Accelerating Database Workloads by Software-Hardware-System Co-design

Rajesh R. Bordawekar
IBM T. J. Watson Research Center

Mohammad Sadoghi
IBM T. J. Watson Research Center

Abstract—The key objective of this tutorial is to provide a broad, yet an in-depth survey of the emerging field of co-designing software, hardware, and systems components for accelerating enterprise data management workloads. The overall goal of this tutorial is two-fold. First, we provide a concise system-level characterization of different types of data management technologies, namely, the relational and NoSQL databases and data stream management systems from the perspective of analytical workloads. Using the characterization, we discuss opportunities for accelerating key data management workloads using software and hardware approaches. Second, we dive deeper into the hardware acceleration opportunities using Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) for the query execution pipeline. Furthermore, we explore other hardware acceleration mechanisms such as single-instruction multiple-data (SIMD) that enables short-vector data parallelism.

I. TUTORIAL OBJECTIVES

The core part of the tutorial is to sketch a roadmap for database hardware-acceleration based on a comprehensive discussion and classification of existing work that not only highlights strengths and novelties of related work, but also critically identifies the limitations for their wide-adoption in practice and open problems from both industry and academic perspectives.

This tutorial features four parts: (1) overview of system-level characterization of database systems with the special attention to areas with high potential for hardware acceleration; (2) overview of the SIMD, GPU, and FPGA architectures and programming models, e.g., GPGPU programming using the Nvidia ecosystem and FPGA programming using the Hardware Description Language (HDL) and its synthesis/deployment process; (3) acceleration of key database kernels using SIMD, GPUs, and FPGAs (guided by existing research); and (4) discussion of the chief limitations and open problems for deploying hardware-acceleration to large-scale data management systems in practice.

For the GPU portion, we overview the GPU architecture and outline the data parallel programming models such as CUDA and OpenCL. Subsequently, we focus specifically on Nvidia’s GPUs and describe key features of the CUDA toolkit through code snippets and examples for database kernel operations such as hashing, sorting, and joins. Similarity, for the FPGA portion, we overview the FPGA architecture and the HDL hardware programming model, which demands a different way of thinking about the problem, a major divergence from the traditional software development mindset. In particular, we describe the FPGA synthesis process and the pros and cons of reconfigurability and reprogrammability of FPGAs through examples and code snippets. This material serve as perquisite, before providing an in-depth explanation and analysis of hardware acceleration for key database kernels and operations (e.g., sorting, joins, filtering, and compression).

II. DETAILED TUTORIAL OUTLINE

System-level characterization of database systems: We begin the tutorial by highlighting key database technologies that go beyond traditional relational systems and the acceleration challenges imposed by today’s architectural limitations. In the data processing space, we discuss row- and column-oriented query execution pipelines, data stream computation, and frequent pattern matching/mining. We also highlight the key architectural challenges such as Moore’s law physical limitation, memory wall and Von Neumann’s bottleneck, large and complex control units, and power consumption.

Hardware programming ecosystem: We provide a brief introduction to data-parallel execution using SIMD instructions, general-purpose computing using GPUs (GPGPUs) and FPGAs, and HDL programming. In this phase, we will cover the basics of SIMD, GPU, and FPGA architectures and programming models. We also describe the entire pipeline and life-cycle for synthesis and re-programmability of FPGAs and discuss the key differences between designing software threading model and hardware logical flow. In particular, our discussion is focused on the Nvidia ecosystem for GPUs. More specifically, we describe how to use the CUDA toolkit for programming GPUs. For FPGAs, we will focus on the Xilinx ecosystem and toolkit and how to design a logical flow based on a high-level hardware description language, such as Verilog, VHDL, or SystemVerilog. These languages enable to specify the circuit behaviorally using traditional C/C++-style constructs for condition, loop, and data types.

After providing an architectural overview of GPGPU and FPGA, e.g., heterogeneous system design, memory subsystem, and compute architecture, we discuss the evolution and life-cycle of these different hardware acceleration paradigms. For GPGPUs, we highlight programming evolution from sh to OpenCL and CUDA while focusing on single-instruction multiple-thread (SIMT) model and the hybrid off-loading in GPU programming. For FPGAs, we discuss programming life-cycle starting from Verilog to synthesis and re-programming of FPGAs (Xilinx toolkit). We further explain the re-programmability using lookup tables (LUTs), the routing architecture and interconnect, utilization of block memory (FPGA on-chip memory) organization and coupling, and comparison of data parallel vs. pipeline programming model. For
SIMD, we present how to enable short-vector data parallelism using Intel x86 and Power architectures.

Lastly, we explore advance topics including communication optimization by focusing on the impact of PCI connectivity, the host-device memory sharing, the multi-GPU/FPGAs programming and communications, and the power consumption and other energy-related issues. We also outline a set of performance optimization methodologies including thread mapping vs. custom logic circuit on hardware strategies, exploiting (distributed) shared memory and shared virtual space, understanding off- and on-chip memory (processor and data coupling), and improving GPUs and FPGAs device utilization.

Accelerating data management workloads using GPUs/FPGAs: The core of the tutorial will be dedicated to surveying existing work and discussing the key opportunities for exploiting GPUs and FPGAs for relational analytical workloads and streaming applications. We will discuss key database kernels such as hashing, sorting, join, and OLAP operations such as aggregation. For these kernels, we analyze their implications on system design, e.g., how does cost-based query optimization affect the hybrid system design, what are the impacts on the data layout strategies (row vs. column organizations), or how to identify off-loadable database kernels for hardware acceleration.

Additionally, we discuss the role of hardware acceleration for data streams. In particular, we highlight major works in the area of SQL query execution using sliding window semantics, regular expression pattern matching using finite-state machines, complex-event processing using high-dimensional indexing and Rete network, and SQL query compiler design and multi-query optimization opportunities. Lastly, we explore advance topics such as (1) best-effort computing including pre-filtering approaches for dividing computation in the co-processor model; (2) supporting online changes to query/data workloads on hardware (i.e., trade-offs between query performance and flexibility) such as modification to existing queries, addition/removal of queries, schema changes, online query topology re-construction for multi-query optimization; and (3) co-processor execution model by combining CPUs, GPUs, and FPGAs as either primary data storage units or within clustered systems.

Open problems and practical limitation of hardware accelerations: We conclude the tutorial with final remarks on benefits and challenges of employing modern hardware accelerators in practice. We offer our views and discuss open problems, namely, closing the gap between the software flexibility and the performant, but inflexible, hardware solutions, from the perspectives of both industry and academic research. In particular, we raise and explore the following questions. (1) How to achieve line-rate data processing (what level of parallelism can be attained)? (2) How to overcome the hardware inflexibility and development cost challenges? (3) How/Where to place hardware accelerators in query execution pipelines in practice? (4) What are the power and energy consumption benefits of hardware acceleration?

Acknowledgments
We wish to thank Hans-Arno Jacobsen for many insightful discussions and invaluable feedback in the earlier stages of this work.

References