Exploration & Production Technology
delivering breakthrough solutions

FreeUSP & FreeDDS

Richard Clarke
Outline

• Introduction
  − why open source?
• FreeUSP
• FreeDDS
  − using
  − developing new applications
• Looking to the future
Why Open Source?

• FreeUSP was released as open source in 2001.
• FreeDDS followed in 2003.

• “to foster education, understanding and collaboration”
  − collaboration with academia
  − technology deployment via contractors

→ fast application of technology
Industry scale research...

• USP & DDS enable ideas to be tested, verified, then scaled
• 99% of the usage of one of the world’s largest research computing systems
  - used for applied research – NOT production processing
  - 450 teraflops, almost 5000 times bigger than 1999
  - 3500 computers with over 40,000 CPUs
  - 230 terabytes of physical memory
  - 15 petabytes of raw disk storage
  - optimized systems:
    - computers with big memory allow fast development
    - clusters to allow cost-effective scaling
  - team of geophysicists, mathematicians, computer scientists
FreeUSP

- “Unix Seismic Processing”
- Started in ~1980
- Used for processing Amoco/bp data for ~30 years
- Unix based
- Command-line driven (there is a GUI, but no-one uses it)
- Unix like syntax
  
  utop –N input.usp –O output.usp –dt 4

- ~600 codes, mostly in signal analysis and preprocessing
- 3D cubes, designed for fast access
- Similar format to SEGY
  - Line header, with processing history
  - 256 byte trace header, followed by samples
FreeDDS *(next release coming in a few days)*

- “Data Dictionary System”
- Started in ~1990
- Used for processing & imaging Amoco/bp data for ~20 years
- Unix based
- Command-line driven
- SepLib/SU/Madagascar like syntax

```bash
editd in= input 3s=100 3e=110 out=records_100_110
```
- ~100 codes in FreeDDS, ~400 not yet released, annual updates
- Covers signal analysis, preprocessing, imaging...
- Up to 9 dimensional cubes, designed for fast access
- There is no “DDS format”!
“There is no DDS format!”

• DDS is heavily influenced by SEPlib, so resembles Madagascar
• Normal use is with a dictionary file + binary file
  – Dictionary is text; line header type info describing the binary
  – Binary file can be fcube (samples only), USP, SEGY, SU, ASP,…
• Autodetects format
  – No dictionary needed for USP, SEGY.
• Every program can change format on the fly
  – `compute in= input.usp scale=10 out_format=segy out_data=output.segy`
Mixing Software Packages: DDS+USP+SU+…

- We can pipe between different software packages
  - DDS can act as the glue between processing packages
- Often preprocessing uses USP format because we need headers, then we use fcube (we separate the headers) when we are imaging.

```plaintext
editd \n  in= input.segy 3s=100 3e=200 \n  out_format=usp out_data=stdout: |\nutop \n  –dt 4 |\nbridge \n  out_format=su out_data=stdout: |\nsuximage \n  legend=1 …
```

**DDS: select records 100-200**

**USP: set the sample rate to 4ms**

**DDS: convert to SU**

**SU: display the data**
Multi-file Support

- The data can be in several different files
  - “sliced” across 1 axis (e.g. separate offset volumes),
  - or across several axes (e.g. organized as many sub-cubes)
- The dictionary allows to access it as 1 big file, or many small files.

Change the axis order* to change
the effective order of the data:
axis=t comp h cdp
axis=t h comp cdp
axis=t h cdp comp

*on the command-line or by editing the
dictionary. For example:
editd in=input.segy axis=t x y h size.y=10 size.h=10 4e=10 | compute ...
Viewing Data: command line

- **ddsploit**
  - Emulates most matlab plots, plus a few extra
  - Can create plots on top of plots
  - Reads DDS/ascii; writes post-script

- Joe Dellinger uses vplot

- **SU Tools**
  - Wrapper scripts that convert the data to SU format:
    
    "bridge in=$in out_format=su out_data=stdout: | suximage legend=1..."
Viewing Data: GUI

- XSD
  - 2D Graphical viewer
  - Data server/client – designed for use with a remote HPC and low bandwidth
  - Interleave, animations, picking,…

Note: watch for missing X libraries during installation that can cause XSD to not compile!
Displaying data: XSD

- Once seismic data is selected to be opened, the following window appears:
Current DDS applications... +30 in next release

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoHTMLIndex</td>
<td>Automatically build HTML documents</td>
</tr>
<tr>
<td>ChklMHeaders</td>
<td>Check the headers for syntax errors</td>
</tr>
<tr>
<td>ascidump</td>
<td>Utility to dump binary data</td>
</tr>
<tr>
<td>boxdesign</td>
<td>An interactive aid to design</td>
</tr>
<tr>
<td>bridge</td>
<td>Convert seismic data between different formats</td>
</tr>
<tr>
<td>checkcordery</td>
<td>Check that the hydrophone checks correctly</td>
</tr>
<tr>
<td>chgaxis</td>
<td>Rename axis names</td>
</tr>
<tr>
<td>chldata</td>
<td>Check/Correct data for channel data</td>
</tr>
<tr>
<td>compute</td>
<td>General trace computations</td>
</tr>
<tr>
<td>create_hdr</td>
<td>Create trace headers to 3D-convert</td>
</tr>
<tr>
<td>ddsServer</td>
<td>Remote data server for access</td>
</tr>
<tr>
<td>dds_defn</td>
<td>Utility to return a definition</td>
</tr>
<tr>
<td>dds_if</td>
<td>Conditionally change if</td>
</tr>
<tr>
<td>dds_in</td>
<td>Utility to return axes</td>
</tr>
<tr>
<td>ddsrepair</td>
<td>A utility program for bulk data repair</td>
</tr>
<tr>
<td>ddsresample</td>
<td>Resample first axes of an N-dimensional dataset</td>
</tr>
<tr>
<td>dds_reverse</td>
<td>Reverse order any selected axes</td>
</tr>
<tr>
<td>dds_rotate</td>
<td>2D coordinate rotation</td>
</tr>
<tr>
<td>dds_extract</td>
<td>Extract either a well or a horizon</td>
</tr>
<tr>
<td>interp_headers</td>
<td>Interpolate arbitrary points</td>
</tr>
<tr>
<td>mbs_list</td>
<td>Utility to list all channels</td>
</tr>
<tr>
<td>mbs_show</td>
<td>Utility to list and display</td>
</tr>
<tr>
<td>mbs_report</td>
<td>Emulates the Matlab plot</td>
</tr>
<tr>
<td>mbs-which</td>
<td>Utility to list the channels</td>
</tr>
<tr>
<td>mcmpbalance</td>
<td>Record consistent interference</td>
</tr>
<tr>
<td>mcapfilt</td>
<td>Apply a multi-channel filter</td>
</tr>
<tr>
<td>mclean</td>
<td>Kill multi-component</td>
</tr>
<tr>
<td>mcrecoconc</td>
<td>Record consistent interference</td>
</tr>
<tr>
<td>mdef</td>
<td>Apply mute (DDS)</td>
</tr>
<tr>
<td>mno</td>
<td>Apply/remove hyperbolic normal move</td>
</tr>
<tr>
<td>mscft</td>
<td>Apply weights and stack traces (DDS)</td>
</tr>
<tr>
<td>mth</td>
<td>An X window seismic display program (DDS)</td>
</tr>
<tr>
<td>mth_reverse</td>
<td>Reverse order any selected axes</td>
</tr>
<tr>
<td>mth_rotate</td>
<td>2D coordinate rotation</td>
</tr>
<tr>
<td>mth_extract</td>
<td>Extract either a well or a horizon</td>
</tr>
<tr>
<td>mth_report</td>
<td>Emulates the Matlab plot</td>
</tr>
<tr>
<td>mth-which</td>
<td>Utility to list the channels</td>
</tr>
<tr>
<td>mthcap</td>
<td>Apply a multi-channel filter</td>
</tr>
<tr>
<td>mthclean</td>
<td>Kill multi-component</td>
</tr>
<tr>
<td>mthdef</td>
<td>Apply mute (DDS)</td>
</tr>
<tr>
<td>mthm</td>
<td>Apply/remove hyperbolic normal move</td>
</tr>
<tr>
<td>mthms</td>
<td>Apply weights and stack traces (DDS)</td>
</tr>
<tr>
<td>mth해서</td>
<td>An X window seismic display program (DDS)</td>
</tr>
<tr>
<td>mth_reverse</td>
<td>Reverse order any selected axes</td>
</tr>
<tr>
<td>mth_rotate</td>
<td>2D coordinate rotation</td>
</tr>
<tr>
<td>mth_extract</td>
<td>Extract either a well or a horizon</td>
</tr>
<tr>
<td>mth_report</td>
<td>Emulates the Matlab plot</td>
</tr>
<tr>
<td>mth-which</td>
<td>Utility to list the channels</td>
</tr>
<tr>
<td>mthcap</td>
<td>Apply a multi-channel filter</td>
</tr>
<tr>
<td>mthclean</td>
<td>Kill multi-component</td>
</tr>
<tr>
<td>mthdef</td>
<td>Apply mute (DDS)</td>
</tr>
<tr>
<td>mthm</td>
<td>Apply/remove hyperbolic normal move</td>
</tr>
<tr>
<td>mthms</td>
<td>Apply weights and stack traces (DDS)</td>
</tr>
<tr>
<td>mth해서</td>
<td>An X window seismic display program (DDS)</td>
</tr>
<tr>
<td>mth_reverse</td>
<td>Reverse order any selected axes</td>
</tr>
<tr>
<td>mth_rotate</td>
<td>2D coordinate rotation</td>
</tr>
<tr>
<td>mth_extract</td>
<td>Extract either a well or a horizon</td>
</tr>
<tr>
<td>mth_report</td>
<td>Emulates the Matlab plot</td>
</tr>
<tr>
<td>mth-which</td>
<td>Utility to list the channels</td>
</tr>
<tr>
<td>mthcap</td>
<td>Apply a multi-channel filter</td>
</tr>
<tr>
<td>mthclean</td>
<td>Kill multi-component</td>
</tr>
<tr>
<td>mthdef</td>
<td>Apply mute (DDS)</td>
</tr>
</tbody>
</table>
• If you are not a software developer, then you can start to nap now....
Writing a DDS Program

• API’s in C & Fortran
  - Template examples with “core” DDS functions on the website
  - Matlab API in progress

• Compilers
  - Requires a recent version of gfortran (that supports pointers) or Intel, PGI,…

• **Not** thread safe
  - “Those geophysicists would shoot themselves with it anyway” - Comp. Sci Guru
  - Many codes use both OpenMP & MPI

• MPI
  - Template, convenience routines
  - Parallel input and output
  - Largest OpenMP+MPI job to date had 20,000 threads/processes.
Cat Herding…

“It compiled and ran. I’m done!”

- Documentation
  - Man pages required!
  - Automated template man page
- Regression testing
  - On-line test results (run with memory checking)
- Run-time Logging
  - Internal only!
  - Connections (who has used the program I need help with?)
  - “Risk” warning (production/test/where did you get this?)
What you are probably thinking…

“That all sounds complicated.”

• It’s not!
• Let’s look at a simple Fortran example ignoring headers…
  – record by record processing
  – can read input files of different formats
• Using headers is only slightly more complicated
  – requires extra trace buffer, and calls to extract headers/samples
Reading and writing parameters…

#include <fdds.h>

If( fdds_openpr(prog,"") .gt.0 ) call help()

ier=fdds_dict('par:','scan')

verbose = (fdds_switch('verbose',0) .eq.1)

ntaper=10
ier=fdds_scanf('ntaper nt','%d\0',ntaper)
ier=fdds_printf('ntaper=%d\0',ntaper)

ptr_taper=fdds_malloc8(dble(ntaper)*SIZEOF_REAL)

Open the print file, unless the user specified help= or -?

Scan from the parfile or command-line (not an input file)

Logical parameter: yes/no/true/false

Read integer parameter, with synonym. Write value to log file.

Allocate (maybe >2Gb)
fd_in=fddx_in('in','stdin:','title')
if (fd_in.lt.0) then
  ier=fdds_prterr('Unable to open input data file!\n\0')
else
  ier=fdds_scanf('size.axis(1)','%d\0',n1)
  ier=fdds_scanf('size.axis(2)','%d\0',n2)
n3=fdds_axis_prod(3)
endif

fd_out=fddx_out('out','stdout:','title',fd_in)
if (fd_out.lt.0) then
  ier=fdds_prterr('Unable to open output!\n\0')
endif

If ( fdds_errors().gt.0 ) fdds_closepr()
do i3 = 1,n3
  if (fddx_read(fd_in,record,n2).ne.n2) then
    ier=fdds_prterr('Reading record %d\n\0',i3)
    ier=fdds_closepr()
  endif

  call do_some_stuff()

  if (fddx_write(fd_out,record,n2).ne.n2) then
    ier=fdds_prterr('Writing record %d\n\0',i3)
    ier=fdds_closepr()
  endif
end do

ier=fdds_closepr()
Simple makefile…

Name := my_program
FSrcs := my_fortran_code.F

#DEBUG := TRUE
#WARN := TRUE
#CHECK := TRUE

MP := TRUE
MPI := TRUE

include ${DDSROOT}/etc/MakeVariables
include ${DDSROOT}/etc/MakeRules

• “make” to compile
• “make help” for options
• “make newtest” to create a test template
• “make tests” to run the tests
• “make man” to create a template man page

Useful flags for debugging
(commented out here)
• If you were napping, you might want to wake up now…
Looking ahead…

• R&D platform for seismic processing & imaging in bp

• Simplification
  – We are trying to move from USP to DDS
  – Less maintenance, training; more flexible formats (less use of integer2!)

• More user-friendly
  – GUI integrating a script builder with more powerful graphical QC tools

• Improvements to DDS
  – Parallel I/O with compressed data?
  – Thread safe?
  – Stub libraries for other packages?
  – Looking for opportunities to collaborate
“DDSGUI” – in progress

- 3D viewer for:
  - Slices of N-dimension data
  - Very large datasets
  - Surfaces
  - Well paths and Markers
  - Lease block lines
  - Scriptable

- Script builder
  - Ability to save and reload scripts
  - Very easy to add new functions (through XML)
  - Foreach loops
  - SGE job submission
  - Framework for error checking

- Based on Qt and VTK
Acknowledgments

• USP developers:
  - Paul Garossino, Paul Gutowski, Don Wagner, Martin Smith, Greg Partyka, Joe Wade, Ganyuan Xia, Bryan Helvey, Richard Crider…
  - Many people who worked at the Amoco Research Center in Tulsa.

• DDS developers
  - Randy Selzler, Jerry Ehlers
  - John Etgen, Joe Dellinger, Mike O’brien, Dan Whitmore,…
  - Many members of the bp Advanced Seismic Imaging Team

• BP management for permission to make USP & DDS open source and for permission to deliver this presentation.
For information about USP and DDS:

FreeUSP.org

Note: FreeDDS is currently hosted on FreeUSP.org