The Next 700 Heterogeneous OLAP Systems: A Framework to Answer What-if Design Questions

Faeze Faghih
Systems Group, TU Darmstadt
faeze.faghih@tu-darmstadt.de

Zsolt István
Systems Group, TU Darmstadt
zsolt.istvan@tu-darmstadt.de

Florin Dinu
Huawei Munich Research Center
florin.dinu@huawei.com

1 BACKGROUND AND MOTIVATION

Near-Data-Processing (NDP) is often proposed as a solution to solve the data movement bottleneck in database systems. NDP offloads part of the computation close to where the data is stored and reduces the amount of data that needs to be moved.

If we look at the NDP proposals in the literature, various types of hardware are used, and it is not trivial to generalize their results. One reason is that the amount of data reduction determines the usefulness of a solution: using a particular architecture may result in performance improvement if the data reduction exceeds a certain threshold, otherwise, it could lead to a slowdown. However, data reduction is not the only concern. Today's servers offer numerous heterogeneous resources and the processing elements (PEs) chosen for running offloaded computation (e.g. microprocessors, FPGAs embedded in SSDs, SmartNICs) typically have lower computation power in comparison to a server-class CPU. Across the different heterogeneous resources, the system may experience a slowdown instead of a speedup due to computational overhead.

In related works, the choice of hardware and work partitioning is typically done by hand by domain experts. In this work, we propose a framework to automate such design decisions in an NDP system. Our framework examines a combination of metrics (i.e. amount of data reduction in the workload, processing rate of resources, and link bandwidths) to answer what-if questions in designing NDP OLAP systems with heterogeneous hardware offloading such as: under what conditions of relative throughput and data reduction, would adding a new PE make sense?

2 FRAMEWORK OVERVIEW AND NEXT STEPS

To build a framework that automates determining and comparing viable offload design choices, we need a way to abstract away low-level details of systems and model them in a common representation. In related works, different PE types are used to run the offloaded computation (e.g. microprocessor [1] and FPGA [2] in SSDs), this brings our first challenge: to find a way to model various types of PEs in a similar fashion. In addition, the internal structure of devices could also play a role in the usefulness of NDP. For example, NVIDIA Bluefield-2 SmartNIC provide different bandwidths depending on how it is used (doing processing in its processor or bypassing it), or the SSD in Biscuit [1] has more than one type of PE (i.e. a microprocessor and additional hardware pattern-matchers) inside it. The second challenge is capturing the internal structure in the model without "overfitting". Our systems of interest are Online Analytical Processing (OLAP) systems. As a result, our primary focus is on throughput and data reduction at each PE. Based on these, we want to determine the bottleneck PE, or link, in the overall architecture.

Defining a model. One of the well-studied modeling approaches that we can adapt to reach our goal, is Network-of-Queues (NoQ). In an NoQ model, a system consists of multiple devices, each of which has its own queue and can be single-threaded or parallel. We represent all links and PEs of the real system as devices in the NoQ model. An NoQ model considers jobs, which in our case map to batches of records (e.g., the size of a disk page). The inputs to the NoQ model are: *(i)* the external arrival rate, which is equivalent to the overall throughput of an NDP system, *(ii)* the number of times each job gets service from each device (visit ratio), which can be derived from the relative selectivity of each processing step as data is retrieved from storage, and *(iii)* the service time of each device, which we can set by using the inverse of the maximum processing/transfer rate of each PE/link.

In an NoQ model, the topology of how devices are connected is abstracted away but in our case, we need to derive visit ratios based on selectivity and how devices are connected. An example scenario: we have a device running a process that filters out 20% of the data (i.e. 80% of the input data leaves that device); if all the input jobs visit this device (i.e. visit ratio of the device equals 1), only 80% of the input data visit the next device. As a result, the visit ratio of the downstream devices will be 0.8.

How we use the model. We use the NoQ model "in reverse" to find the saturation point of a system (i.e. the point at which the utilization of one of the devices approaches saturation and becomes the bottleneck) and achieve the maximum throughput that the system is capable of handling. By changing the model parameters slightly, e.g., adjusting bandwidths or selectivities, it is possible to understand how the system would behave under slightly different conditions and whether it would be useful, for instance, to consider a faster in-storage processor.

An NoQ model can also output response times and queue length statistics at different load levels – in our case, however, these are not of interest because our goal is to find the bottleneck device in a heterogeneous system, and to reason about how the bottleneck would change under slight changes to the system properties.

Next Steps. In our next step, we will use the described framework to create a design space exploration (DSE) tool. With more candidate devices to run offloaded computations and more workloads using the OLAP system, the practical design options increase. With a set of devices and workloads (e.g. set of SQL queries), determining whether and how to use these devices is not trivial, and our DSE tool will help solve this problem by automating the process.

REFERENCES

[1] Boncheol Gu et al. 2016. Biscuit: A framework for near-data processing of big data workloads. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 153–165.
[2] Joo Hwan Lee et al. 2020. SmartSSD: FPGA Accelerated Near-Storage Data Analytics on SSD. *IEEE Computer Architecture Letters* 19, 2, 110–113.