# Mathematical Approach to the Performance Evaluation of Matrix Multiply Algorithm

Luisa D'Amore[1(✉)], Valeria Mele[1], Giuliano Laccetti[1], and Almerico Murli[1,2]

[1] Department of Mathematics and Applications "Renato Caccioppoli", University of Naples Federico II, Via Cintia, 80126 Naples, Italy
{luisa.damore,valeria.mele,giuliano.laccetti,almerico.murli}@unina.it
[2] SPACI, University of Naples Federico II, Via Cintia, 80126 Naples, Italy
http://matematica.dip.unina.it

**Abstract.** Matrix multiplication (MM) is a computationally-intensive operation in many algorithms used in scientific computations. Not only one of the kernels in numerical linear algebra, the problem of matrix multiplication is also fundamental for almost all matrix problems such as least square and eigenvalues problem. The performance analysis of the MM needs to be re-evaluated to find out the best-practice algorithm on novel architectures. This motivated the analysis which is presented in this article and which is carried out by means of the new modelling framework that the authors have already introduced (L. D'Amore et al. *On a Mathematical Approach for Analyzing Parallel Algorithms*, 2015). The model exploits the knowledge of the algorithm and the multilevel parallelism of the target architecture and it could help the researchers for designing optimized MM implementations.

**Keywords:** Matrix-matrix multiply · Performance analysis · Multilevel paralllelism

## 1 Introduction

The authors proposed a performance model for analysing parallel algorithms. The model assumes that the (parallel) algorithm is represented as a set of operators related to each other according to a rule of dependence. Furthermore, the model has a parameterized formulation intended to exploit the different characteristics of the computing machines such as reconfigurable hardware devices [13].

Here we consider the matrix multiplication (MM) algorithm and we apply the performance model. The algorithm is simple and has not any ambition of optimization (many efforts are spent in the field of linear algebra and recent examples can be found in [1,9,11,12]), instead, our aim is to discuss how easily some implementation choices could be addressed giving rise to different performance results. The focus is on the "opportunity" of implementing the algorithm in hybrid distributed/shared memory computing environments, obtaining the most important information *before* the implementation. The implementations

R. Wyrzykowski et al. (Eds.): PPAM 2015, Part II, LNCS 9574, pp. 25–34, 2016.
DOI: 10.1007/978-3-319-32152-3_3

of MM algorithm will be composed from multiplications with sub matrices. The general MM algorithm can be decomposed into multiple calls to matrix multiplication. These themselves can be decomposed into multiple calls to inner-kernels. The aim now is to understand how these lowest level kernels can attain high performance, then so will the MM algorithm. This paper attempts to describe how to apply the performance model that the authors have developed so as to make it accessible to a broad audience.

## 2 Matrix Multiplication

Given two matrices $A$, $B \in \Re^{n\times n}$ and the computational problem

$$\mathcal{B}_{n^2} \equiv MM_{n\times n} := A \cdot B, \tag{1}$$

we introduce the sub problems $matmul^i_{\frac{n}{3}\times\frac{n}{3}}$, for $i = 0, \ldots, 26$ which are defined as follows:

$$\mathcal{B}_{\frac{n}{3}\times\frac{n}{3}} \equiv matmul^i_{\frac{n}{3}\times\frac{n}{3}} := C_i + A_i \cdot B_i, \tag{2}$$

with $A_i \in \Re^{\frac{n}{3}\times\frac{n}{3}}$, $B_i \in \Re^{\frac{n}{3}\times\frac{n}{3}}$ and $C_i \in \Re^{\frac{n}{3}\times\frac{n}{3}}$ blocks of $A$, $B$ and $C$, respectively. Finally, we introduce the decomposition

$$D_{27}(MM_{n\times n}) := \{matmul^i_{\frac{n}{3}\times\frac{n}{3}}\}_{0\le i<27}. \tag{3}$$

From (1)–(3), the decomposition matrix is:

$$M_D = \begin{bmatrix} matmul^0_{\frac{n}{3}\times\frac{n}{3}} & matmul^1_{\frac{n}{3}\times\frac{n}{3}} & matmul^2_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^8_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^9_{\frac{n}{3}\times\frac{n}{3}} & matmul^{10}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{11}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{17}_{\frac{n}{3}\times\frac{n}{3}} \\ matmul^{18}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{19}_{\frac{n}{3}\times\frac{n}{3}} & matmul^{20}_{\frac{n}{3}\times\frac{n}{3}} & \cdots & matmul^{26}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix} \tag{4}$$

The set $D_{27}(MM_{n\times n})$ is made of 27 subproblems $matmul^{(i+j+k)}_{\frac{n}{3}\times\frac{n}{3}} \in D_{27}$, and the problem $MM_{n\times n}$ has concurrence degree $r_D = 9$ and dependence degree $c_D = 3$.

Suppose that the computing environment can be represented by means of the machine $\mathcal{M}_{1,1}$ which has

- $P = 1$,
- $Op_{\mathcal{M}_{1,1}} = \{\otimes, ...\}$ where $\otimes :=$ matrix-matrix multiply,
- $L = 2$ two memory levels,
- $rmem_i$ (read) and $wmem_j$ (write) as memory accesses operators on blocks of size $\frac{n}{3} \times \frac{n}{3}$,
- $tmem_1 := tblock_{mem}$,
- for each $\otimes$, 1 read (before the execution) and 1 write (after the execution) are needed.

According to $D_{27}$, the sequential algorithm $A_{D_{27},M_{1,1}}$ on $\mathcal{M}_{1,1}$ is made of the 27 operators $\otimes$ corresponding to the 27 sub-problems. The execution matrix corresponding to $A_{D_{27},M_{1,1}}$ on $\mathcal{M}_{1,1}$ has $r_E = 27$ rows and only one column, i.e. $c_E = 1$. It is the following matrix:

$$M_E = \begin{bmatrix} \otimes_0 \\ \otimes_1 \\ \vdots \\ \otimes_{26} \end{bmatrix} \tag{5}$$

while the memory matrix $AM_{A_{D_{27},\mathcal{M}_{1,1}}}$ has $r_{MEM} = 52$ rows and $c_{MEM} = 1$ column, and it can be described in the following way:

$$AM_{A_{D_{27},\mathcal{M}_{1,1}}} = \begin{bmatrix} rmem_0(\cdot) \\ wmem_0(\cdot) \\ rmem_1(\cdot) \\ wmem_1(\cdot) \\ \vdots \\ rmem_{26} \\ wmem_{26} \end{bmatrix} \tag{6}$$

The execution time of algorithm $A_{D_{27},M_{1,1}}$ is

$$T(A_{D_{27},\mathcal{M}_{1,1}}) = r_E \cdot T_r \tag{7}$$

where $T_r$ is the execution time of the row $r$ of the matrix given in (5). It is equal to the execution time of the $\otimes$ operator (since they are all the same).
Let $C(\otimes)$ denote the complexity of $\otimes$ operator, then (7) becomes:

$$T(A_{D_{27},\mathcal{M}_{1,1}}) = 27 \cdot C(\otimes) \cdot tcalc \tag{8}$$

The memory access time of the software corresponding to $A_{D_{27},M_{1,1}}$, is

$$T_M(SW(A_{D_{27},\mathcal{M}_{1,1}}(A))) = r_{mem_1} \cdot tblock_{mem} = 54 \cdot tblock_{mem}, \tag{9}$$

and its execution time is

$$\begin{aligned} T(SW(A_{D_{27},\mathcal{M}_{1,1}})) &= T(A_{D_{27},\mathcal{M}_1}) + T_M(SW(A_{D_{27},\mathcal{M}_{1,1}})) \\ &= 27 \cdot C(\otimes) \cdot tcalc + 54 \cdot tblock_{mem} \end{aligned} \tag{10}$$

### 2.1 The Algorithm at the First Level of Decomposition

We consider the machine $\mathcal{M}_{9,9}$ such that

- $P = 9$ (which we call nodes), which are organized in a $3 \times 3$ logical grid,
- $Op_{\mathcal{M}_{9,9}} = \{\otimes, ...\}$ where $\otimes$ = matrix-matrix multiply,
- $L = 3$ (two memory levels plus one level for communications),
- $trans_i$ denotes the memory access operator which moves a block of size $\frac{n}{3} \times \frac{n}{3}$ in time $tblock_{com}$[1],

[1] Note that typically $tblock_{com} >> tblock_{mem}$.

– each node can transfer a single block concurrently, that is the machine can transfer 9 blocks at the same time.
– for a broadcast step, each node performs a transfer (one send, other eight receive).
– for a rolling step, each node performs two transfers (send and receive one block).

Starting, each node has a $\frac{n}{3} \times \frac{n}{3}$ block of each matrix. If $matmul(p \cdot i)$ is the subproblem $matmul^{p \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D_{27}$, the algorithm $A_{D_{27},\mathcal{M}_{9,9}}$ is the following (i.e. the so called Broadcast Multiply Rolling (BMR) Algorithm [10]) (Fig. 1):

```
for p=1 to 3
   nodes on the i-th grid diagonal broadcast their A block
      to all its grid row;
   the node i solves  matmul(i*p);
   if p<3, each node sends its B block to the upper one
      on the same column of the grid (rolling).
endfor
```

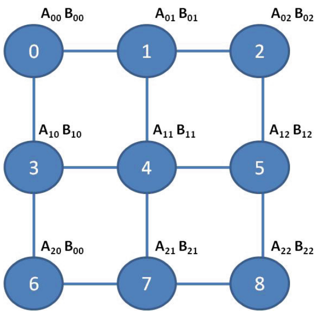

**Fig. 1.** The starting matrices blocks distribution among the nodes.

The execution matrix of $A_{D_{27},\mathcal{M}_{9,9}}$ is

$$M_E = \begin{bmatrix} \otimes_0 & \otimes_1 & \otimes_2 & \otimes_3 & \otimes_4 & \otimes_5 & \otimes_6 & \otimes_7 & \otimes_8 \\ \otimes_9 & \otimes_{10} & \otimes_{11} & \otimes_{12} & \otimes_{13} & \otimes_{14} & \otimes_{15} & \otimes_{16} & \otimes_{17} \\ \otimes_{18} & \otimes_{19} & \otimes_{20} & \otimes_{21} & \otimes_{22} & \otimes_{23} & \otimes_{24} & \otimes_{25} & \otimes_{26} \end{bmatrix} \tag{11}$$

and it is perfectly parallel. The memory matrix is

$$AM_{A_{D_{27},\mathcal{M}_{9,9}}} = \begin{bmatrix} trans_0(\cdot) & trans_1(\cdot) & trans_2(\cdot) & \dots & trans_8(\cdot) \\ trans_9(\cdot) & trans_{10}(\cdot) & trans_{11}(\cdot) & \dots & trans_{17}(\cdot) \\ trans_{18}(\cdot) & trans_{19}(\cdot) & trans_{20}(\cdot) & \dots & trans_{26}(\cdot) \\ trans_{27}(\cdot) & trans_{28}(\cdot) & trans_{29}(\cdot) & \dots & trans_{35}(\cdot) \\ trans_{36}(\cdot) & trans_{37}(\cdot) & trans_{38}(\cdot) & \dots & trans_{44}(\cdot) \\ trans_{45}(\cdot) & trans_{46}(\cdot) & trans_{47}(\cdot) & \dots & trans_{53}(\cdot) \\ trans_{54}(\cdot) & trans_{55}(\cdot) & trans_{56}(\cdot) & \dots & trans_{62}(\cdot) \end{bmatrix} \tag{12}$$

The execution time of each row of $M_E$, is the execution time of the $\otimes$ operator. If $r_E = 3$ is the number of rows of $E_{A_{D_{27},\mathcal{M}_{9,9}}}$, the execution time of the BMR algorithm $A_{D_{27},\mathcal{M}_{9,9}}$ is

$$T(A_{D_{27},\mathcal{M}_{9,9}}) = r_E \cdot T_r = 3 \cdot C(\otimes) \cdot tcalc \tag{13}$$

Since $r_{mem} = 7$, the memory access time of the software $SW(A_{D_{27},\mathcal{M}_{9,9}})$ is

$$T_M(SW(A_{D_{27},\mathcal{M}_{9,9}})) = r_{mem_9} \cdot tblock_{com} = 7 \cdot tblock_{com} \tag{14}$$

and its execution time is

$$\begin{aligned} T(SW(A_{D_{27},\mathcal{M}_{9,9}})) &= T(A_{D_{27},\mathcal{M}_{9,9}}) + T_M(SW(A_{D_{27},\mathcal{M}_{9,9}})) \\ &= 3 \cdot C(\otimes) \cdot tcalc + 7 \cdot tblock_{com}. \end{aligned} \tag{15}$$

Finally, the speed up of the software $SW(A_{D_{27},\mathcal{M}_{9,9}})$ is

$$Sp(SW(A_{D_{27},\mathcal{M}_{9,9}})) = \frac{T(SW(A_{D_{27},\mathcal{M}_{1,1}}))}{T(SW(A_{D_{27},\mathcal{M}_{9,9}}))} = \frac{26 \cdot C(\otimes) \cdot tcalc + 52 \cdot tblock_{mem}}{3 \cdot C(\otimes) \cdot tcalc + 7 \cdot tblock_{com}} \tag{16}$$

## 2.2 The Sequential Algorithm at the Second Level of Decomposition

Consider the subproblem $matmul^i_{\frac{n}{3}\times\frac{n}{3}}$ and the decomposition

$$D'_{\frac{n}{3}-1} = \{matvec^i_{\frac{n}{3}\times\frac{n}{3}}\}_{0\le i<(\frac{n}{3}-1)} \tag{17}$$

where

$$\begin{aligned} matvec^i_{\frac{n}{3}\times\frac{n}{3}} :=& \text{multiply of a block } A_i \text{ of } \frac{n}{3}\times\frac{n}{3} \\ & \text{elements and a vector } B_i \text{ of } \frac{n}{3} \text{ elements.} \end{aligned} \tag{18}$$

All the subproblems are independent, so the decomposition matrix of $matmul^i_{\frac{n}{3}\times\frac{n}{3}}$ is

$$M_{D'_{\frac{n}{3}-1}} = \left[matvec^0_{\frac{n}{3}\times\frac{n}{3}}\ matvec^1_{\frac{n}{3}\times\frac{n}{3}}\ \dots\ matvec^{\frac{n}{3}-1}_{\frac{n}{3}\times\frac{n}{3}}\right] \tag{19}$$

and $matmul^i_{\frac{n}{3}\times\frac{n}{3}}$ has concurrence degree $\frac{n}{3}$ and dependence degree 1.

Let us introduce the machine $\mathcal{M}'_{1,1}$ corresponding to a generic node of $\mathcal{M}_{9,9}$. Suppose that $\mathcal{M}'_{1,1}$ is such that

- $P = 1$,
- $Op_{\mathcal{M}'_{1,1}} = \{\boxtimes, \dots\}$ where $\boxtimes =$ matrix-vector multiply,
- $L = 2$,
- $rmemv_i$ (read) or $wmemv_j$ (write) denote the memory accesses operators moving a vector of size $\frac{n}{3}$ in time $tmem := tvec_{mem}$[2].

[2] Typically $tvec_{mem} \le tblock_{mem}$.

Since all the subproblems must be solved one after another, the execution matrix of $A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}$ is

$$M_E = \begin{bmatrix} \boxtimes_0 \\ \boxtimes_1 \\ \vdots \\ \boxtimes_{\frac{n}{3}-1} \end{bmatrix} \tag{20}$$

Since we assume that for the execution of each operator, it is required one read (before the execution) and one write (after the execution) of a vector of size $\frac{n}{3}+1$, the memory matrix has

$$r_{mem,D'_{\frac{n}{3}-1}} = \left(\frac{n}{3}+2\right)\cdot\frac{n}{3}$$

rows. The execution time of each row of the matrix in (20) is the execution time of the $\boxtimes$ operator. If

$$r_{E,D'_{\frac{n}{3}-1}} = \frac{n}{3}$$

is the number of rows of $E_{A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}}$, the execution time of the algorithm $A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}$ is

$$\begin{aligned} T(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}) &= C(\otimes)\cdot tcalc = r_{E1D'_{\frac{n}{3}-1}}\cdot T_r \\ &= \frac{n}{3}\cdot C(\boxtimes)\cdot tcalc = \frac{n}{3}\cdot 2\cdot\left(\frac{n}{3}\right)^2\cdot tcalc \\ &= 2\cdot\left(\frac{n}{3}\right)^3\cdot tcalc \end{aligned} \tag{21}$$

and the memory access time of the software $SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}})$ is

$$T_M(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}})) = r_{mem,D'_{\frac{n}{3}-1}}\cdot tvec_{com} = \left(\frac{n}{3}+2\right)\cdot\frac{n}{3}\cdot tvec_{mem}. \tag{22}$$

Finally, the execution time of the software $SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}})$ is

$$\begin{aligned} T(SW(A_{D'_{\frac{n}{3}-1}},\mathcal{M}'_{1,1})) &= T(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}) + T_M(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}})) \\ &= 2\cdot\left(\frac{n}{3}\right)^3\cdot tcalc + \left(\frac{n}{3}+2\right)\cdot\frac{n}{3}\cdot tvec_{mem} \end{aligned} \tag{23}$$

## 2.3 The Parallel Algorithm at the Second Level of Decomposition

We consider the machine $\mathcal{M}'_{1\cdot 8}$ made of 8 cores/threads for each node of $\mathcal{M}_{9,9}$. Let us assume that $\mathcal{M}'_{1\cdot 8}$ is such that

- $P = 8$,
- $Op_{\mathcal{M}'_{1\cdot 8}} = \{\boxtimes, ...\}$ where $\boxtimes$ = matrix-vector multiply,
- $L = 2$,

– $rmemv_i$ (read) or $wmemv_j$ (write) denote the memory access operators on a vector of $\frac{n}{3}$ elements concurrently in time $tvec_{mem}$ between the memory levels. Note that $tvec_{mem} \leq tblock_{mem}$.

Then, if $matvec(t \cdot i)$ denotes subproblem $matvec^{t \cdot i}_{\frac{n}{3} \times \frac{n}{3}} \in D'_{\frac{n}{3}-1}$, we get the Multi Thread Matrix multiply Algorithm $A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{1 \cdot 8}}$:

```
for i:=1 to n/(9*8)
  each thread t solves matvec(t*i)
endfor
```

The first 8 of the $\frac{n}{3}$ subproblems can be solved independently by the 8 cores, and so on until they are all completed. Hence, the execution matrix of the algorithm $A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{1 \cdot 8}}$ has $r_E = \frac{n}{3 \cdot 8} = \frac{n}{24}$ rows and if we assume that $\frac{n}{3}$ is a multiple of 8[3], the algorithm is perfectly parallel.

Assuming that for the execution of each operator, it is required to read (before the execution) and to write (after the execution) the vector of size $\frac{n}{3} + 1$ and that the cores can transfer their vectors concurrently, that is the machine can concurrently transfer 8 vectors, the memory matrix $AM_{A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}}}$ has

$$r_{mem, D'_{\frac{n}{3}-1}} = \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24}$$

rows. The execution time of each row of the execution matrix is the execution time of the $\boxtimes$ operator. If $r_E = \frac{n}{24}$ is the number of rows of $E_{A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{1 \cdot 8}}}$, the execution time of the algorithm $A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{1 \cdot 8}}$ is

$$T(A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{1 \cdot 8}}) = r_E \cdot T_r = \frac{n}{24} \cdot C(\boxtimes) \cdot tcalc = \frac{n}{24} \cdot 2 \cdot \left(\frac{n}{3}\right)^2 \cdot tcalc. \tag{24}$$

If we denote by $r_{mem, D'_{\frac{n}{3}-1}} = \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24}$ the number of rows of the memory access matrix of the algorithm $A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}}$, the memory access time of the software $SW(A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}})$ we are going to implement is

$$T_M(SW(A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}})) = r_{mem, D'_{\frac{n}{3}-1}} \cdot tvec_{mem} = \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem} \tag{25}$$

and the execution time of the software $SW(A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}})$ is

$$\begin{aligned} T(SW(A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{1 \cdot 8}})) &= T(A_{D'_{\frac{n}{3}-1}, \mathcal{M}'_{8,8}}) + T_M(SW(A_{D'_{\frac{n}{3}-1}, \mathcal{M}_{8,8}})) \\ &= \frac{n}{24} \cdot 2 \cdot \left(\frac{n}{3}\right)^2 \cdot tcalc + \left(\frac{n}{3} + 2\right) \cdot \frac{n}{24} \cdot tvec_{mem}. \end{aligned} \tag{26}$$

[3] There is no loss of generality.

Finally, the speed up is

$$\begin{aligned} Sp(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1\cdot 8}})) &= \frac{T(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}'_{1,1}}))}{T(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}_{1\cdot 8}}))} \\ &= \frac{2\cdot\left(\frac{n}{3}\right)^3\cdot tcalc+\left(\frac{n}{3}+2\right)\cdot\frac{n}{3}\cdot tvec_{mem}}{\frac{n}{24}\cdot 2\cdot\left(\frac{n}{3}\right)^2\cdot tcalc+\left(\frac{n}{3}+2\right)\cdot\frac{n}{24}\cdot tvec_{mem}} > 1 \end{aligned} \tag{27}$$

Let $A'_{D_{\frac{n}{3}-1},\mathcal{M}_{9\cdot 8}}$ denote the algorithm that uses 9 nodes and 8 cores per node. We get the following expression of the speed up the algorithm that uses 1 level of parallelism in $M_{9,9}$

$$\begin{aligned} Sp(SW(A_{D_{27},\mathcal{M}_{9,9}})) &= \frac{T(SW(A_{D_{27},\mathcal{M}_{1,1}}))}{T(SW(A_{D_{27},\mathcal{M}_{9,9}}))} \\ &= \frac{26\cdot C(\otimes)\cdot tcalc+52\cdot tblock_{mem}}{3\cdot C(\boxtimes)\cdot tcalc+7\cdot tblock_{com}} \end{aligned} \tag{28}$$

which should be compared to the speed up of the algorithm that uses 2 levels of parallelism in $M_{9\cdot 8}$

$$\begin{aligned} Sp(SW(A_{D'_{\frac{n}{3}-1},\mathcal{M}_{9\cdot 8}})) &= \frac{T(SW(A_{D_{27},\mathcal{M}_{1,1}})}{T(SWA_{D'_{\frac{n}{3}-1},\mathcal{M}_{9\cdot 8}})} \\ &= \frac{26\cdot\left(\frac{n}{3}\cdot C(\boxtimes)\cdot tcalc+(\frac{n}{3}+2)\cdot tvec_{mem}\right)+52\cdot tblock_{mem}}{3\cdot\frac{n}{24}\cdot\left(C(\boxtimes)\cdot tcalc+\left(\frac{n}{3}+2\right)\cdot tvec_{mem}\right)+7\cdot tblock_{com}} \end{aligned} \tag{29}$$

By specializing the parameters we can estimate the performance gain that we get using two levels of parallelism instead of one.

## 3 Conclusion

Matrix multiplication is one of the fundamental kernels in numerical linear algebra, for almost all matrix problems such as least square problem eigenvalue problem and data assimilation problem [5–8,14]. Future designs of microprocessors and large HPC systems will be heterogeneous in nature, relying on the integration of two major types of components. On the first hand, multi/many-cores CPU technology have been developed and the number of cores will continue to escalate because of the desire to pack more and more components on a chip while avoiding the power wall, instruction level parallelism wall, and the memory wall. On the other hand special purpose hardware and accelerators, especially Graphics Processing Units (GPUs) are in commodity production, and have outpaced standard CPUs in floating point performance in recent years, and have become as easy, if not easier to program than multi-core CPUs. Finally, reconfigurable architectures such as Field programmable Gate Arrays (FPGAs) offer several parameters such as operating frequency, precision, amount of memory, number of computation units, etc. These parameters define a large design space that must be explored to find efficient solutions.

To address this scenario, it is undoubted that performance analysis of MM algorithm should be re-evaluated to find out the best-practice algorithm on novel architectures. This motivated the work to investigate the performance of the standard MM algorithm, by means of the new modelling framework that the authors have introduced.

This paper attempts to describe how to apply the performance model that the authors have developed so as to make it accessible to a broad audience. The model exploits the knowledge of the algorithm and the target architecture and it could help the researchers for designing optimized implementations on emerging computing architectures, such as that one developed in [3,4].

## References

1. Ballard, G., Demmel, J., Holtz, O., Schwartz, O.: Minimizing communication in numerical linear algebra. SIAM J. Matrix Anal. Appl. **32**(3), 866–901 (2011)
2. Cuomo, S., D'Amore, L., Murli, A., Rizzardi, M.: Computation of the inverse Laplace transform based on a collocation method which uses only real values. J. Comput. Appl. Math. **198**(1), 98–115 (2007)
3. D'Amore, L., Laccetti, G., Romano, D., Scotti, G., Murli, A.: Towards a parallel component in a GPU-CUDA environment: a case study with the L-BFGS Harwell routine. Int. J. Comput. Math. **92**(1), 59–76 (2015)
4. D'Amore, L., Casaburi, D., Galletti, A., Marcellino, L., Murli, A.: Integration of emerging computer technologies for an efficient image sequences analysis. Integr. Comput.-Aided Eng. **18**(4), 365–378 (2011)
5. D'Amore, L., Murli, A.: Regularization of a Fourier series method for the Laplace transform inversion with real data. Inverse Prob. **18**(4), 1185–1205 (2002)
6. D'Amore, L., Arcucci, R., Carracciuolo, L., Murli, A.: DD-OceanVar: a domain decomposition fully parallel data assimilation software in mediterranean sea. Procedia Comput. Sci. **18**, 1235–1244 (2013)
7. D'Amore, L., Arcucci, R., Carracciuolo, L., Murli, A.: A scalable approach to variational data assimilation. J. Sci. Comput. **61**, 239–257 (2014)
8. D'Amore, L., Arcucci, R., Marcellino, L., Murli, A.: HPC computation issues of the incremental 3D variational data assimilation scheme in OceanVar software. J. Numer. Anal. Ind. Appl. Math. **7**(3–4), 91–105 (2012)
9. Demmel, J., Eliahu, D., Fox, A., Kamil, S., Lipshitz, B., Schwartz, O., Spillinger, O.: Communication-optimal parallel recursive rectangular matrix multiplication. In: Proceedings of the IEEE 27th International Symposium on Parallel and Distributed Processing (IPDPS 2013), pp. 261–272. IEEE Computer Society, Washington, D.C. (2013)
10. Fox, G., Otto, S., Hey, A.: Matrix algorithms on a hypercube I: matrix multiplication. Parallel Comput. **3**(5), 17–31 (1987)
11. Gunnels, J.A., Henry, G.M., van de Geijn, R.A.: A family of high-performance matrix multiplication algorithms. In: Alexandrov, V.N., Dongarra, J., Juliano, B.A., Renner, R.S., Tan, C.J.K. (eds.) ICCS-ComputSci 2001. LNCS, vol. 2073, pp. 51–60. Springer, Heidelberg (2001)

12. Gunnels, J.A., Gustavson, F.G., Henry, G.M., van de Geijn, R.A.: FLAME: formal linear algebra methods environment. ACM Trans. Math. Softw. **27**(4), 422–455 (2001)
13. Kuon, I., Tessier, R., Rose, J.: FPGA architecture: survey and challenges. Found. Trends Electron. Des. Autom. **2**(2), 135–253 (2007)
14. Murli, A., Cuomo, S., D'Amore, L., Galletti, A.: Numerical regularization of a real inversion formula based on the Laplace transform's eigenfunction expansion of the inverse function. Inverse Prob. **23**(2), 713–731 (2007)