A. Summary

- In the area of ***Evaluation and Exploration of Next Generation Systems for Applicability and Performance,*** over the period of 07/01/10 through 06/30/11 the NCSA Innovative Systems Lab team conducted investigation of the applicability of GPU-based acceleration technology for data-oriented applications. We have ported image characterization algorithm implemented in doc2learn application to GPUs using both CUDA C targeting NVIDIA GPUs and OpenCL targeting NVIDIA and AMD GPU architectures. We have also implemented this algorithm as a stand-alone application and used all implementations to perform an energy efficiency study on host CPU and GPU platforms and demonstrated that a GPU-based implementation is not power-efficient. We integrated GPU implementation of Scale-Invariant Feature Transformation (SIFT) algorithm with *doc2learn* and *Versus* software and demonstrated its performance for image comparison within these two frameworks. We analyzed two classes of data-intensive applications: *computation of checksums* used for data integrity verification, encryption, and data comparison, and *lossless data compression* and concluded that while these algorithms can benefit from GPU acceleration to some degree, their practical use on GPUs is limited due to the PCIe bandwidth bottleneck between the host and the GPU. We conducted XtremeData dbx data analytics appliance evaluation using NARA-ICAT database and concluded that the system is capable of executing complex queries an order of magnitude faster than a traditional database engine.

B. Evaluation and Exploration of Next Generation Systems for Applicability and Performance (Volodymyr Kindratenko, Guochun Shi)

XtremeData dbx data analytics appliance evaluation

1 Summary

Over the period of 7/1/10 through 9/30/10, we have ported image characterization algorithm implemented in doc2learn application to the Graphics Processing Unit (GPU) platform using both CUDA C targeting NVIDIA GPUs and OpenCL targeting NVIDIA and AMD GPU architectures. We also implemented doc2learn image analysis algorithm in C targeting microprocessor architecture. Our conclusion is that doc2learn image processing part can be accelerated up to 4 times using NVIDIA GTX 480 GPU, but 1) the speedup depends on the image size and 2) other parts of doc2lear application dominate the execution time.

Over the period of 10/1/10 through 12/30/10, we focused on replacing the Java-based doc2learn framework with a light-weight C-based implementation. Specifically, we used xpdf-3.02 library and modified one of its applications, pdfimage, to compute image histograms identical to the histograms computed by the original doc2learn Java code. We used the software developed over two quarters to perform an energy efficiency study on the host CPU and GPU platforms and demonstrated that a GPU-based implementation is not power-efficient.

Over the period of 01/01/11 through 03/31/11, we have integrated a GPU-based implementation of Scale-Invariant Feature Transformation (SIFT) algorithm for detecting distinctive image features for image matching and recognition with doc2learn pdf file comparison software as a replacement for the probability density function based image comparison algorithm previously used in doc2learn. This allows us to compare images embedded in pdf files based on their actual content rather than on the color probability density function. We also have integrated a GPU-based implementation of SIFT algorithm with Versus framework developed by the Image Spatial Data Analysis Group at NCSA.

Over the period of 04/01/11 through 06/30/11, we performed XtremeData dbx data analytics appliance evaluation. We used a database supplied by NARA consisting of a collection of approximately 79 million records. We also used a database with randomly generated records, ranging from 1 million to 1 billion records. We measured database deployment time as well as the time to run complex queries involving joining tables. We compared our measurements with the results supplied by NARA.

2 XtremeData dbX data analytics appliance

2.1 Hardware

The system that was made available to use by XtremeData is referred to as dbX *Foundation* configuration[1]. It consists of two rack-mounted physical nodes connected via a direct InfiniBand (IB) link (Figure 1). One node is referred to as *head node*; the other node is called *data node*. Both nodes include two six-core AMD Opteron 2431 processors, 32 GB of RAM, and ~12 TB of storage. Number of data nodes can be increased up to 1024.

Figure 1. XtremeData data analytics appliance.

The head node runs “front-end” processes, such as administrator, user sessions, SQL, compiler, optimizer, plan generator, etc. The data node performs “back-end” query execution and handles all external I/O, such as indexed and sequential scan of disks and inter-node communications.

2.2 Software

The system runs Linux OS, version 2.6.18-194.11.4.el5. The database engine is a modified version of PostgreSQL. In addition to the database engine, a command-line based and web-based database administration interfaces are provided (Figure 2).

---

[1] DBX product brief, http://www.xtremedata.com/images/pdf/DBX_2011_Product_Brief_Final.pdf

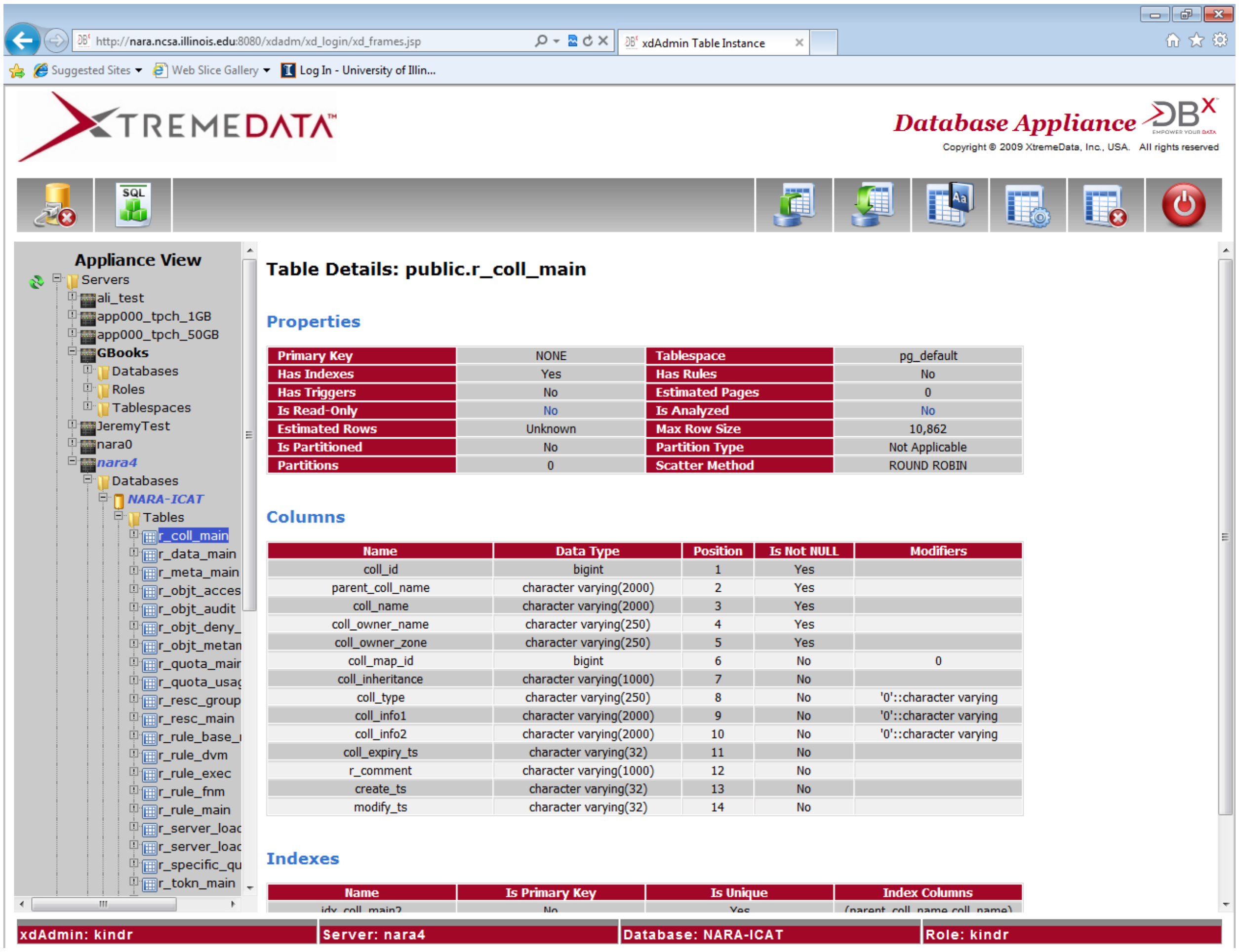

Figure 2. XtremeData data analytics appliance web-based administration tool.

3 Test datasets

In this study, we used two databases: NARA-ICAT database provided by NARA and a custom database consisting of semi-random records stored in two tables modeled after the two main tables in the NARA-ICAT database.

NARA-ICAT database consists of 32 tables, but only two of them, r_coll_main and r_data_main, carry principal data. Tables I and II provide characteristics of these two database tables.

Table Ia. r_coll_main properties

Primary Key	NONE	**Tablespace**	pg_default
Has Indexes	Yes	**Has Rules**	No
Has Triggers	No	**Estimated Pages**	0
Is Read-Only	No	**Is Analyzed**	No
Estimated Rows	Unknown	**Max Row Size**	10,862
Is Partitioned	No	**Partition Type**	Not Applicable
Partitions	0	**Scatter Method**	ROUND ROBIN

Table Ib. r_coll_main columns

Name	Data Type	Position	Is Not NULL	Modifiers
coll_id	bigint	1	Yes	
parent_coll_name	character varying(2000)	2	Yes	
coll_name	character varying(2000)	3	Yes	
coll_owner_name	character varying(250)	4	Yes	
coll_owner_zone	character varying(250)	5	Yes	
coll_map_id	bigint	6	No	0
coll_inheritance	character varying(1000)	7	No	
coll_type	character varying(250)	8	No	'0'::character varying
coll_info1	character varying(2000)	9	No	'0'::character varying
coll_info2	character varying(2000)	10	No	'0'::character varying
coll_expiry_ts	character varying(32)	11	No	
r_comment	character varying(1000)	12	No	
create_ts	character varying(32)	13	No	
modify_ts	character varying(32)	14	No	

Table Ic. r_coll_main indexes

Name	Is Primary Key	Is Unique	Index Columns
idx_coll_main2	No	Yes	(parent_coll_name,coll_name)
idx_coll_main2_xdglobal	No	Yes	(parent_coll_name,coll_name)
idx_coll_main3	No	Yes	(coll_name)
idx_coll_main3_xdglobal	No	Yes	(coll_name)
idx_coll_main1	No	No	(coll_id)

Table IIa. r_data_main properties

Primary Key	NONE	**Tablespace**	pg_default
Has Indexes	Yes	**Has Rules**	No
Has Triggers	No	**Estimated Pages**	0
Is Read-Only	No	**Is Analyzed**	No
Estimated Rows	Unknown	**Max Row Size**	6,918
Is Partitioned	No	**Partition Type**	Not Applicable
Partitions	0	**Scatter Method**	ROUND ROBIN

Table IIb. r_data_main columns

Name	Data Type	Position	Is Not NULL	Modifiers
data_id	bigint	1	Yes	
coll_id	bigint	2	Yes	
data_name	character varying(1000)	3	Yes	
data_repl_num	integer	4	Yes	
data_version	character varying(250)	5	No	'0'::character varying
data_type_name	character varying(250)	6	Yes	
data_size	bigint	7	Yes	
resc_group_name	character varying(250)	8	No	
resc_name	character varying(250)	9	Yes	
data_path	character varying(2000)	10	Yes	

data_owner_name	character varying(250)	11	Yes	
data_owner_zone	character varying(250)	12	Yes	
data_is_dirty	integer	13	No	0
data_status	character varying(250)	14	No	
data_checksum	character varying(1000)	15	No	
data_expiry_ts	character varying(32)	16	No	
data_map_id	bigint	17	No	0
r_comment	character varying(1000)	18	No	
create_ts	character varying(32)	19	No	
modify_ts	character varying(32)	20	No	
data_mode	character varying(32)	21	No	

Table IIc. r_data_main indexes

Name	Is Primary Key	Is Unique	Index Columns
idx_data_main2	No	Yes	(coll_id,data_name,data_repl_num,data_version)
idx_data_main2_xdglobal	No	Yes	(coll_id,data_name,data_repl_num,data_version)
idx_data_main1	No	No	(data_id)
idx_data_main3	No	No	(coll_id)
idx_data_main4	No	No	(data_name)
idx_data_main5	No	No	(data_repl_num)

3.1 Database deployment options

A dbX database can be deployed on a user-specified subset of data nodes. We evaluated two deployment configurations:

- **Configuration A**: Single virtual node executed on the head node
- **Configuration B**: Four virtual nodes; two of which are executed on the head node and two on the data nodes.

3.2 NARA-ICAT deployment time

Timing measurements performed in this study have been collected using one the following procedures, depending on which procedure is more convenient for a given operation:

- Time measurement method 1: Using *time* Linux utility, i.e.,
  - time xdusql sql nara0 NARA-ICAT < query.sql
- Time measurement method 2: Using *sql* timing utility, i.e.,
  - xdusql sql nara0 NARA-ICAT; \timing on

NARA-ICAT database was deployed on dbX system by restoring a database dump provided by NARA. Table III contains timing for different stages of the database deployment on the two deployment configurations.

As seen from Table III, data ingestion time remains unchanged for either of the two configurations. Since there is a single file that is feed into the database, the data ingestion time depends on the speed with which this file can be read from the disk.

Time to index the tables is decreased by a factor of ~3.3 when the database is spread across four virtual nodes instantiated on the two physical nodes instead of just one virtual node. And the analysis time is decreased by a factor of ~2.

Table III. Database setup time.

Deployment configuration A: Single virtual node (one process)				
	Create tables	Load data	Analysis	Indexing
real	0m2.351s	20m43.036s	6m11.476s	417m51.229s
user	0m0.017s	11m0.309s	0m0.009s	0m0.011s
sys	0m0.013s	8m1.247s	0m0.021s	0m0.020s
Deployment configuration B: Four virtual nodes				
	Create tables	Load data	Analysis	Indexing
real	0m2.239s	20m33.278s	3m31.286s	127m14.555s
user	0m0.012s	11m12.962s	0m0.014s	0m0.011s
sys	0m0.019s	8m0.183s	0m0.020s	0m0.020s

4 Performance evaluation

4.1 Using NARA-ICAT database

In this study, we run queries against two databases deployed using deployment configuration A or B. In the tables below, timing for these two configurations is referred to as *TimeA* (Configuration A: Single virtual node executed on the head node) and *TimeB* (Configuration B: Four virtual nodes; two of which are executed on the head node and two on the data nodes.). When available, we also provide timing for the same queries measured and provided by NARA on IRODS database. This timing measurement is referred to as *TimeN*.

Table IV. Count number of collections in the archive.

Query1	SELECT COUNT(coll_name) FROM r_coll_main WHERE coll_name LIKE '/nara-cpk/home/maconrad/National_Archives%';				
Result	count 1467811				
TimeA	real 0m4.237s	**TimeB**	real 0m2.697s	**TimeN**	N/A
Query2	SELECT COUNT(coll_name) FROM r_coll_main WHERE coll_name LIKE '/nara-cpk/home/maconrad%';				

Result	count 1637884				
TimeA	real 0m4.065s	**TimeB**	real 0m2.684s	**TimeN**	N/A

Table V. Count number of files in collections.

Query3	SELECT COUNT(data_name) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives%';				
Result	count 61218583				
TimeA		**TimeB**	real 1m30.698s	**TimeN**	real 2m59.244s
Query4	SELECT COUNT(data_name) FROM r_coll_main WHERE coll_name LIKE '/nara-cpk/home/maconrad%';				
Result	count 79097137				
TimeA		**TimeB**	real 1m36.764s	**TimeN**	real 3m8.671s

Table VI. Count number of files and their combined size in a given collection.

Query5	SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 266 - Records of the Securities and Exchange Commission%';				
Result	count | sum 8603389 | 1491679205975				
TimeA	real 1m25.817s	**TimeB**	real 1m0.610s	**TimeN**	real 2m27.993s
Query6	SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 560 - Records of the Transportation Security Administration%';				
Result	count | sum 273 | 131450733				
TimeA	real 1m0.472s	**TimeB**	real 0m43.799s	**TimeN**	real 0m1.405s

<table>
<tr><td>Query7</td><td colspan="5">SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 034 - Records of the Federal Deposit Insurance Corporation%';</td></tr>
<tr><td>Result</td><td colspan="5"><code>count | sum</code>
<code>619 | 22008579</code></td></tr>
<tr><td>TimeA</td><td>real 1m5.162s</td><td>TimeB</td><td>real 0m44.257s</td><td>TimeN</td><td>real 0m1.596s</td></tr>
<tr><td>Query8</td><td colspan="5">SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 563 - General Records of the Department of Homeland Security%';</td></tr>
<tr><td>Result</td><td colspan="5"><code>count | sum</code>
<code>299 | 8860332</code></td></tr>
<tr><td>TimeA</td><td>real 1m0.660s</td><td>TimeB</td><td>real 0m42.703s</td><td>TimeN</td><td>real 0m1.463s</td></tr>
<tr><td>Query9</td><td colspan="5">SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records%';</td></tr>
<tr><td>Result</td><td colspan="5"><code>count | sum</code>
<code>61211325 | 14696973201297</code></td></tr>
<tr><td>TimeA</td><td>real 2m1.075s</td><td>TimeB</td><td>real 1m5.771s</td><td>TimeN</td><td>real 3m26.528s</td></tr>
<tr><td>QueryA</td><td colspan="5">SELECT COUNT(data_name), SUM(data_size) FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad%';</td></tr>
<tr><td>Result</td><td colspan="5"><code>count | sum</code>
<code>74829764 | 45431130232531</code></td></tr>
<tr><td>TimeA</td><td>real 2m9.477s</td><td>TimeB</td><td>real 1m11.844s</td><td>TimeN</td><td>real 3m50.299s</td></tr>
</table>

Table VII. Find if a particular file exists in the collection.

<table>
<tr><td>QueryB</td><td>SELECT data_name, coll_name, data_path FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND data_name LIKE 'seastar.jpg';</td></tr>
<tr><td>Result</td><td><code>data_name | coll_name | data_path</code></td></tr>
</table>

	(3 rows, Query Total: 3) seastar.jpg | /nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images | /irodsvault/WV/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images/seastar.jpg seastar.jpg | /nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images | /irodsvault/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images/seastar.jpg seastar.jpg | /nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images | /irodsvault/maconrad.nara/National Archives/Federal Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images/seastar.jpg				
TimeA	real 1m2.241s	**TimeB**	real 0m33.328s	**TimeN**	real 0m0.179s
QueryC	SELECT data_name, coll_name, data_path FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND data_name LIKE 'nonexisting.file';				
Result	data_name | coll_name | data_path (0 rows, Query Total: 0)				
TimeA	real 1m4.783s	**TimeB**	real 0m30.388s	**TimeN**	real 0m1.128s

Table VIII. Queries involving r_objt_access table.

QueryD	SELECT COUNT(object_id) FROM r_objt_access WHERE object_id IN (SELECT data_id FROM r_data_main WHERE data_path='/irodsvault/WV/home/maconrad/National_Archives/Federal_Records/RG 079 - Records of the National Park Service/Acadia National Park Photo Galleries/www.nps.gov/acad/kids/images/seastar.jpg');				
Result	count 11				
TimeA	Time: 115.961 s	**TimeB**	Time: 74.402 s	**TimeN**	Time: 81.092 s
QueryE	SELECT COUNT(object_id) FROM r_objt_access WHERE object_id IN (SELECT data_id FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 560 - Records of the Transportation Security Administration%');				
Result	count 1001				
TimeA	Time: 107.533 s	**TimeB**	Time: 82.795 s	**TimeN**	Time: 531.095 s

QueryF	SELECT COUNT(object_id) FROM r_objt_access WHERE object_id IN (SELECT data_id FROM r_data_main, r_coll_main WHERE r_data_main.coll_id=r_coll_main.coll_id AND coll_name LIKE '/nara-cpk/home/maconrad/National_Archives/Federal_Records/RG 266 - Records of the Securities and Exchange Commission%');				
Result	count 47852744				
TimeA	Time: 118.867 s	**TimeB**	Time: 86.592 s	**TimeN**	Time: 2482.763 s

4.2 Scalability study

In this study, we created a database consisting of semi-random records stored in two tables modeled after the two main tables in the NARA-ICAT database. We started by creating one collection and filling it in with 1M records, and run a query similar to those shown in Table VI. Next, we add another collection consisting of 1M records and run a similar query. And so on until we add 100 collections, 1M records in each, or total of 100M records. Each time we run a search query for counting the number of files and their combined size in a random collection. Figure 3 (left plot) presents results of this test.

We conducted another study in which we generated and sequentially added 10 collections of size 100M records each, or 1B records in total. After adding a new collection, we run a search query similar to the previous test. Figure 3 (right plot) presents results of this test.

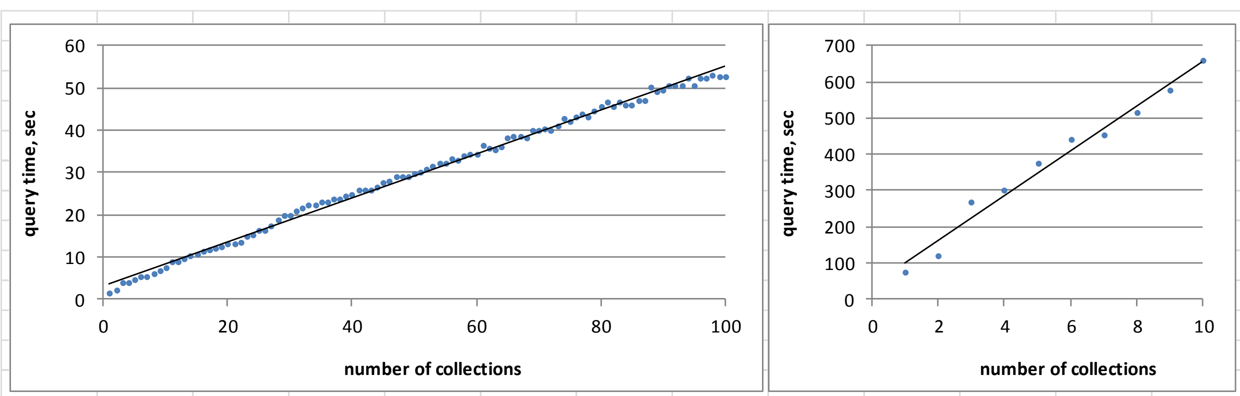

Figure 3. XtremeData data analytics appliance scalability study.

4.3 Analysis

In most cases, XtremeData dbx data analytics appliance executed queries faster than NARA’s database engine. A few exceptions are queries 6, 7, 8, B and C. In the case of these queries, NARA’s database engine returned results within 2 seconds whereas dbx database engine returned results within 10s of seconds. All these queries involved only a small number of records. One possible explanation is that the query results already are already cashed on the NARA database server.

Queries that involved joining 2 tables and returning very large number of records (hundreds of thousands to millions) were generally executed about two times faster by the dbx database than by the NARA database engine.

Queries that involved joining 3 tables executed up to 28 times faster by the dbx database than by the NARA database engine.

Scalability study shows linear increase in query time as the database size grows, which is ideal for this type of analysis.

5 Future Work

For the remainder of the project, we plan to investigate GPU-accelerated pattern matching algorithms based on regular expression matching. Such algorithms are key in searching text files and databases, deep network packets analysis, computer virus scanners, and bioinformatics applications, to name a few areas of potential impact. Existing CPU-based implementations run at MB/sec data rates. Their port to a GPU-based platform, if successful, can run in GB/sec data rates range.