Embedding Applications within a Storage Appliance

Roger D. Chamberlain
Exegy Inc. and Dept. of Computer Science and Engineering, Washington University in St. Louis
roger@wustl.edu

Introduction

Storage densities for magnetic media have improved over the last decade at a rate that exceeds that of Moore's Law. The pace of this technological advance has resulted in dramatic cost decreases for hard disk storage systems, and as a result, many organizations find it less expensive to simply "save everything" rather than incur the expense to determine what must be saved and what can be safely discarded. Emails, memos, customer correspondence, current product information, obsolete product information, raw sensor data from the plant control systems, literally all of it is stored on magnetic disk, and all of it gets saved.

The "save everything" mentality addresses the short-term issue of what should be archived vs. what can be deleted. It, however, raises an important follow-on question: "Now, how do I find the information I need in all of these terabytes of saved stuff?"

Traditional storage systems simply retain information. We have constructed a storage system that can search through the data stored within it and provide answers to sophisticated queries. In effect, we have provided application-level computing capabilities to the storage system.

The Exegy K•Appliance™

Exegy Inc., in collaboration with Washington University in St. Louis, has developed the K•Appliance [1,2], a network appliance that provides network-attached storage augmented with a high-performance, application-level computing capability. Traditional network-attached storage systems provide a file system interface to client computers (e.g., desktop systems). Our system includes reconfigurable logic within the appliance, providing the ability to process data stored on the appliance without moving them across the network.

As an appliance, users are not expected to develop applications directly, but to simply invoke functionality provided by the appliance. The library of functions currently available includes: exact and approximate text search, record/field selection operations on structured data, encryption, decryption, and signature hashing. Functions currently in development include: biosequence search, signal processing (e.g., filtering, FFT), and various data mining primitives.

Typical usage is via the following pattern. Data are stored on the appliance using writes to the traditional file system interface. The data are queried by a network socket interface that invokes the requested application and returns the query results via the network to the requesting client system. Of course, traditional file system reads are supported as well.

Figure 1 illustrates the internal architecture of an individual appliance. Data flows off the disks into an FPGA. The FPGA provides reconfigurable logic that has its function specified via firmware. We have designed and built a standard firmware socket that handles data movement requirements into and out of the FPGA, providing a consistent application interface to firmware application modules that are deployed within the FPGA. Results of the processing performed on the FPGA are delivered to the processor(s). By delivering the high-volume data directly to the FPGA, the processor(s) can be relieved of the requirement of handling the bulk of the original data set.

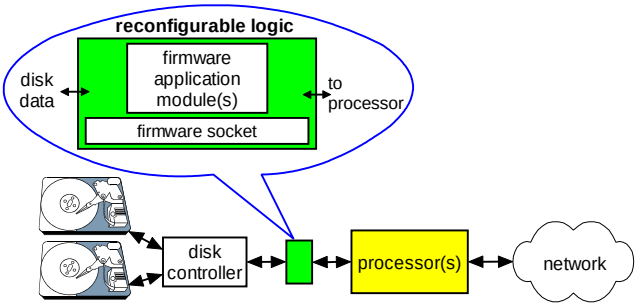

Figure 1: Exegy K•Appliance architecture

Development Environment

While application development is typically not the responsibility of the end user, a robust and efficient development environment is still essential. Figure 2 shows the framework for the development of applications on the K•Appliance. The top three layers represent functionality that is executed in software on the appliance's processor(s), while the bottom two layers represent functionality that is executed in firmware on the FPGA.

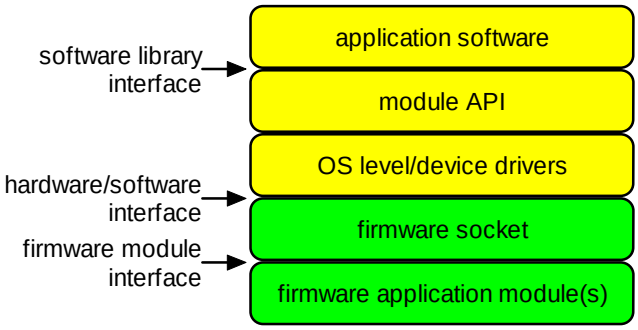

Figure 2: Application development environment

The author is on leave from Washington University in St. Louis at Exegy Inc., 3668 S. Geyer Rd., Suite 300, St. Louis, MO 63127, www.exegy.com

The central three layers of this framework are fairly application independent, supporting data movement between the application software at the top of the framework and the firmware application module(s) at the bottom of the framework.

Another way to view the application decomposition is shown in Figure 3. The top portion of the pyramid represents application functionality performed in software, and the bottom portion of the pyramid represents application functionality deployed in firmware. The firmware, with its greater execution speed, is well-matched to application functions that need to examine a greater data volume. The software, with its ease of programmability, is well suited to application functions that have greater algorithmic complexity.

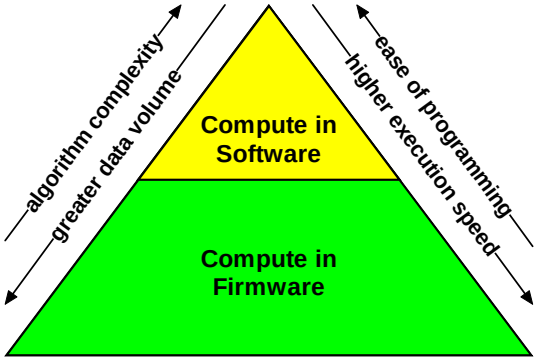

Figure 3: Hardware/software application partitioning

We illustrate this decomposition using the example of an approximate text search application. Consider a search (within a set of unstructured text files) for matches on the following query:

(Czar AND Nicholas) NEAR Crimea

In addition, it can be specified that the first word may have up to two characters different than the literal string (so as to enable matching the text "Tsar" as well as "Czar") and that the NEAR operator is parameterized to require a match within 1000 characters (i.e., its arguments must reside within 1000 characters of one another).

In our current implementation of this approximate search application, the individual keywords in the query (i.e., "Czar," "Nicholas," and "Crimea") are found through firmware (including the approximate matches) and the logical combinations of these results (i.e., the AND and NEAR operators) are computed in software. In this way the firmware considers the bulk of the raw data, and the software operates on results returned by the firmware. The client system only receives results that match the complete query.

Next we consider the more general application decomposition required in a data mining operation. The firmware is assigned responsibility for the simple, high data-rate, repetitive operations that must examine the entire data set, effectively filtering the raw data into a more manageable size prior to the higher-level processing. The software is assigned responsibility for processing the filtered output looking for high-level semantic information. This can include probability modeling, knowledge-base modeling, etc.

Applications

Our first set of applications for the K•Appliance focused on unstructured text search. Both exact and approximate keyword search applications were developed [3], and performance improvements of greater than 200-fold were reported when compared to a high-end Xeon processor [2].

Our next set of applications is aimed at structured data sets. When the data are organized as logical records with individual fields within each record, common low-level operations to be performed include record selection, in which individual records are examined for inclusion or exclusion in the return set, and field selection, in which particular fields are designated for later inspection. Performance characterization of these applications is currently ongoing.

We have also deployed the 3DES encryption/decryption application as well as the MD5 signature hashing application on the appliance. The use of the system for biosequence similarity search is described in [4], and several signal processing primitives (including FFT and FIR filtering) are under development.

Conclusions

We describe the Exegy K•Appliance, a network-attached storage device that supports application-level processing of very large data sets. Providing as much as 4 TB of storage, the K•Appliance also addresses the needs of users to filter their data into useful knowledge. The goal is to make the data storage a resource, rather than just a repository.

References

[1] R. D. Chamberlain, R. K. Cytron, M. A. Franklin, and R. S. Indeck, "The *Mercury* System: Exploiting Truly Fast Hardware for Data Search," in *Proc. of Int'l Workshop on Storage Network Architecture and Parallel I/Os*, September 2003, pp. 65-72.

[2] M. A. Franklin, R. D. Chamberlain, M. Henrichs, B. Shands, and J. White, "An Architecture for Fast Processing of Large Unstructured Data Sets," in *Proc. of 22nd Int'l Conf. on Computer Design*, October 2004, pp. 280-287.

[3] Q. Zhang, R. D. Chamberlain, R. S. Indeck, B. West, and J. White, "Massively Parallel Data Mining Using Reconfigurable Hardware: Approximate String Matching," in *Proc. of Workshop on Massively Parallel Processing*, April 2004.

[4] P. Krishnamurthy, J. Buhler, R. D. Chamberlain, M. A. Franklin, K. Gyang, and J. Lancaster, "Biosequence Similarity Search on the Mercury System," in *Proc. of the IEEE 15th Int'l Conf. on Application-Specific Systems, Architectures and Processors*, September 2004, pp. 365-375.