Parallel Software Solutions for Processing Hydrographic Data

Dr. M.J. Miller, The Naval Research Laboratory
Geary Layne, The Naval Research Laboratory
Jim Braud, The Naval Oceanographic Office
Dr. Krzysztof Sarnowski, Hydrographic Science Research Center

Naval Research Laboratory
Bldg. 1005, Code 7004.1
Stennis Space Center, MS, 39522, USA
mmiller@nrlssc.navy.mil

Abstract - **The research in this paper focuses on the I/O problem associated with a parallel application writing to a single physical disk. Included in our research are the original ideas that led to the first version of the parallel software, subsequent versions of the software derived from lessons learned from benchmark results, and speedup results of each version. The underlying purpose of this software is to process hydrographic data having a complicated, multi-tiered format. The data processing involves reading tens to hundreds of files containing raw data, filtering out extraneous data values, and writing the filtered data to a single file used in additional processing. The problem is not computationally intensive, but bound by the system's file writing capability. Results show that the more responsible the software was for organizing the data before writing, the better the speedup. The critical factor for writing data efficiently involved the limitation of writing data over a single I/O controller. Our parallel software has fantastic utility where system specifications do not allow for the use of parallel file systems, or writing data over multiple I/O controllers.**

I. INTRODUCTION

The Naval Research Laboratory's (NRL) Code 7440 Production Enhancement Team at the Stennis Space Center has been tasked to develop ways to speedup hydrographic data processing at the Naval Oceanographic Office (NAVOCEANO) [Depn02]. This paper presents the final development of a parallelized version of the Pfm_loader application customized to run on a Beowulf cluster.

This paper specifically covers software development efforts in early FY02 [Sarn02]. In a series of algorithms, called Schemes D, E, and F, a parallel algorithm previously implemented in Scheme C (see Fig. 1) has been integrated with an improved version NAVOCEANO's of the Pure File Magic (PFM) library. Former versions of this software not mentioned, namely Schemes A and B, were used as a design platform for the software architecture found in Scheme C. The main goal of this work was to increase the writing rate of binned data to a physical disk. The final software version is Scheme F.

The final parallel code of Scheme F achieves the best speedup for the largest available test dataset. The simple runs (no filtering) exhibited a speedup of 10 as compared to the original, serial algorithm. Runs with swath filtering showed a top speedup of 8. Runs with area filtering reached a speedup of 6.5. Runs with both swath and area filtering showed a top speedup of 7. In all cases, data strongly suggest that greater speedups could be achieved for larger than tested input datasets.

Section 2 of this paper is devoted to Scheme D. Section 3 presents results on an implemented threaded version of Scheme D. Section 4 deals with results for Scheme E. Section 5 describes Scheme F. Detailed results are presented on the performance of Scheme F when swath and area based filtering are enabled. Section 6 presents results illustrating the effect of the physical disk's I/O throughput rates on speedup results. Section 7 discusses results and future plans.

II. SCHEME D

In Scheme D, as in Scheme C, nodes are assigned one of the following roles — a master node, an I/O node, or a slave node. In general, more of the functionality of the original PFM library has been assigned to the slave nodes in Scheme D than in earlier schemes.

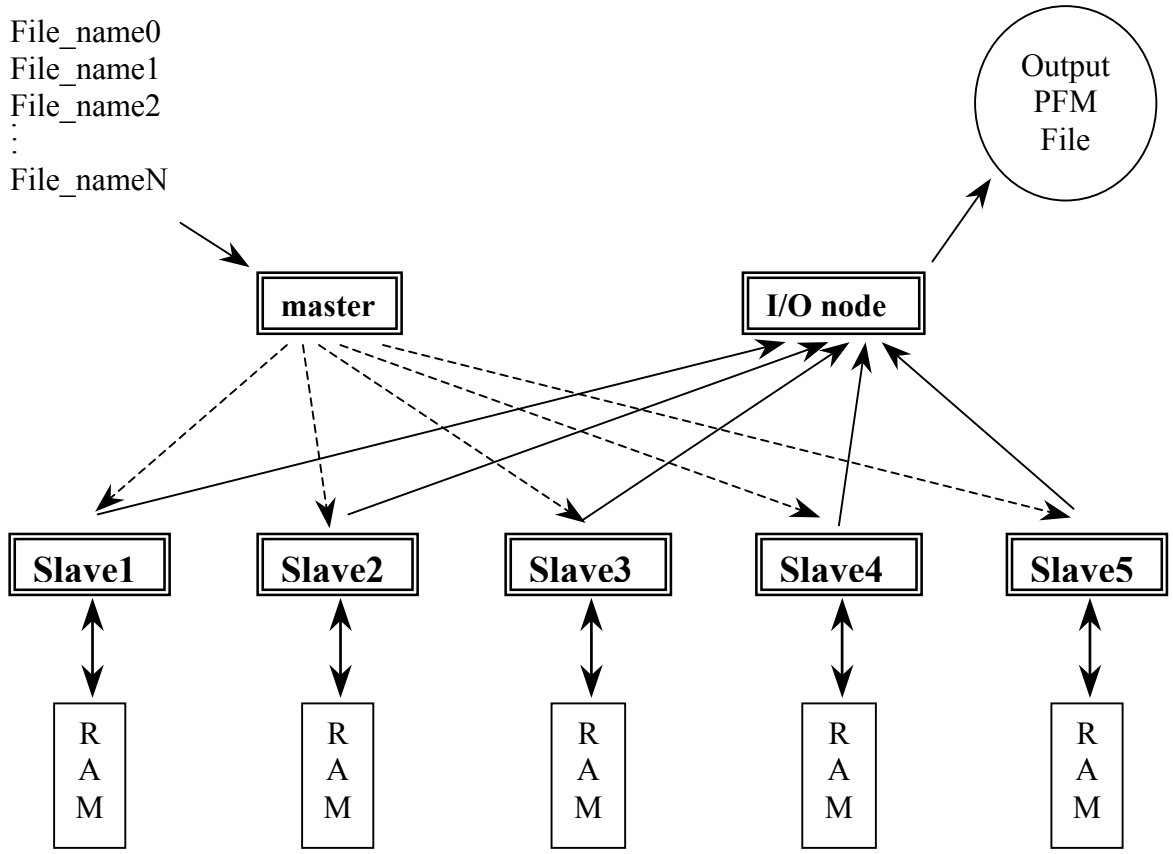

Fig. 1. Scheme C (the precursor to the schemes presented in this paper)

- Each slave node sorts read-in sounding data according to bin index and compresses them in the same way as is done in the original PFM library. The sending buffer is now in the form of a character string.
- The I/O node receives sounding data in the PFM compressed form and copies them into the PFM-style depth blocks consisting of six sounding data and a continuation pointer.
- The master node role has not been changed from Scheme C.

A. Results

Fig. 2 illustrates achieved speedups for different loads. The test datasets are denoted, as previously, by "L" (the Liberty dataset), "12" (12 files), "24" (24 files), "48" (48 files), and "74" (74 files). Timings have been averaged over four runs. Speedup figures are created by comparing parallel and serial runs. In the case of algorithm speedups, the comparisons are made with corresponding timings for three Message Passing Interface (MPI) processes. Speedups exhibited by Scheme D reach about 7 for the four larger dataset loads. For the smallest dataset, Scheme D shows a speedup of around 5, *reducing* the run time from around 2 min to 23 sec. For the largest input dataset tested, the measured speedup is 7 fold, *reducing* execution time from around 24 min to 3 min 20 sec. The optimal number of MPI processes is 9 for this scheme.

III. THREADING

A. Initial Plan

The overall processing flow can be improved by grouping tasks into at least two threads on the I/O node and a slave node. Tasks related to MPI communication can be accomplished by a separate thread. A testing program mixing MPI and threading confirmed that MPI Chamleon (MPICH) allows only a single thread to execute MPI calls. On the I/O node, a separate communication thread would take care of receiving buffers. The other thread would be

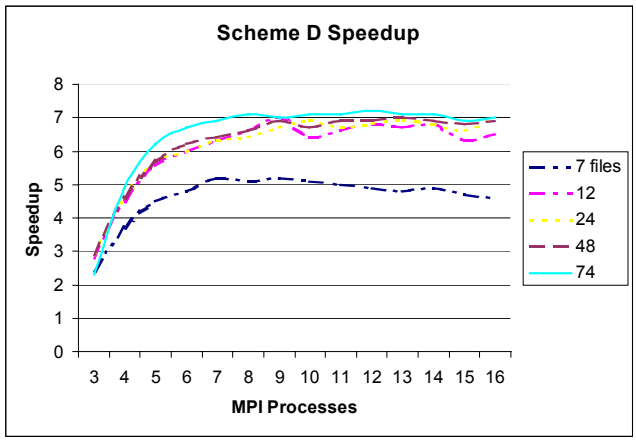

Fig. 2. Achieved speedup of Scheme D for different loads

responsible for writing data to the PFM output file. On the slave node, a communication thread would send full buffers to the I/O node. The second thread would transfer and process sounding data from input files to buffers. Tests are planned to check if the currently used dual-CPU system boards will efficiently support two application threads. A quad system board for the I/O node could be used if that proved beneficial.

B. *Threading the I/O Node: Algorithm*

Only one part of the above plan has been implemented: Scheme D has been tested using Linux Posix threads on the I/O node only. Processing on the I/O node has been separated into two threads. The second thread submits data to an output PFM file. The main thread does all necessary initialization and then creates one additional thread, called the I/O processing thread. The main thread acts as the I/O communication thread. All MPI function calls are done only by the I/O communication thread. The I/O communication thread receives buffers from worker nodes and submits them to a work queue. The I/O processing thread checks the work queue for any full buffers, and uses PFM function calls to write data to the PFM output file. An empty buffer is returned to the work queue area. The algorithm implements reuse of buffers in order to avoid reinitializing buffers. All access to the work queue is safeguarded by "a mutex lock" (Mutually Exclusive Access Lock) mechanism, a standard tool available in the thread library. A set of a few empty buffers is initialized in advance on the I/O communication side. If all empty buffers on the I/O communication node are used, that thread enters the work queue and gets their empty buffers. If no empty buffers are available in the work queue area, the I/O communication thread initializes a fresh buffer. Only if initializing the fresh buffer fails, the I/O communication thread waits for the I/O processing thread to return a buffer. The creation of additional buffers is always expected because PFM library operations are the slowest part of processing (since these library operations require multiple accesses to the physical disk drive).

C. *Threading the I/O Node: Results*

Fig. 3 illustrates achieved speedups for different loads. The test datasets are denoted, as previously, by "L" (the Liberty dataset), "12" (12 files), "24" (24 files), "48" (48 files), and "74" (74 files). Speedup values for the Scheme D-Threaded version are noticeably smaller than the original Scheme D. Such results could be attributed to the additional overhead of the thread library. However, results for the largest test dataset ("74") are especially disappointing, due to the lack of control of the memory usage on the I/O node in the threaded code. This preliminary threaded version has no control on the number of buffers used to keep incoming buffers on the I/O node. Since worker nodes deliver buffers much faster than the I/O node could possibly write them to a (slow) physical disk, incoming buffers forced the operating system on the I/O node to use disk swap space, causing a significant slow down in processing.

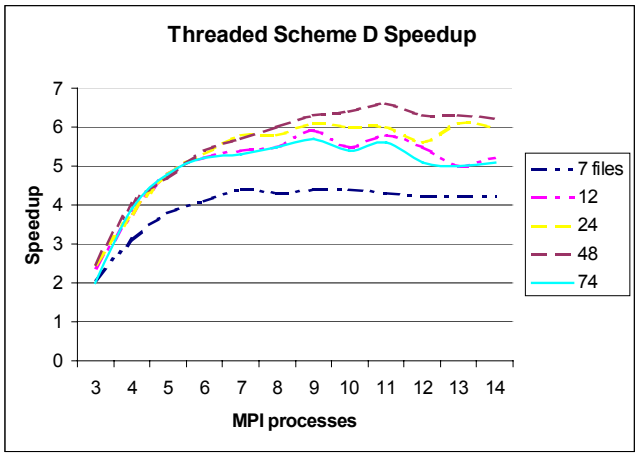

Fig. 3. Achieved speedups of Scheme D-Threaded for different loads

IV. SCHEME E

A. *Grouping Sounding Data into Packages of 6*

The improved version of the PFM library, as well as the original library, writes data to the physical disk in fixed length blocks. Each block can hold up to six sounding values (the value of six is configurable). To take advantage of these blocks, sounding data are sent in groups of six (with the same bin index). At first, slave nodes send depths grouped into packets of six (with same bin indexes) to create full depth records in PFM style. When all input file names have been distributed (and most of the files have been already processed) the finishing slave nodes have some leftover depth data. Separately, each node cannot create a final full depth record containing 6 sounding values for one bin index. Under the direction of the master node, the first slave node to finish processing becomes a sorting and grouping node. Then, the other slave nodes send their leftover data to that sorting node for consolidation. Subsequently, full records are sent to the sorting and grouping node (the one which currently is accepting data) with the final leftover data sent to the I/O node. This additional effort of sorting into groups of six depths has the advantage of increasing speed of writing the data to disk as well as avoiding any need to reread and rewrite partially empty depth records. Further improvements include a new interface function to the PFM library to accommodate direct writing of blocks to the PFM file. Also, slave nodes are given the additional task of creation of complete depth blocks. This improvement reduces the I/O node's task to simply updating the continuation pointers before writing data to disk.

B. *Results*

Fig. 4 illustrates achieved speedups for different loads, compared with the original serial application. The test datasets are denoted, as previously, by "L" (the Liberty dataset), "12" (12 files), "24" (24 files), "48" (48 files), and "74" (74 files). The timings have been averaged from four runs. A comparison between speedup numbers for Schemes D and E shows that values for Scheme E are noticeably smaller, except for the largest test dataset. For the smallest dataset,

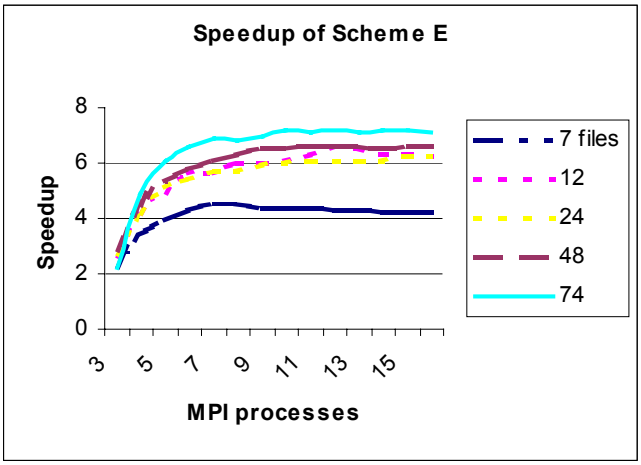

Fig. 4. Achieved speedup of Scheme E for different loads

Scheme E shows a speedup around 4.5, *reducing* the run time from around 2 min to 26 sec. For the largest input dataset tested, the measured speedup is around 7, *reducing* execution time from around 24 min to 3 min 19 sec. The optimal number of MPI processes is 10 for larger datasets in Scheme E.

C. *Filtering, Recomputing Steps and Final Tune-up*

The original filtering functionality works well in the Beowulf cluster environment. The original swath filtering was being handled separately for each input file. Thus the new distributed scheme of processing input files by a group of slave nodes has no effect on swath filtering. The same has been found for the recomputing step and area based filtering, which run without any changes to the code because they are executed in the serial phase, on the I/O node, only after the PFM file has already been created.

V. SCHEME F

A tuned version of the new PFM library was used in Scheme F. Scheme F was tested with four different setups, involving two possible filtering procedures: swath filtering, area filtering, both swath and area filtering, and no filtering (called "simple runs"). Each of these four setups has been tested with the standard five data test loads: "L" (the Liberty dataset), "12" (12 files), "24"

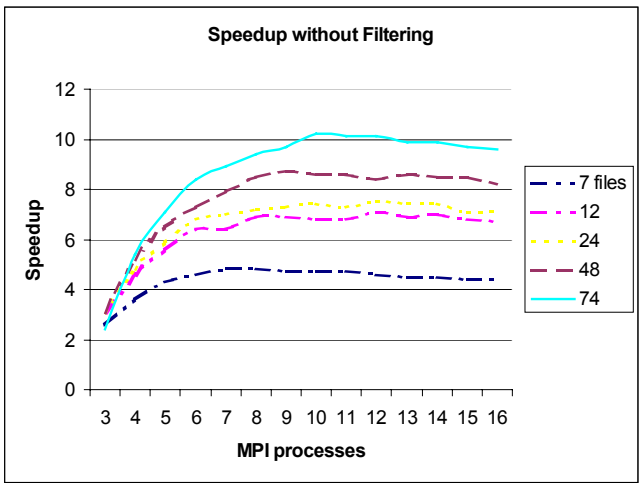

Fig. 5. Achieved speedups of Scheme F with no Filtering

(24 files), "48" (48 files), and "74" (74 files). Results with no filtering are illustrated in Figure 5.

It was found that Scheme F achieves the best speedup for the largest available test dataset. The simple runs (no filtering) exhibited a speedup of 10. Runs with swath filtering enabled showed a top speedup of 8. Runs with area filtering reached a speedup of 6.5. Runs with both swath and area filtering enabled showed a top speedup of 7. In the simple runs, swath filtering and area-based filtering are turned off.

Fig. 6 illustrates achieved speedups for different loads and filter processing. For the smallest dataset, when swath filtering is turned on, worker nodes filter the sounding data by swath before assembling them and sending to the I/O node. Since swath filtering puts additional processing onto worker nodes, such code behavior is to be expected. Worker nodes perform swath filtering on sounding data read from the input files. When area filtering is also turned on, the I/O node also performs area filtering of sounding data. Since area filtering is done exclusively by the I/O node after all sounding data has been written to the PFM file, the area filtering is performed in the serial phase of overall processing.

VI. THE ROLE OF HARD DRIVE PERFORMANCE

This section presents arguments to support the claim that the application Pfm_loader is I/O bound. When Scheme D was developed, upgrading the BIOS on each Beowulf cluster node resulted in significant improvement in the throughput of cluster IDE (ATA 100) disks. These changes significantly affected the run time of parallel codes as well as the original serial code. The changes for the original serial code are in the range from 5% to 11% (see table 1). The resulting speedup is presented in fig. 7. Speedup numbers are derived by comparing parallel and serial runs. In the case of algorithm speedups, the comparisons are done with corresponding timings for three MPI processes. The changes range from 29% to 57%, which at least triples that of corresponding percentage changes for serial runs.

VII. SUMMARY

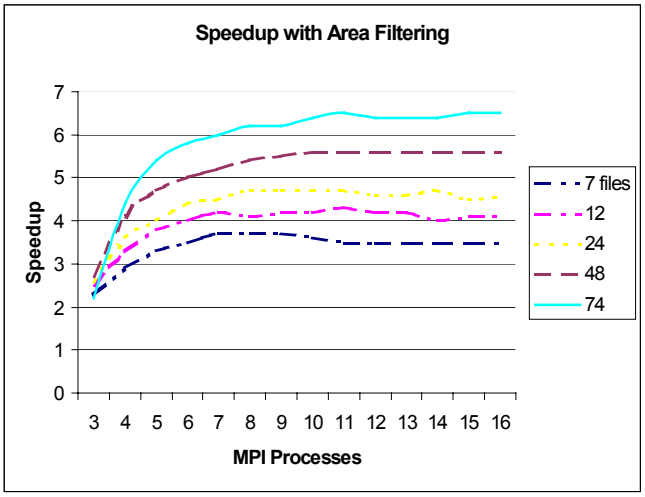

Fig. 6. Scheme F with Area Filtering

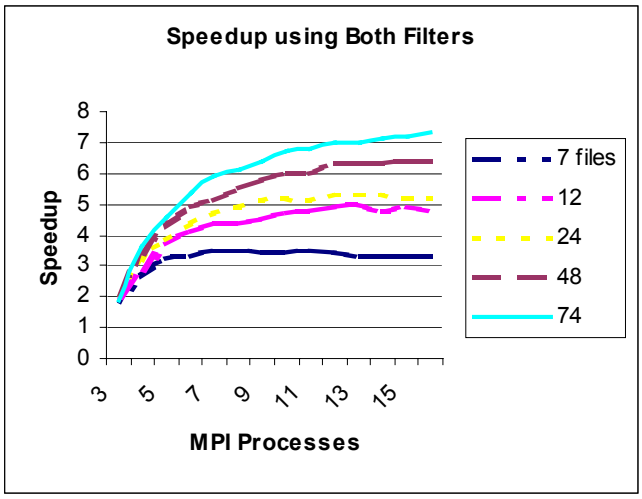

Fig. 7. Scheme F using both filters

The optimal number of MPI processes for the simple runs with large datasets is 9.6. When area filtering is turned on, the I/O node filters sounding data by geographic area. Since area filtering is done exclusively by the I/O node after all sounding data has been already written to the PFM file, this processing adds to the length of the serial phase in the overall processing. The serial processing nature of area filtering causes the greatest reduction in performance. The results generated by testing two of the four types of processing provide arguments for increasing the size of the Beowulf cluster. These two processing types are: (a) processing with swath filtering enabled, and (b) processing with both swath and area filtering enabled. For the largest test dataset, both processing methods achieved their best speedups when running with the maximal possible number of MPI processes. However, the same data indicates the speedup increase would be modest.

In all cases, the data strongly suggests that a speedup greater than 10 could be achieved for larger datasets. This property is a positive characteristic of the current parallel code. It also strongly suggests that the five test datasets have not yet pushed the current code and cluster configuration to its limits.

TABLE 1
Timings of change in hard drive performance

Number of Generic Sensor Format (GSF) input files				
7	12	24	48	74
Timings of serial code, before BIOS upgrade				
125.11	203.57	439.25	946.03	1607.13
Timings of serial code, with BIOS upgrade				
118.44	192.22	412.31	893.73	1433.21
Percentage change				
5.33%	5.58%	6.13%	5.53%	10.82%

ACKNOWLEDGEMENTS

This work was sponsored under Program Element 0603704N by the Oceanographer of the Navy via SPAWAR PMW 155, Captain Bob Clark, Program Manager. The Naval Research Laboratory (NRL) wishes to acknowledge contributions the Naval Oceanographic Office (NAVOCEANO) has made to this research. In particular, NRL thanks Mr. Jim Braud, Mr. Jan Depner, Mr. Dave Fabre, Mr. Roger Meadows, Mr. Dave Richardson, Ms. Barbara Reed, and Mr. Steve Nosalik for contributing their ideas, test data, and original software sources. Additionally, the authors wish to acknowledge contributions made by the University of Southern Mississippi Hydrographic Science Research Center located at Stennis Space Center.

REFERENCES

Depner, J. et al, *Dealing with Increasing Data Volumes and Decreasing Resources*, Oceans MTS/IEEE, 2002.

Sarnowski, K., Miller, M.J., Layne, G., *Project Report FY02*, Hydrographic Research Center, 2002.

Oceans 2003

Celebrating the Past... Teaming Toward the Future

September 22-26, 2003
San Diego, California

Scientific Papers
Chairman's Message
Conference Committee
Exhibitor's List
Author's Index
Conference Program
Table of Contents
Search CD-ROM
Help

CD Rom

Marine Technology Society
IEEE
OES
Scripps Institution of Oceanography UCSD 1903 Centennial 2003

Oceans 2003

Celebrating the Past...
Teaming Toward the Future

September 22-26, 2003
San Diego, California
Town and Country Hotel and Convention Center

Celebrating the 100th Anniversary of
Scripps Institution of Oceanography

Return

OCEANS 2003 MTS/IEEE
CONFERENCE PROCEEDINGS
PUBLISHED BY:
OCEANS 2003 MTS/IEEE CONFERENCE COMMITTEE

Responsibility for the contents rests upon the authors, and not upon the Marine Technology Society, the Institute of Electrical and Electronics Engineers/Oceanic Engineering Society, or its members and Holland Enterprises dba Holland Publications.

Copies of the *OCEANS 2003 IEEE/MTS Proceedings* are available from:

The Marine Technology Society
5565 Sterrett Place, Suite 108
Columbia, MD 21044
410-884-5330
www.mtsociety.org

and

The IEEE Service Center
445 Hoes Lane, POB 1331
Piscataway, NJ 08854-1331
1-800-701-4333
www.ieee.org

All rights including translations, are reserved by the Marine Technology Society and the Institute of Electrical Electronics Engineers/Oceanic Engineering Society. For copying, reprint, or republications permission, write to Publications Manager, MTS 5565 Sterrett Place, Suite 108, Columbia, MD 21044 or IEEE Copyright Manager, IEEE Service Center, 445 Hoes Lane, POB 1331, Piscataway, NJ 08854-1331. Copyright and Reprint Permission: Abstracting is permitted with credit to the source. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

All rights reserved
Copyright © 2003
by
The Marine Technology Society

Softbound Book Version
ISBN: 0-933957-30-0

DVD Version
ISBN: 0-933957-32-7

CD ROM Version
ISBN: 0-933957-31-9

DVD, CD-ROM Layout and Replication by:
Holland Enterprises
Book Layout and Printing by:
Holland Publications
1782 Solana Glen
Escondido, CA 92026-1858
Phone: 760.743.8486
Fax: 760.743.5758
E-mail: vivholland@usa.net

CD Contents

Message from the Chairmen iii

Message from the Executive Vice Chairman iv

Society Liaisons v

Conference Committee vi

Message From The Exhibits Director viii

Exhibitors ix

Author's Index xiii

Posters xxxv

Student Posters xl

Table of Contents / Sessions xlii

Table of Contents

Ecosystem-Based Management-
A New Goal Under NOAAs Strategic Plan 1011
Mr. James P. Burgess, *National Oceanic and Atmospheric Administration, MD*

A Tidal Habitat Restoration Success Story - The Union Slough (WA) Restoration Project 1015
Dr. Jon P. Houghton, James E. Starkes, *Hart Crowser, Inc., WA*

Session 53
Data Access & Vizualization
Co-Chairs **Joe Stinus**
NOAA NCDDC, SC
Lindsay Gee
Interactive Visualization Systems, Canada

SIOExplorer: IT for Research and Education 1016
Mr. Stephen P. Miller, Dr. Hubert Staudigel, Dr. Catherine Johnson, Deborah Day, Dr. John Helly
Scripps Institution of Oceanography, CA

New Developments in Internet-Based Delivery of MetOc Data to Warfighters 1020
Ms. Elizabeth Warner, Dr. Roy Ladner, Udaykiran Katikaneni, *Naval Research Laboratory, MS*
Mr. John Shea, *SPAWARSYSCOM, CA*

Spatio-Temporal Data Access for Information-Based Decision Making 1025
Dr. Roy Ladner, Elizabeth Warner, Dr. Fred Petry, *Naval Research Lab, MS*

EnVis/Hum: High-Resolution Ocean Model Visualization and Display 1030
Mr. Michael A. Chupa, Mr. Jace A. Mogill, Mr. Derek W. Irby, Dr. Robert J. Moorhead
Mississippi State University / ERC, MS
Dr. Jay F. Shriver, Mr. Peter M. Flynn
Naval Research Laboratory, MS

Parallel Software Solutions for Processing Hydrographic Data 1032
Dr. M.J. Miller, *The Naval Research Laboratory,* Geary Layne, *The Naval Research Laboratory,* Jim Braud, *The Naval Oceanographic Office,* Dr. Krzysztof Sarnowski, *Hydrographic Science Research Center, Naval Research Laboratory, Stennis Space Center, MS*

Session 54
Ocean Observatory Programs
Chair **Dr. John A. Orcutt**
Scripps Institution of Oceanography, UCSD, CA

Project Neptune: An Innovative, Regional Cabled Ocean Observatory in the Northeast Pacific 1038
Dr. John R. Delaney, *University of Washington, WA,* Dr. Christopher R. Barnes, *University of Victoria, Canada,* Dr. Patricia Beauchamp, *Jet Propulsion Laboratory, CA,* Dr. Alan Chave, *Woods Hole Oceanographic Institution, MA,* Dr. Marcia McNutt, *Monterey Bay Aquarium Research Insitute, CA,* Dr. John Madden, *Institute of Pacific Ocean Science and Technology, Canada*

Dynamics of Earth and Ocean Systems (DEOS) 1043
Prof. John A Orcutt, *Scripps Institution of Oceanography, CA*

Session 55
Quik SCAT Vector Winds: Validation & Applications
Co-Chairs **Charie Hopkins**
Planning Systems Inc., CA
Dr. Mark A. Bourassa
Florida State Univ., FL

Validation of QSCAT Vector Winds with Air-Sea Interaction Spar Fluxes 1044
Prof. Hans C. Graber, Prof. Mark A. Donelan, *University of Miami, FL,* Michael Caruso, *Woods Hole Oceanographic Institution, MA,* Dr. Vincent J. Cardone, *Oceanweather, Inc., CT*