ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Active storage networks: Using embedded computation in the network switch for cluster data processing

Janardhan Singaraju, Ajithkumar Thamarakuzhi, John A. Chandy *

Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA

HIGHLIGHTS

- Computation within a network switch accelerates data processing.
- FPGA implementations of modules to process data on the fly through the network.
- Data processing modules include data search, sort, *k*-min/*k*-max, and *k*-means clustering.

ARTICLE INFO

Article history:
Received 6 June 2013
Received in revised form
1 October 2014
Accepted 10 October 2014
Available online xxxx

Keywords:
FPGA
Data processing
Active storage

ABSTRACT

High performance data processing clusters use parallelism to accelerate computation. Often, these computation processes require the transmission of data across networks. In this paper, we propose the use of computation within the network switch to perform computation on data on the fly and further accelerate computation. We call networks built with these compute switches Active Storage Networks (ASN) and they provide an opportunity to optimize storage system and computational performance by offloading some computation to the network switch. We present an approach to perform transformation and reduction data operations in a network switch comprised of FPGAs. In this paper, we demonstrate an ASN using representative data processing applications, namely data search, data sort, *k*-min/max, and *k*-means clustering.

1. Introduction

High performance computing often requires computations on large amounts of data that may reside on the cluster computation nodes or on a separate clustered storage system. In order to operate on this data or simply collect the data after a parallel computation, it will require transmission of data over a network. The transmission of data across the network provides an opportunity to do further computation on the data while the data is in transit. We propose taking advantage of this opportunity by embedding compute capabilities into the network in order to do computation on data as it is flowing through the network.

This is the principle behind what we call an active storage network (ASN), a network with embedded computation. The "active storage" nomenclature is to indicate that the network works in concert with active storage systems—i.e. computation at the storage node. An ASN can enhance storage node performance as well as improve the computational performance of the parallel I/O systems. In this paper, we illustrate the power of an ASN by performing data processing in an intelligent switching system which is built using FPGAs. We demonstrate a few data processing applications, namely data sort, data search, *k*-min/max, and *k*-means clustering, and show the performance improvements made by offloading processing from the computation node to the network by comparing the application computation time using an ASN with the computation time when it is performed with a normal switch.

While performing a reduction operation, an ASN switch can perform the operation at the switch level thereby eliminating some of the traffic in the network that could consume the client bandwidth. For example, in a system with 10 storage nodes, if the client wants to perform an operation to find the minimum of a large set of keys stored across these storage nodes, an ASN switch with some computation capabilities could eliminate 90% of the non-minimum keys from each node which would be discarded at the client anyway. This allows the switch to send data to the client at full line rate even though it receives data from the storage node in parallel at 10 times the line rate.

As mentioned above, ASNs build upon the active disk concept [1–3]. In active disks, computation can be offloaded from the

* Corresponding author.
E-mail address: john.chandy@uconn.edu (J.A. Chandy).

http://dx.doi.org/10.1016/j.future.2014.10.020
0167-739X/

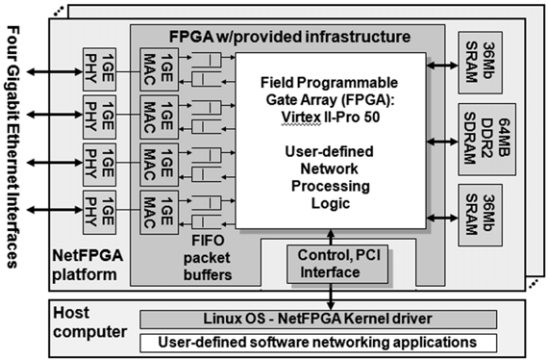

Fig. 1. NetFPGA board architecture.

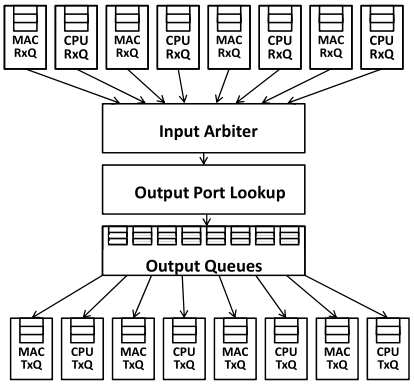

Fig. 2. NetFPGA switch pipeline.

processor to the disk. Previous work has demonstrated the effectiveness of this approach particularly with functions such as storage management, data mining, and multimedia [4]. However, the drawback of active disks in a distributed storage setting is that the data is striped across several storage nodes and each processor at the storage node can only see data residing at that node. Thus, any computation at the storage node cannot operate on the entire set of data spread across storage nodes. For example, when doing a query in a database for the k items closest to a particular key, each of the m storage nodes will return the k closest items in its portion of the data. The requesting client must then search through mk items to determine the k closest items overall. The overall computation is $O(n) + O(mk)$ where n is the number of data items per storage node. In an ASN, the $O(mk)$ search can be offloaded from the client to the network switch which can perform the search at line speed. Processing the data as it flows through the network essentially means the computation comes for free, since we have to pay for that transmission time anyway.

In an ASN, the goal is to move computation to the network which has a better view of data than the individual storage node, thereby optimizing network performance. Processing ability on the network also eases some of the computational workload at the network client. Most of the applications that operate on large sets of data require transforming the data from one form to another. Examples include file compression, video editing, and data encoding/decoding applications. Offloading data intensive parts of these applications to the network could ease client computing resources. It could also reduce network traffic as some of the data transfer operations that read and write data from client to the storage can be avoided. This further provides the impetus to embed data processing in a network.

As with an active disk model, the ASN is intended to be programmable. In other words, as the user's application changes the ASN could be reprogrammed to implement specific data processing components. In this paper, we present four specific application kernels, namely data sort, data search, k-min/max, and k-means clustering. However, the ASN can easily be reconfigured to implement many other computational kernels that may be found in HPC applications. One can conceive of a number of possible kernels including reduction operations like average, sum, and compression and transformational operations like FFTs and encryption. A kernel is suitable for implementation in an ASN if the data is stored on distributed nodes and the kernel either does a large scale reduction like a search query or does a data transformation such as a sort.

Some other applications that could benefit from the idea of ASN include redundancy optimizations and file system caching. Parallel I/O systems provide data protection through replication and parity across nodes in the cluster. With the use of an active storage network, the switch can offload parity computations from the client. Distributed file system performance often depends on aggressive caching to reduce network traffic. However, managing a cache across multiple clients can be problematic. ASNs offer an opportunity to present a global cache shared amongst all clients. By placing a centralized cache at the switch, we can free up memory at the client for other purposes. In this paper, we focus on the computational aspects of an ASN.

The rest of the paper is organized as follows: Section 2 describes the ASN architecture built using the Stanford NetFPGA and a 2-dilated flattened butterfly (2DFB) switch, Section 3 introduces some example data processing applications built using the ASN, and finally we close in Section 4 with an evaluation of the performance improvements made by computing on data within the network.

2. ASN architecture

A critical component of an ASN is the network switch, since the switch implements the data processing on data as it is aggregated and distributed from multiple sources. Typically, custom silicon is used to build gigabit and multi-gigabit switches and these switches offer the best performance. While ASICs could be used to implement the ASN switch, we would lose the ability to reprogram the switch for different application needs. Microprocessors offer programmability but cannot keep up with multi-gigabit speeds in a switch. FPGAs, on the other hand, provide an intermediate design point by offering network processing programmability while still achieving high performance. For this reason, our ASN uses FPGAs to implement the core network switch.

The ASN switch is built using a NetFPGA board designed by Stanford University and Digilent Inc., to help build prototypes of hardware-accelerated networking systems [5].

2.1. NetFPGA

2.1.1. NetFPGA architecture

The NetFPGA is a PCI card that contains a Xilinx Virtex-II Pro (XC2VP50) FPGA and is specifically designed for network applications by a research group at Stanford University. It has four 1 Gb/s Ethernet (GigE) interfaces and two SATA ports which make it suitable to build a switching network. It also has four banks of locally-attached static and dynamic random access memory (SRAM and DRAM).

Fig. 1 [6] shows the full resources available on the NetFPGA.

2.1.2. Reference switch pipeline

Fig. 2 shows the design of the reference Ethernet switch provided in the NetFPGA package [5]. It is a five stage pipeline structure where each module communicates using a simple packet based synchronous FIFO push interface which makes it easy to add additional modules to the structure for the purpose of packet

Please cite this article in press as: J. Singaraju, et al., Active storage networks: Using embedded computation in the network switch for cluster data processing, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.020