Nano-Structural Requirements for Artificial Intelligence & Blockchain Applications

Răzvan Mihai*, Mihaela Malița, Gheorghe M. Ștefan*****

* Politehnica University of Bucharest
E-mail: razvan.mihai.phd@stud.etti.upb.ro
** Saint Anselm College
E-mail: mmalita@anselm.edu
*** Politehnica University of Bucharest
E-mail: **gheorghe.stefan@upb.ro**

Abstract *– Artificial Intelligence and Blockchain are among the most demanding technologies requesting tremendously powerful computation engines capable to deliver fast, cheap and energy aware solutions. These technologies provide the safe computational environment for making intelligent decisions related to complex issues. Are presented the functional aspects and the structural requirements for the emerging intelligent world – the world dominated by technologically assisted consensual decisions and self-enforced regulations. Finally, are asserted the improvements required from the emerging nano-technologies.*

Keywords *– artificial intelligence; blockchain; parallel architecture; power aware technology; map-reduce.*

1. Introduction

The functional approach in the domains involving emergent nano-technologies must start from high level requirements defined at the market level. The current technology driven approach must shift gradually toward a market driven approach. In this context, the architectural level is of maximal importance by its intermediary position. A resurrected technology – the Artificial Intelligence (AI) – and an emergent technology – the Blockchain (BC) – *form a very promising couple* for the near future. Indeed, in our increasingly *complex* world, *intelligent* decisions should be taken only in a *safe* environment. *Complexity-Intelligence-Safety* represents the magic triad we face in our emergent technologically dominated world. Under these conditions, requirements appearing at the highest functional level need to be reflected at the deepest structural levels. In this paper we will try to cover the space between function and structure, with a special emphasis on the architectural level.

In the next section we present some aspects of the new applications using both, AI and BC which provide advanced tools capable to safely manage complex problems. The safety and complexity are supported by specific highly intensive computation. The third section emphasizes the specific architectural requirements for a computing system able to deal with AI and BC using energy aware and area efficient solutions. The fourth section describes the structural solutions. Some preliminary evaluations are provided in the fifth section. We conclude with the main improvement expected at the nano-technological level.

2. Where and How AI & BC are Involved

BC and AI have presently become the two most talked about technologies having the potential to disrupt almost any imaginable vertical industry. Both technologies meet the definition of general-purpose technologies (GPT) as defined by Bresahan and Trajtenber [1] (cited in [2]). One of the key characteristics of a GPT is that it catalyzes innovation and that it is complementary to other emerging technologies. While each technology comes with its own degree of complexity, they both seemed to have passed the hype phase and are now moving on to the productivity/maturity phase on the Gartner technology development scale.

BC's reputation, as a general-purpose technology, is largely due to its first application – Bitcoin – an peer-to-peer cash system proposed by Satoshi Nakamoto back in

2008 [3]. Nakamoto presented a solution to the double-spending problem using a peer-to-peer distributed timestamp server. The term “distributed” has since been arguably considered an underlying feature of an authentic BC architecture, in line with the original intent of the Bitcoin creator.

AI has made an impressive comeback in recent years thanks to significant advances in computing power. The most famous implementation of AI includes personalized user recommendations, advertisements targeting, or Machine Learning (ML) algorithms to prevent fraud. There are other applications in logistics, data mining, medical diagnoses, or automotive, to name just a few.

AI relies on large amount of data sets in order to train and improve its algorithms. Hence, the quality and accuracy of collected data are fundamental for the AI development. At the same time, BC is well-known for its capability to handle immutable, tamper free and consensus validated data. It is precisely at this point where BC and AI technology intersect and complement one another.

In contrast to BC, AI is at the present time highly centralized in the hands of corporations such as Facebook, Google or Amazon who have been able to collect impressive amounts of data sets. Combining these two technologies has the potential not only to improve the quality and reliability of input data necessary to ML algorithms, but to democratize the computational power of AI in a distributed peer-to-peer system such as the one proposed by BC.

3. Architectural Requirements

The two technologies – AI and BC – suppose different architectural requirements, but we must accommodate them on the same physical organization.

A. AI Architecture

There are two distinct approaches in AI: one is explicitly rule-based developed usually in Lisp, another – currently in vogue – involving the ML technique based on Deep Convolutional Neural Networks (DCNN). In a complex application both techniques are used.

A.1 Rule-Based AI Architecture

The Lisp language is developed starting from the lambda-calculus proposed by Alonzo Church [4]. The main distinct operations performed in Lisp are searches, inserts and deletes in lists, plus a very efficient implementation of a huge stack. In a sequence of data, where the previous operations are performed, it is possible to implement one or more stacks. On a sequence of symbols $<s_0 s_1 \ldots s_{n-1}>$ we apply operations such as:

- `SEARCH(name)`: all the occurrences of the sequence `name` are located
- `INSERT(name)`: in the *first* located place the sequence `name` is inserted
- `DELETE`: the symbol from the *first* located place is deleted
- `READ`: the symbol from the *first* located place is accessed moving the access to the next one.

A.2 DCNN-Based AI Architecture

The main functions used in implementing ML on DCNN are: convolution, pooling, and fully connected neural layers. All these functions are implemented on vectors:

```
V[1] = [v[11],v[12],… v[1p]]
V[2] = [v[21],v[12],… v[2p]]
                 …
V[m ]= [v[m1],v[m2],…v[mp)]]
```

as following:

- predicated vector operations:

```
v[ij]<=cond[j]?op(v[ij],v[kj]):v[ij]
```

where `op` is an arithmetic or logic operation applied on the components where the condition `cond` is fulfilled

- predicated reduction operations:

```
redOp(V[i]) = Op(…,(cond[j]?v[ij]:-),…)
```

where: `j = 1,2,…,p` and `Op` is an associative operation; it provides a scalar starting from the selected components of a vector

- predicated scan operations:

```
V[k] <= scanOp(…,(cond[j]?v[ij]:-),…)
```

return a vector starting from the selected components of a vector.

B. BC Architecture

The mining process is the most time and energy consuming process associated with the BC technology. It consists in hashes reconsidered until the requested condition is fulfilled. For the mining process the implementation is embarrassingly parallel. Two types of computations are involved:

- map computations performing hashing functions (for example SHA256) on blocks of data each prefixed with a different `nonce`
- reduce operations to find for what `nonce` the condition is fulfilled.

A many-cell hardware receives a block of data and distributes it in each cell. The nonce is selected randomly and incremented in each cell with the index of the cell. A SIMD-like program runs. At the end, the cell where the condition is fulfilled, if any, sends back its index using the reduction network.

4. Structural Proposal

The structure we propose is based on the mathematical computational model of partial recursive functions proposed by Stephen Kleene [5]. The structure has a Map-Scan-Reduce organization [6] [7] [8] [9]. The solution we propose is an accelerator as part of a hybrid computing system (see Fig. 1).

The structure of the accelerator (see Fig. 2) is centred on a linear array of *n* cells, MAP, which receives, in each clock cycle through the *log*-depts network DISTRIBUTE, an instruction issued by CONTROLLER to be executed in all its active cells. Two loops are closed over MAP: one through the *log*-depts network SCAN, and another through: *log*-depts network REDUCE, CONTROLLER and *log*-depts network DISTRIBUTE.

In each cell there is an accumulator centred execution unit, `eng[i]`, and a local data memory, `mem[i]`, for `i=1,…,n`. Thus, along the cells are distributed the previously defined *horizontal vectors* `V[j]`, for `j=1,…,m`. The content of each local memory represents a *vertical vector*

```
W[i] = [v[1i],v[2i],…v[mi]]
```

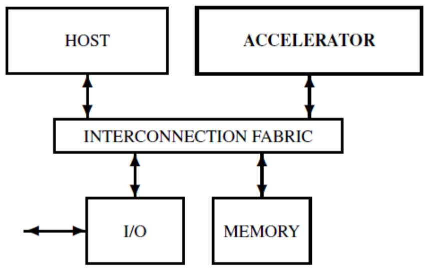

Fig. 1. Hybrid computing system organization [6].

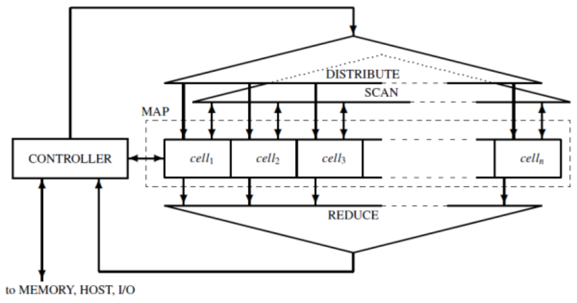

Fig. 2. Map-Scan-Reduce organization [6].

Besides these vectors are implemented:

- the accumulator vector:
  `acc = [acc[1], acc[2], …, acc[n]]`
- the Boolean vector used to select the active cells:
  `b = [b[1], b[2], …, b[n]]`
- the cell index vector:
  `ix = [1, 2, …, b]`

Besides the standard logic and arithmetic instructions executed on vectors in the active cells (**`where`**`(b[i]==1)`), in the MAP array are performed specific ***global operations***, for all `i`, exemplified by the followings:

- spatial selections, for example:
  - **`where`**`(cond)`:
    `b[i] = (cond) ? 1 : 0;`
  - **`endwhere`**: `b[i]==1;`
- search and modify operations:
  - **`search`**`(x)`:
    `b[i] = (acc[i]==x) ? 1 : 0;`
  - **`csearch`**`(x)`:
    `b[i] = (acc[i]==x)&(b[i-1]==1) ? 1 : 0;`
  - **`insert`**`(x)`: insert in the **first** active position the value `x` in the vector `acc` and save `acc[n]`
  - **`delete`**`(x)`: delete the symbol from the **first** active cell and `acc[n]<= x`

- shift operations applied on `acc`
- scan operations:
  - **`first`**: emphasizes the first active cell; it is applied to the Boolean vector `b`
  - **`pool`**: on vector `acc`, returns only the odd components aligned to the left
- reduction operations
  - **`redadd`**: $\sum_1^n$`(b[i]?acc[i]:0)`
  - **`redmax`**: MAX_1^n`(b[i]?acc[i]:0)`
  - **`redor`**: OR_1^n`b[i]`

The size of ACCELERATOR is in *O(n)*. The latency of the global loops is in *O(log n)*.

5. Evaluation

The proposed architecture is validated being implemented in a few versions. There are three silicon implementations, while a 28nm version is evaluated for 2048 32-bit cells with m = 1024. Running at 1 GHz and 85°C the silicon die of 9.2×9.2mm^2 is powered at 12W.

The performance in implementing DCNN for ML are evaluated in simulation [6]. The acceleration provided for all types of layers is in *O(n)*. For some operations the constant associated to the magnitude order is >1. For example, the multiplication of a matrix with a vector, besides the parallelism introduced at the MAP level, benefits from the parallelism between: multiplication in MAP, addition in REDUCE and loop control in CONTROL.

Regarding SHA256 used in BC, the performance on the previously described 28nm implementation is 46.8 MH/sec/Watt, which is 3.37× the performance of a Nvidia chip produced in 28nm [10].

6. Concluding Remarks

From the emergent nano-technologies, going under 7nm, for the physical embodiment of the proposed hybrid computing system, we request support for improving or adding the following features to the accelerator part:

- ***On chip dynamic memory*** buffer big enough to avoid as much as possible data transfers between MEMORY and ACCELERATOR (see Fig. 1)
- ***Multi-chip module*** with memory organized as chip-stack, to reduce the energy and time for transfers between MEMORY and ACCELERATOR
- ***Faster internal connections*** to reduce the latency of the global loops.

Acknowledgements - The authors got support from the technical contributors to the development of the ConnexArray*TM* technology: E. Altieri, F. Ho, B. Mîțu, M. Stoian, D. Thiebaut, T. Thomson, D. Tomescu.

References

[1] T. F. Bresnahan, M. Trajtenberg, "General Purpose Technologies: Engines of Growth?", *Journal of Econometrics*, 65(1), 83–108, 1995.

[2] C. Catalini, J. S. Gans, "Some Simple Economics of the Blockchain", 2016. At: https://www.nber.org/papers/w22952.pdf

[3] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008. At: https://bitcoin.org/bitcoin.pdf

[4] A. Church, "An unsolvable problem of elementary number theory", *The American Journal of Mathematics* 58, pp.345-363, 1936.

[5] S. Kleene, "General recursive functions of natural numbers", *Mathematische Annalen*, 112(5):727–742, 1936.

[6] M. Malița, G. V. Popescu, G. M. Ștefan, "Heterogenous Computing System for Deep Learning" in Witold Predricz, Shyi-Ming Chen (eds), *Deep Learning: Concepts and Architectures*, Springer (in print), 2019.

[7] M. Malița, G. M. Ștefan, D. Thiebaut. "Not multi-, but many-core: Designing integral parallel architectures for embedded computation", *ACM SIGARCH Computer Architecture News*, 35(5):32–38, Dec. 2007.

[8] G. M. Ștefan, M. Malița, "Can one-chip parallel computing be liberated from ad hoc solutions? A computation model-based approach and its implementation", *18th Inter. Conf. on Circuits, Systems, Com. and Comp.*, pp 582–597, 2014.

[9] G. M. Ștefan, et al., "The CA1024: A Fully Programmable System-On-Chip for Cost-Effective HDTV Media Processing", *Stanford University: Hot Chips: A Symposium on High Performance Chips*, August 2006. At: https://youtu.be/HMLT4EpKBAw at 35:00.

[10] MSI GeForce RTX 2070 Ventus 8GB GDDR6 Review (Turing TU106 GPU), 2018. At: https://www.geeks3d.com/20181130/msi-geforce-rtx-2070-ventus-8gb-gddr6-review-turing-tu106-gpu/#4_15