Development of the Geophysical Data Base Variable Resolution (GDBV) Version 2 using HDF5

Chad A. Steed
Mapping, Charting, and Geodesy
Naval Research Laboratory
Stennis Space Center, MS 39529
Email: csteed@acm.org

Chiu-Fu "Tiger" Cheng
Exploration and Science
Lockheed Martin
Stennis Space Center, MS 39529
Email: ccheng@nrlssc.navy.mil

David W. Harvey
Acoustic Modeling and Databases Division
The Naval Oceanographic Office
Stennis Space Center, MS 39529
Email: david.w.harvey@navy.mil

***Abstract*—The Geophysical Data Base Variable resolution Version 2 (GDBV V2) provides a high performance database format for ocean bottom characterization by utilizing a flexible data model implemented using the Hierarchical Data Format Version 5 (HDF5). GDBV V2 has been developed by the Naval Research Laboratory (NRL) and Lockheed Martin in close collaboration with the Naval Oceanographic Office (NAVOCEANO). GDBV has been designed to support the parameters stored in NAVOCEANO's ocean bottom database products: Low Frequency Bottom Loss (LFBL), High Frequency Bottom Loss (HFBL), and Bottom Sediment Type (BST). In this paper, we describe the initial study to select the HDF5 format, as well as details regarding the GDBV data model, utilities, and implementation. Using the current GDBV V2 release candidate, we also present the initial performance statistics, which reveal excellent performance as compared to the current NAVOCEANO database formats.**

I. Introduction

The Geophysical Data Base Variable resolution Version 2 (GDBV V2) is a modern, environmental database that facilitates standardized storage for high performance ocean bottom characterization. Developed in close collaboration with the Naval Oceanographic Office (NAVOCEANO), the GDBV V2 has been developed by the Naval Research Laboratory (NRL) and Lockheed Martin to support NAVOCEANO's low frequency, high frequency, and bottom sediment type products. Specifically, GDBV V2 will support the parameters contained in the following ocean bottom database products that are currently available within the Oceanographic and Atmospheric Master Library (OAML): Low Frequency Bottom Loss (LFBL), High Frequency Bottom Loss (HFBL), and Bottom Sediment Type (BST). In the remainder of this paper, these databases will be collectively referred to as the legacy OAML databases.

The GDBV development is motivated by the lack of standardization in the bottom characterization databases, as well as limitations on extensibility and scalability with the legacy database formats. GDBV V2 provides high performance, standardized data storage to streamline maintenance tasks at NAVOCEANO and usability issue for its customers.

GDBV V2 must have data extraction speeds that are nearly equal to or better than the current legacy database access libraries, which are written in the C programming language. In the prior version of GDBV, GDBV V1, data compression and extensibility were the most critical factors, which led to the decision to implement its implementation in the Java programming language. For GDBV V2, the software has been re-developed using a new underlying data format with access libraries written in the C programming language. Since the extraction speed is the most critical measure for GDBV V2, the initial focus of this research was devoted to the selection of the most suitable file format.

In order to identify the most suitable format, we conducted an exploratory investigation of scientific data file formats to identify the best performing format with the best set of features. We developed and compared four prototype applications using the following file formats: the Hierarchical Data Format Version 5 (HDF5)[1], the Network Common Data Form (NetCDF)[2], a binary flat file format, and a Java object serialization file format. The binary flat file format simulates the legacy OAML database formats and the Java-based format simulates the GDBV V1 database. Based on the results of this study, HDF5 was selected as the underlying format for GDBV V2. The results of this preliminary study are presented in the current work, as well as our rationale for selected the HDF5 format.

Using the HDF5 Application Programmer's Interface (API), a hierarchical data model has been formulated and implemented for the GDBV. To simplify GDBV data access, the high level API has been developed over the HDF5 API to abstract away the mechanical details of accessing the data. In the current work, we provide an overview of the conceptual data model along with some implementation details. The current work also describes a comprehensive set of software utilities that we have developed to streamline GDBV V2 creation, maintenance, and data extraction capabilities.

At the onset of this development, NRL and NAVOCEANO developed a comprehensive set of tests for evaluating the performance of the GDBV relative to the legacy formats [1]. These tests are designed to measure the extraction performance of the GDBV for track line and area-based queries. The execution of these tests has revealed that the current release

[1]HDF5 software and documentation is available at http://www.hdfgroup.org/HDF5

[2]NetCDF software and documentation are available at http://www.unidata.ucar.edu/netcdf

0-933957-38-1/09/$20.00

candidate for the HDF5 GDBV V2 significantly exceeds the legacy formats' extraction speeds. The latest extraction performance statistics are presented in this paper. The enhanced performance and flexibility of the GDBV format demonstrate its promise to provide more efficient geophysical database storage and distribution.

II. Background

GDBV V1 was originally developed to support the Space and Naval Warfare Command (SPAWAR) sponsored efforts to develop and transition geoacoustic inversion algorithms to the Navy Fleet. These algorithms were designed to feed a variable resolution, worldwide geoacoustic database to support propagation loss models. In particular, the effort called for local storage of newly defined ocean bottom parameters in a standard data format for use in acoustic models and tactical decision aids (TDAs).

In close collaboration with NAVOCEANO and the Commander, Naval Meteorology and Oceanography Command (CNMOC), NRL developed GDBV V1 in 2003 and 2004 [2]. By utilizing the Java object serialization capabilities, GDBV V1 offered a single geophysical database format that supported both a dynamic (in-situ) role and a static (historical) role [3]. With GDBV V1, the most critical requirements were to provide enhanced compression and extensibility [4] and the resulting database successfully achieved these goals by requiring only 33% of the disk space necessary for the original databases and providing new capabilities for storing data derived on-scene by inversion algorithms locally for assimilation with historical data. The on-scene storage capabilities were successfully demonstrated in the data management software for the AQS-20 system, where GDBV was used to store bottom sediment data types and bathymetry for subsequent post-mission analysis [5]–[8].

A disadvantage of the GDBV V1 system was that compression, generality, and Java implementation resulted in a system with slower extraction speeds than the original software, which employed for a flat file binary database using C software routines. With GDBV V2, the requirements are focused on the development of a standardized database with extraction routines that are equal to or better than the legacy OAML database software.

The Ocean Bottom Characterization Initiative (OBCI) was initiated to augment NAVOCEANO bottom loss and backscatter classification capabilities in specific geographical regions. The OBCI strategic plan [9] identifies the GDBV as the data repository for geoacoustic data for platforms that engage in ocean bottom characterization. The OBCI is supporting the GDBV V2 development efforts to ensure the database is ready to support the OBCI data storage needs. The GDBV V1 data model was the starting point for this new effort. GDBV has been re-developed to improve data extraction performance and to meet NAVOCEANO's data extraction requirements.

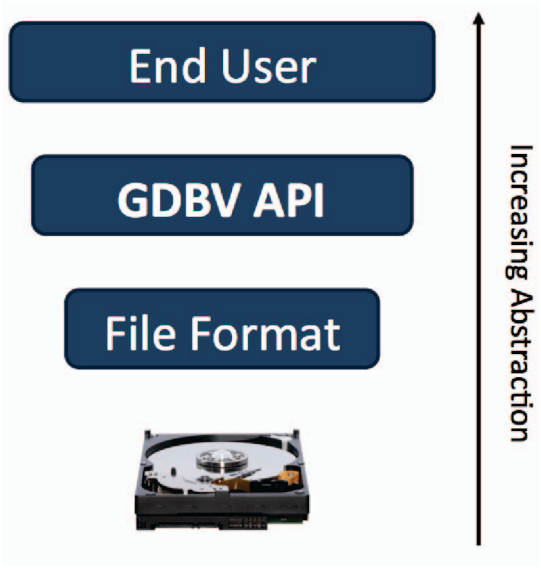

Fig. 1. The levels of abstraction in storing the GDBV V2 database from raw disk storage to end user application access can be viewed as a pyramid with increasing abstraction at the top. The disk is the lowest level of abstraction while the end user application is the highest level of abstraction.

III. File Format Study

In scientific data formats, the data is maintained and accessed at different levels of abstraction through various APIs as illustrated in Figure 1. At the lowest level of abstraction is the file format API. This API handles writing and organizing the data using the input/output capabilities of the operating system. The Hierarchical Data Format Version 5 (HDF5) and the Network Common Form (NetCDF) are examples of scientific data format APIs. Above the file format layer is the product-specific API. This API provides a view of the file that is tailored to a specific use for a particular product, such as the GDBV. That is, the features of the file format API are utilized behind the scenes so that the user does not have to deal with the details and they enjoy a customized view of the data.

The file format layer handles low-level tasks, which frees the developer to focus on higher level issues, such as geophysical data storage. The end user, shown at the highest level in Figure 1, gains access to the GDBV via its API, which offers streamlined capabilities to access the data in specialized ways.

Because of the importance of selecting the GDBV V2 file format layer, we developed prototypes to compare three different formats. The first format we considered was a modified flat file format that uses the OAML LFBL as the starting point. The LFBL APIs are currently implemented in the C programming language and the database is stored on disk in a custom file format that has been developed and maintained over the years by NAVOCEANO scientists.

The next two approaches seek to abstract away the details of raw data storage by utilizing standardized, scientific data formats maintained by third parties. One approach we investigated was to use the NetCDF format that is actively developed by the Unidata University Corporation for Atmospheric

TABLE I
FILE FORMAT PERFORMANCE TEST RESULTS FOR WRITING DATA.

(All timing values are times in seconds.)

Trial	Flat File	NetCDF	HDF5	Java
1	27.934	69.406	26.511	35.047
2	27.946	52.192	26.605	37.841
3	29.732	55.369	24.757	42.005
4	26.066	53.748	25.474	37.910
5	26.818	55.005	25.969	37.288
6	29.646	53.972	25.520	37.741
Average	**28.024**	**56.615**	**25.806**	**37.972**

TABLE II
FILE FORMAT PERFORMANCE TEST RESULTS FOR READING DATA.

(All timing values are times in seconds.)

Trial	Flat File	NetCDF	HDF5	Java
1	3.571	4.944	3.542	19.947
2	3.538	4.577	4.145	19.952
3	3.563	4.608	3.555	19.861
4	3.545	4.603	4.940	20.147
5	5.687	4.574	3.536	19.959
6	3.566	4.554	3.556	19.877
Average	**3.912**	**4.643**	**3.879**	**19.957**

TABLE III
HDF5 COMPRESSION PERFORMANCE TEST RESULTS.

(All timing values are times in seconds.)

	Uncompressed		Compressed	
Trial	Read	Write	Read	Write
1	0.336	1.309	0.863	4.507
2	0.332	2.362	0.856	5.437
3	0.334	1.693	0.858	5.426
4	0.337	1.482	0.858	5.419
5	0.338	1.541	0.854	5.628
6	0.337	1.169	0.860	4.274
Average	**0.336**	**1.593**	**0.858**	**5.115**

Research (UCAR). The NetCDF is the product of over 20 years of development and UCAR offers APIs in many different programming languages, including C. At the time of this study, NetCDF version 3.6.2 was the most current version available. Another alternative that we explored was to use the HDF5 format that was developed by the HDF Group. This format has also been actively developed for more than 20 years. At the time of our study, HDF5 version 1.6.5 was the most current version available. Two NAVOCEANO environmental databases, the OAML Digital Bathymetry Data Base Variable resolution (DBDBV) and the OAML Bottom Sediment Type (BST), are also distributed using the HDF5 format.

We measured the performance of our prototypes in write and read tests for a 3-dimensional array of 4-byte floating point values were. In addition to the 3 file formats mentioned above, a Java file format was included to simulate the GDBV V1 performance. The array dimensions used in these tests were 200 x 1500 x 1000 for a total of 300,000,000 elements. In each test, the array was written or read as a single block of data. In both the read and write tests, 6 trials were executed, for a total of 12 trials per file format, and timed using the Linux *time* utility. The tests were executed on a Linux-based (Fedora Core 6) computer with a 3.20 GHz Intel Pentium 4 processor (with hyper-threading enabled) and 4 GB of Random Access Memory (RAM).

Table I lists the test results for the 6 write trials along with average times. From these results, the HDF5 file format appears to be the fastest at writing data to the system disk. The flat file format was about 2 seconds slower than the HDF5 format and the NetCDF file format exhibits the poorest performance of all the formats.

Table II lists the test results for the 6 read trials along with average times. These results show that the HDF5 format was slightly faster than the flat file format (with 0.033 seconds difference). Although not as close as the flat file format, the NetCDF format was only 0.764 seconds slower than the HDF5 format. From these results, it appears that the HDF5, flat file, and NetCDF formats are nearly equal in read performance. However, the Java format exhibited significant differences from the other formats as it required, on average, about 16 more seconds than the HDF5 format.

The HDF5 file format includes built-in compression capabilities, which will help reduce the disk storage and transmission costs of GDBV V2. In order to quantify the performance cost associated with the HDF5 compression, an additional set of tests were executed. In these tests an array of floating point values with dimensions of 300, 300, and 300 for the first, second, and third dimensions, respectively, was written and read from the disk with and without the HDF5 "deflate" compression property enabled. In these tests, the compression algorithm effort parameter was set to a value of 9 – the most aggressive level of compression. Six trial runs were executed and timed for reading and writing the data with compression enabled and disabled for a total of 24 trial runs.

Four average times were calculated and these statistics along with the individual trial timings are listed in Table III. Based on the averages, the ratio of compressed to uncompressed write time is 3.2 and the ratio of compressed to uncompressed read time is 2.6. It is also important to note that the "deflate" compression capability can be expected to reduce the file to about one-third of its uncompressed size.

Based on these results, and an evaluation of the features available for each of the file formats, we selected the HDF5 format for the GDBV V2 low level format. One reason we selected the HDF5 format is because it exhibited the best performance in our prototypes test cases. In addition, the HDF5 format provided several attractive built-in capabilities such as compression, filtering mechanisms, portability between different operating systems, and a wide variety of open source and third party tools. Another benefit to adopting the HDF5 format is that the HDF Group also maintains a separate API to support parallel file access in a message passing environment, which can be used to enhance database access

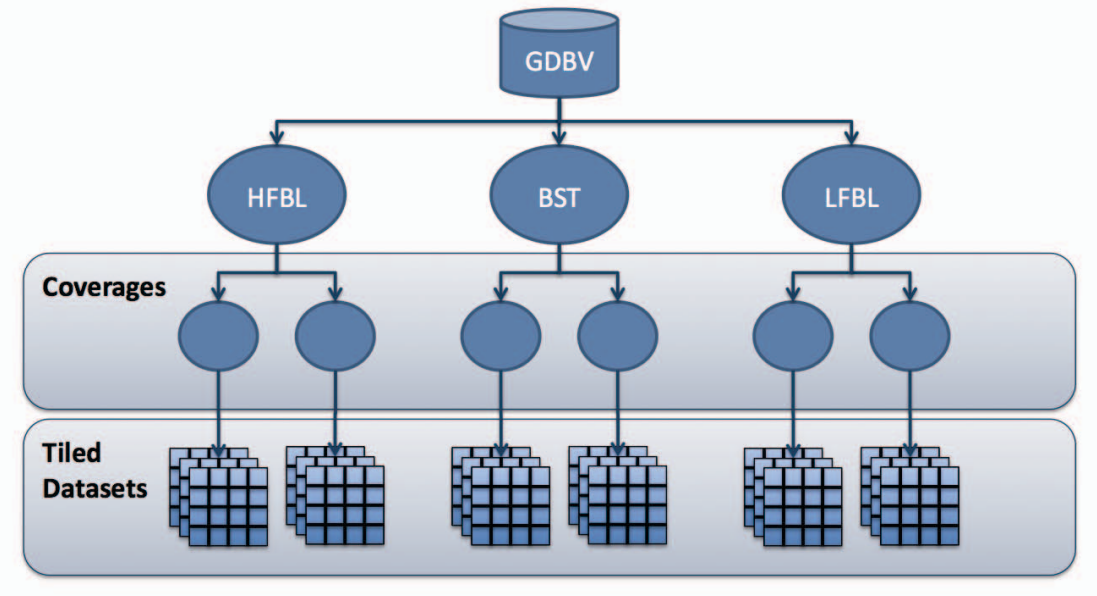

Fig. 2. GDBV V2 conceptual data model is a hierarchical organization that represents the data at different levels of abstraction.

on multiprocessor systems. HDF5 was also chosen because NAVOCEANO has built up expertise in using the format as evidenced by the use of HDF5 in the DBDBV and BST databases. In 2008, NRL and NAVOCEANO re-evaluated the results of this 2007 study and re-affirmed the decision to use HDF5.

IV. HDF5 Data Model

After selecting the HDF5 file format, we revisited the conceptual data model that was originally developed for GDBV V1 [2]. The GDBV V1 data model has been modified to remove the dynamic storage layer that supports point-based data. Like the original model, the new model, which is shown graphically in Figure 2, is hierarchical with the top level representing the HDF5 file root. The next level encapsulates the various data sets that are stored in the GDBV (i.e. HFBL, LFBL, and BST). Each data set group contains one or more coverage groups, which separate the data sets into spatial coverages. For HFBL and LFBL, the data set coverages are defined by areal extents in the form of a geographical minimum bounding rectangle. Each coverage in LFBL and HFBL can also have its own grid resolution. With the BST coverages, the coverages are defined by the resolution of the data grid and each coverage has worldwide geographical extents.

Beneath the coverage groups, the actual data are stored in tiled data sets using the HDF5 data set entities. To implement the tiling, the data sets utilize the HDF5 "chunking" mechanism that partitions a large grid into sub-regions, or tiles. Space for these tiles are only allocated when data is written within the tile's region of the overall grid, thereby reducing the memory and disk space to store the data. The tiling also helps improve the reading of the data by avoiding the need to read the entire data set, and permitting HDF5 caching, indexing, and filtering features. At each level of the hierarchy, the data model also supports the storage of descriptive metadata as HDF5 attributes. This metadata information describes areal coverages, grid spacing, and general distribution attributes.

V. GDBV Toolkit

In addition to the GDBV API, a set of command line utilities (see Figure 3) have been developed to facilitate practical access to the database. The source code for these utilities will also provide end users with specific examples that demonstrate how to interface with the GDBV V2 API functions. These utilities provide the core functionality that are necessary to create, maintain, and read from the database.

GDBVCreate is used to create a database group prior to loading the data and it accepts as arguments the database path, the name of the data set group, and the data compression level. *GDBVLoad* is a utility that populates a data set with data from an external file. *GDBVDump* is a diagnostic utility that dumps the contents of a database to the disk. *GDBVExtractArea* is used to extract data from a particular coverage in GDBV by the geographical extents of a minimum bounding rectangle (in latitude and longitude coordinates). *GDBVExtractPoint* is used to extract data from a coverage as point(s). *GDBVSubset* is used to create a new copy of the GDBV database that contains only the data within a particular minimum bounding rectangle. *GDBVSetSecurity* is a utility to modify the database security and distribution metadata fields. *GDBVStatus* is a program that prints information about which data types (LFBL, HFBL, and BST) are loaded.

VI. Performance Results

After completing the initial release candidate GDBV V2, we executed a series of tests to determine how the database API and utilities perform relative to the legacy databases.

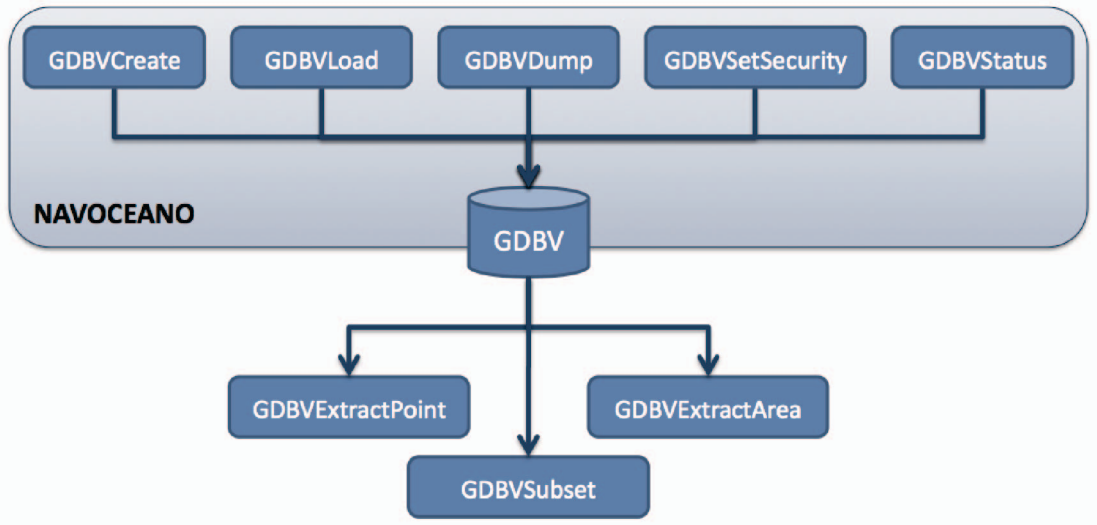

Fig. 3. The GDBV V2 toolkit facilitates practical access to the data through several command line utilities. In addition, the source code for these utilities provide examples for how to interact with the GDBV API.

These tests consisted of area- and point-based extractions from HFBL, LFBL, and BST. For each database, extractions were executed for the legacy OAML formats and the new HDF5 GDBV format. The descriptions for the area and point extraction test cases are as follow:

- Area Extraction Test Cases
  - BST
    - Area 1: 301 x 301 (from 6-second coverage)
    - Area 2: 301 x 501 (from 6-second coverage)
    - Area 3: 7,801 x 1,801 (from 6-second coverage)
    - Area 4: 37 x 37 (from 5-minute coverage)
    - Area 5: 85 x 85 (from 5-minute coverage)
    - Area 6: 241 x 361 (from 5-minute coverage)
  - HFBL
    - Area 1: 1201 x 1201
    - Area 2: 25 x 25
    - Area 3: 97 x 61
  - LFBL
    - Area 1: 1200 x 1200
    - Area 2: 898 x 1199
    - Area 3: 600 x 650
    - Area 4: 1200 x 1250
    - Area 5: 708 x 602
- Point Extraction Test Cases
  - BST
    - 60 tracklines (48,370 points)
  - HFBL
    - 50 tracklines (19,241 points)
  - LFBL
    - 60 tracklines (48,370 points)

Using these test cases, we compared the performance of the HDF5 GDBV to the legacy OAML database formats. For each database, the total time to execute the test cases was recorded.

TABLE IV
AREA AND POINT EXTRACTION PERFORMANCE TEST TIMES.

(All timing values are times in seconds.)

AREA Extractions

	GDBV V2	OAML	Difference	% OAML
BST	0.20	1.01	0.81	20%
HFBL	0.01	0.16	0.15	6%
LFBL	7.58	82.41	74.83	9%

POINT Extractions

	GDBV V2	OAML	Difference	% OAML
BST	0.93	1.02	0.09	91%
HFBL	0.005	0.20	0.195	3%
LFBL	5.91	9.80	3.89	60%

A comparison of the resulting timing are listed in Table IV for both the area and point extractions.

The extraction performance results reveal the superior performance of the GDBV V2 HDF5 versus the legacy OAML formats. In every case, the time required to extract data from the GDBV V2 format was less than the legacy OAML format counterpart. For HFBL, the point extraction only required 3% of the OAML HFBL point extraction time. Also, the area extractions for HFBL and LFBL in GDBV V2 only required 6% and 9% of the OAML HFBL and LFBL extraction times, respectively. In addition to faster data extraction times, the adoption of the underlying HDF5 format will also yield greater extensibility for future expansion of the database.

VII. Conclusions and Future Work

The GDBV V2 provides an extensible database format for NAVOCEANO ocean bottom characterization products by effectively harnessing the HDF5 format. The performance tests presented in the current work demonstrate the superior performance of the GDBV V2 relative to the legacy database formats

that it is expected to eventually replace at NAVOCEANO: OAML HFBL, OAML LFBL, and OAML BST. Additionally, GDBV supports future expansion to include more data types and its software can be used on either the Linux or Windows operating systems. In the future, we will add additional support for other ocean bottom datatypes. We will also explore new ways to represent the information in these databases using multidimensional visualization techniques to enable more effective data analysis. By combining exceptional performance with standardized data access, GDBV V2 promises to greatly enhance ocean bottom data storage, processing, and dispensation.

ACKNOWLEDGMENTS

This research was sponsored by Program Element No. 0603207N by the Oceanographer of the Navy via PEO-C4I PMW-120, Mr. Marcus Speckhahn, Program Manager. We would like to thank Mr. Bruce Northridge (CNMOC), Mr. Ricky Williamson (NAVOCEANO NP53), and Ms. Sheri Carbonette (Lockheed Martin) who provided support and valuable input into the GDBV V2 developments.

REFERENCES

[1] C. A. Steed and D. W. Harvey, "Geophysical Data Base Variable Resolution Version 2: Planning report," Naval Research Laboratory, Stennis Space Center, MS, Technical Report NRL/FR/7440–09-10,173, Jan. 2009.

[2] C. A. Steed, "Geophysical Database Variable Resolution (GDBV): Database definition document," Naval Research Laboratory, Stennis Space Center, MS, Technical Report NRL/FR/7440–03-10,063, Dec. 2003.

[3] ——, "GAIT GDBV quick start guide," Naval Research Laboratory, Stennis Space Center, MS, Technical Report NRL/FR/7440–04-8772, May 2004.

[4] C. A. Steed, D. W. Harvey, K. A. Koehler, and B. Northridge, "Geophysical Database Variable Resolution (GDBV): An object-oriented database for dynamic geo-acoustic data storage," in *Proceedings of MTS/IEEE Oceans 2003*, San Diego, CA, Sep. 2003, pp. 132–140.

[5] Michael M. Harris, William E. Avera, Chad A. Steed, Leonard D. Bibee, Warren T. Wood, William D. Morgerson, and Christopher S. Robinson, "Through-The-Sensor determination of AN/AQS-20 sensor performance demonstration 1, December 13 through 17, 2004," Naval Research Laboratory, Stennis Space Center, MS, Technical Report NRL/FR/7440–05-10,106, Jun. 2005.

[6] Michael M. Harris, William E. Avera, Chad A. Steed, and John T. Sample, "AN/AQS-20 environmental data collection demonstration 2, March 21-25, 2005," Naval Research Laboratory, Stennis Space Center, MS, Formal Report NRL/FR/7440–05-10,113, Dec. 2005.

[7] Michael M. Harris, William E. Avera, John T. Sample, Chad A. Steed, Leonard D. Bibee, and Dave Morgerson, "AN/AQS-20 environmental data collection: End-to-end demonstration 3, tactical sensor to tactical decision aid, June 1 through 3, 2005," Naval Research Laboratory, Stennis Space Center, MS, Formal Report NRL/FR/7440–06-10,134, Jul. 2006.

[8] M. Harris, W. Avera, C. Steed, J. Sample, L. Dale Bibee, D. Morgerson, J. Hammack, and M. Null, "AQS-20 Through-The-Sensor (TTS) performance assessment," in *Proceedings of MTS/IEEE Oceans 2005*, vol. 1, Washington, D.C., Sep. 2005, pp. 460–465.

[9] M. D. Huff, "Ocean Bottom Characterization Initiative (OBCI) execution strategy," PEO C4I PMW–180, San Diego, CA, Letter Ser. PMW 180/7-19, Feb. 2007.