The *NOPE* Manifesto

The path towards lasting impact in programmable hardware research

Zsolt István
https://zistvan.github.io

Where are we now? The performance of database management systems, distributed or single node, is important to all of us in the database community. We care about throughput and response time metrics and in today's world of stagnating CPU performance, there is a real incentive to extract every last bit of computing power from the CPUs and to use server components, such as NICs and NVMe devices, as efficiently as possible. This stagnation of CPU performance gave rise to the colorful space of programmable hardware accelerators for databases, such as Field Programmable Gate Arrays (FPGAs). Even though they demonstrate significant acceleration capabilities, typically they are not *yet* part of the server architecture. Hence, focusing only on throughput or response time disregards the fact that adding new components to servers increases power consumption, development times and adds integration overheads. Even if 100x speedups can be shown for some workload in a prototype, unless the database can *always* benefit from the additional hardware, the true impact of the underlying research work will be marginal in reality. At the same time, if used to its true potential, programmable hardware can be disruptive: examples include turning compute-bound operators into bandwidth-bound ones, offering parallel processing that is skew-resistant, or processing data directly in the caches, near memory.

The purpose of the *NOPE* approach is to focus **n**ot **o**nly on **pe**rformance when researching ideas on heterogeneous hardware and instead cast a wider net. To ensure that the proposed hardware elements are *always* useful and can have lasting impact, we must flip priorities: look for improvements in non-performance-related aspects first and, once identified, work to deliver these without hurting performance.

NOPE
A DB RESEARCHER SKIPS A PERFORMANCE-ONLY IDEA
NOPE
10% TPS increase with FPGA
NOPE

Saying *NOPE* is not saying no to faster systems. It's saying *yes to more rewarding challenges.* We all know, focusing only on performance improvements is a Sisyphean task: there will always be an other marginally faster solution.

Saying *NOPE* will ensure that research on specialized hardware will result in novel ideas with lasting effects in more forward-looking applications. It will also lead to specialized hardware being used only when and where it can truly make a difference!

Join the *NOPE* movement! As a researcher, adopting the thinking behind *NOPE* will help in deciding which problems to tackle: the ones that are not only focused on performance but instead bring new, perhaps even groundbreaking, properties and functionalities to the underlying DBMS.

Examples of other metrics and functionality to consider include:

Energy efficiency: Replacing a large number of CPU cores with a smaller amount of specialized-hardware based components has the potential to reduce power consumption of the underlying server dramatically. For most workloads, both run in the cloud and on premise, the power consumption of the server hardware is one of the main cost sources. Reducing power consumption while *at the same time* increasing performance is a very meaningful research direction. In order to achieve this, however, the programmability and general applicability of the hardware solutions has to increase. One cannot rely on "one trick ponies" and instead has to design with run-time flexibility in mind.

Security and privacy: There is a growing body of databases and analytics systems aiming to provide higher levels of security and privacy. Many of the existing solutions, however, result in a steep reduction of performance. If one could use heterogeneous hardware to provide these properties *without* introducing latency or throughput overhead, it would enable wide-spread adoption of such privacy-ensuring databases – perhaps making such features default in the future!

Exploring different algorithms / complexity classes: When using specialized hardware to implement a given operator, the interesting question is whether the hardware could be used to implement it *fundamentally differently* from its CPU counterpart. Perhaps, some property of the hardware (e.g., massive parallelism) allows one to switch to an other family of algorithms with different trade-offs. It is objectively more useful to investigate the applicability of a wider range of algorithms to the problem, than to "optimize the commonly used ones to death".

As a reviewer, you have tremendous power in determining what research directions make it into our community's bloodstream. Given that programmable hardware, such as FPGAs, is a mature technology and we have already a good understanding of its benefits and shortcomings in terms of raw performance, one could (1) discourage papers which focus only on performance and raise questions to the authors related to cost of integration and programmability; and (2) judge proposed systems not only on performance but instead look for and cherish other types of insights and lessons learned in papers. As an example: even if an idea using a SmartNIC does not make the prototype faster than the fastest existing OLAP system, perhaps it opens up exciting new opportunities in scaling databases to a larger number of nodes in the future. It is through trial-and-error and open-minded discussion that we can move ahead the state of the art in databases running on heterogeneous hardware.