The analysis of constructing and evaluating tensor operation paralleling algorithms

Rusul J. Alsaedi¹, Isam K. Shallal²,* and Qasim S. Kadhim³

¹ Al-Qasim Green University, Babil, Iraq
² University of Anbar, Anbar, Iraq
³ University of Babylon, Babil, Iraq

E-mail: esam.khamis@uoanbar.edu.iq

Abstract. The algorithms of tensors' summing, multiplying and collapsing are observed in that issue from the perspectives of those paralleling possibilities. The graphs of these algorithms are developed and analyzed from the point of the forecasted values of the acceleration and efficiency. It is assumed that the time of execution for all computing operations is same and equal to a unit of time, and data transfer between computer devices is performed instantaneously without any time consuming (it is acceptable, for example, a parallel computing systems with shared memory). In particular, it is shown that for the tensors’ addition the time of the fastest execution of algorithm for an unlimited number of processors is equal to the length of the maximum path in the graph. In other words, the minimum time of the algorithm will be achieved when the number of processors is equal to the number of components of the tensor. A similar analysis was performed for the algorithms of multiplication and convolution of tensors.

Keywords: Tensor operations; parallel computations; the algorithm’s graph; speed; efficiency.

1. Introduction

Tensor calculations have a wide range of applications niya. They are a necessary tool in various natural sciences, such as geometry, physics, mechanics, quantum chemistry, crystal physics and many others. In particular, the development of the quantum theory of relativity, the theory of combined fields, the theory of nanostructures is carried out mainly by the methods of tensor calculus [1]. The use of the tensor apparatus implies a large amount of calculations that are extremely difficult to do manually, therefore it is necessary Automation when performing tensor operations. Most often, when conducting research, the following approaches to automating tensor calculations are used: development of software systems using high-level languages; use of specialized mathematical packages; modeling operations on tensors on neural networks [2]. The development of software systems is usually used to solve a specific problem (or class of problems) and does not imply universality. All information about the tensors used and the operations performed on them are entered at the design stage, and the user only sets certain parameters for the calculations. Examples of such complexes are: a software package for automating the simulation of non-stationary processes in mechanical systems and systems of a different physical...
nature PRADIS (module "Elements of a continuous medium", in which the calculation of the stress tensor components is used to calculate the stress intensity); ESPRI 2013 (sections "Steel structures" and "Reinforced concrete structures"), in which, based on the values of the stress tensor, the values of the principal deformations, the angles of inclination of the principal stresses to the current axes, as well as the angle of inclination of the crack plane to the X-axis are determined. means it is necessary to automate the basic operations on tensors: addition, multiplication, multiplication by a number, convolution, summarization, alternation and covariant differentiation of tensors. To perform such operations, it is possible to use mathematical packages. The most popular software packages for working with multidimensional objects are Maple, Mathcad, Matlab, and Mathematica. There are no specialized tools for working with tensors in the Mathcad, Matlab and Mathematica packages, but a tensor can be represented as a multi-index (nested) array, and operations on it are performed as operations on matrices. Performing operations on tensors, especially in multidimensional spaces, makes the question relevant about their parallelization. As one of the possibilities for parallelization, the papers [5, 6, 7 and 8] considered models of neural networks for the implementation of tensor algebra operations: addition, multiplication of tensors, multiplication of a tensor by a scalar, summarization and alternation of a tensor.

2. Materials Research Methods
Consider algorithms for performing some algebraical operations on tensors from the point of view of parallelization. One of the most important points in the development of parallel computational algorithms is the assessment of the resulting acceleration of the computational process (reducing the execution time of calculations)[4]. We will represent the algorithms in the form of an acyclic directed graph of the algorithm $G = (V, R)$, where $V$ is the set of graph vertices, $R$ is the set of graph arcs [1, 3]. The operations of the algorithm correspond to the vertices of the graph. The arc $r = (i, j)$ belongs to the graph only if operation $j$ uses the result of operation $i$. For simplicity of the model, we will assume that the execution time of any computational operations is the same and is equal to a certain unit of time. In addition, we will assume that data transfer between computing devices is performed instantly without any time expenditure (which is permissible, for example, for parallel computing system with shared memory). Algorithm operations, between which there is no path within the chosen calculation circuits, can be performed in parallel. Let us set the set $H_p = \{(i, P_i, t_i): i \in V \}$ (where $p$ is the number of processors used), in which the torus for each operation $i \in V$ indicates the number of the processor $P_i$ used for the operation and the start time of the operation $t_i$. In this case, one processor cannot perform different operations at the same time, and by the appointed time of the operation, all the data used for it must be calculated. The graph $G$ together with the set $H_i$ can be considered as a model of a parallel algorithm executed using $p$ processors. In this case, we can talk about the following estimates:

$$T_i(G, H_i) = \max_{i \in V}(t_i + 1)$$ - parallel algorithm execution time;

$$T(G) = \min_{H_p} T_p(G, H_p)$$ - The minimum execution time of the algorithm for the selected computation scheme when using the optimal schedule $H_p$;

$$T_p = \min_{G} T_p(G)$$ - Minimum execution time of the algorithm when using the best computational scheme;

$$T_{\infty} = \min_{P_{2,1}} T_p$$ - The minimum possible execution time of a parallel algorithm when using an unlimited number of processors;

$$T_i$$ - Execution time of the algorithm when using one processor (time of execution of the sequential algorithm for solving the problem).

Acceleration obtained when using parallel algorithm for $p$ processors, in comparison with the sequential version of the computations is determined by the value $S_p = T_i / T_p$, i.e. as the ratio of the time of the sequential solution of the problem to the execution time of the parallel algorithm.
Average fraction of the execution time of the algorithm, during which processors are actually involved in solving the problem is called the efficiency of using the parallel algorithm of processors when solving the problem and is determined by the relation

$$E_p = \frac{T_s}{pT_p} = \frac{S_p}{p}$$  \hspace{1cm} (1)

3. Research Results and Their Discussion

1. Tensor addition operation

Tensor addition is defined by the formula

$$c_{ijk}^{rsK} = a_{ijk}^{rsK} + b_{ijk}^{rsK}$$  \hspace{1cm} (2)

where $a_{ijk}^{rsK}, b_{ijk}^{rsK}, c_{ijk}^{rsK}$ — tensors, valences $g + q$, where $g$ is the number of subscripts, $g$ is the number of superscripts, and each of the subscripts takes a value from 1 to $n$, where $n$ is the dimension of the space in which the tensors are specified. Thus, when two tensors are added, their respective components are added. The result will be a tensor of the same valence as the term tensors. The calculation of each of the tensor components is performed independently of the other components. The number of components of such a tensor, and, accordingly, the required number of addition operations for tensor components to perform addition is equal to $ng + q$. The graph of the algorithm for performing this operation is shown in Fig. 1.

Let us analyze the proposed computational scheme. The sequential execution time of this algorithm will be equal to the number of circuit vertices excluding input vertices, i.e. $T_s = ng + q$. The time of the fastest execution of the algorithm with an unlimited number of processors will be equal to the length of the maximum path of the graph, i.e. $T_{\infty} = 1$. The execution time of the algorithm, comparable to the minimum time, can be achieved with the number of processors $p \geq \frac{T_s}{T_{\infty}}$ [3], in this case $p \geq \frac{n^{g+q}}{1} = n^{g+q}$. That is, the minimum execution time of the algorithm will be achieved with the number of processors equal to the number of tensor components [9].

When the number of processors is less than the number of tensor components, the execution time of the algorithm can be estimated by the equality $T_p = \left\lceil \frac{n^{g+q}}{p} \right\rceil$, where the sign $\lceil \rceil$ denotes the integer nearest from the top. Then the acceleration value of the algorithm

$$S_p = \frac{n^{g+q}}{\left\lceil \frac{n^{g+q}}{p} \right\rceil}$$  \hspace{1cm} (3)
When $n^{q \times q}$ is a multiple of $p$, the acceleration when using parallel computations will be equal to $p$, otherwise it will be close to the number $p$, that is, the inequalities will be satisfied
\[
\frac{n^{q \times q} \cdot p}{n^{q \times q} + p} < S_p \leq p
\]
The efficiency of the algorithm can be assessed on inequalities
\[
\frac{n^{q \times q} \cdot p}{n^{q \times q} + p} < E_p \leq 1
\]
For comparison, based on the simulation operations of addition of tensors on a neural network, when calculating the sum of tensors, it is possible to reduce the time $2np$ times, where $p$ is the valence of the tensors being added, $n$ is the dimension of space.

2. Operation of multiplication of tensors: The operation of multiplication of tensors is defined by the formula
\[
C_{ijk}^{rs} = A_{ij}^{rs} \cdot B_{jk}^{rs}
\]
where
\[
A_{ij}^{rs} - \text{valence tensor } h,
\]
\[
B_{jk}^{rs} - \text{valence tensor } l
\]
\[
C_{ijk}^{rs} - \text{valence tensor } h + 1
\]
where $n$. Moreover, each of the indices takes a value from 1 to $n$ dimension of the space in which tensors are specified.

\[
\text{Figure 1. Algorithm graph for tensor multiplication operation.}
\]

That is, the operation of multiplying two tensors is reduced to multiplying each component of the first tensor by each component of the second tensor. Thus, the number of graph input vertices must correspond to the number of components of two tensor factors, that is, equal to $nh + nl$, and the number of multiplication operations will correspond to the number of components of the sought tensor, that is, equal to $nh + l$. The graph of the algorithm for performing this operation is shown in Fig. 2. Let’s analyze the proposed computational scheme. Since the algorithm for the parallelization structure is similar to the algorithm for the addition operation, similar estimates will be obtained. The sequential execution time of the algorithm will be equal to $T_1 = nh + l$. The minimum execution time of the algorithm for an unlimited number of processors is $T_\infty = 1$. The execution time of the algorithm, comparable with the minimum time, can be achieved when the number of processors $p \geq nh + l$. The execution time of the algorithm when using $p$ processors can be computed as
\[
T_p = \left\lceil \frac{n^{h+1} \cdot p}{p} \right\rceil
\]
The acceleration of the algorithm corresponds to the inequalities
\[
\frac{n^{h+1}}{n^{br+2}} < S_p \leq p, \text{ and efficiency - to inequalities}
\]
\[
\frac{n^{h+1}}{n^{h+1} + p} < E_p \leq 1.
\]
The operation of multiplying a tensor by a number is quotient the case of the operation of multiplication of tensors. When simulating the operation of multiplication of tensors over the field of complex numbers on a neural network, the number of network inputs should be twice the number of components of two tensor factors, that is, it should be equal to 2 (np + nq), where p is the total valence of the first tensor, q is the total valence of the second tensor, n is the dimension of space. The number of network inputs is 2np + q Note the special case when one of the two multiplied tensors, one, for example the first, has zero valence, that is, it is just a number (scalar). Then the operation of multiplying this scalar by the second tensor reduces to multiplying all coordinates of the second tensor by a scalar. It follows from this that the operation of subtracting tensors of the same structure is reduced to multiplying the subtracted tensor by (-1) and then adding the components of these tensors.

4. Convolution of Tensor
Tensor \(a_{ij...}^{rs...}\) obtained from tensor \(a_{ij...}^{rs...}\) by the formula \(a_{ij...}^{rs...} = a_{ij...}^{as...}\) is called the convolution of tensor \(a_{ij...}^{rs...}\) \[2\], where \(a_{ij...}^{rs...}\) is the tensor of valence q. The operation of convolution of a mixed tensor by one upper and one lower index consists in summing the corresponding components of the tensor and leads to a decrease in its valence by 2. Thus, to perform the convolution of the tensor, it is necessary to calculate \(nq-2\) sums, each of which contains \(nq-2\) terms. Let us first consider an algorithm for parallelizing the addition of \(m\) numbers. When the summation operation is performed sequentially, at each step, one of the terms is added to the sum. To add \(m\) numbers, you need \(m-1\) time units. This algorithm does not allow parallelization; therefore, to calculate the sum in parallel, it is necessary to use another algorithm, for example, the doubling algorithm. This algorithm consists in the fact that at the first step of its execution, the initial data is divided into pairs and an addition operation is performed between them. In the next steps, all the sums obtained are also divided into pairs, and the next summation is performed. This computational scheme is based on the associativity property of the addition operation.

Graphs for the sequential algorithm and for the algorithm for doubling the addition of numbers for \(m = 8\) are shown in Figure 3. When using one processor, the execution time operations will be equal to
\[
T_1 = \frac{m}{2} + \frac{m}{4} + \cdots + 1 = m - 1
\]
Obviously, for the parallel implementation of such an algorithm, the number of processors is \(p = \frac{m}{2}\). Minimum time
Completing algorithm $T_p = \lceil \log_2 m \rceil$. In this case, the acceleration and efficiency indicators of the doubling algorithm for summation can be estimated by the expressions

$$S_p = \frac{m-1}{\lceil \log_2 m \rceil}$$

$$E_p = \frac{m-1}{p\lceil \log_2 m \rceil} = \frac{m-1}{\frac{m}{2}\lceil \log_2 m \rceil}$$

respectively. Thus, the algorithm shows high acceleration, but as $m$ increases, the processor utilization efficiency will decrease (since $\lim_{m \to \infty} E_p = 0$). To improve this situation, a modified doubling algorithm [3] can be used. Let us return to the problem of tensor convolution. For each of the sums $m = nq-2$. In addition, the number of such sums is equal to $nq-2$. Then $T_p = (n^{q-2}-1)n^{q-2}$

$$P = \frac{n^{q-2}n^{q-2}}{2} = \frac{n^{q-2}}{2}$$

$$T_p = \lceil \log_2 n^{q-2} \rceil$$

will remain unchanged since the number of operations at each stage of the computational scheme will increase, but the number of stages will not change. The acceleration of the convolution algorithm can be estimated by the expression

$$S_p = \frac{n^{q-2}(n^{q-2}-1)}{\lceil \log_2 n^{q-2} \rceil}$$

and efficiency by the expression

$$E_p = \frac{n^{q-2}(n^{q-2}-1)}{\frac{p\lceil \log_2 n^{q-2} \rceil}{\lceil \log_2 n^{q-2} \rceil}} = \frac{2(n^{q-2}-1)}{n^{q-2}\lceil \log_2 n^{q-2} \rceil}$$

When modeling this operation on a neural network it is necessary to design the network in such a way that the number of its inputs is twice the number of tensor components. The number of network outputs and, accordingly, the number of neurons in the layer will be twice as many different components of the sought tensor. Based on the modeling performed, it can be argued that the use of a neural network for convolution of the tensor of general valence $p$ in a complex $n$-dimensional space will reduce the computation time by $2n2$ times [9].

5. Conclusion

Having analyzed the proposed algorithms, we can conclude that the operations of addition and multiplication of tensors lend themselves well to parallelization and show high acceleration while maintaining the efficiency of using computational elements. The operation of convolution of a tensor
during parallelization gives a high acceleration, however, with an increase in the valence of the tensor and the dimension of the space, the efficiency of using the computational elements will be low.

References

[1] Voevodin, V. V., and VI V. Voevodin. "Parallel’nye vychisleniya." SPb.: BHV-Peterburg 608 (2002): 608.
[2] Gel'fand, Izrael' Moiseevich. "Lectures on linear algebra." (1989).
[3] Quinn, Michael J. Parallel computing theory and practice. McGraw-Hill, Inc., 1994.
[4] Dimitrienko, Yurii I. Tensor analysis and nonlinear tensor functions. Springer Science & Business Media, 2013.
[5] Makokha, A. N., and T. E. Tyshlyar. "Imitacionnye modeli nejronnyh setej, realizuyushchih operacii tenzornoj algebry nad polem kompleksnyh chisel (Simulation models of neural networks, realizing the operation of the tensor algebra over the field of complex numbers)/Nejrokompyutery: razrabotka i primenenie." M.: Izdatel’mstvo «Radiotekhnika 9 (2010): 64-70.
[6] Salam Hussein Ewaid et al 2021 IOP Conf. Ser.: Earth Environ. Sci. 790 012075
[7] Kim, Joon-Hwi, Maverick SH Oh, and Keun-Young Kim. "Boosting vector calculus with the graphical notation." American Journal of Physics 89.2 (2021): 200-209.
[8] Adelman, Menachem, and Mark Silberstein. "Faster neural network training with approximate tensor operations." arXiv preprint arXiv:1805.08079 (2018).
[9] Salam Hussein Ewaid et al 2021 IOP Conf. Ser.: Earth Environ. Sci. 722 012008
[10] Shen, Xu, et al. "Transform-invariant convolutional neural networks for image classification and search." Proceedings of the 24th ACM international conference on Multimedia. 2016.
[11] Nados, Wessam L., Rusul J. Alsaedi, and Qasim S. Kadhim. "THE APPLICATION OF THE GRAPH THEORY FOR DEVELOPING AND TESTING THE SOFTWARE".