Acceleration of Complex Network Analysis

Athanasios Grivas, Terrence Mak
School of Electrical, Electronic & Computer Engineering, Newcastle University
a.grivas, terrence.mak{@ncl.ac.uk}

Abstract

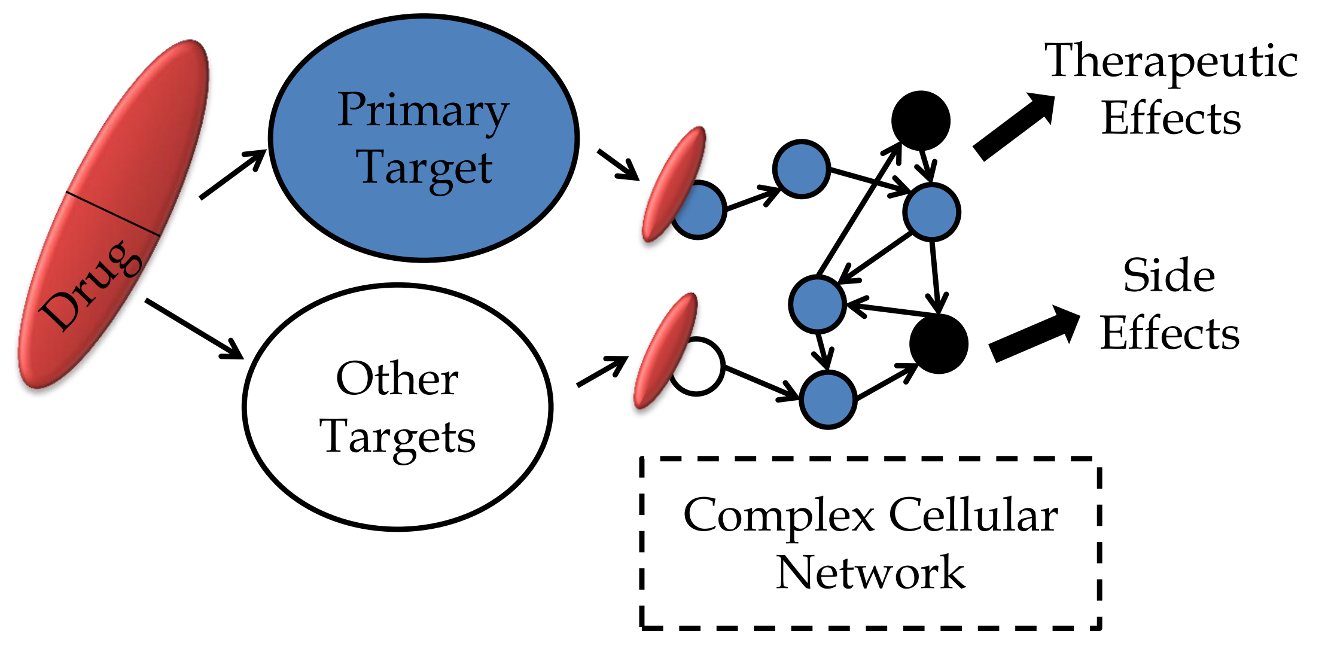
The scientific role of complex networks nowadays is of great importance. Their universal characteristics can be adopted for use from all over the scientific fields. This is the reason that scientists are pushing to the limits the network analysis. But the conduct of such analysis in CPU models it can be dramatically slow. They are not able to process multiple computations on a small time frame. There is need for acceleration of complex network analysis where the time execution of the used algorithms will be decreased in a large scale. Until now these algorithms were executed to conventional CPUs in sequential way. The breakthrough is the use of GPUs and parallel computing in order to accelerate the whole process. The CUDA (Compute Unified Device Architecture) architecture is used as the main tool in order to achieve this purpose. The transformation of common algorithms as matrix multiplication to a parallel model has shown large acceleration, which is a promising point for the field of network analysis.

Complex Networks

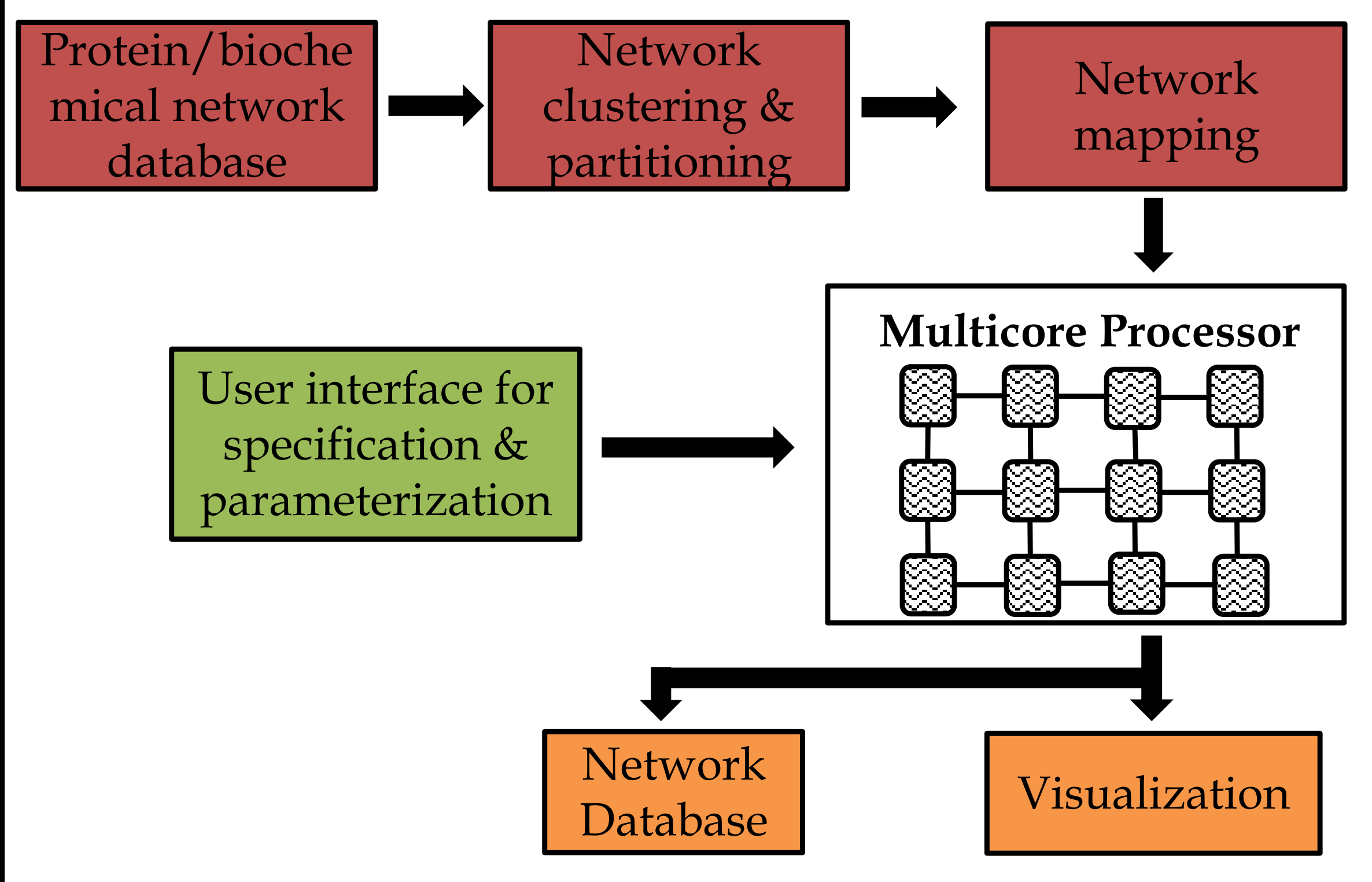
Genomics
Astrophysics
Artificial Intelligence

Many areas of science demand techniques to explore large data sets which, are in most cases represented by **large graph abstractions.**

Network Pharmacology: Drug discovery through analysis of massive protein interaction networks .



Modelling Platform


Protein/biochemical network database
Network clustering & partitioning
Network mapping
Multicore Processor
User interface for specification & parameterization
Network Database
Visualization

Building a **new platform** that will be capable to partition and map a massive network to multicore processors in such way that **parallel processing** will guide to acceleration of network analysis.

Possible Analysis

- **All-Pairs Shortest Path through Matrix Multiplication (APSP)**

Assume that the input graph $G = (V, E)$ has n vertices, so that $n = |V|$. Suppose that we **<u>present the graph by adjacency matrix</u>** $W = (w_{ij})$.

Let $l_{ij}^{(m)}$ be the minimum weight of any path from vertex i to j that contains at most m edges.

When $m = 0$, $l_{ij}^{(0)} = \begin{cases} 0 & if\ i = j, \\ \infty & if\ i \neq j. \end{cases}$

For $m \geq 1$, $\boldsymbol{l_{ij}^{(m)}} = min\left(l_{ij}^{(m-1)},\ \min_{1\leq k\leq n}\left\{l_{ik}^{(m-1)} + w_{kj}\right\}\right)$ (1)

$$= \boldsymbol{\min_{1\leq k\leq n}\left\{l_{ik}^{(m-1)} + w_{kj}\right\}}.$$

The actual shortest-path weights are therefore given by:
$\delta(i,j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = l_{ij}^{(n+1)} = \cdots$

We now compute a series of matrices $L^{(1)}, L^{(2)}, \cdots\cdots, L^{(n-1)}$, where for $m = 1, 2, \cdots, n-1$, we have $L^{(m)} = \left(l_{ij}^{(m)}\right)$. The final matrix $L^{(n-1)}$ contains the actual shortest path weights.

Now we can see the **relation to matrix multiplication**. Suppose we wish to compute the matrix product $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ of two $n \times n$ matrices A and B. Then, for $i, j = 1,2, \cdots, n$, we compute: $c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$. (2)

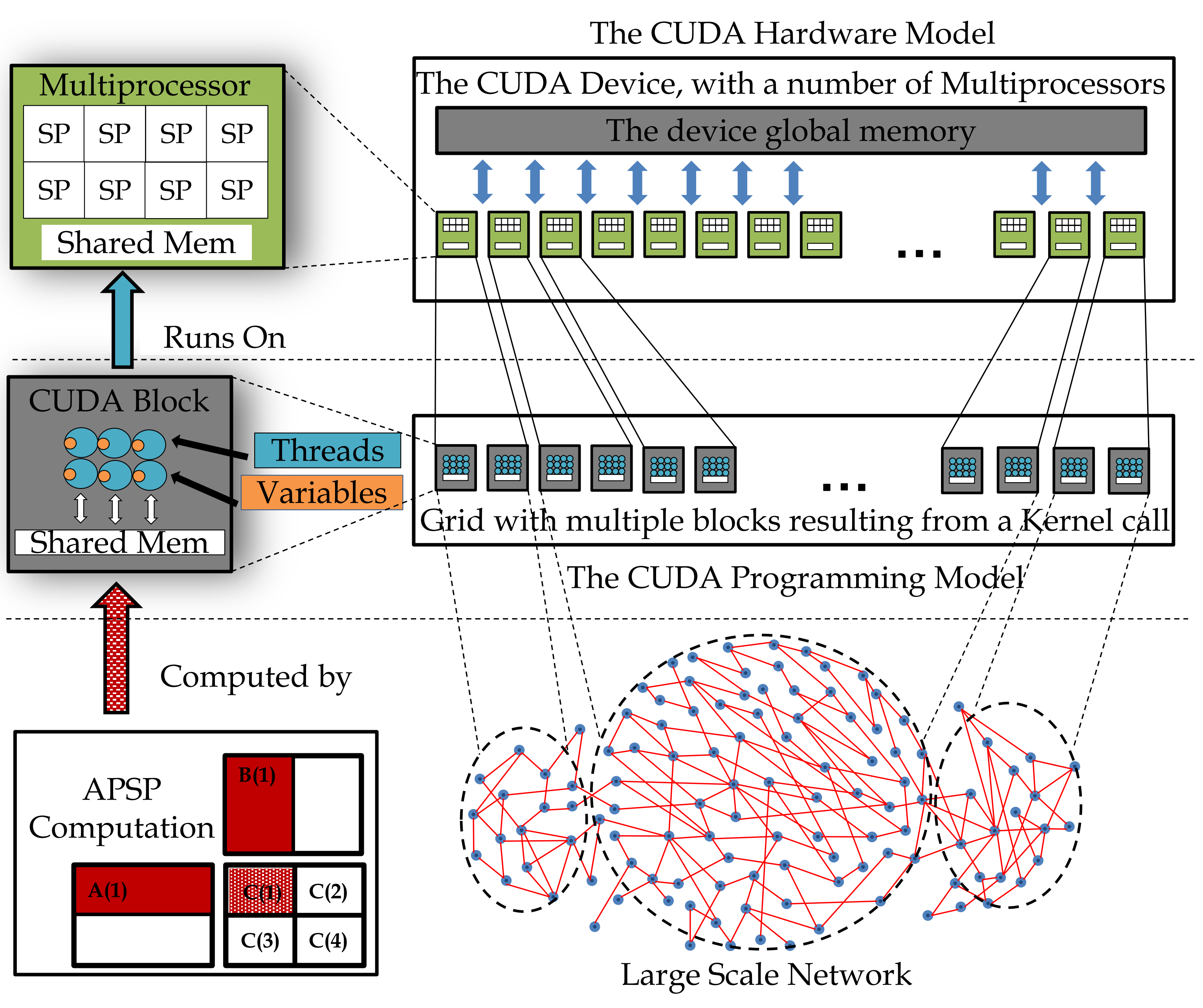
By doing the following substitutions at equation (1): $l^{(m-1)} \rightarrow a$, $w \rightarrow b$, $l^{(m)} \rightarrow c$, $min \rightarrow +$, $+ \rightarrow \times$ we observe that we obtain equation (2).

Consequently, all-shortest path can be calculated by multiplying network's adjacency matrix by its own self for n - 1 times :

$$\left.\begin{aligned} L^{(1)} &= L^{(0)} \times W = W, \\ L^{(2)} &= L^{(1)} \times W = W^2, \\ &\vdots \\ L^{(n-1)} &= L^{(n-2)} \times W = W^{n-1} \end{aligned}\right\}$$

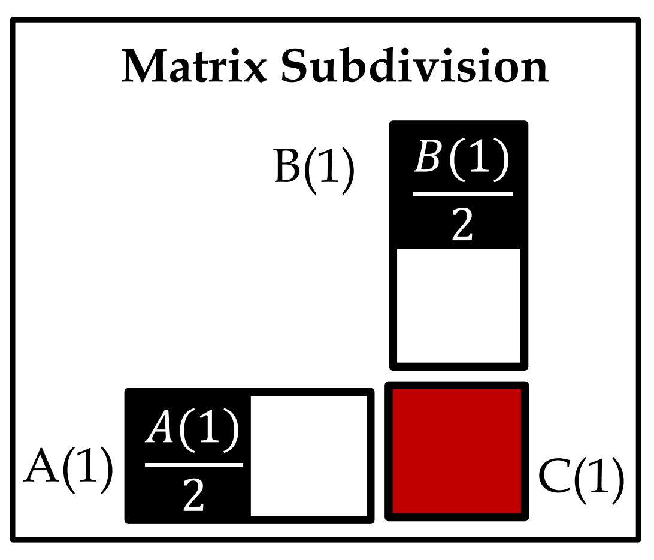
The <u>last Matrix</u> $\boldsymbol{L^{(n-1)}} = \boldsymbol{W^{n-1}}$ contains the shortest path weights.

Parallel Computing

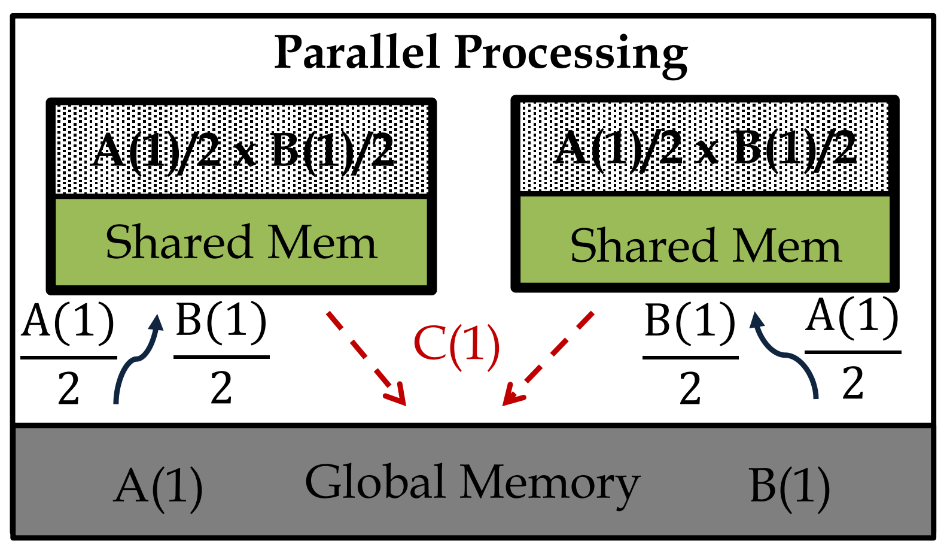


- **Multithreading Computing**
  - ✓ Graphics Processing Unit (GPU) as a multicore processor provides the ability of multithreading.
  - ✓ GPU is fully programmable through NVidia's CUDA tool.
- **Network Analysis**
  - ✓ Every network can be represented through an adjacency matrix.
  - ✓ Finding the APSP to such network can be done through matrix multiplication.
- **Mapping Network**
  - ✓ **General view:** Network is divided to several parts and mapped on different processors.
  - ✓ **Implementation:** Each matrix sub-computation is done through multiple blocks of threads that are running on multiple cores.

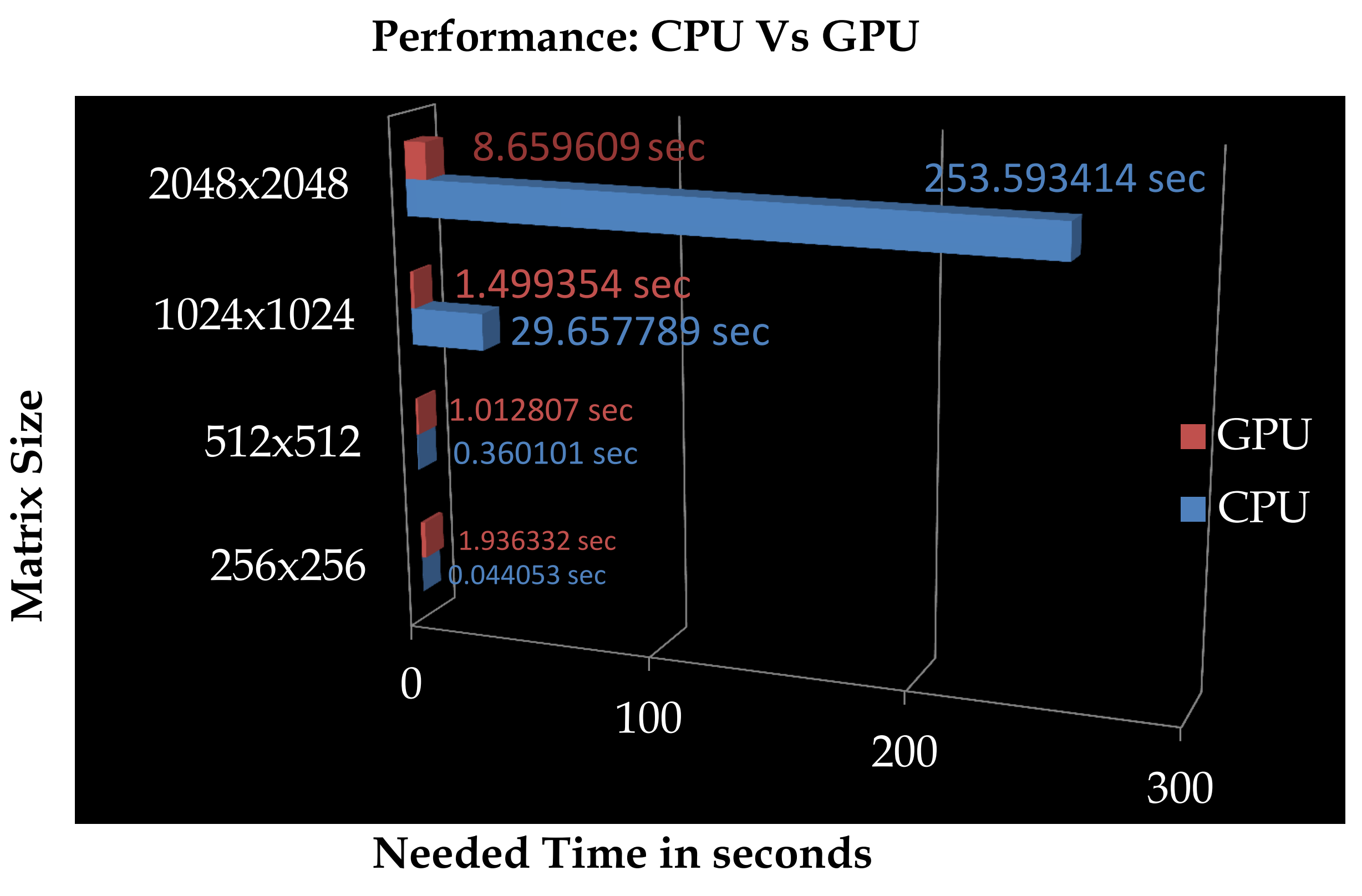
Multicore Mapping

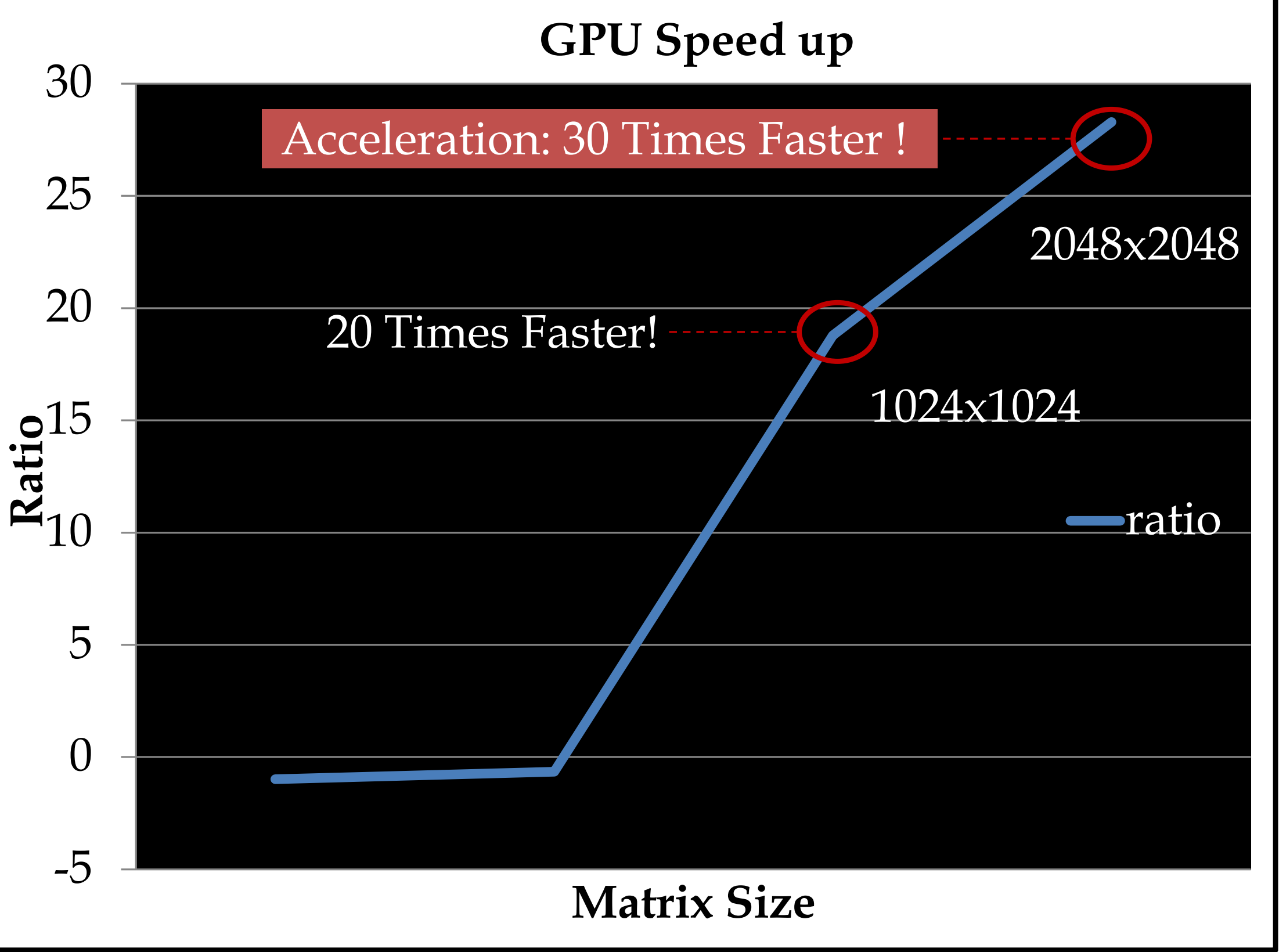


The initial sub-matrices are divided to further sub-matrices. Therefore, matrix multiplication is subdivided to more parts and **mapped to more cores**. The data are loaded from the global to shared memory. They are processed and then their combination provides the produced result of its part of the new matrix. This parallel processing guides to **massive acceleration** of the whole computation.



Results





Discussion

CPU's execution time is far slower. GPU achieved a 30 times speed up. This performance is surely the most promising factor. Real time analysis of large scale graphs can take advantage of this massive parallelism and boost its own acceleration. The main effort should be focused on the way that algorithms as APSP can be mapped in the most optimal way to multicore architectures. New techniques are needed to be discovered that improve the time consuming transfers of memory in order to exploit in maximum multiprocessors' performance.

References

[1] Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (2009) Introduction to Algorithms, Third edition, USA: The MIT Press
[2] Brodtkorb, A., Dyken, C., Hagen, T., Hjelmevrik, J. and Storaasli, O. (2010) 'State-of-the-art in heterogeneous computing', *Scientific Programming 18*, 1-33
[3] Berger, S. and Lyengar, R. (2009) 'Network analyses in systems pharmacology', *Bioinformatics,* vol. 25, no. 19, July, pp. 2466-2472