Paper critique on

A Computational Stack for Cross-Domain Acceleration

By **Brandon Saldanha**

Summary

With the push towards machine learning greater than ever, Domain Specific Accelerators have gained importance in recent years. Damian Specific Accelerators (DSA) have greater performance over general-purpose processors because of their design which is capable of executing a particular task with the highest efficiency. DSAs lose to general-purpose processors in the later's flexibility in handling diverse requirements. To bridge this gap between good flexibility and good efficiency, the paper suggests PolyMath, a cross computational stack. A cross-domain stack, PMLang is defined on this stack.

End-to-end applications require multiple different sub-processes to work in tandem to realize the entire process. With the current push towards development of DSAs, these individual sub-processes are becoming more and more isolated - making application development that depends on these sub-processes onerous. The performance boost attained by using DSAs is undeniably superior to general-purpose processors, leaving users to choose between usability and performance.

PMLang exploits the mathematical similarities between different domains such as Robotics, Graph Analytics, DSP, Data Analytics and Deep Learning to achieve a modular and reusable language that is useful across Robotics, Graph Analytics, DSP, Data Analytics and Deep Learning. Even though, PolyMath loses in performance to a completely ideal application design, it makes up for this in its usability and versatility. Given that PolyMath has higher ease of use than python, the minor performance dip is acceptable.

The results in the paper show that PolyMath has a good balance of usability and efficiency, while enabling cross-domain multi-acceleration. It is imperative to design cross domain stacks instead of multiple highly specialized domain-specific accelerators. PolyMath does a good job in this regard by taking advantage of domain-specificity while setting a standard across cross-domain applications.

Strengths

PolyMath has an extensible, modular and open-source computation stack that invites other to innovate and explore cross-domain stacks. This enables others to add other domains which have similar mathematical constructs as PolyMath.

The paper does a good job explaining the newly defined language PMLang along with examples that make understanding the underlying structure easy.

Weakness

PMLang is essentially a new language and while it may be easier to code in PMLang than Python this is only after you clear the learning curve of PMLang. Users who have worked with other domain specific languages and have experience with building multi-domain systems may not switch to PMLang.

PMLang definitely has its benefits as shown in the paper but currently is not as widely used as other domain specific languages. Pre-existing languages have much more support, tutorials, examples, etc than PMLang. As it is right now, PMLang may not be an ideal substitute to these.

While PMLang makes things easier overall, it still has a 23% gap in performance to the older, ideal implementations - this performance gap is big enough to convince users to stick to the old approach and not use PMLang for system design as it stands right now.

The paper compares PMLang to general purpose architecture such as Xeon CPU, Jetson Xavier and Titan XP but comparison with a full fledged development with uniquely written parts is missing. Such a comparison with an ideal design environment would help to understand how much PMLang has to be optimized to be the optimal route for multi-domain application development.

Future scope

To have wide adoption of PMLang, it is necessary to have more tutorials and example applications developed using these. It is also possible to optimize PMLang to diminish the current performance gap between a PMLang implemented system and an ideal implementation with multiple domain-specific accelerators.