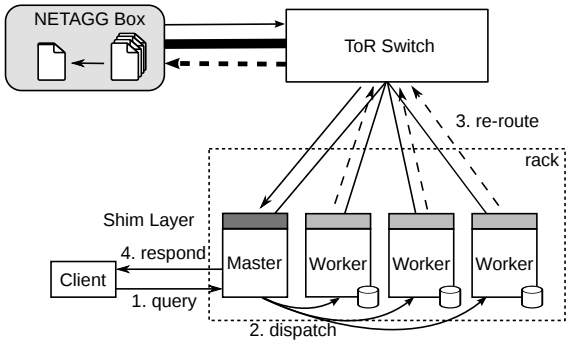

Figure 1: An example using NETAGG in DC applications.

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 cooperating 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).