W-2 DDS, A Seismic Processing Architecture

RANDALL L. SELZLER, JERRY EHLERS, *JOSEPH A. DELLINGER
RSelzler@Data-Warp.com Jerry.Ehlers@BP.com dellinja@BP.com

Abstract: The Data Dictionary System (DDS[1]) is a seismic processing architecture. It has proved its value in a *production* environment at a leading High Performance Computing center over the last ten years. It also provides the flexibility needed for advanced *research* in seismic imaging. The software has been ported to more than a dozen hardware platforms and the data sets can be efficiently accessed across any combination of them. One unique feature of DDS is its ability to inter-operate with other processing systems. This is accomplished using a combination of generic data formats, interface emulation and on-the-fly conversion. Inter-operation can protect your investment in existing software, leverage the unique capabilities of other systems, facilitate collaboration with partners and provide an evolutionary path to the future that complements the open source philosophy.

DDS was released by BP America in 2003 under an open source license. The goal of this paper is to reveal useful innovations so that subsequent projects can benefit from DDS.

Introduction: The Stanford Exploration Project[2] was one of the first to provide seismic processing software under an open source license. Equally important is Seismic Un*x[3]. They were conceived in the 70's and 80's within an academic environment and have been used to some extent commercially. FreeUSP[4] is a contemporary package with a commercial origin. It was made available to the public in 2001 (before DDS) by BP America.

Development of DDS began in 1995 at the Amoco Research Center, in Tulsa Oklahoma. Many of its concepts were borrowed from other systems, including SEPlib, USP, SU and Cogniseis DISCO. The infrastructure emphasizes a generic seismic format, interfaces to other processing systems and high-performance disk I/O.

Seismic processing architectures can be divided into two broad categories. Architectures such as DISCO and ProMax use an executive program to control the flow of data and call coroutines to operate on it. Architectures such as SEPlib, SU and FreeUSP rely upon the OS to execute independent processes and transfer data among them. DDS falls into the later category.

DDS has one underlying philosophy: Data handling is fundamental. Get it right and other issues will fall into place. Get it wrong and the system is doomed to mediocrity.

DDS software supports a variety of platforms. It is currently used by BP on workstations (Sun-Solaris, Intel-Linux), servers (SGI-IRIX) and high-performance clusters (Intel-Linux, HP Itanium-Linux, SGI Altix-Linux). BP's HPC center has thousands of compute nodes, many terabytes of memory and petabytes of raid disk storage (managed by 5 systems administrators!). DDS has also been used on Cray, Thinking Machines and Convex super-computers. It provides an Application Program Interface (API) for C and Fortran. Data sets are portable. Files and I/O streams created on one platform are automatically converted as needed when accessed by another platform.

A persistent challenge faced by seismic processing software is efficiency, flexibility, portability and inter-operation with other processing systems. The architecture used for data handling has a major impact on all of these issues. The importance of the first three is obvious and most processing systems solve them to varying degrees. I believe the last one, inter-operation, is very important, but is often neglected. DDS provides a solution to all four issues.

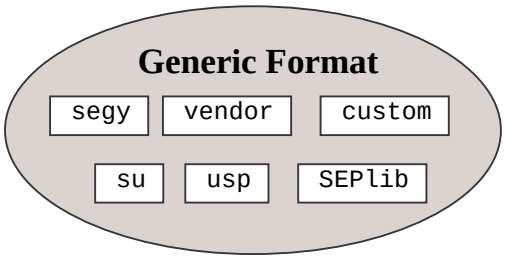

Figure 2: Generic format.

In a perfect world, *everyone* would just use *my* favorite processing system. In the real world, inter-operation with other processing systems has significant value. It can protect your investment in existing software. It allows you to leverage the unique value provided by other systems. It facilitates collaboration with partners in industry and academia. It also provides an evolutionary path to the future that complements open source software.

Data Dictionary: DDS normally keeps binary data and dictionary information in separate files (Figure 1). The importance of this concept is reflected in the name, *Data Dictionary* System. This concept was patterned after SEPlib circa 1994. It kept trace samples in a binary file (typically a cube of numbers), and meta-data in a history file (in plain text). This division of responsibility is simple and flexible.

Multi-dimension: DDS is designed to handle multi-dimension data sets. Each dimension can be described by an extensible set of attributes. This allows many problems to be accurately described in their most natural terms. The dictionary in Figure 1 shows an excerpt for a simple data set with three dimensions. The number of dimensions, their names and nominal sizes are specified in a simple and intuitive syntax. The format of the binary data and its filename are also obvious.

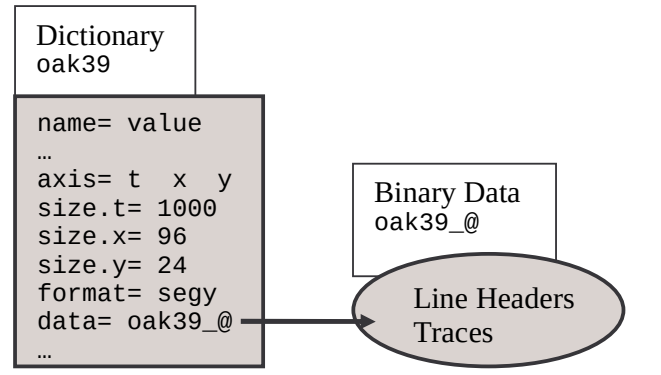

Figure 1: Dictionary and data file for 3D data set.

The axis names and nominal sizes are required by the DDS API. Meaningful names can be chosen for time (`t`), space (`x, y, z`), transforms (`w, kx, ky, kz`) and field geometries (`shot, station, channel, cdp, offset`). In addition to size, most dimensions have attributes for physical coordinates (`origin.x, delta.x, units.x`) or index schemes (`base.x, step.x`). Variable grids (`grid.z= 1 2 4 8 16` ...) and non-numeric coordinates (`label.earth= Vp Vs density`) can also be defined. A suite of applications may require specific axis names or attributes in order to cooperate, but the DDS architecture does not. The architecture can accommodate unforeseen processing domains and dimension counts as easily and naturally as conventional ones.

Lesson 1: Avoid unnecessary constraints and assumptions.

Generic Format: Traces (binary data) are described by a *generic format*. The name, type and offset of *headers* can be specified, much like *fields* within a C structure. These fields within an *I/O record* can be very simple or arbitrarily complex. In fact, *conventional* formats such as SEGY, SU, SEP, and USP are a subset of the generic format that is supported by DDS (Figure 2). This provides an important foundation for portable software, portable data, operational efficiency and leveraging of non-DDS software.

Generic formats are very flexible. Fields within an I/O record may be a scalar or an array. The `Samples` field is a vector and is handled by DDS the same way as any other header. I/O

records may contain samples without any headers, or just the opposite. Specialized formats can also be created. For example, multi-component traces might have three sample vectors that all share one set of headers.

DDS provides recipes for *creating* formats, such as segy, usp and fcube. The recipes can be adjusted at run time for special cases, without compiling or linking. For example, end users could define "`fmt:*:segy.SEGY_TRACE_SPARE= typedef struct { float4ibm muf; float4 foo; float4x bar; } SEGY_TRACE_SPARE;`". This adjustment adds three custom fields (`muf, foo, bar`) to the segy trace format (union for the spare area). Their types are three flavors (IBM, IEEE big endian, IEEE little endian) of 4 byte floats. This feature allows DDS to handle bizarre variations of SEGY data. Arbitrary formats can be created by end users with other recipe mechanisms.

Lesson 2: Generic formats provide important flexibility.

Header Maps: Header values can be changed when I/O records are read and written. For example, the external format on disk can be converted to an internal format when read. Changes may be as simple as converting the endian of one specific field or as complicated as converting from SEGY to USP format.

DDS provides rules for *mapping* the content of headers. The rules can be adjusted at run time, without compiling or linking. For example, end users could define "`map:segy:usp.SrPtXC= 3.2808 * SrcX`". This rule would convert a source point x coordinate from meters to feet, when changing from SEGY to USP format. In general, the map expression (right hand side) can be a C-like scalar expression. Unfortunately, DDS does not support vector arithmetic. If it did, maps could be applied to sample vectors too, because all fields are *created equal*. Implementation of this powerful concept is planned for PSEIS[5].

Lesson 3: Maps provide important processing flexibility.

Processing: Most DDS applications can directly read data in a foreign format, convert it on-the-fly to an internal format and apply an algorithm to the headers and samples. If the subsequent processing step is not DDS-enabled, the internal format can be converted again as needed before it is written. DDS can also use disk or tape media for storage, or data can be sent through I/O streams (pipes and sockets).

Generic formats, field maps and media flexibility can boost processing efficiency. This is particularly important when individual data sets are measured in tera-bytes. Flexibility can be used to eliminate processing steps, save precious disk bandwidth and reduce peak storage (Figure 3). DDS can also randomly access disk files that have a foreign format. Other processing systems can not, if they rely upon format converters within a pipeline.

Lesson 4: Synergy is increased when individual features are integrated.

Proprietary formats: Data in a vendor format could be directly read and written by DDS applications. Amoco had a developer's license for the CogniSeis DISCO package. DDS was enhanced to use the DISCO I/O routines, but present the data in the generic format. No changes were needed to DDS applications for them to access Yet Another Format (YAF).

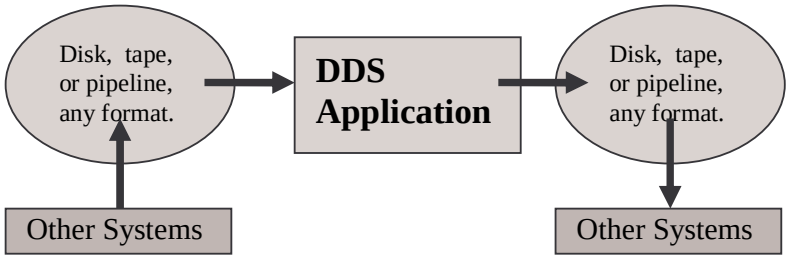

Figure 3: DDS plays nicely with others to increase value.

Care must be taken to avoid litigation when mixing open source and

proprietary software. Mixed licenses raise many questions and are often incompatible. This is one reason that DISCO support was removed from DDS before it was released publicly.

Lesson 5: Know the software pedigree and honor its license.

Emulation: DDS could emulate the API used by other processing systems. For example, in 1998 a wrapper library was created that allowed USP applications to be DDS-enabled, by simply re-linking them. Data was automatically converted between the USP format (used by the application) and other foreign formats. This emulation was surprisingly efficient too. Emulated I/O speed ranged between 90% and 101% of that obtained by the native USP I/O. Yes, for special cases, emulation was *faster* than native, when accessing its own USP format!

Lesson 6: Other systems can leverage functionality provided by DDS.

Information can be lost when DDS converts data between formats. This includes meta-data and trace header information. Foreign formats may lose numerical precision or they may not have a well defined place for certain information. The generic format can typically capture all the information provided by a foreign format, but applications may have difficulty using it. For example, how should a DDS application handle a SEGY "trace identification code" between 9 and 32,767 (non-standard "optional use values")?

Lesson 7: Format conversion and emulation is not a substitute for modernization.

Future: DDS has been successful, but it can be significantly improved. The Parallel Seismic Earth Imaging System (PSEIS) is a new package that will be "OSI Certified open source software[6]". Improvements currently envisioned are described in a companion paper and two poster sessions. "PSEIS, A Processing Architecture Blueprint", describes a collection of enhancements that are designed to increase functionality, convenience and robustness. "PSEIS, Blueprint for Parallel Processing", describes a parallel processing scheme for distributed-memory machines. "PSEIS, Meta-Data Dictionaries" describes a syntax for plain text that has functionality similar to a file system.

Conclusion: DDS provides an efficient, flexible and portable seismic processing architecture that can inter-operate with other processing systems. It has proved its value for research and production work over the last ten years. It is now available as open source software.

Acknowledgment: I'm grateful to the DDS user community within BP and many colleagues from the Amoco Research days for their support. Without the guidance and encouragement of John Etgen, Joe Dellinger, Dan Whitmore and Kurt Marfurt, this project would not have been possible. Special thanks are due Jerry Ehlers for doing a marvelous job of maintaining and enhancing DDS since 1998. Thanks are also due BP America for releasing the DDS software and for permission to publish this paper.

References:

[1] DDS http://FreeUSP.org/DDS/index.html
[2] SEPlib http://sepwww.stanford.edu
[3] CWP/SU http://www.cwp.mines.edu/cwpcodes
[4] FreeUSP http://FreeUSP.org
[5] PSEIS http://PSEIS.org
[6] OSI http://www.opensource.org