Supporting Application-specific In-network Processing in Data Centres

Luo Mai
Imperial College London
luo.mai11@imperial.ac.uk

Lukas Rupprecht
Imperial College London
lr12@imperial.ac.uk

Paolo Costa
Microsoft Research
paolo.costa@microsoft.com

Matteo Migliavacca
University of Kent
m.migliavacca@kent.ac.uk

Peter Pietzuch
Imperial College London
prp@doc.ic.ac.uk

Alexander L. Wolf
Imperial College London
a.wolf@imperial.ac.uk

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications; Distributed Networks

General Terms

Design, Performance

Keywords

Data Centres, In-network Processing, Network-as-a-Service

1. INTRODUCTION

Modern data centers (DCs) host a range of distributed applications that process large data sets, such as Google MapReduce and Microsoft Scope, or respond to user requests for online services, such as search engines [1], distributed data stores [13], and interactive query systems [12]. These DC applications manage to handle large amounts of data and high numbers of concurrent requests through "scale out" – computation is executed in a parallel fashion by distributing data across a large number of machines.

Typically, such applications operate on data following a *partition-aggregate* paradigm: In the *partition* phase, a master node partitions a job or user request into sub-tasks, which are scheduled to different worker nodes for concurrent execution. Each worker operates on a subset of data and generates partial results locally. For example, in Hadoop MapReduce, a large dataset is divided and dispatched to mappers to execute sub-tasks independently. In the *aggregate* phase, the partial results generated by the mappers are collected and aggregated by one or more reducers.

During the aggregation phase a large number of many-to-few network flows are generated. This traffic pattern is hard to support with traditional infrastructures due to network over-subscription in the core (1:20 or higher is not unusual [8]) and the limited bandwidth available at the end host (typically 1 Gbps or 10 Gbps at most). This is consistent with recently published studies on real-world clusters, showing that a significant part of a job is spent on network activity. For example, Facebook's traces [4] show that 42% of MapReduce jobs spend more than 50% time in the shuffle phase.

Many solutions have been proposed to improve the performance of such network-bound applications in DCs. New DC network topologies [8, 9] offer full bisection bandwidth but require complex mechanism to fully exploit the available bandwidth, e.g., [2, 14]. More critically, their performance is still bound by the limited bandwidth at the edge. Other proposals focus on reducing bandwidth usage by employing overlay networks to distribute data [4] and to perform partial aggregation [11]. However, this requires applications to reverse-engineer the physical network topology to optimise the layout of overlay networks. Even with full network visibility, an optimal mapping from a logical topology to the physical network is hard to obtain.

Recently, triggered by the availability of multi-core CPUs at low cost, there has been a renewed interest in software-based routers [7, 10], which replace the traditional operations of switches and routers with custom software implementations. We argue that the flexibility provided by these implementations should be offered to tenants to implement part of the application logic in the network. We refer to this approach as "Network-as-a-Service" (NaaS) [6] because it exposes networking as a service to the application, e.g., in-network data aggregation [5] or content-based routing [3], to improve network usage and thus mitigate bandwidth scarcity.

By modifying the content of packets on-path, tenants can implement customised services that are *application-specific*, as opposed to *application-agnostic*. For example, batch processing jobs like MapReduce would greatly benefit when mapping results are aggregated on paths towards reducers, thus improving processing throughput [5]. Data intensive web applications such as distributed search engines [1] can reduce network traffic by moving aggregation of partial query results close to data sources instead of frontend servers.

2. ON-GOING WORK

As a first step in this direction, we prototype a general-purpose software platform that allows users to deploy in-network data aggregation services (NETAGG) for partition / aggregate applications. NETAGG exploits the observation that many DC applications can aggregate data more efficiently – without introducing bottlenecks at edge machines – if they take the network topology into account and aggregate in-network, e.g., along the network paths. This permits the aggregation phase to utilise higher link bandwidth that exist at the upper tiers of network topologies (aggregation and core

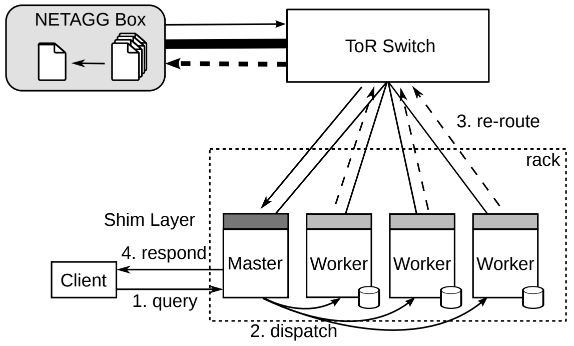

Figure 1: An example using NETAGG in DC applications.

switches) and thus reduces the overall network traffic and improving application performance.

To convey the idea behind our approach, we give an example of a NETAGG use case (see Figure 1). In our sample set-up, a ToR switch is connected to machines in a rack via 1 Gbps links and a collocated NETAGG box via a 10 Gbps link. The master node receives user requests from a client application and dispatches sub-requests to worker machines that index data. The worker responses can generate an aggregation flow back to the master of up to 3 Gbps; but are capped at 1 Gbps because of the over-subscribed edge link. By using NETAGG, the flow is redirected to the NETAGG box through the 10 Gbps link, and only aggregation results, which are usually smaller in size, are sent back to the master. This allows to support the full aggregation bandwidth of the workers and removes the bottleneck to the master.

The design of NETAGG consists of the following aspects:

(1) *deployment cost*: to foster the adoption of in-network aggregation, it should be realised with affordable changes to the DC networking hardware. In particular, it should be compatible with existing DC topologies and adaptable to an incremental deployment. Since programmable switches are expensive and only provide low-level interfaces that are difficult to expose to users, we exploit a software-only approach to custom networking to keep deployment costs low. We develop a dedicated in-network machine, called NETAGG box, where arbitrary aggregation functions tailored for different applications can be easily deployed by users. These boxes are commodity machines connected via high bandwidth links to switches at the upper levels of a DC network topology. Multiple co-operating NETAGG boxes can exist in a network topology to form an aggregation tree;

(2) *application transparency*: a wide range of DC applications should benefit from in-network aggregation without the need for substantial modifications. NETAGG deploys *shim-layer*s at edge machines to transparently intercept application traffic at the socket layer so that it can be redirected to and processed by NETAGG boxes;

(3) *aggregation performance*: to improve the performance of applications, in-network aggregation must be able to process data at rates higher than the link capacity provided by edge machines. For example, in a typical DC network with 1 Gbps edge links, the application can harvest performance gain only when the aggregated incoming throughput to a NETAGG box grows above 1 Gbps. To aggregate data at the highest possible rate, NETAGG minimises the functionality executed on NETAGG boxes. For example, the shim layers transcode data from application-specific protocols to an efficient binary representation in order to reduce data parsing overhead at NETAGG boxes;

(4) *multi-tenancy*: As shared facilities, DCs need to accommodate multiple applications to conduct in-network aggregation concurrently. NETAGG provide a programming interface to execute multiple application-specific aggregation functions. It schedules the processing of data belonging to different applications at NETAGG boxes. This permits NETAGG to prioritise data aggregation of latency-sensitive applications such as online services over throughput-sensitive batch data processing applications.

So far we have enabled NETAGG to improve the throughput of the Apache Hadoop map/reduce framework and the Apache Solr distributed text search engine [1]. For evaluation, NETAGG is deployed on a testbed that consists of a 16-core 2.9 GHz Xeon machine (the NETAGG box) connected to a ToR switch via a 10 Gbps link and 10 edge machines connected to the switch via 1 Gbps links. Evaluation results show that network traffic is reduced significantly and, as a consequence, application goodput is improved in both scenarios.

3. REFERENCES

[1] Apache Solr. `http://lucene.apache.org/solr`.
[2] AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B., HUANG, N., AND VAHDAT, A. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI* (2010).
[3] CARZANIGA, A., AND WOLF, A. L. Forwarding in a Content-Based Network. In *SIGCOMM* (2003).
[4] CHOWDHURY, M., ZAHARIA, M., MA, J., JORDAN, M. I., AND STOICA, I. Managing Data Transfers in Computer Clusters with Orchestra. In *SIGCOMM* (2011).
[5] COSTA, P., DONNELLY, A., ROWSTRON, A., AND O'SHEA, G. Camdoop: Exploiting In-network Aggregation for Big Data Applications. In *NSDI* (2012).
[6] COSTA, P., MIGLIAVACCA, M., PIETZUCH, P., AND WOLF, A. L. NaaS: Network-as-a-Service in the Cloud. In *Hot-ICE* (2012).
[7] DOBRESCU, M., EGI, N., ARGYRAKI, K., CHUN, B.-G., FALL, K., IANNACCONE, G., KNIES, A., MANESH, M., AND RATNASAMY, S. RouteBricks: Exploiting Parallelism To Scale Software Routers. In *SOSP* (2009).
[8] GREENBERG, A., HAMILTON, J. R., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D. A., PATEL, P., AND SENGUPTA, S. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM* (2009).
[9] GUO, C., LU, G., LI, D., WU, H., ZHANG, X., SHI, Y., TIAN, C., ZHANG, Y., AND LU, S. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *SIGCOMM* (2009).
[10] HAN, S., JANG, K., PARK, K., AND MOON, S. PacketShader: A GPU-Accelerated Software Router. In *SIGCOMM* (2010).
[11] LOGOTHETIS, D., TREZZO, C., WEBB, K. C., AND YOCUM, K. In-situ MapReduce for Log Processing. In *USENIX ATC* (2011).
[12] MELNIK, S., GUBAREV, A., LONG, J. J., ROMER, G., SHIVAKUMAR, S., TOLTON, M., AND VASSILAKIS, T. Dremel: Interactive Analysis of Web-scale Datasets. In *VLDB* (2010).
[13] NISHTALA, R., FUGAL, H., GRIMM, S., KWIATKOWSKI, M., LEE, H., LI, H. C., MCELROY, R., PALECZNY, M., PEEK, D., SAAB, P., STAFFORD, D., TUNG, T., AND VENKATARAMANI, V. Scaling Memcached at Facebook. In *NSDI* (2013).
[14] RAICIU, C., BARRE, S., PLUNTKE, C., GREENHALGH, A., WISCHIK, D., AND HANDLEY, M. Improving Datacenter Performance and Robustness with Multipath TCP. In *SIGCOMM* (2011).