} \times B_{22} \qquad T_3 = A_{21} \times B_{12} \qquad T_4 = A_{22} \times B_{22}$$

2. 4 additions

$$U_1 = S_1 + S_2 \qquad U_2 = T_1 + T_2 \qquad U_3 = S_3 + S_4 \qquad U_4 = T_3 + T_4$$

3. Result is $\begin{bmatrix} U_1 & U_2 \\ U_3 & U_4 \end{bmatrix}$

Strassen-Winograd Algorithm

Multiply square 2-power matrices: $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

1. 8 additions

$$S_1 = A_{21} + A_{22} \quad S_2 = S_1 - A_{11} \quad S_3 = A_{11} - A_{21} \quad S_4 = A_{12} - S_2$$
$$T_1 = B_{12} - B_{11} \quad T_2 = B_{22} - T_1 \quad T_3 = B_{22} - B_{12} \quad T_4 = T_2 - B_{21}$$

2. 7 *recursive* multiplications

$$R_1 = A_{11} \times B_{11} \quad R_2 = A_{12} \times B_{21} \quad R_3 = S_4 \times B_{22} \quad R_4 = A_{22} \times T_4$$
$$R_5 = S_1 \times T_1 \quad R_6 = S_2 \times T_2 \quad R_7 = S_3 \times T_3$$

3. 7 final additions

$$U_1 = R_1 + R_2 \quad U_2 = R_1 + R_6 \quad U_3 = U_2 + R_7 \quad U_4 = U_2 + R_5$$
$$U_5 = U_4 + R_3 \quad U_6 = U_3 - R_4 \quad U_7 = U_3 + R_5$$

4. Result is $\begin{bmatrix} U_1 & U_5 \\ U_6 & U_7 \end{bmatrix}$

Standard vs Strassen-Winograd

For matrices of order $n = 2^k$

	# additions	# multiplications
Standard	$n^3 - n^2$	8^k
SW	$n^3 - \frac{n^2}{2} + \sum_{i=1}^{k} 7^i 4^{k-i}$	7^{k}

Static Padding

Add rows and columns to obtain matrices of 2-power order

$$
\begin{array}{cc}
\overbrace{\hphantom{a_{1,1} \; \cdots \; a_{1,n}}}^{2^{k-1} < n < 2^k} & \overbrace{\hphantom{0 \; \cdots \; 0}}^{2^k - n}
\end{array}
\qquad\qquad
\begin{array}{cc}
\overbrace{\hphantom{b_{1,1} \; \cdots \; b_{1,n}}}^{2^{k-1} < n < 2^k} & \overbrace{\hphantom{0 \; \cdots \; 0}}^{2^k - n}
\end{array}
$$

$$
\left[\begin{array}{ccc|ccc}
a_{1,1} & \cdots & a_{1,n} & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
a_{n,1} & \cdots & a_{n,n} & 0 & \cdots & 0 \\
\hline
0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right]
\qquad
\left[\begin{array}{ccc|ccc}
b_{1,1} & \cdots & b_{1,n} & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
b_{n,1} & \cdots & b_{n,n} & 0 & \cdots & 0 \\
\hline
0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right]
$$

Dynamic Padding

Add one row and one column when order is odd

$$\begin{array}{cc} \overbrace{\hphantom{a_{1,1} \quad \cdots \quad a_{1,n}}}^{n=2k-1} & \overbrace{\hphantom{0}}^{1} \end{array}$$

$$\left[\begin{array}{ccc|c} a_{1,1} & \cdots & a_{1,n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right] \qquad \left[\begin{array}{ccc|c} b_{1,1} & \cdots & b_{1,n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ b_{n,1} & \cdots & b_{n,n} & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]$$

Dynamic Peeling

"Remove" one row and one column when order is odd

$$\left[\begin{array}{ccc|c} \multicolumn{3}{c|}{\overbrace{\hphantom{a_{1,1} \quad \cdots \quad a_{n-1,n-1}}}^{n-1=2k}} & \overbrace{\hphantom{a_{n-1,n-1}}}^{1} \\ a_{1,1} & \cdots & a_{1,n-1} & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} & a_{n-1,n-1} \\ \hline a_{n,1} & \cdots & a_{n,n-1} & a_{n,n} \end{array}\right] \quad \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{\overbrace{\hphantom{b_{1,1} \quad \cdots \quad b_{n-1,n-1}}}^{n-1=2k}} & \overbrace{\hphantom{b_{n-1,n-1}}}^{1} \\ b_{1,1} & \cdots & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots \\ b_{n-1,1} & \cdots & b_{n-1,n-1} & b_{n-1,n-1} \\ \hline b_{n,1} & \cdots & b_{n,n-1} & b_{n,n} \end{array}\right]$$

Block matrix multiplication

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11} \times B_{11} + A_{12} \times B_{21} & A_{11} \times B_{12} + A_{12} \times B_{22} \\ A_{21} \times B_{11} + A_{22} \times B_{21} & A_{21} \times B_{12} + A_{22} \times B_{22} \end{bmatrix}$$

Strassen-Winograd with MapReduce

Execution

- 2 phases: *deconstruction* and *combination*
- Each phase is run (at most) $F = \log_2(d)$ times

Deconstruction phase

- Split matrices until the desired dimension (8 additions)
- Compute matrix multiplications

Combination phase

- Combine results of multiplications (7 additions)

Strassen-Winograd with MapReduce

Preprocessing

- Run on both matrices $A, B \in \mathbb{R}^{d \times d}$
- Each $a_{i,j} \in A$ gives key-value pair $(0, (\mathtt{A}, i, j, a_{i,j}, d))$
- Each $b_{j,k} \in B$ gives key-value pair $(0, (\mathtt{B}, j, k, b_{j,k}, d))$

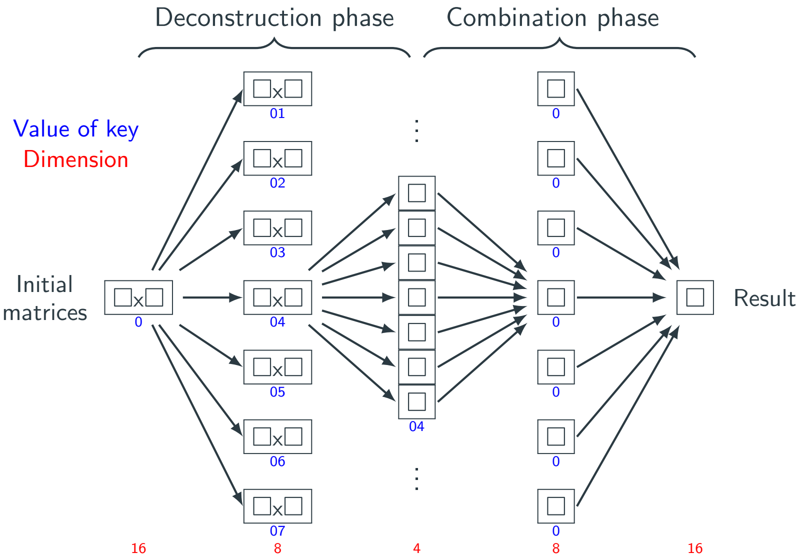
Reduce function (Deconstruction)

- Update value of key according to the recursive multiplication
- Perform additions and multiplications

Reduce function (Combination)

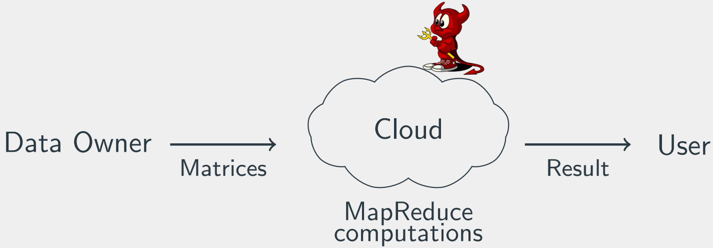
- Update value of key to build intermediate matrices
- Perform additions and combine matrices

SW Matrix Multiplication with MapReduce



Security Model

Cloud is **honest-but-curious**



Without security, Cloud learns:

- Content of both matrices
- Result

Cryptographic Tools

Additive homomorphic scheme

- $\mathcal{E}(pk, x_1) \otimes \mathcal{E}(pk, x_2) = \mathcal{E}(pk, x_1 + x_2)$
- $\mathcal{E}(pk, x_1) \oslash \mathcal{E}(pk, x_2) = \mathcal{E}(pk, x_1 - x_2)$
- Example: Paillier, Okamoto-Uchiyama cryptosystems...

Interactive homomorphic multiplication[4]

How to obtain $\mathcal{E}(pk, m_1 m_2)$ from $c_1 = \mathcal{E}(pk, m_1)$ and $c_2 = \mathcal{E}(pk, m_2)$?

$$\alpha_1 = c_1 \otimes \mathcal{E}(pk, r_1),\ \alpha_2 = c_2 \otimes \mathcal{E}(pk, r_2)$$

$$\begin{array}{lcl} \beta_1 = \mathcal{D}(sk, \alpha_1),\ \beta_2 = \mathcal{D}(sk, \alpha_2) & \xleftarrow{\alpha_1, \alpha_2} & \\ c = \mathcal{E}(pk, \beta_1 \times \beta_2) & \xrightarrow{c} & \end{array}$$

$$\mathcal{E}(pk, m_1 m_2) = c \oslash (\mathcal{E}(pk, r_1 r_2) c_1^{r_2} c_2^{r_1})$$

[4] Cramer et al. *Multiparty Computation from Threshold Homomorphic Encryption*. In the proceedings of EUROCRYPT 2001.

Secure SW Matrix Multiplication with MapReduce

8 additions

Use additive homomorphic properties $A \otimes B$ or $A \oslash B$

7 recursive multiplications

- Use interactive homomorphic multiplication
- Only applied during the *last* round of deconstruction phase

7 additions

Use additive homomorphic properties $A \otimes B$ or $A \oslash B$

Experimental Results

Settings

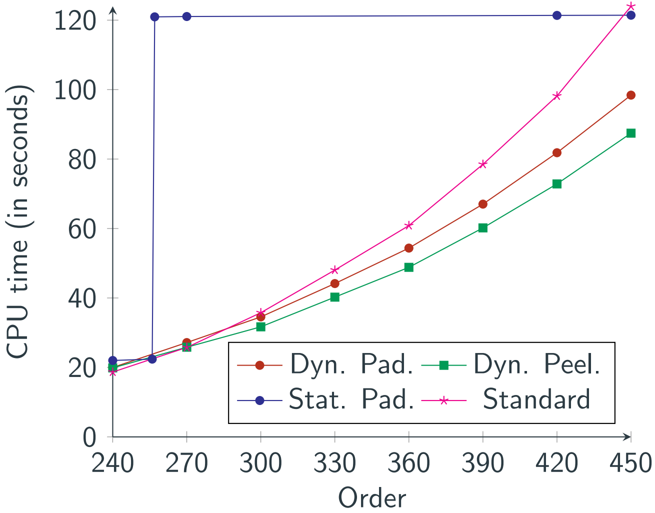
Software

- hadoop 3.2.0 / Standalone mode / Streaming
- ubuntu 16.04 LTS
- Map and Reduce functions in GO

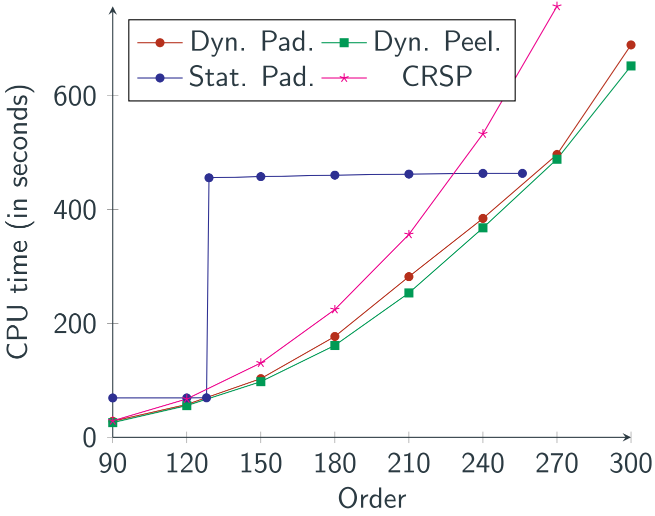
Hardware

- 4 CPU @ 2.4 GHz
- 80 Gb of disk
- 8 Gb of RAM

Experimental Results



Experimental Results



Conclusion

Conclusion

- Design MapReduce approach of SW algorithm
- Design a secure approach of SW algorithm with MapReduce
- Comparison with state-of-the-art MapReduce protocols

Future works

- Apache Spark experiment
- Malicious cloud
- Collusion resistance

Thank you for your attention

Any questions?

`matthieu.giraud@uca.fr`

Credit: Pascal Lafourcade