Editors: Volodymyr Kindratenko, kindr@ncsa.uiuc.edu
Pedro Trancoso, pedro@cs.ucy.ac.cy

XTREMEDATA DBX: AN FPGA-BASED DATA WAREHOUSE APPLIANCE

By Todd C. Scofield, Jeffrey A. Delmerico, Vipin Chaudhary, and Geno Valente

FPGA-based architectures are known for their applicability to embedded systems. Nevertheless, recent developments make it possible to exploit this technology's benefits for large-scale systems targeting compute- and data-intensive applications.

Green computing—low power, "eco-friendly"—is a hot topic these days and will influence server design for the coming years. Green computing's importance was firmly recognized by the research community with the first semiannual publication of the Green500 List (www.green500.org) in November 2007. The Green500 identifies the most energy-efficient supercomputers in the world by measuring the performance per watt of the TOP500 Supercomputers list (www.top500.org). The current top 10 in the November 2009 Green500 list all use low-power accelerators or processors—such as a Cell Processor, a graphical processing unit (GPU), a greatly reduced array of processor elements (GRAPE)[1] or a PowerPC—to achieve their high performance-per-watt results.

Traditional Linux clusters are great for scaling problems and leveraging commodity components to solve large computing problems. Nevertheless, they're not power efficient. Linux clusters are built from general-purpose CPUs that typically contain more features and hardware than developers need to solve certain problems. This underutilization results in unnecessary power consumption. In addition, clusters are usually configured with large memory and mechanical storage, thus wastefully burning even more power. Finally, the software leveraged on these platforms typically can't take full advantage of multicore CPUs (let alone tomorrow's many-core versions). Overall, the system's general-purpose nature results in large inefficiency in terms of power consumption. A power-efficient solution is to design systems that are more specialized to the applications they're running.

As with embedded computing, for high-performance computing (HPC), we should be looking at architectures that are built for a specific purpose as opposed to generic architectures—such as Linux clusters—that can handle different applications. If the cost, power, and size of these specific machines are substantially less than the generic solution, then it's likely that the market will shift back to such application-specific architectures.

Rather than design a new system for each application, however, it's possible to customize a system with dedicated accelerators to better match the application's requirements, thus resulting in better efficiency. Accelerators, if abstracted from the IT user, can be successfully used in data-intensive supercomputers (DISC). IT users can benefit tremendously from an appliance that transparently provides performance, lower power, and ease of scalability along with a proper mix of storage, CPU, software, and acceleration.

Products based on field-programmable gate arrays have focused on solving some of the industry's toughest digital problems. Over the past decade, very large FPGAs have been implemented in applications such as radar, cryptography, WiMax/Long Term Evolution (LTE), and software-defined radio (SDR). Factors including cost pressure and stringent size, weight, and power (SWaP) requirements have created constrained environments that require high-performance and high-efficiency architectures. FPGA-based systems can satisfy these requirements. We believe that the best computing configuration currently available consists of x86 CPUs coupled to FPGA-based accelerators.

FPGAs have several attractive key features:

- They're highly power efficient, outperforming CPUs on the performance/watt metric by two to three orders of magnitude.[2]
- They closely follow the CPU's semiconductor process technology, typically lagging by no more than six to 12 months (and thus FPGAs accrue all of the Moore's law benefits).
- They're in-system reconfigurable. Unlike fixed architecture accelerators, FPGAs can be instantly reloaded to optimally match application requirements.

Today, it's generally accepted that developers can achieve scale-up on a large scale only through the loosely coupled massively parallel processing (MPP) approach. We believe that this scale-up philosophy, complemented with accelerators (such as FPGAs), will be leveraged in next-generation data warehouse architectures. Using FPGAs as HPC accelerators has been widely explored in the literature. Keith D. Underwood and his colleagues cover the current state well[3] and forward a case for the simplicity of a *computing appliance* model. An important element of this model is placing a known and simplified software development environment (C, C++, SQL) between the user/developer and the complexities of traditional FPGA development environments.

The research accelerator for multiple processors (RAMP) project developed an FPGA-based hardware and software environment that simplifies the design and development of next-generation supercomputers (see http://ramp.eecs.berkeley.edu). The developers are effectively applying the RAMP system to the design of next-generation petascale supercomputers.[4] The benefits of performance/price, performance/power, "under the hood" acceleration, and ease of use are the key factors that will drive the design of future database and data appliance systems.

FPGA-based Data Intensive Supercomputer Architecture

We based the XtremeData DISC system architecture on a commodity Linux cluster augmented with direct-attached distributed storage, high-speed interconnect network (InfiniBand), and a re-engineered open-source database engine (PostgreSQL). The latter supports a shared-nothing, parallel query execution model and uses FPGA hardware to accelerate common database operations ("SQL in Silicon").

Our system has the following characteristics:

- scalable, shared-nothing MPP architecture;
- hardware-accelerated parallel SQL processing;
- efficient, zero-copy, data exchange on high-bandwidth network; and
- dynamic load balancing at runtime.

At XtremeData, we implemented FPGA-based accelerators in the *in-socket accelerator* (ISA) module (www.xtremedata.com/products/accelerators/in-socket-accelerator), which is pin compatible with the CPU socket. The ISA lets developers easily integrate FPGAs with servers from many vendors and can generate an order of magnitude performance improvement over the replaced CPU at a fraction of its power consumption. FPGAs are reprogrammable in-system and offer a versatile platform for implementing processing engines. Key SQL operations are accelerated under the hood inside the FPGA. These operations include large data movement and time-consuming functions such as `Joins`, `Sorts`, `GroupBy`, `OrderBy`, and `Aggregations`.

The XtremeData database analysis appliance (dbX) that realizes the DISC system architecture consists of a `Head Node` coupled to *N* `Data Nodes`. Each `Data Node` is a self-contained slice of the database appliance encapsulating local storage, local computing (CPU + FPGA), and the database query execution engine. The appliance is easily scalable by adding `Data Nodes` (up to a maximum of 1,024 nodes) within a single tower or across multiple towers. An appliance that spans multiple towers will have the `Head Node` on the first tower and a `Coordinator Node` on each of the remaining towers. Each `Coordinator Node` will offload some `Head Node` functions, such as staging during data load/unload and the computation of interim aggregates up to the tower level. As a true appliance, its deployment is simple: the user simply powers up the system, connects to the network, loads data, and begins to explore the data using SQL.

Figure 1 shows a block diagram of the dbX system. Each system node (`Head` and *N* `Data`) runs a standard distribution of the Linux OS, clustered via an InfiniBand network. The `Head Node` performs traditional database engine front-end functions, including management of external connections, user sessions, metadata, query parsing, and query execution-plan generation. Each `Data Node` performs the database's back-end functions, including table storage and access management, indexes, query execution, and data exchange among nodes. The dbX system is largely data-model agnostic, and users aren't burdened with using data partitioning and colocation schemes to address load-balancing considerations. We designed the dbX engine to automatically analyze data streams at query runtime and dynamically guarantee load balancing. Multiple parallel high-bandwidth pathways facilitate high-speed loading and unloading of data into and out of the database. The dbX environment also efficiently accomplishes data profiling for quality or audit purposes, eliminating the need for external systems and third-party tools.

Case Study

Our investigation was motivated by biostatisticians' need to perform

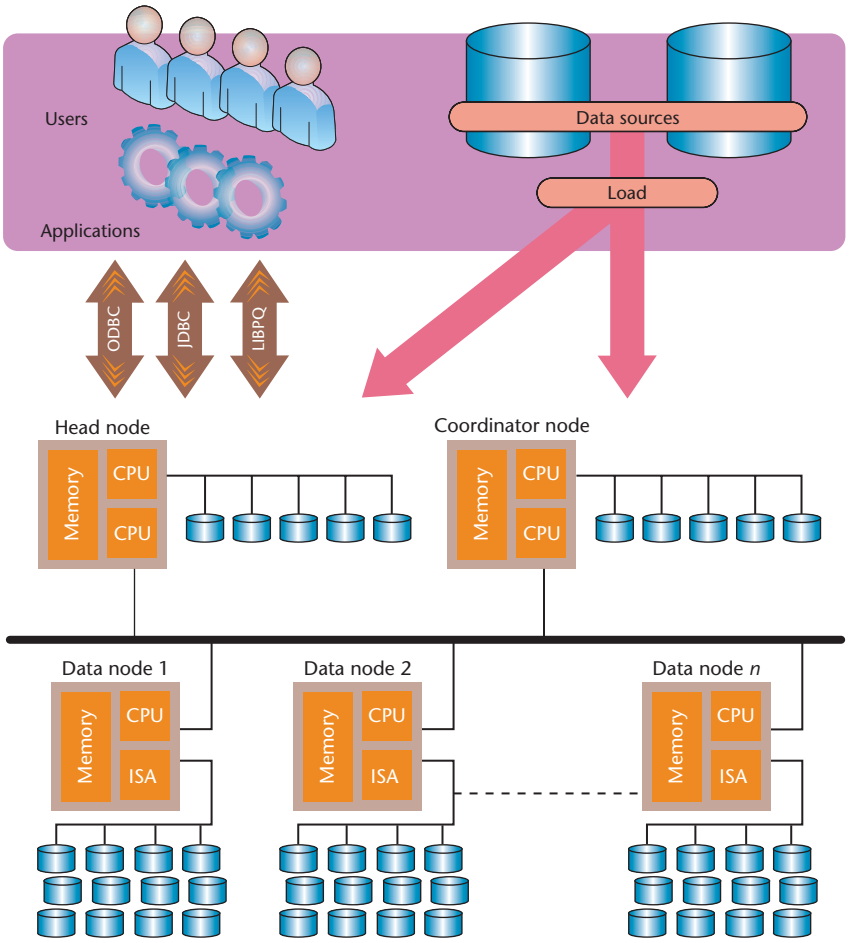

Figure 1. XtremeData's dbX system. Each system node (`Head` and *N* `Data`) runs a standard distribution of the Linux OS, clustered via an InfiniBand network. The `Head Node` performs database front-end functions, while each `Data Node` performs database back-end functions, including table storage and access management, indexes, query execution, and node-to-node data exchange.

correlation analysis on data produced from array comparative genomic hybridization (aCGH, also known as chromosomal microarray analysis). In this technique, DNA genes from a cancer cell and a healthy cell are tagged with different fluorescent markers, and the ratio of their fluorescent intensities is calculated for each gene. The data produced represents the ratio of the gene copy number of the cancer cell to the control cell, a good indicator of the level of that gene's expression in the individual.

Newer microarray machines can measure 244,000 or even 1,000,000 different locations within the genome. The initial data set is large, but not unwieldy. However, an important analysis of that data set is to measure the correlation of each gene's score with the scores of all the others. This method is applied to investigate the genetic causes of many types of cancer, for example.[5,6] For a microarray with N probes, this requires the calculation of an N entry by N entry correlation matrix. At double precision, such arrays can reach hundreds of gigabytes or more—prohibitively large sizes, even on some shared high-performance computing resources.

To enable this analysis even for newer higher-resolution microarray hardware, we developed software for computing the row-wise correlation and for performing queries against the correlation data for the retrieval of significant values. This application has been implemented on both a Linux cluster architecture and the specialized XtremeData dbX hardware. Although an end user's ability to perform this analysis on a traditional cluster is limited only by the available hardware resources, the performance comparison indicates the need for a paradigm shift in the implementation of such data-intensive applications.

We present methods and performance analyses of several implementations of this application on individual clusters (of up to 128 cores/nodes) with

- a storage area network (SAN),
- an IBRIX parallel file system, and
- with Hadoop,

and using a 16-node/64 core XtremeData dbX model 1016 FPGA-enabled data warehouse appliance. Each dbX node has a dual-FPGA accelerator (with a total of 32 FPGAs). Our results offer benchmarks for the performance of data-intensive applications within these distributed computing paradigms.

The Pearson product-moment correlation coefficient (the *correlation*) for two random variables represents the degree to which they're linearly related. We can express a pair of random variables (g_1, g_2), each with a mean μ and standard deviation σ, as

$$\rho g_1 g_2 = \frac{\mathrm{cov}(g_1 g_2)}{\sigma_1 \sigma_2},$$

where

$$\mathrm{cov}(g_1, g_2) = E((g_1 - \mu_{g_1})(g_2 - \mu_{g_2})).$$

We consider the copy number ratio for each gene location in a data

set as a random variable. Data consist of multiple measurements for each location, resulting in a 2D data set of N rows of measurements (one for each gene location) of m samples each. For all of our implementations of this application, we calculated the correlation for every pair of gene locations in a data set using the following steps:

- Compute the mean (μ) and standard deviation (σ) for each row in the data set.
- For each measurement, calculate its deviation from its row mean.
- Multiply the corresponding deviations for the two rows of data.
- Find the mean of those products and divide by the product of the two rows' standard deviations.

Because we're considering pair-wise correlation among N different genes, this results in an $N \times N$ correlation matrix if we compute all possible pairs. However, because

$$\rho_{g_1 g_2} = \rho_{g_1 g_2},$$

only the matrix's upper triangle contains distinct values. For all implementations, the correlation degree doesn't affect the performance of either generating or querying the correlation matrix. We must perform the same number of computations to calculate each entry in the resulting correlation matrix and perform a full scan of that matrix, regardless of the values' correlation.

To parallelize this computation for a cluster, we leveraged a set of routines developed by one of the authors (Delmerico) for working with large data sets.[5] This software consists of several standalone applications as well as bindings for the R Statistical Computing Package, which allow for the manipulation and analysis of large data sets that exceed the user's memory resources. The standalone applications accept a text-based data set—in this case, a microarray experiment's output—generate the correlation matrix, and then decompose it into smaller submatrices to "stripe" the data out over a storage array. Finally, a query routine scans the decomposed data and returns the values with the largest correlation.

All of these steps are performed in parallel, with the work being divided into blocks of rows for each processor. So, for an $N \times N$ correlation matrix, each of the P processes computes the correlation of NP gene locations with all of the others, producing an $NP \times N$ submatrix, the union of which contains all of the row-wise correlation values. Then, each process decomposes its correlation submatrix into smaller submatrices and writes them out to disk.

For a 2D data array, we use a block-cyclic decomposition. Users can specify the tile size to optimize performance for the system on which it's running. The decomposition program's output is a group of files—each storing an individual tile from the decomposition—that are archived on disk for subsequent analysis. During decomposition, each individual tile's dimensions and coordinates are calculated and output to users and a machine-readable metadata file. This file is later used during stored-data retrieval. By decomposing this correlation matrix into many smaller, more tractable submatrices and storing the decomposition's metadata, users can load subsets of the entire matrix into memory for individual analysis and scanning. Finally, each process scans the submatrix files from its portion of the matrix for the largest values, identifying the most correlated gene locations.

XtremeData dbX Approach to the Problem

Given that dbX is a database system, we were able to convert the microarray gene correlation problem's solution to a set of SQL statements (queries) and load the data into the database. This approach is enabled by dbX's MPP infrastructure. A data set of 19,116 $\times$ 43 of actual microarray output had been provided. For other input sizes, we created a simple C program that generated three columns of data consisting of two integers and a random double-precision floating-point number. The integers represented the gene and sample numbers. We converted the real data set to the same three-column format for standardization sake. For example, if there were 19,000 genes and 43 samples, the data file would have 19,000 $\times$ 43 records.

The basic process is to first load the data into the dbX 1016 system—a 16-node MPP version of the dbX data warehouse appliance. We accomplished this with a simple database `COPY` command. Once the data is in the database, the system performs three steps, all leveraging dbX's built-in SQL command line interface:

1. Convert the three-column table into a table with the number of samples, plus one column.
2. Compute each row's standard deviation and mean and, in the same step, calculate the difference from mean for each column and store the results in a temporary table.

The SQL code (truncated for clarity) for steps 1 and 2 is

```
select A.o_row as c_row,
  ((A.val_0 - A.avg_o) /
    (A.std_o * sqrt((43)-1)))
    as nv_0,
  ...,
  ((A.val_[n] - A.avg_[n]) /
    (A.std_[n] * sqrt((43)-1)))
    as nv_[n]
into adjustments
from
 (     select o_row,
       avg(o_value) as avg_o,
       stddev(o_value) as std_o,
       min(o_value) as min_o,
       max(o_value) as max_o,
       max(case when
         o_sample=0 then o_value
         else null end) as val_0,
       ...,
max(case when o_sample=[n]
  then o_value else null end)
  as val_[n]
from observations
      group by o_row
) as A
;
```

3. Compute the temporary table's Cartesian product, with the restriction that the first reference's gene either isn't equal to that of the second reference or is less than that of the second. The truncated SQL Code for Step 3 is

```
select       LROW as l_row,
      RROW as r_row,
      prod_0 + ... + prod_[n]
        as cor
into  correlations
from
( select A.c_row as LROW,
       B.c_row as RROW,
A.nv_0*B.nv_0 as prod_0,
      ....
      A.nv_[n]*B.nv_[n]
        as prod_[n],
from  adjustments A,
        adjustments B
where A.c_row < B.c_row
) as C;
```

The process output is another database table with three columns: gene A, gene B, and the correlation. The largest result set was the 1,000,000 × 43 generated with the "not equal" approach. This created a 999,999,000,000 record table that was 18 Tbytes in size.

The final step in the testing process was to run an "interrogation" query against the result sets. The query we chose returned the 200 most correlated genes from the result sets.

The entire procedure occurs in seven steps:

- Create a raw data table with three columns: gene id, sample id, value.
- Convert data into a loadable format or generate simulation data (ASCII text that matches raw data table structure).
- Load data using dbX `Copy`.
- Create a wide table with one row per gene and one column per sample.
- Create a worktable that stores the standard deviation, mean, and variance from mean over each column, for each row.
- Compute the correlation of every row to every other row in the worktable using a cross product: the SQL join of the worktable with itself. The cross product is restricted with one of two conditions; the first ensures that we don't calculate the correlation of X to X, and the second that we don't compute the correlation of both X and Y and Y and X. The queries ordered the results and stored the top correlations within the system for subsequent querying.
- Query the 200 most correlated gene–gene sets from the result set as an example of result interrogation performance.

Two engineers from XtremeData implemented the dbX approach in less than a week. We could perform any subsequent querying of the result set on an ad hoc basis using SQL; the longest running query tested against the 18 Tbytes result set took 16 minutes and 22 seconds.

Because dbX is a share-nothing MPP database engine, the system's user interface is simply SQL or the supported Java Database API (JDBC) and Open Database Connectivity (ODBC) drivers. The dbX system's FPGAs were abstracted from the user and are coprocessors to the CPU's execution engine running on every data node. Traditionally, long FPGA development times and a lack of high-level language support have limited FPGA uptake in the HPC marketplace. To address this, we preprogrammed the FPGA by accelerating key SQL operators, data movement capabilities, and statistics gathering. Ultimately, this created a high-performance FPGA-accelerated DISC without users having to know how to program FPGAs or deal with data partitioning. By removing the FPGA programming barrier, dbX gives scientists and statisticians the benefits of FPGA technology without their needing to know how to program in a hardware descriptions language such as VHDL or Verilog.

Case Study Results

We performed the following tests on the University at Buffalo's Center for Computational Research U2 cluster (www.ccr.buffalo.edu/display/WEB/U2)

and on a dbX 1016 appliance. For each data set, we computed the correlation matrix and then performed a query to return the 200 most correlated results. In addition to the real-world data set of 19,116 × 43, we synthetically produced data sets of various sizes for both the cluster and dbX (see Table 1). We chose these sizes partially to provide scalability estimates, but also to approximate the data set sizes that actual microarray experiments might produce.

In addition to the above experiments, we attempted to run further tests using larger data set sizes on the Hadoop cluster and dbX. Nevertheless (and without loss of generality), we can't present those results here as we couldn't execute the experiments on the cluster due to limited shared resources.

Although the cluster performance scaled adequately for the application's compute-bound generation portion, the I/O portions performed poorly, being restricted to single-digit speedups on any sizeable data set. In contrast, the dbX machine achieved large speedups over the 128 node cluster—up to 65 times for the generation step (see Figure 2) and up to 165 times (with the use of a C-code user defined function, UDF, to augment the SQL) for the query step (see Figure 3). The speedup that occurs with the dbX with increasing data set sizes is particularly noteworthy.

We've explored two implementations of an application intended to enable correlation analysis of microarray data, the volume of which is rendering it intractable on all but high-performance clusters and specialized hardware. Although the cluster version demonstrates that this application can be implemented on more conventional hardware, the performance is lacking precisely because a cluster is designed for general-purpose computing. Not only is a specialized machine such as the XtremeData dbX able to tackle these problems and scale them up beyond a cluster's capabilities, the performance is significantly better, even for smaller problems.

Finally, dbX has distinct power advantages. The ISA that houses the SQL In Silicon technology is based largely on FPGA. XtremeData's measurements on the ISA's peak power consumption have shown it to be less than 40 watts during real query operation. Given that the ISA replaces a quad-CPU with a peak power consumption of 120 watts, a node of dbX has an 80-watt advantage against its Hadoop 2-socket-per-node CPU-only counterpart. Thus, the DISC technology is not only faster and denser for such computational problems, but also has distinct performance-per-watt and power-per-square-foot advantages. Table 2 compares a U2 cluster rack and dbX appliance.

Table 1. Data sets used for performance evaluation.

Data set name	Number of genes	Number of samples
19K	19,116	43, 80, 120
40K	40,000	43, 80, 120
80K	80,000	43, 80, 120
120K	120,000	43, 80, 120

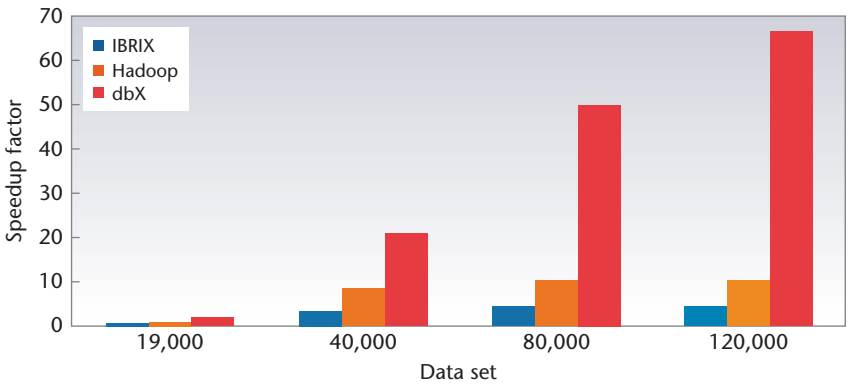

Figure 2. The generation step speedup over a storage area network for 128 cluster nodes and 16-node-64-core dbX. The sample size was 43. As the chart shows, dbX achieved a speedup of up to 65 times.

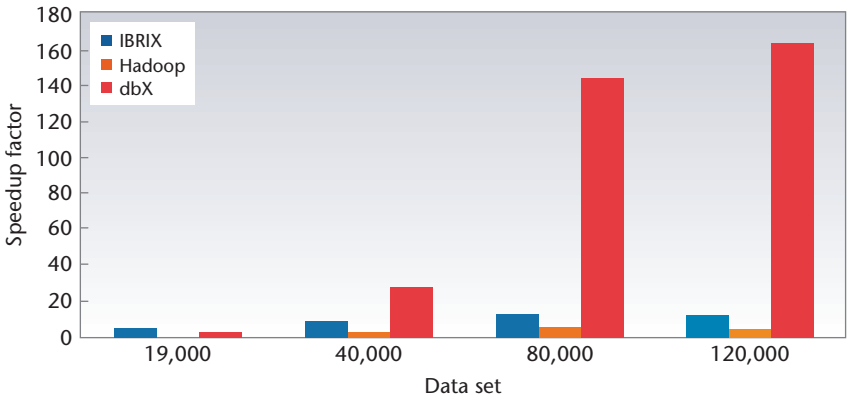

Figure 3. The query step speedup over a storage area network for 128 cluster nodes and 16-node-64-core dbX. The sample size was 43. As the chart shows, dbX achieved a speedup of up to 165 times.

Table 2: Rack-by-rack comparison of U2 Hadoop cluster and the XtremeData dbX appliance.

	U2 cluster	dbX 1016**
Size	1 rack (42U 19")	1 rack (42U 19")
Number of nodes/rack	32*	16
Node type	Dell SC1425 Dual socket—2 core	Data node 4 core + ISA-FPGA
Number of CPU cores/rack	64	64
Number of FPGAs/rack	0	16
Power/rack	14.4 kilowatt (est. peak)	8.6 kWatt (est. peak)

**128 nodes or 4 × U2 racks were used in the actual test results in Figures 2 and 3*

***A single-rack dbX 1016 was used in the actual test results in Figures 2 and 3*

Data-centric applications such as this can potentially drive development of this new computing paradigm forward, as well as benefit from improved performance and extended capabilities. The major challenges analysts face today are large and growing data sets and legacy systems that work well for predictable queries (operational reporting), but poorly for ad hoc querying. This isn't surprising given that historic spending has been directed toward optimizing traditional business intelligence. As a consequence, an entire industry now exists to provide workarounds, such as implementing query-aware data models; carefully constraining query workloads, complexity, and schedules; colocating data partitions sympathetically with the "most important" joins; and segregating ad-hoc querying into a separate environment. Ultimately, these are still patches or workarounds rather than real solutions, which should address the spectrum of requirements for ad hoc exploration, including

- unpredictable data access patterns;
- data sets that can be very large due to long time series, archived data, and subtransactional data;
- large data sets that have low information density, and therefore can't justify the ROI of a major system implementation; and
- large data sets that must be quickly loaded and unloaded from the analytics system.

The ideal solution for data exploration must satisfy these requirements, as well as

- provide access via industry standard SQL,
- execute queries extremely fast,
- be scalable to petabytes,
- provide predictable performance, and
- have a low total cost of ownership (TCO).

A solution that satisfies these requirements will naturally tend toward a fully integrated "appliance" that incorporates all the components of analytics systems: storage, computing, interconnect network, and database engine. An integrated appliance from a single vendor obviates the need for system configuration, integration, performance tuning, and the numerous workarounds mentioned earlier. This eliminates a significant labor cost component from the system and results in two immediately realizable benefits: fast deployment and a low operational TCO.

As enterprise customers and research organizations continue growing their data requirements, these requirements will demand four essential benefits: usability, scalability, cost-efficiency, and eco-friendliness. Obviously, a system that can store all data, perform all queries in seconds, cost no money, and take up no power or space would be ideal, but isn't possible. Still, the desire is there. The sweet spot in the ad hoc analytic market is for systems that allow cost-effective ways to scale beyond a petabyte, are simple and fast to use, allow unrestricted ad hoc access, and are green. No small feat, but by correctly applying the proper mix of commodity, open-source, and acceleration technology, XtremeData's novel architecture has created a data-intensive supercomputer that has achieved a previously unattainable level of performance.

Acknowledgments

We thank Andrew E. Bruno, Matthew D. Jones, and Steven M. Gallo of the Center for Computational Research, University at Buffalo, and Ali Sajanlal, director of analytic applications at XtremeData, for contributing to the benchmarking effort.

References

1. J. Makino, "Specialized Hardware for Supercomputing," *SciDAC Rev.*, no. 12, Spring 2009; www.scidacreview.org/0902/html/hardware.html.
2. J. Williams et al., "Computational Density of Fixed and Reconfigurable Multi-Core Devices for Application Acceleration," *Proc. 4th Reconfigurable Systems Summer Inst.*, Nat'l Center for Supercomputing Applications, 2008; www.rssi2008.org/proceedings/papers/presentations/10_Williams.pdf.
3. K.D. Underwood, K.S. Hemmert, and C.D. Ulmer, "From Silicon to Science: The Long Road to Production Reconfigurable Supercomputing," *Reconfigurable Computing: Architectures, Tools and Applications Technology*, LNCS 4943, Springer Verlag, 2008, p. 2, doi:10.1007/978-3-540-78610-8.
4. D. Donofrio et al, "Energy-Efficient Computing for Extreme Scale Science," *Computer*, vol. 42, no. 11, 2009, pp. 62–71.

5. M. Bredel et al., "High-Resolution Genome-Wide Mapping of Genetic Alterations in Human Glial Brain Tumors," *Cancer Research*, vol. 65, no. 10, 2005, pp. 4088–4096.
6. H. Lee, S.W. Kong, and P.J. Park, "Integrative Analysis Reveals the Direct and Indirect Interactions between DNA Copy Number Aberrations and Gene Expression Changes," *Bioinformatics*, vol. 24, no. 7, 2008, pp. 889–896.

Todd C. Scofield is managing director of Big Data Fast and founder and codirector of the University at Buffalo, SUNY, Data Intensive Discovery Initiative, an academic, government, and industry consortium focused on data-intensive science and engineering. His research interests include the application of novel technologies and methods to solving complex problems, and the psychology and dynamics of successful project, virtual, and interorganization team building. Scofield has a BS in marketing from the University of Connecticut. Contact him at todd.scofield@bigdatafast.com.

Jeffrey A. Delmerico is a PhD student in the Department of Computer Science and Engineering at the University at Buffalo, SUNY. His research interests include computational science, data-intensive computing, and computer vision. Delmerico has an MS in mathematics from the University at Buffalo, SUNY. Contact him at jad12@buffalo.edu.

Vipin Chaudhary is the CEO of Computational Research Laboratories Ltd., a wholly owned subsidiary of Tata Sons, Ltd. He also directs the Data Intensive Discovery Initiative and is an associate professor of computer science and engineering at the University at Buffalo, SUNY. His research interests are in cloud, grid, and high-performance computing and their applications to science, engineering, and medicine, as well as in data-intensive computing and computer-assisted diagnosis and interventions. Chaudhary has a PhD in electrical and computer engineering from the University of Texas at Austin. Contact him at vipin@buffalo.edu.

Geno Valente is vice president of worldwide sales and marketing at XtremeData. His research interests include field-programmable gate array acceleration, high-performance computing, business intelligence/data warehousing, and low-latency algorithmic trading. Valente has a BS in electrical and computer engineering from the University of Illinois at Urban Champaign. Contact him at geno.valente@xtremedata.com.