A Coding Based Optimization for Hadoop

Zakia Asad
The Edward S. Rogers Sr.
Department of Electrical and
Computer Engineering
University of Toronto
z.asad@mail.utoronto.ca

Mohammad Asad
Rehman Chaudhry
Soptimizer
Canada
masadch@soptimizer.org

David Malone
The Hamilton Institute
Maynooth University
David.Malone@nuim.ie

1. INTRODUCTION
The rise of the cloud and distributed data-intensive (“Big Data”) applications puts pressure on data center networks due to the movement of massive volumes of data [2, 3, 4]. Reducing volume of communication is pivotal for embracing greener data exchange by efficient utilization of network resources [5, 6, 7, 8, 9]. We propose a system for optimizing Big Data processing by exploiting a mixing technique, a variant of index coding problem [10, 11], working in tandem with software-defined network control for dynamically-controlled reduction in volume of communication. We present a motivating use-case and developed a proof-of-concept implementation of the proposed system in a real world data center. We use Hadoop as our target framework and Terasort, and Grep as workloads for performance evaluation of the proposed system. The experimental results exhibit coding advantage in terms of the volume of communication, goodput, as well as number of bits that can be transmitted per Joule of energy.

2. PROPOSED CODING BASED DATA FLOW FOR HADOOP
We introduce three new stages, namely sampler, coder, and preReducer to the traditional Hadoop MapReduce. The primary objective of the sampler is to gather side information. Similarly the primary objective of the coder is to code, and of the preReducer is to decode. The overall architecture is shown in Figure 1, while it shows only two nodes it is in fact replicated across all the nodes.

3. PERFORMANCE EVALUATION
We developed a prototype as well as a testbed to evaluate the performance of the proposed coding based approach. We use data from Hadoop shuffle to benchmark our proposed solution. The Hadoop jobs consisted of the following two industry standard benchmarks.

1. Terasort
2. Grep (Global Regular Expression)

3.1 Prototype
We have prototyped parts of the system in a data center as an initial proof of concept implementation. Our testbed consisted of 96 cores arranged in 8 identical blade-servers. Each server was equipped with twelve x86.64 Intel Xeon cores, 128 GB of memory, and a single 1 TB hard disk drive. The servers were arranged in three racks. Furthermore, the servers were connected in a typical data center configuration with OpenFlow enabled IBM RackSwitch G8264 as Top-of-Rack switches, and OpenFlow enabled Pronto 3780 as Aggregation switches. One server was used as the middlebox. The components were implemented using Java, and Octave [12]. All the servers were running Red Hat Enterprise Linux 6.2 operating system [13].

We use the following metrics for quantitative evaluation:

- Job Gain, defined as the increase (in %) in the
number of parallel Hadoop jobs that can be run simultaneously with coding based shuffle compared to the number of parallel Hadoop jobs achieved by standard Hadoop.

- **Utilization Ratio**, defined as the ratio of link-level packet transmissions when employing coding based shuffle to the number of link-level packet transmission incurred by the standard Hadoop implementation.

Our experimental study shows that for both of the tested benchmarks, the overhead to implement coding based shuffle (in terms of transmission of extra bookkeeping data in packet headers) was less than 4% in all the experiments. Table 1 shows the results across the two metrics for the two benchmarks. The results show significant performance of our scheme compared to the standard Hadoop implementation.

Noting the fact that our coding based scheme just requires XORing of packets which is computationally very fast operation and given high memory bandwidth of the servers, we were able to process closer to line rate. Specifically in the experimental setup, even during the worst case scenario, the throughput of the coder was 809 Mbps on a 1 Gbps link.

### 3.2 Testbed

Our testbed consisted of eight virtual machines (VMs), each running CentOS 7 as the operating system [14]. We used Citrix XenServer 6.5 as the underlying hypervisor [15]. Citrix XenCenter was used to manage the XenServer environment and deploy, manage and monitor VMs and remote storage [16]. Open vSwitch [17] was used as the underlying switch providing network connectivity to the VMs. Rest of the software implementation was same as used in Section 3.1.

Moreover we have implemented a stand-alone split-shuffle, to provide better insights into shuffle dynamics, where receiver service instances (e.g., reducers) fetch file spills from sender service instances (e.g., mappers) in a split fashion using standard Hadoop http mechanism.

To investigate performance of the proposed scheme as well as middlebox placement in different scenarios, we used following two commonly-used data center topologies:

1. Tree topology with middlebox at bisection (Top-1).
2. Tree topology with NIC-Teaming. Moreover, in this topology the middlebox is placed at first L2-switch (Top-2).

We use the following performance metrics:

- **Volume-of-Communication (VoC)**, defined as the amount of data crossing the network bisection.
- **Goodput (GP)**, defined as the number of useful information bits delivered to the receiver service instance per unit of time.
- **Bits-per-Joule (BpJ)**, defined as the number of bits that can be transmitted per Joule of energy.

**Acknowledgements:** Authors would like to thank Kostas Katrinis from IBM Research for his help and support.

### 4. REFERENCES

[1] Zakia Asad, Mohammad Asad Rehman Chaudhry, and David Malone. Codhoop: A system for optimizing big data processing. In
Figure 4: Goodput versus link rates for sorting benchmark for topology Top-1.

Figure 5: Goodput for different oversubscription ratios using sorting benchmark for topology Top-1 with link rate at 1000 Mbps

Figure 6: BpJ versus link rates using sorting benchmark for topology Top-1.