INTERACTIVE SUPERCOMPUTING WITH IN-SITU VISUALIZATION OF OCEAN BIG DATA

Desislava Ivanova
Technical University of Sofia, Faculty of Applied Mathematics and Informatics,
8 Kliment Ohridski,
1000 Sofia, Bulgaria, E-mail: d_ivanova@tu-sofia.bg

Abstract: The paper proposes an interactive approach with In-situ visualization for analysis and post-processing of ocean big data. The NEMO ocean engine simulates the ocean thermodynamics, sea-ice dynamics and biogeochemical processes. All these ocean processes are with big impact on climate change. In HPC simulation of ocean is important to have quick access to the simulation data, to explore the results and to retrieve the knowledge in real-time. The NEMO code is written in FORTRAN. The visualization of high-performance computer system simulations is the bottleneck with respect to storing and retrieving data in disk storage. The other challenging part of the approach is the integration of FORTRAN code with the modern GPU accelerators for in-situ visualization. The proposed interactive approach with In-situ visualization will trace the path to couple the simulation with visualization and will provide the visualization whilst the simulation is running.

Key words: In-situ Visualization, NEMO Ocean Engine, Interactive Supercomputing, Ocean Big Data

1. Introduction

Ocean modeling such as NEMO ocean engine is a key approach toward better understanding of complex environmental systems. The real-time visualization of NEMO OPA simulations can help the scientists to discover the events like ocean gyres, flooding, and ocean margins etc. with big impact on forecasting and climate change. The problem is that the post-processing of big data from HPC ocean simulations is a bottleneck with respect to storing and retrieving data in disk storage [1].

The scientific community is currently witnessing unprecedented growth in the quality and quantity of big data coming from HPC simulations and experiments. In addition, recording the results of numerical simulations on a disk has been an obstacle to high performance computing. In fact, the main barrier to traditional visualization is due to I/O, connecting to the fact that multiple HPC nodes are used for simulations, all or some of them store the data, and a smaller number of nodes on a separate system read and visualized the data. It is obvious, that the effective access and retrieval of the scientific content of such large-scale datasets requires appropriate tools and techniques [2].

The way to avoid the difficulties of storing and retrieving data in a storage disk is to combine simulations and visualization. In-situ visualization enables the users to connect directly to a running simulation, view the data, create queries, and create a graphical output in real time as the simulation is running. For this reason, In-situ visualization is of particular interest and importance to the scientific community.

The paper proposed the interactive supercomputing with In-situ visualization for analysis and post-processing of ocean big data in real-time that will allows to explore the results and to retrieve the knowledge from the simulation in real-time.

2. Techniques and Tools for In-situ Visualization

In-situ visualization allows coupling a simulation with visualization in order to visualize data produced by the HPC simulation remotely in real time, whilst it is still running. There are different techniques for in-situ visualization depends on data produced by the HPC simulations, on user needs and post-processing. The In-situ visualization techniques can be considered as *loosely coupled, tightly coupled and hybrid.* [3]

The *loosely coupled technique* is characterised with visualization and analysis run on concurrent resources and access data over network. The *tightly coupled technique* is with direct access to the memory of the simulation code, analysis algorithms run on the same nodes as the simulation. Finally, the *hybrid technique is* characterised with data is reduction in a tightly coupled setting and sent to a concurrent resource. [4, 5, 6]

Nowadays, there are some widely used tools and frameworks for in-situ visualization like *ParaView, ICARUS plug-in and VisIt.*

ParaView and VisIt In-situ visualization tools are designed from the ground up for massively parallel, remote real-time visualization in HPC simulations. They tackle data sets so huge that they need to be distributed across multiple nodes of a high-performance computer system, and all the visualization steps, including filtering, rendering and compositing can take advantage of multiple processors. The both tools have a client-server architecture that allows remote visualization, providing a responsive user interface without requiring transfers of large datasets from the supercomputing facility.

ParaView is an application designed for data parallelism on shared-memory or distributed-memory multicomputers and clusters. It can also be run as a single-computer application. The client - server architecture of ParaView facilitates remote visualization of datasets, and generates level of detail (LOD) models to maintain interactive frame rates for large datasets. It is an application built on top of the Visualization Toolkit (VTK) libraries. The ParaView code base is designed in such a way that all of its components can be reused to quickly develop vertical applications. ParaView runs parallel on distributed and shared memory systems using MPI. Moreover, it is also used as a server of a web application aimed to provide a collaborative remote web interface for 3D visualization. [7]

ICARUS plug-in allows the transfer of data in parallel between two different codes, or between a simulation and a post-processing application, such as a visualization tool. ICARUS is a ParaView plug-in for steering and in-situ visualization. Furthermore, it is able to transparently send data across a network in parallel to a distributed shared memory buffer allocated on different nodes of the same machine, or even of a completely different machine but accessible via the network. Data transfer between the simulation and the ParaView server takes place using a virtual file driver for HDF5 that bypasses the disk entirely by directly communication between the coupled applications in parallel, simulation and visualization tool. It implements a loosely coupled and push-driven approach, where the link between simulation and visualization is decoupled to have a shared memory mapped file in between. It minimizes modification of existing simulation codes but requires data written using HDF5 format. [8]

Other option is visualization with ParaView co-processing plugin which allows an application code to be instrumented to connect to a ParaView server in order to execute a visualization pipeline. The pipeline can produce either images in a variety of formats or VTK XML parallel file format data sets. A user creates a pipeline in the ParaView client using a sample. This pipeline is then exported using the CoProcessor plugin as a Python script which will be loaded by the application. [9]

The other in-situ vitalization tool for massively parallel in-situ visualization is VisIt. It is an open source interactive parallel visualization and graphical analysis tool for viewing scientific data. VisIt contains a rich set of visualization features to enable users to view a wide variety of data including scalar and vector fields defined on two- (2D) and three-dimensional (3D) structured, adaptive and unstructured meshes. Due to its distributed and parallel architecture, VisIt is able to handle very large datasets interactively. In addition, VisIt is extensible, allowing users to add data loaders or additional analysis tools. Owing to its customizable plugin design, VisIt is capable of visualizing data from over 120 different scientific data formats. VisIt provides an easy to use GUI as well as C++, Python and Java interfaces. The Python scripting interface gives users the ability to batch process data using a powerful scripting language. This feature can be used to create extremely sophisticated animations or implement regression suites. [10]

3. Nemo Ocean Framework

NEMO is an ocean modeling framework which is composed of "engines" nested in an "environment". The "engines" provide numerical solutions of ocean, sea-ice, tracers and biochemistry equations and their related physics. It includes four engines: OPA - the ocean engine. LIM2: the previous version of the Louvain-la-Neuve sea-ice model. LIM3: the latest version of the Louvain-la-Neuve sea-ice model including a new thermodynamics and a C-grid Elasto-ViscoPlastic rheology. TOP2: the passive tracer package including a transport component (TRP), and source/sink associated to CFC and LOBSTER and PISCES biogeo-chemical models. [11]

The NEMO code is written in FORTRAN. For the moment there are 6 available configurations which could be used with free download datasets: 1) ORCA2_LIM: a coupled ocean / sea-ice configuration based on the ORCA tripolar grid at 2° horizontal resolution with climatological forcing. 2) ORCA2_LIM_ PISCES: above configuration with PISCES biogeochemical model.
3) ORCA2_OFF_PISCES: PISCES model forced with output form an ORCA2 simulation.
4) GYRE: an idealized double gyres configuration on a beta-plane at 1° horizontal resolution with analytical forcing.
5) GYRE_LOBSTER: above configuration with LOBSTER biogeochemical model.
6) POMME: an Open Boundary Configuration on the POMME experiment zone.

4. Interactive Approach with In-situ visualization for Analysis and Post-processing of Ocean Big Data.

The proposed approach used high-performance computer system for massively parallel simulations and CUDA servers for in-situ visualization via SSH tunnels. Both platforms are communicated with each other and the client is associated with both platforms. The high-performance computer system data server reads the client set files and applies the selected filter. The data server transmits the data to the image generating server. The image generating server processes its transmitted geometry and forward pixels to the client. The result of simulation files is displayed on the client. The proposed approach is visualized on Fig. 1.

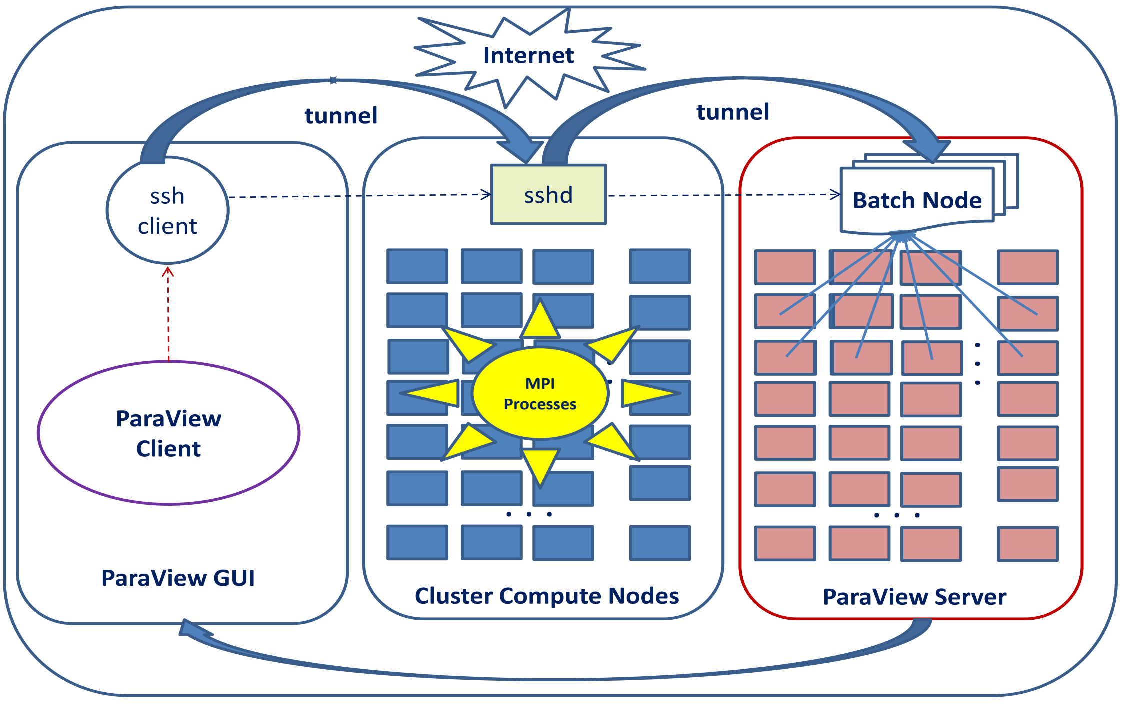

Fig. 1 Interactive Approach with In-situ visualization for HPC Post-processing

The visualization of HPC simulations requires parallel systems with lots of memory and fast GPU accelerators. The data cannot be moved off system where it is computed. It is unsuitable to move the data over WAN, and costly to move over LAN.

The proposed interactive approach with In-situ visualization for HPC post-processing of ocean big data solved all these problems. First, it solved the issue with integration of NEMO FORTRAN code with modern GPU clusters and couple the simulation with visualization whilst the simulation is running. Second, the proposed approach solved the huge problem with data movement as eliminate the single points of failure with respect to storing and retrieving data in disk storage.

4.1. Experimental Framework.

The experimental framework is based on a high-performance computer system for HPC simulations and CUDA cluster for in-situ visualization. The experiments are performed for NEMO ocean OPA model with ORCA2 dataset.

High-performance computer system supports a message passing programming model for parallel implementations.

The CUDA cluster is located at Technical University of Sofia. It is based on Tesla M2090 accelerators with 665 Gigaflops peak double precision floating point performance, 512 CUDA cores, 6 GBytes GDDR5 and 177 GBytes/sec memory bandwidth.

4.2. ParaView: Settings and Configuration for Interactive Super-computing with In-situ visualization of Ocean Big Data.

For the configuration and implementation of ParaView a CMake version 2.8.8 or higher version and a working compiler is required. Linux operating systems required *Make* while Windows requires *Visual Studio* (8 or higher version). To build the *ParaView User Interface, Qt* is required, version 4.7 (4.8 recommended). It can be used either *LGPL* or commercial versions of *Qt* to compile ParaView. To run ParaView in parallel, MPI is also required. With respect to the script usage, Python is necessary. ParaView uses OpenGL graphics drivers and maps from the user's workstation. When it is running the ParaView server on a platform that does not include OpenGL hardware support, it must be used MESA to emulate this hardware in the software area. When the ability to write AVI files is desirable and writing these files is not supported by the operating system, attach ParaView to the FFMPEG library, like all the necessary compilers (C, C ++, MPI, FORTRAN, etc.).

For the access to the machine where the ParaView will be configured, the SSH client *PUTTY* or a console mode of a local machine can be used. Once the latest version of ParaView is downloaded then, it is necessary to be created a new directory in the folder where it is located and the source folder with the example name *ParaView-built.* Then the configuration will start once the following command is running, into a newly created folder:

ccmake ../ ParaView-x.x.x-Source

The graphical interface will appear, indicating that the cache is empty. The configuration will start and after its completion, all the necessary packages and compilers will be automatically found, if it does not, it must be manually specified.

The new window represents all the settings of the current configuration (if you change an option, it may appear new depending on it).

Table 1. ParaView configuration features

Variable	Value
CMAKE_INSTALL_PREFIX	Path to directory /paraview-bin
PARAVIEW_ENABLE_ PYTHON	ON
PARAVIEW_USE_MPI	ON
BUILD_SHARED_LIBS	ON

Variable	Value
PARAVIEW_BUILD_QT_GUI	OFF
CMAKE_BUILD_TYPE	Release
PARAVIEW_ENABLE_ COPROCESSING	ON
BUILD_COPROCESSING_ ADAPTORS	ON
BUILD_FORTRAN_ COPROCESSING_ADAPTORS	ON
PYTHON_ENABLE_MODULE _vtkCoProcessorPython	ON

4.2. High-performance computer system: Settings and Configuration Features.

With respect to applying the interactive supercomputing with in-situ vitalization of ocean big data on high-performance computer system, there are some settings and configuration feature need to be taken into account and they are presented in Table 2.

Table 2. High-performance computer system configuration features

Variable	Value
VTK_OPENGL_HAS_OSMESA	ON
OSMESA_INCLUDE_DIR	NOT set up on the OpenGL directory for header files
OPENGL_INCLUDE_DIR	NOT set up on the OpenGL directory for header files
OSMESA_LIBRARY	Set the library MESA
OPENGL_gl_LIBRARY и OPENGL_glu_LIBRARY	ON
VTK_USE_OFFSCREEN	ON
PARAVIEW_ENABLE_ PYTHON	ON

4.2. ParaView Client-Server: Settings and Configuration Features.

The plugin for generating Python scripts for Co-Processing is client-side and it must be running on. Because this is a client-side plug-in, the *PARAVIEW_BUILT_QT_GUI* option must be turned on. After setting up the configuration, the configuration needs to be completed and to generate the make files. If everything has gone smoothly, the compiling can be started and setting for parallel compilation.

The next step is to start the servers in a single group of processes. The data server is started with the default port: 11111. The image generating server is running with the default port is 22221.

The Client is located at */path to the directory ParaView-buil /bin/paraview*. Particularly when connecting client to a graphical cluster, the tunnel must be created by using: *putty -ssh -N -L 11111:graph2:11111 gw.tu-sofia.bg*
Where *graph2* is the name of the node on which the *pvserver* is executed. After this operation, the server is available as a localhost and port 11111 for its name.

Once the client is connected, files from the data server are accessed through the File-> Open menu and all operations such as applying filters, saving to different file formats etc. are available from the user's graphical interface.

5. In-situ Visualization Results and Analysis

The interactive approach with In-situ visualization for post-processing of ocean big data is applied to NEMO ORCA2 configuration. A 4D-Var data assimilation technique is applied to an ORCA-2 configuration of the NEMO in order to identify the optimal parameterization of the boundary conditions on the lateral boundaries as well as on the bottom and on the surface of the ocean.

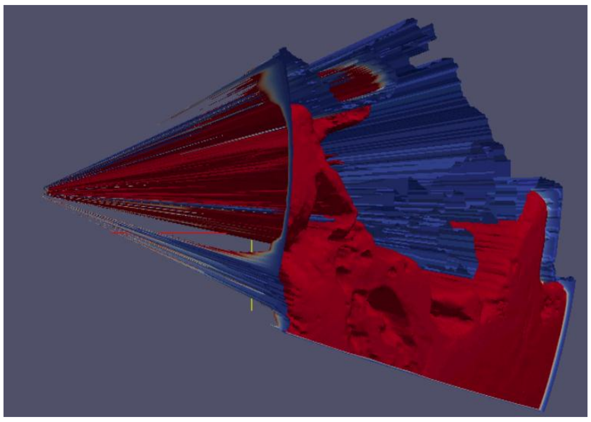

Fig. 2 In-situ Visualization of

Ocean Boundaries

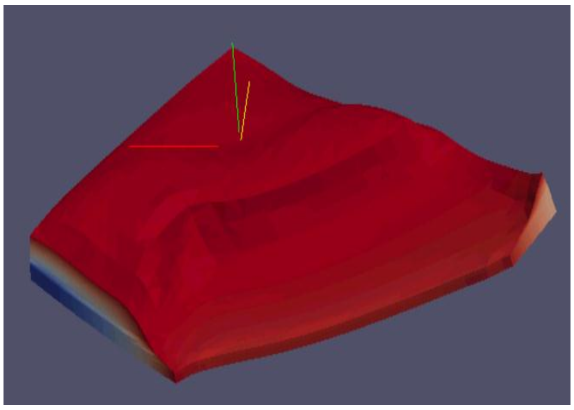

Fig. 3 In-situ Visualization of Grid Ocean

The test case shows that during the tests no particular overheads are noticed. The most important part of the test and only issue is that both client and server must be up and running during the HPC ocean simulations.

The proposed interactive approach with In-situ visualization for HPC post-processing of ocean big data solved the problems with integration of NEMO FORTRAN code with modern GPU clusters and couple the simulation with visualization whilst the simulation is running. Moreover, the proposed approach solved the huge problem with data movement as eliminate the single points of failure with respect to storing and retrieving data in disk storage.

The proposed interactive approach enables scientists in oceanology to analyze the results in real time and to make easy the verification of the NEMO simulation output data and to discover the events like ocean gyres, flooding, and ocean margins etc. which are with big impact on forecasting and climate change at early stage. That is of significant importance for the human life. The ability to save the results from visualization as popular video and picture formats makes it easy to spread the results among an even larger audience without additional software tools.

Acknowledgement

The results reported in this paper are part of a research project № 171ПР0015-11, supported by the Research and Development Sector, Technical University of Sofia.

References:

[1] Biddiscombe, J., Soumagne, J., Oger, G., Guibert, D., Piccinali, J.G.. 2011,”Parallel computational steering and analysis for HPC applications using a paraview interface and the HDF5 DSM virtual file driver”. In Proceedings of the 11th Eurographics conference on Parallel Graphics and Visualization (EGPGV '11), Torsten Kuhlen, Renato Pajarola, and Kun Zhou (Eds.). Eurographics Association, Airela-Ville, Switzerland, Switzerland, 91-100.

[2] Rivi, M., Calori, L., Muscianisi, G.,”In-situ Visualization State-of-the-art and Some Use Cases”, 2012, PRACE White Papers.

[3] J. Biddiscombe, J. Soumagne, G. Oger, D. Guibert, J-G. Piccinali, Parallel Computational Steering and Analysis for HPC Applications using a ParaView Interface and the HDF5 DSM Virtual File Driver. EGPGV 2011, 91-100.

[4] A. Henderson, ParaView Guide, A Parallel Visualization Application. Kitware Inc., 2005. Web site: http://www.paraview.org

[5] J. Soumagne, J. Biddiscombe, J. Clarke, An HDF5 MPI virtual file driver for parallel In-situ post-processing. In: Recent Advances in the Message Passing Interface, R. Keller, E. Gabriel, M. Resch, J. Dongarra, (eds.), Springer Berlin / Heidelberg, 2010, vol. 6305 of Lecture Notes in Computer Science (2010) 62-71.

[6] J. Biddiscombe, J. Soumagne, G. Oger, D. Guibert, J-G. Piccinali, “Parallel computational steering and analysis for HPC applications using a ParaView Interface and the HDF5 DSM Virtual File Driver. EGPGV 2011, 91-100.

[7] ParaView at the Large-Scale Data Analysis and Visualization Symposium, Kitware blog: https://blog.kitware.com/paraview-at-the-large-scale-data-analysis-and-visualization-ldav-symposium/

[8] ICARUS ParaView plug-in, Project: https://hpcforge.org/projects/icarus/

[9] PraView CoProcessing: https://www.paraview.org/Wiki/CoProcessing

[10] Visit Tutorial In-Situ: http://visitusers.org/index.php?title=VisIt-tutorial-in-situ

[11] Pl. Borovska, D. Ivanova, Code optimization and performance analysis of oceanographic software package NEMO for GPGPU systems, 2014 International Conference on Circuits, Systems, Signal Processing, Communications and Computers, Venice, Italy, ISBN: 978-1-61804-228-6, pp. 156-16.