3B.2

DAP-ENABLED SERVER-SIDE DATA REDUCTION AND ANALYSIS

Daniel L. Wang,* Charles S. Zender, and Stephen F. Jenks
University of California, Irvine

1. INTRODUCTION

Despite the frenetic pace of technology advancement towards faster, better, and cheaper hardware, terascale data reduction and analysis remain elusive for most. Disk technology advances now enable scientists to store such data volumes locally, but long-haul network bandwidth considerations all but prohibit frequent terascale transfers. Bell et al. (2006) have noted that downloading data for computation is worthwhile only if the analysis involves more than 100,000 CPU cycles per byte of data, meaning that a 1GB dataset is only worth downloading if analysis requires 100 teracycles, or nearly 14 hours on a 2GHz CPU. In data-intensive science, data volume rather than CPU speed drives analysis, pointing to a need for a system that locates colocates computation with data.

Our system provides a facility for colocating computation with data sources, leveraging shell script-based analysis methods to specify details through an interface piggy-backed over the Data Access Protocol (DAP) protocol, implemented through a custom OPeNDAP data handler (Cornillon, 2003). Scripts of netCDF Operator (NCO) (Zender, 2006b) commands are sent through an interface extended from DAP's subsetting facility and processed by a server-side execution engine. Resultant datasets may be retrieved in the same DAP request, or deferred for later retrieval. Additional processing efficiency is available through script-based parallelism. Our execution engine optionally parses scripts for data-dependencies and exploits parallelism opportunities from the extracted dataflow. With this capability, existing analyses can better utilize the available parallelism of high-capacity datacenter hardware.

2. RELATED WORK

Other systems for remote computation exist in many areas and can be characterized by their generality and computation size. Grid computation engines remain the most general, allowing the widest variety of heavy computational tasks to be run on a heterogeneous set of remote systems (Foster and Kesselman, 1998). Unfortunately, their application independence means that, in practice, data-dependencies cannot be specified automatically, and that data locality is generally ignored in favor of greater scheduling flexibility for higher throughput. The Globus toolkit for grid systems allows users to define input and output files to be staged to and from compute nodes (Foster and Kesselman, 1997), but this capability may not be desired for data-intensive computation, where data volume is more significant than computation complexity. Systems for remote data access that specific to geoscience data processing are also in wide use. While such systems succeed in providing simple, lightweight access to large, remote datasets, their interfaces remain optimized for operations on small volumes of data, prompting scientists to utilize other tools such as NCO for larger-volume data analysis and reduction.

The Open-source Project for a Network Data Access Protocol (OPeNDAP) server serves a significant fraction of available geoscience data (Cornillon, 2003). It provides metadata querying and subsetting capabilities as well as access to raw data. The Earth System Grid II (ESG) project aims to address concerns similar to those of this project, and is in development of filtering servers that permit data to be processed and reduced closer to its point of residence (Foster et al., 2002). The NCO tools supported by our system operate on data stored in network Common Data Form (netCDF) (Rew and Davis, 1990). netCDF is a self-describing, machine-independent format for representing scientific data, and is the most popular format for exchanging ocean-atmosphere model output.

3. OVERVIEW

Our system, currently called SSDAP (Server-Side DAP), is implemented as a wrapper to the netCDF data handler in an OPeNDAP instance. Just as in generic DAP protocol transactions, SSDAP requests are sent to an `httpd`, where they are shunted to the OPeNDAP CGI handler, which parses the request and forwards the result to a data-handler specific to the dataset file format. Our customized netCDF data handler processes the requested script, if available, forwarding the request

*Corresponding author address: Daniel L. Wang, Department of EECS, 425 Engineering Tower, Irvine, CA; e-mail: wangd@uci.edu to the standard netCDF data handler otherwise.

From a client's perspective an SSDAP-enabled OPeNDAP server instance is indistinguishable from a vanilla OPeNDAP installation, except for additional script-processing capabilites. To initiate server-side analysis, the client makes a standard DAP request using a special constraint expression that signals the existence of a script, sent in the same request via HTTP POST. Simple directives in scripts flag files as "output" or "temporary", facilitating proper execution and transmission of results.

4. EXAMPLE SCRIPT

We tested our system with a script that resamples Community Atmospheric Model simulation data into time-steps that can be better compared against observed NASA Quick Scatterometer (QuikSCAT) data (Tsai et al., 2000). In this script, ten years of data at 20-minute timesteps are masked for their surface wind speed values at 6:00AM and 6:00PM, the local times from the QuikSCAT satellite passes. The script contains over 14,000 NCO command-lines for masking, averaging, concatenating, and editing, which produce 228MB of resultant data from 8230MB of input data, and generate 26GB of temporary intermediate files in the process. A dataflow summary of this example is shown in Figure 1.

5. USAGE

Usage of SSDAP is designed to require minimal training. Shell scripts of NCO commands (Zender, 2006a) require only small modifications to flag command outputs as temporary or output files. Figures 2 and 3 illustrate these modifications, which are necessary to eliminate unnecessary transfer of unwanted intermediate data. Once the script is appropriately modified, it can be sent via HTTP to a standard OPeNDAP URL, tagged with a special OPeNDAP constraint expression signalling the presence of a computation script via HTTP POST. Computation proceeds remotely at the OPeNDAP instance and produces a result that is transmitted back in the standard DAP fashion. Overall, using SSDAP requires two things: modification of scripts to signal temporary and output files, and submission to a modified OPeNDAP server rather than executing on the local workstation. Future work will eliminate manual script modification in the majority of cases.

Our simplified usage depends on an execution engine implemented as a OPeNDAP data handler. This execution engine parses the user script for basic correctness and dataflow information, and

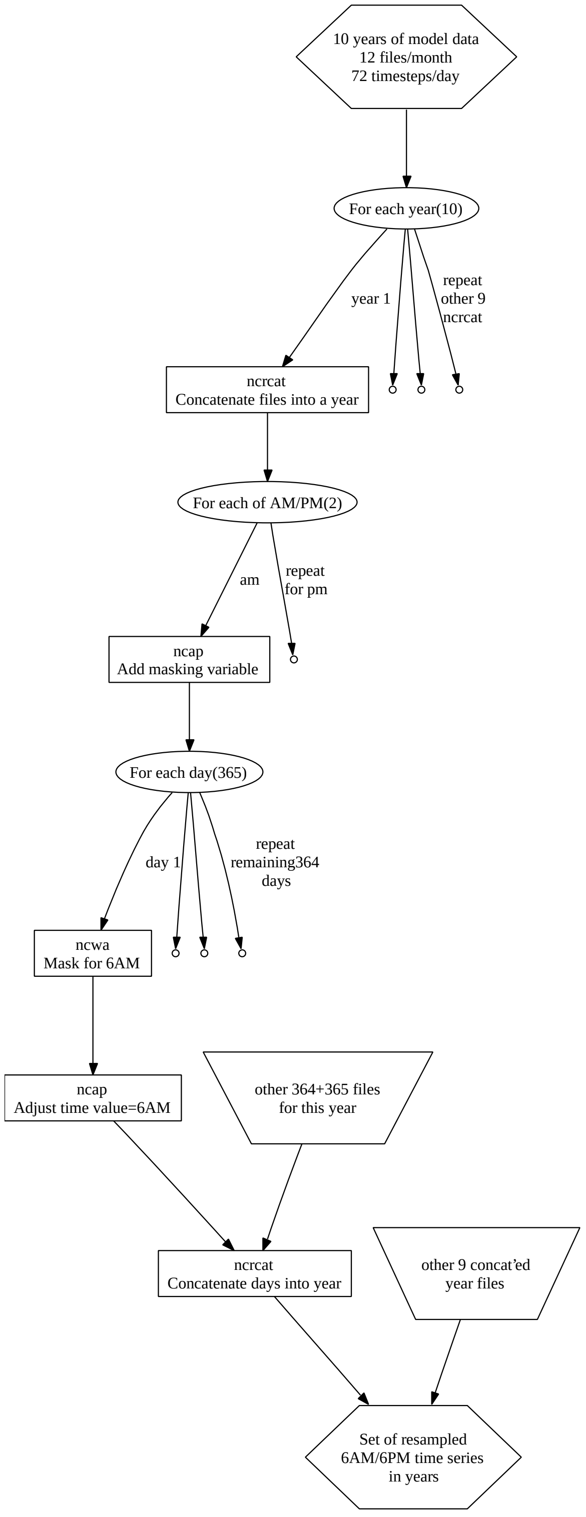

FIG. 1: Example script dataflow

```
## original script that obtains a yearly average
ncrcat -O model.1990*.nc model.yr.nc
ncwa -O model.yr.nc model.avg.nc

## SSDAP-version of original script
ncrcat -O model.1990*.nc %tempf_model.yr.nc%
ncwa -O %tempf_model.yr.nc% %outfile_model.avg.nc%
```

FIG. 2: Original and SSDAP-modified script comparison

manages execution of the script commands, optionally detecting and exploiting parallelism where available. The example script contains numerous parallelism opportunities, easily providing the four cores of our dual Opteron 270 test machine with constant work from start to finish. When all work is completed, the resultant data file can be immediately returned to the user, or, alternatively retrieved later, if there are multiple files. The DAP protocol does not have the facility to transmit multiple files in a single transaction.

6. RESULTS

We tested our system with the above example script on a dual Opteron 270 with 16GB of memory with dual 500GB SATA drives in RAID 1, running a CentOS 4.3 kernel. In our example, a scientist can avoid downloading nearly 8GB, by obtaining just 228MB of output rather than the entire input dataset. This illustrates the potential savings possible from elimination of unnecessary network traffic. Even given a decent 1MByte/s bandwidth to the data source, saving 8GB of downloading saves 136 minutes of transfer time.

Figure 4 summarizes the test results, estimating transfer time using an average bandwidth of 2^{20}bytes/sec. Computation cost on our test system was measured as roughly 53 minutes in local (non server-based) serial execution, with overall speedups of 0.80 (67 minutes) and 1.93 (28 minutes)for server-based execution in serial and parallel modes, respectively. The performance penalty in the server-based serial execution mode illustrates overhead from parsing and database queries necessary to manage execution. However, with network transfer costs included, we see speedups of 2.69 (71 minutes) and 6.06 (31 minutes) in serial and parallel modes, clearly showing the significance and benefit of eliminating unnecessary network traffic. Currently, execution speed seems limited by I/O parallelism, limiting the benefit of additional processors. Future work will include confirmation and characterization of this bottleneck and explore possible solutions.

SSDAP filename	Description
path/name	path to file on the server
%tempf_*name*%	an intermediate file
%outfile_*name*%	a result file desired by the user

FIG. 3: SSDAP basic syntax

	Local serial	SSDAP serial	SSDAP parallel
Transfer size	8230MB	228MB	228MB
Transfer time	137:11	3:48	3:48
Compute time	53:18	67:02	27:38
Total time	190:29	70:50	31:26
Speedup (Transfer)	1.00	36.1	36.1
Speedup (Compute)	1.00	0.80	1.93
Speedup (Total)	1.00	2.69	6.06

FIG. 4: Timing results (Times shown as mm:ss, 2^{20}bytes/sec bandwidth)

Our system targets scientists with compute capacity or network connectivity less than what a data center offers, which we believe should include most scientists. Data centers should benefit as well from reduced external network usage, which is often more costly than CPU speed. Since scripts are transferred to data sources, additional insight may result from the aggregated knowledge of scientists' data usage patterns.

7. CONCLUSION

A server-side data reduction and analysis system saves scientists time and bandwidth, enabling them to exploit potentially greater computing resources with minimal additional effort. We leverage existing script-based methods of analysis and the widely used DAP protocol to provide simple distributed computing to non-computer-scientists. Initial tests and use cases show the potential for significant bandwidth savings and faster results generation. While performance of the current implementation provides a significant speedup, future implementations will further exploit clustering and parallelism available at the data center, further enhancing performance. Systems such as ours that colocate computation with data will be well poised to meet the demands of more comprehensive, more detailed, and more frequent analyses, and will facilitate data-intensive science.

8. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grants ATM-0231380 and IIS-0431203.

REFERENCES

Bell, G., J. Gray, and A. Szalay, 2006: Petascale computational systems. *IEEE Computer*, **39**(1), 110–112. 1

Cornillon, P., 2003: OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal*, **2**, 164–174. 1, 2

Foster, I., E. Alpert, A. Chervenak, B. Drach, C. Kesselman, V. Nefedova, D. Middleton, A. Shoshani, A. Sim, and D. Williams, 2002: The Earth System Grid II: Turning Climate Datasets Into Community Resources. In *Proceedings of the 18th International Conference on Interactive Information and Processing Systems for Meteorology*, American Meterological Society, AMS Press, Boston, MA. 2

Foster, I. and C. Kesselman, 1997: Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, **11**(2), 115–128. 2

Foster, I. and C. Kesselman, 1998: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA. 2

Rew, R. K. and G. P. Davis, 1990: NetCDF: an interface for scientific data access. *IEEE Computer Graphics and Applications*, **10**(4), 76–82. 2

Tsai, W.-Y., M. Spencer, C. Wu, C. Winn, and K. Kellogg, 2000: SeaWinds on QuikSCAT: Sensor Description and Mission Overview. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 3, Honolulu, HI, 1021–1023. 4

Zender, C. S., 2006a: NCO User's Guide, version 3.1.4. http://nco.sf.net/nco.pdf. 5

Zender, C. S., 2006b: netCDF Operators (NCO) for analysis of self-describing gridded geoscience data. *Submitted to Environ. Modell. Softw.* Available from `http://dust.ess.uci.edu/ppr/ppr_Zen07.pdf`. 1