To Ship or Not to (Function) Ship

Feilong Liu, Niranjan Kamat, Spyros Blanas, Arnab Nandi
The Ohio State University
{liu.3222, kamat.14, blanas.2, nandi.9}@osu.edu

***Abstract*—Sampling is often used to reduce query latency for interactive big data analytics. The established parallel data processing paradigm relies on function shipping, where a coordinator dispatches queries to worker nodes and then collects the results. The commoditization of high-performance networking makes *data shipping* possible, where the coordinator directly reads data in the workers' memory using RDMA while workers process other queries. In this work, we explore when to use function shipping or data shipping for interactive query processing with sampling. Whether function shipping or data shipping should be preferred depends on the amount of data transferred, the current CPU utilization and the sampling method. The results show that data shipping is up to 6.5× faster when performing clustered sampling with heavily-utilized workers.**

I. Introduction

In big data analysis, sampling is often used to reduce query latency for interactive query execution [6]. Current database systems use function shipping in query execution, where the coordinator distributes query plans to the workers for execution then collect results from the workers. The cost of function shipping includes the computation cost of executing queries in workers and the communication cost of transferring results from workers to the coordinator. In function shipping, sampling methods do not affect the communication cost, but affect the computation cost. For example, random sampling accesses the whole data set while cluster sampling only accesses part of the data set during query execution.

Commodity clusters are now commonly equipped with fast networks with Remote Direct Memory Access (RDMA) support [1]. RDMA enables user applications to directly access memory in remote machines without involving the operating kernel and offers higher throughput than TCP/IP sockets [2]. Data shipping is possible with one-sided memory access provided by RDMA. In data shipping, the coordinator uses RDMA Read to read data from workers and executes query locally while the workers remain passive. The cost of data shipping includes the computation cost of executing queries in the coordinator and the communication cost of transferring data from workers to the coordinator. In data shipping, sampling methods not only affect the computation cost, but also affect the communication cost. In cluster sampling only the sample of the data set is transferred to the coordinator, however, the whole data set is transferred to the coordinator in random sampling.

This research was partially supported by the National Science Foundation under grants IIS-1422977, IIS-1527779, CAREER IIS-1453582, CCF-1816577 and CNS-1513120.

We discuss the trade-offs between function shipping and data shipping that are afforded by the advent of RDMA and look at how sampling influences this decision. Whether function shipping or data shipping should be preferred depends on the amount of data transferred, the current CPU utilization and the sampling method. The result shows that data shipping has better performance when the computing resources are limited in workers for both sampling methods and data shipping improves performance by up to 6.5×.

II. System Design

We use the single coordinator multiple workers design, where queries are sent to the coordinator, which directs the query to the workers. In data shipping, the worker returns the raw data to the coordinator and the coordinator executes the query on the received data. In function shipping, the worker executes the query on its data and returns the result back to the coordinator. A final aggregation to combine the results from workers is performed at the coordinator.

A. Function Shipping vs Data Shipping

In function shipping, the worker executes the query and performs RDMA Write to send the result to the coordinator. In data shipping, the coordinator uses RDMA Read to read the data and executes the query on received data. The worker is passive in data shipping. The costs are as follows:

$$\texttt{COST}(DS) = C_{Read} + C_{Sample} + C_{CExec} \quad (1)$$

$$\texttt{COST}(FS) = C_{Sample} + C_{WExec} + C_{Write} + C_{CAgg} \quad (2)$$

where $\texttt{COST}(DS)$ is the cost of data shipping, C_{Read} is the cost of reading data from workers, C_{Sample} is the cost of sampling, C_{CExec} is the cost of executing queries at the coordinator, $\texttt{COST}(FS)$ is the cost of function shipping, C_{WExec} is the cost of executing queries at the worker, C_{Write} is the cost of writing the result to the coordinator, and C_{CAgg} is the cost of aggregating results. Increasing the number of workers reduces the data to be processed in each worker, which decreases the sampling cost C_{Sample} and the execution cost at the workers C_{WExec} in Equation 2, and hence favors function shipping. Data shipping is preferred when the size of the result is large or the computation load on the worker is high.

B. Sampling

Our system uses online sampling which supports two sampling modes, simple random sampling and cluster sampling [5]. In simple random sampling, every tuple has an equal probability of being included in the sample. In the absence of indexes, this involves accessing every tuple of the data set. We

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

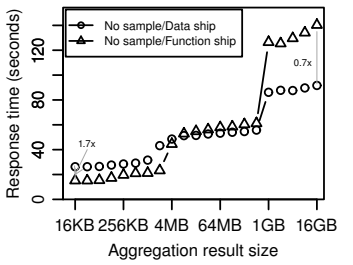

Fig. 1. As distinct cardinality increases, function shipping becomes expensive due to result size increase.

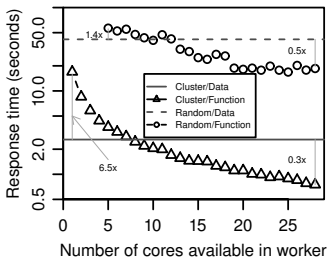

Fig. 2. As the computation resources available at worker increases, function shipping gets cheaper.

use Bernoulli sampling semantics for simple random sampling. In cluster sampling, different clusters are chosen randomly and all tuples within a cluster are included in the sample. This avoids accessing every tuple in the data set. The pros and cons of both sampling strategies are as follows.

1) Execution Speed: In function shipping, performing simple random sampling involves adding Bernoulli sampling-based scan operator and accessing the whole data set, while in cluster sampling, only a fraction of tuples are accessed. In data shipping, to perform simple random sampling, the entire data set needs to be transferred to the coordinator as the worker lacks computing resources required to perform sampling. The coordinator then samples the received data. For cluster sampling, the coordinator only accesses a sample of the data, resulting in less network traffic. Thus, cluster sampling is cheaper than simple random sampling.

2) Result Quality: Simple random sampling usually results in better sample quality than cluster sampling if the tuples are stored in non-random order. A clustered index stores the data in a sorted order. If the `GROUP BY` or `WHERE` clause contains any of the clustered index columns in order, cluster sampling can result in tuples and groups being respectively missed, causing sampling error to be large.

III. Experiments

We extended our open-source RDMA-aware query engine Pythia [4] with sampling support. We currently use a single coordinator and a single worker setup. They each have 512 GB of memory across two NUMA nodes, with each NUMA node having one Intel Xeon E5-2680v4 14-core processor. They are connected by an EDR (100 Gb/s) InfiniBand network.

Our data set has one table `R` with 10 billion tuples, with each tuple having two long integers `R.a` and `R.b` as attributes. We evaluate the SQL query `SELECT R.b, COUNT(*) FROM R GROUP BY R.b`, in which records with the same value in `R.b` are aggregated to a single record. Hence the number of records in the result is the same as the number of distinct values of `R.b` in the data.

A. Changing Cardinality of Results

This section evaluates how the size of the result affects the response time (Section II-A).

As the result size is non-deterministic with sampling, we turn off sampling in this experiment. We vary the distinct cardinality of `R.b` from 1 thousand to 1 billion. At the coordinator, we use all 28 cores for query execution, while the worker only uses 14 cores to simulate the additional workload in the worker node. Figure 1 shows that when the result size is less than or equal to 4 MB, function shipping has lower response time than data shipping. This is because the size of the result which is transferred in function shipping is not large. When the result size is equal to or larger than 8 MB, the saving in network traffic decreases and function shipping has higher response time than data shipping. Hence, data shipping is preferred when the result size is large.

B. Changing Load on the Workers

How does the load on the worker and the choice of sampling method affect the choice of shipping method?

We simulate different loads on the worker by varying the number of available cores from 1 to all 28 cores, and keeping the number of cores at the coordinator fixed at 28. We set the distinct cardinality of `R.b` to be 2, and the query timeouts at 60 seconds. The sampling rate is 10% and we compare both cluster sampling and random sampling. The result is shown in Figure 2. The number of available worker cores has no impact on data shipping due to our use of RDMA. The response time for function shipping decreases when the number of worker cores increases. When the number of cores in the worker is 8 and 9, data shipping has higher performance for random sampling but has lower performance for cluster sampling. This is because random sampling is more computation intensive and favors data shipping when the worker has limited computing resources. For the same sampling method, we can see that data shipping has lower response time when the number of cores is small and is up to $6.5\times$ faster than function shipping, as the saving in network traffic is offset by the slow workers in query execution. Data shipping will be preferred when the result size increases.

IV. Conclusions and Future Work

In this paper, we compare how RDMA and fast networks affect query execution strategies for interactive queries with sampling. While function shipping was the norm, interactive big data analytics should take the amount of data transferred, the CPU utilization and the sampling methods into account when choosing query execution strategies. Looking ahead, one possible direction is to build a cost model which takes these factors into account to predict the cost of different execution strategies and to pick the optimal execution strategy. One can find detailed discussion, related work and the evaluation of executing multiple queries in the extended version [3].

References

[1] C. Barthels et al. Rack-Scale In-Memory Join Processing Using RDMA. *SIGMOD*, 2015.
[2] P. W. Frey et al. Minimizing the Hidden Cost of RDMA. *ICDCS*, 2009.
[3] F. Liu, N. Kamat, S. Blanas, and A. Nandi. To ship or not to (function) ship (extended version). arXiv:1807.11149.
[4] F. Liu, L. Yin, and S. Blanas. Design and Evaluation of an RDMA-aware Data Shuffling Operator for Parallel Database Systems. *Eurosys*, 2017.
[5] S. Lohr. *Sampling: Design and Analysis*. 2009.
[6] F. Olken. Random Sampling from Databases. 1993.

978-1-5386-5989-2/18/$31.00 ©2018 IEEE