PARALLELIZATION OF THE FVCOM COASTAL OCEAN MODEL

Geoffrey W. Cowles

DEPARTMENT OF FISHERIES OCEANOGRAPHY,
SCHOOL FOR MARINE SCIENCE AND TECHNOLOGY,
UNIVERSITY OF MASSACHUSETTS-DARTMOUTH,
(GCOWLES@UMASSD.EDU)

Abstract

The Finite Volume Coastal Ocean Model (FVCOM) is a publicly available software package for simulation of ocean processes in coastal areas. The unstructured grid approach used in the model is highly advantageous for resolving dynamics in regions with complex shorelines such as estuaries, embayments, and archipelagos. A growing user community and a demand for large-scale, high resolution simulations has driven the need for the implementation of a portable and efficient parallelization of the FVCOM core code. The triangular grid approach used in FVCOM precludes the utilization of schemes used previously in the parallelization of popular structured grid ocean models. This paper describes recent work on a SPMD parallelization of FVCOM. The METIS partitioning libraries are employed to decompose the domain. Parallel operations are programmed with the Message Passing Interface (MPI) standard interface. Updates for flow quantities near the interprocessor domain boundaries are performed using a mixture of halo and flux summation approaches to minimize communication overhead. Evaluation of the implementation efficiency is made on machines comprising several parallel architectures and interconnect types. The implementation is found to scale well on medium-sized (~ 256 processor) clusters. An execution time model is developed to expose bottlenecks and extrapolate the performance of FVCOM to increasingly available large MPP machines. Application to a model of water circulation in the Gulf of Maine shows that the parallelized code greatly increases the capabilities of the original core scheme by extending practical model simulation timescales and spatial resolution.

Key words: ocean model, parallelization, MPI, performance model, unstructured grid, FVCOM

The International Journal of High Performance Computing Applications,
Volume 22, No. 2, Summer 2008, pp. 177–193
DOI: 10.1177/1094342007083804

Figures 1, 4–7, 9–16 appear in color online: http://hpc.sagepub.com

1 Introduction

The Finite Volume Coastal Ocean Model (FVCOM) is a computational fluid dynamics code with application to the study of geophysical flows in coastal regions (Chen, Liu and Beardsley 2003). A solution of the hydrostatic primitive equations (HPE) is obtained on unstructured triangular grids with a finite-volume flux discretization and explicit time-stepping integration. The model is readily applied to the simulation of processes in coastal regions, including estuarine zones, shelf break fronts and tidal mixing fronts.

Recently, an effort has been undertaken to apply FVCOM to a high resolution study of the circulation in the Gulf of Maine region over decadal time scales. This study will provide the physical fields that can help to understand the dependency of biological fields on annual variations in the external forcing (e.g. wind, heat flux). Simulations of long-term circulation such as this can help provide links between the state of regional ecosystems and future climate change. A second burgeoning area of application is the development of coupled ocean/atmospheric forecasting tools for coastal regions, e.g. Aikman et al. (1996), Blumberg, Khan and St. John (1999) and Xue et al. (2005). These tools can provide pertinent information about the coastal ocean environment for fisherman and mariners, realtime trajectory information for the Coast Guard, estimated dispersal of harmful pollutants following a spill, as well as predictions of storm surge useful for coordinating emergency preparations.

Both of these application areas can demand significant computational resources because of the high grid density required to resolve dominant physical processes in the coastal area. In addition, forecasting tools face a practical integration time constraint. The runtime for these high resolution models on modern desktop machines can be extreme. For example, the current FVCOM model used to study the interannual variability of water mass transport in the Gulf of Maine region requires roughly six months of compute time to produce one year of simulated data on a desktop machine. In order to reduce runtime, the resolution would have to be significantly decreased, and the dominant processes affecting the variability would be left unresolved.

In order to ameliorate the issue of impractical compute time for proposed research projects, a parallelization of the FVCOM core code was implemented. The objectives of the parallelization were threefold: achieve good scalability on low cost distributed computing platforms, maintain portability across parallel architectures, and utilize an implementation that is relatively transparent to the user. These objectives were primarily driven by target user needs. FVCOM is a community model and the serial version enjoys use among a rapidly growing user group, most of whom have limited access to expensive shared memory supercomputers.

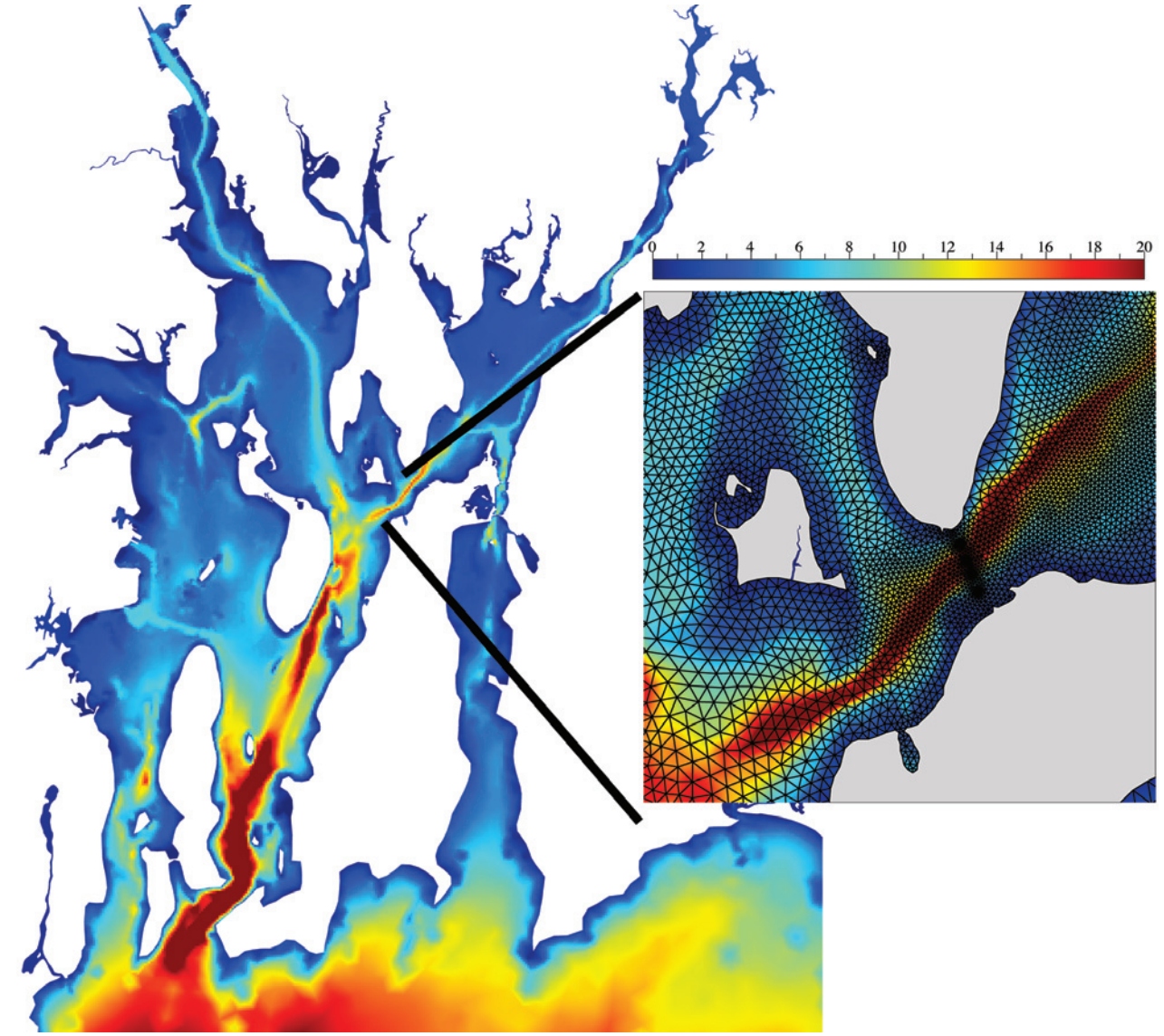

Fig. 1 Surface grid and bathymetry (m): Narragansett Bay, Rhode Island (detail: Hog Island).

Parallelization implementations have been made in several community global and regional ocean models, including MICOM, ROMS, MOM, and more recently POM, and SUNTANS (Bleck et al. 1995; Beare and Stevens 1997; Boukas et al. 1999; Wang et al. 2005; Fringer, Gerritsen and Street 2006). This paper focuses on the design of an implementation specific to the unstructured grid model FVCOM and a detailed analysis of the resulting performance using an execution time model.

To achieve our parallelization objectives as outlined above, an explicit Single Program Multiple Data (SPMD) parallelization method was implemented. Due to the complex nature of grid geometries in coastal and estuarine applications, as illustrated in Figure 1, a high quality partitioning scheme is necessary for good scalability. Interprocessor communication is performed using the Message Passing Interface (MPI) standard (MPI Forum 1993). The majority of the communication uses non-blocking sends and receives. The major parallel constructs, including exchanges of quantities across interprocessor boundaries and global collection of data for output are encapsulated in subroutines which shield the users from the MPI calls.

Benchmarks were performed for the completed code on several parallel machines comprising both distributed and shared memory architectures. The scalability shows that the implementation is efficient for a typical FVCOM model application. The resulting code extends greatly the potential of FVCOM to the high resolution simulation of long term circulation in coastal regions.

2 FVCOM Model Description

In an explicitly programmed parallelization, the timing and volume of interprocessor communication is dependent on the exact formulation of the time stepping and spatial discretization of the model. A brief description of FVCOM is provided here.

2.1 Mathematical Models

FVCOM solves the hydrostatic primitive equations (HPE) which consist of time dependent advection-diffusion equations for the two horizontal components of the three-dimensional velocity u and v, the scalar fields of temperature T and salinity s, and an advection equation for the height of the water surface ζ. Here, horizontal refers to a plane tangent to the Earth's surface and is normally cast in Cartesian coordinates x and y. Solution of these prognostic equations for u, v, T, s and ζ in addition to diagnostic equations for density ρ, pressure P, and vertical velocity w form the core of FVCOM model. In order to better resolve gradients in the fast-varying bottom topography of the coastal ocean, the equations are recast in a terrain following the σ coordinate system (Philips 1957). The equations are advanced in time using an explicit mode-splitting approach (Madala and Piacsek 1977; Simons 1974). In this approach, the barotropic (vertically independent) mode is integrated separately from the fully three-dimensional baroclinic motion using smaller time steps. To construct the barotropic mode, the equations for horizontal velocity are vertically averaged to give advection-diffusion equations for the vertically averaged velocities $\bar{u}$ and $\bar{v}$.

2.2 Spatial Discretization

Following the general procedures used in the finite-volume formulation, the governing differential equations are integrated over an arbitrary volume V and the divergence theorem is applied to transform the spatial derivatives to spatial fluxes.

The horizontal fluxes for the barotropic mode and baroclinic mode are discretized using a second-order accurate formulation (Kobayashi, Pereira and Pereira 1999). In order to improve the consistency and conservation properties of the scheme, a staggered-mesh is used

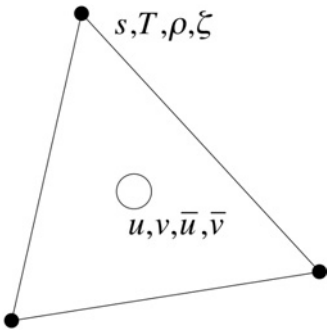

Fig. 2 Location of primary variables in the horizontal grid.

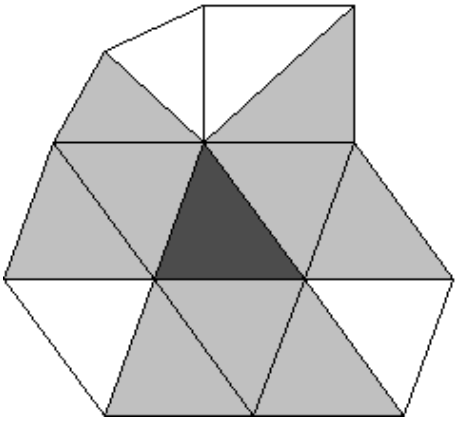

Fig. 3 Elements required for momentum update (*dark shade*: element control volume; *light shade*: elements in momentum stencil; *transparent*: elements not used in update).

as shown in Figure 2 (Arakawa and Lamb 1977). The scalar quantities T, s and ρ are stored at the nodal points and the velocity quantities are stored at the element centroid. This generates the necessity for two types of stencils for flux formulation: an element-based stencil for calculation of the flux through element edges and a node-based stencil for calculating fluxes through internal edges. Figure 3 shows the element-based stencil used in the calculation of horizontal fluxes in the baroclinic and barotropic velocity equations. The flux stencil for the advective and diffusive fluxes of the scalar quantities is shown in Figure 4. In both stencils, the required data is very local to the element to be updated. This choice of compact spatial discretization scheme benefits the parallelization efficiency by reducing the required volume of

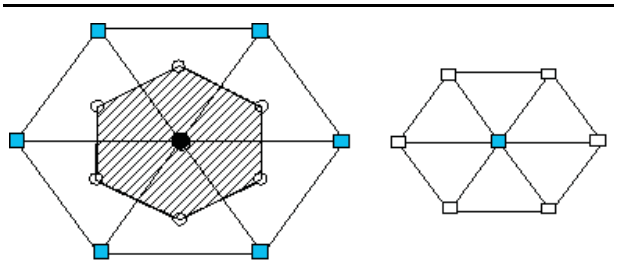

Fig. 4 Nodes required for scalar update. Left (*dark node*: node to be updated; *hashed area*: control volume; *filled rectangle*: primary stencil nodes required for flux computations, center of secondary stencil. Right (secondary stencil for computing data for primary stencil nodes).

interprocessor communication. The setup of this communication is dependent on the specifics of the numerical flux implementation and is discussed in Section 4.

2.3 Time Stepping

The barotropic mode is integrated in time using a second-order accurate four-stage explicit Runge-Kutta (ERK) scheme. Following the update of the primary barotropic mode variables $\bar{u}$, $\bar{v}$ and ζ, an adjustment is performed to the baroclinic velocity field u, v so that its subsequent vertical average is consistent with the barotropic field. The three-dimensional baroclinic mode is then integrated one time step using a forward-Euler time-stepping scheme resulting in updated quantities for the baroclinic velocity field components u, v, w, temperature T, salinity s, and density ρ. In a typical FVCOM application, the barotropic mode will be integrated using ten subiterations for each time step. A skeleton of the core solution process is as follows:

Serial Code Skeleton

```
initialize model
DO i=1,Nsteps (baroclinic mode loop)
    DO i=1,10 (barotropic mode loop)
        DO i=1,4 (Runge-Kutta loop)
            update(ζ, ū, v̄)
        END DO
    END DO
    adjust(u, v with ū, v̄)
    calc_fluxes(T, s)
    update(T, s, ρ)
    calc_fluxes(u, v)
    update(u, v)
END DO
output results
```

The vertical diffusion terms in the three-dimensional velocity scalar equations are integrated using a fully implicit Crank-Nicholson scheme and are solved using a Thomas algorithm. The computational work is small compared with that required to integrate the horizontal terms.

3 Domain Decomposition

A domain decomposition method is used to divide the computational work among processors in the parallelized FVCOM code. This decomposition is performed in the horizontal and not in the vertical. This is due to the difference in the discretization of horizontal and vertical fluxes. At each (x, y) location in the horizontal, data at all σ layers are assigned to a given processor. The explicit horizontal spatial flux discretization provides relatively compact data dependence which minimizes communication overhead. Data in vertical layers is highly interdependent because of the implicit nature of the vertical (σ) discretization. Decomposition in the vertical would incur larger interprocessor communication costs.

With an explicit time integration scheme, the volume of communication is proportional to the interprocessor boundary edge length in our two-dimensional decomposition. Thus a partitioning scheme that contributes to parallelization efficiency will create subdomains of roughly equal size while minimizing this length. With a structured grid, achieving a good load balance is fairly straightforward; the mesh is divided into equal size blocks along computational coordinates. With unstructured grids, however, the task becomes more difficult since the mesh consists of arbitrarily connected regular elements and the domain itself can be multiply connected. For coastal ocean applications with highly irregular coastlines, the mesh topology can be very complex. For this work, the domain is decomposed using the METIS graph partitioning software (Karypis and Kumar 1998a, 1998b). METIS is easily interfaced with FVCOM, produces partitions with low edge-cut, and executes in negligible wall clock time. Figure 5 displays the domain decomposition of a mesh used for the study of the circulation in the Satilla River in Georgia in the Southeastern United States. The multiple branches of the estuary demonstrate the challenge of achieving high quality partitions and motivate the need for using a graph-based partitioning algorithm.

4 Interprocessor Data Exchange

In a domain decomposition method, interprocessor communication is necessary to update quantities on and near subdomain boundaries. As discussed in Section 2.2, FVCOM uses element- (Figure 3) and node-based control volumes (Figure 4) for computing momentum and scalar terms respectively. In addition, the arbitrary connectivity of the

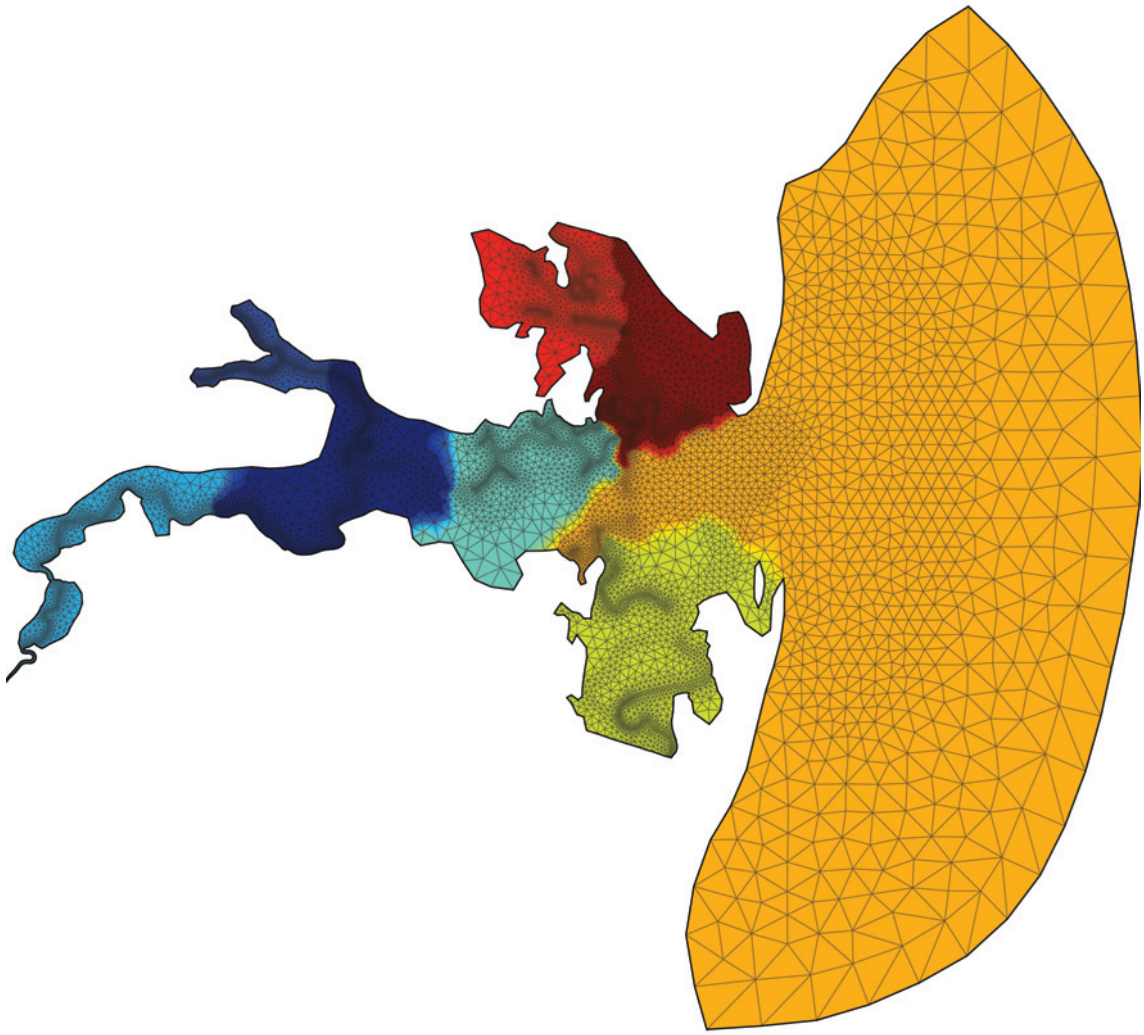

Fig. 5 Accumulation of horizontal flux of scalar quantity on interprocessor boundary node: partial fluxes are computed by processor A (*hashed zone*) and by processor B (*gridded zone*) and added with an MPI call (*arrowed line*: interprocessor boundary).

unstructured mesh necessitates a more complex algorithm for mapping interprocessor communication patterns.

In this work, the boundary fluxes of element-based variables u, v are computed using a traditional halo approach as shown in Figure 6. The compact nature of the discretization discussed in Section 2.2 requires only a single layer of element data along the interprocessor boundary to be exchanged. For scalar quantity updates, the discretization of diffusion terms generates a larger stencil of dependency. The update of nodes residing on the interprocessor boundary requires data that resides two layers deep in the neighboring processor's domain. Setting up this exchange in an arbitrarily decomposed unstructured grid can be tedious. To alleviate these issues with scalar quantities, a flux accumulation approach is used in place of the traditional halo approach. The scalar control volume is actually just a collection of fluxes over the internal edges of elements surrounding a node. If a node is an interprocessor boundary node (IBN), the internal edges will be divided between (or among) the domains sharing the node. Thus partial fluxes for the update of scalar quantities can be uniquely computed by the adjoining processors and summed at the node point using MPI calls. The domains which share the IBN will compute the same

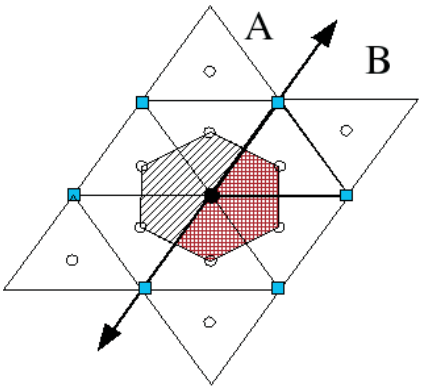

Fig. 6 Halo elements required for momentum update (*dark*: element to be updated; *hashed*: halo elements needed for update of dark triangle; *light*: additional elements in the halo of processor B; *arrowed line*: interprocessor boundary).

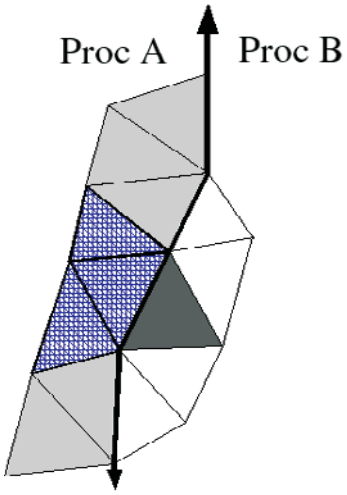

Fig. 7 8-way partitioning of Satilla River estuary model using METIS.

total flux and thus compute the same updated scalar value. The partial flux calculation is shown graphically in Figure 7. The communication cost of the flux collection is proportional to the number of IBNs on the domain boundary. Computation of the partial fluxes does not require two layers of data from the neighboring processor and thus communication costs and setup complication are reduced.

Exchanges of dynamic quantities are made after they are updated to maintain exactness with the serial code. Flux collections for scalar quantities are made between flux computation and update. The skeleton of the parallelized code is shown below for clarity. Parallel procedures are shown in bold. The goal was to minimize modification of the serial code while maintaining exactness of the output. Preprocessing directives are used to enable removal of all MPI routines before compilation so that users who did not have the MPI libraries available could compile FVCOM.

Parallelized Code Skeleton

initialize model
DO i=1,Nsteps (baroclinic mode loop)
  DO i=1,10 (barotropic mode loop)
    DO i=1,4 (Runge-Kutta loop)
      update(η, $\bar{u}$, $\bar{v}$)
      exchange(ζ, $\bar{u}$, $\bar{v}$)
    END DO
  END DO
  adjust(u, v with $\bar{u}$, $\bar{v}$)
  calc_fluxes(T, s)
  collect_fluxes(T, s)
  update(T, s, ρ)
  calc_fluxes(u, v)
  update(u, v)
  exchange(T, s, ρ, u, v)
END DO
output results

5 Parallel Performance Results

The performance of the parallelized FVCOM code has been evaluated on a variety of multiprocessor machines (Table 1). The benchmark case consists of a pared down version of a model setup currently used to study the circulation of water in the Gulf of Maine region in which the complex boundary forcing (e.g. wind stress, heat flux, and freshwater river runoff) has been removed and no assimilation of observed data is made. The model was simplified to make it more compact and easily transferable to various machines for testing. The scalability results of the benchmark case reflect strongly the performance of the realistic model. The benchmark mesh contains 60 K triangular elements in the horizontal and is discretized vertically using 31 σ-levels for a total of $1.8e^6$ control volumes (Figures 8 and 9).

The primary metric for evaluating the efficiency of the parallelization implementation used in this work is the speedup $S_p(P) = T_s/T_p(P)$ where T_s is the compute time on a serial machine and $T_p(P)$ is the compute time of the parallel code on P processors. Ideal speedup is defined as $S_p = P$. Comparison of speedup on the test machines

Table 1
Benchmark machines.

Machine	Arch	Nodes/Procs	Proc	Network
Desktop	SMP	1/2	P4 3.2 GHz	–
Altix 3000	SMP	1/64	I2 1.3 GHz	NUMA Link
Myrinet cluster	Cluster	128/256	P4 3.06 GHz	Myri2K-D
Medium IB cluster	Cluster	128/256	EM64T P4 3.4 GHz	Infiniband 4X
Small GigE cluster	Cluster	21/84	Xeon Dual-Core 2.8 GHz	GigE
Small IB cluster	Cluster	15/60	Xeon Dual-Core 3.0 GHz	Infiniband 4X

shows that the Altix SMP machine gives the highest efficiency over its processor range (Figure 10). The machine exhibits linear speedup to 8 processors and a superlinear speedup up to 48. The superlinear measurement is due to cache effects. The large cache (3 MB) of the Itanium2 processor coupled with the ultra-low latency of the Numalink connection make this possible. The Infiniband-networked machines provide the best performance of the

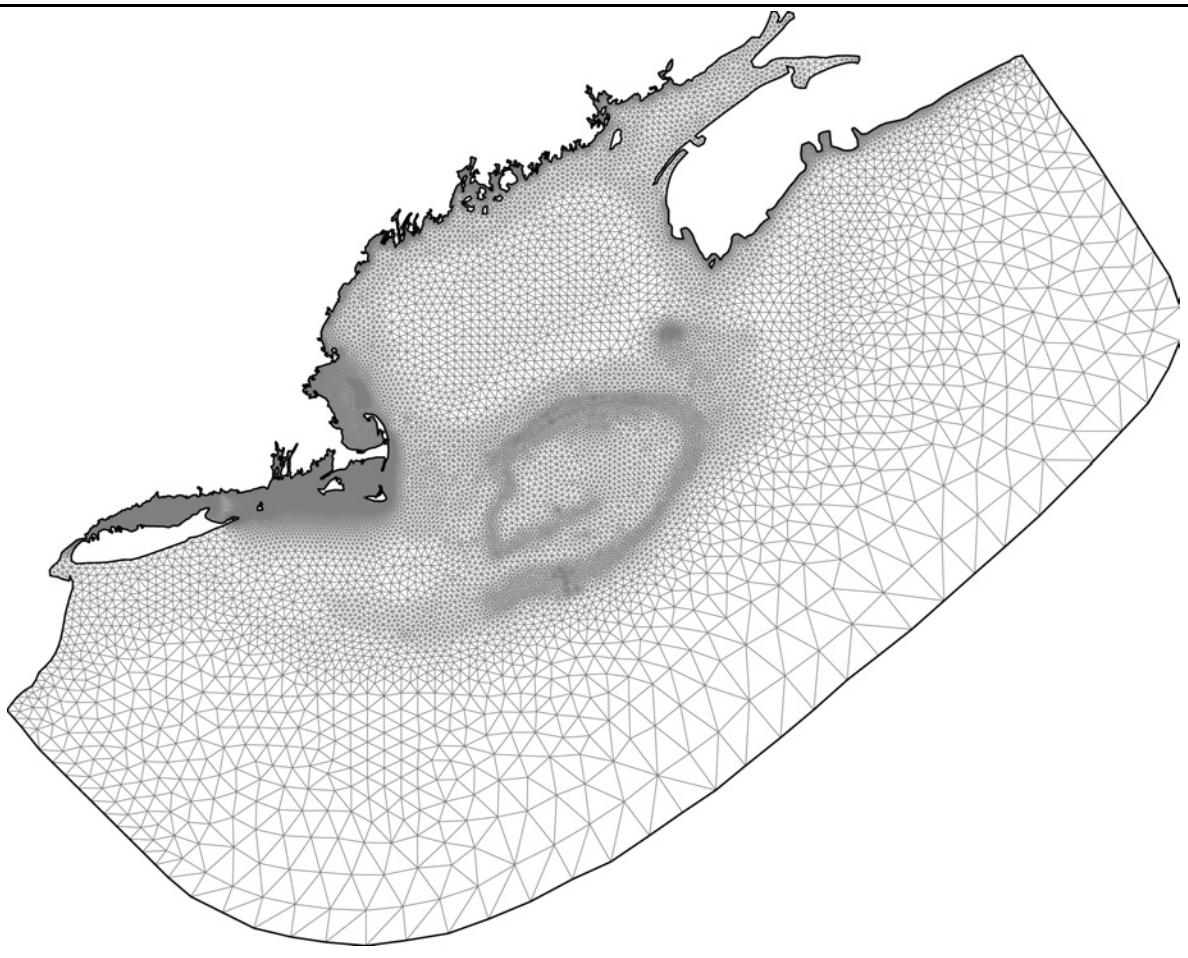

Fig. 8 Surface grid: FVCOM-GoM Model.

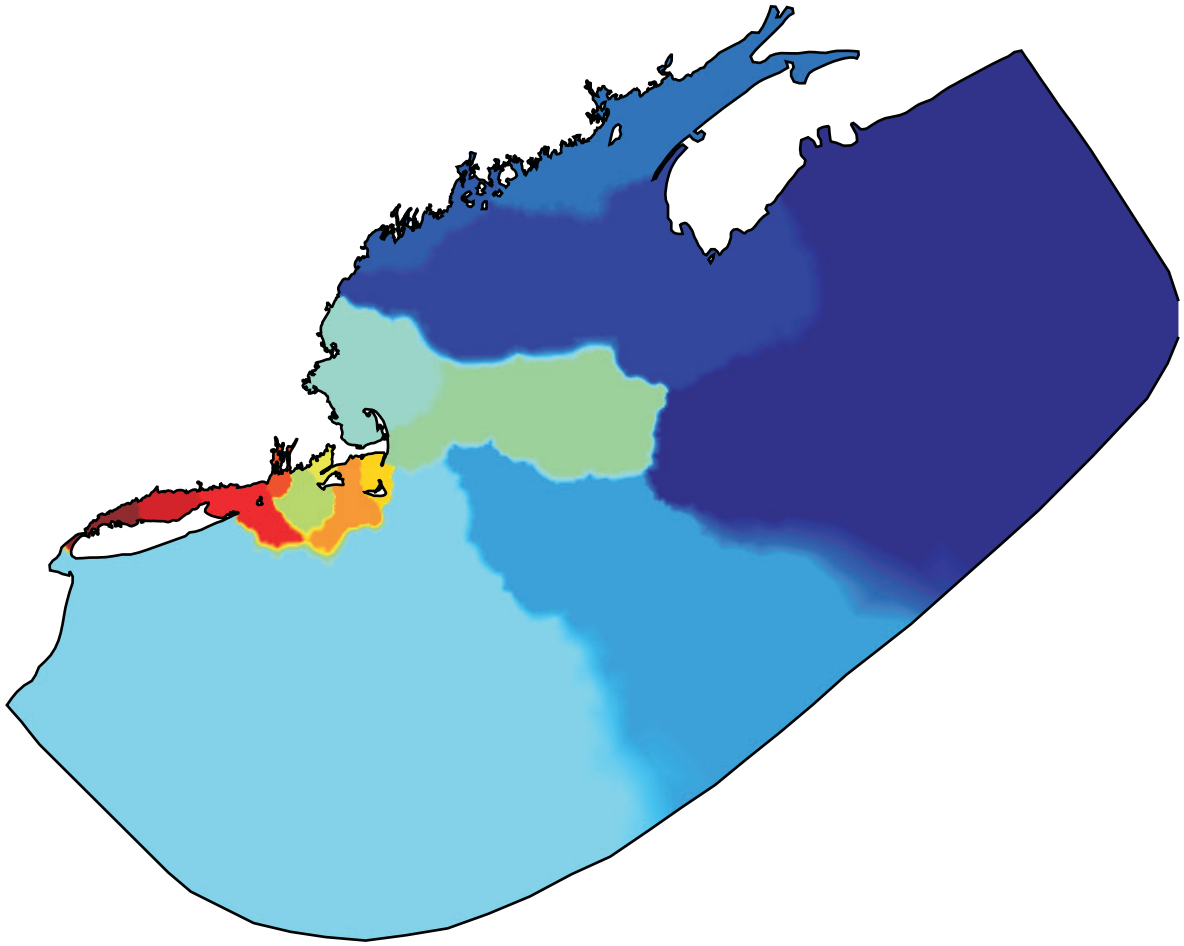

Fig. 9 8-way partitioning of FVCOM-GoM model using METIS.

clustered computers and the 10/100 Ethernet machine the worst. Most importantly for the FVCOM community, the parallelized code scales reasonably well on low cost computing platforms typical of marine science departments consisting of 4–8 dual processor nodes connected by a low cost commodity interconnect such as Gigabit Ethernet. However, a comparison of the small IB and small GigE cluster results indicate the performance boost of the high speed interconnect is worth the additional ~ 20% in cost per node for clusters larger than 16 cores. The Myrinet-based cluster gives slightly lower performance than the medium IB cluster and exhibits a strange kink in the speedup curve. The curve was reproducible but the time allotted for testing was not sufficient to determine the cause. The measured speedup on the Myrinet machine is $S_p = 99$ on 128 processors and $S_p = 155$ on 256. On the medium IB cluster, the speedups are $S_p = 115$ and $S_p = 172$ for 128 and 256 processors respectively. It should be noted that the definition of speedup does not include the effects of bus contention for the multi-processor/node based clusters. The baseline measurement T_s is a single processor measurement while all parallel measurements utilize all available processors/cores in a node. The speedup curve thus gives a lower bound for parallelization efficiency of FVCOM cluster machines with multiprocessor nodes.

Table 2 compares the execution time for the FVCOM Gulf of Maine model benchmark on a variety of machines. On modern parallel machines such as the Infiniband cluster, a substantial reduction in compute times is obtained. Simulations of the annual circulation in the Gulf of Maine have been reduced to less than one day of compute time.

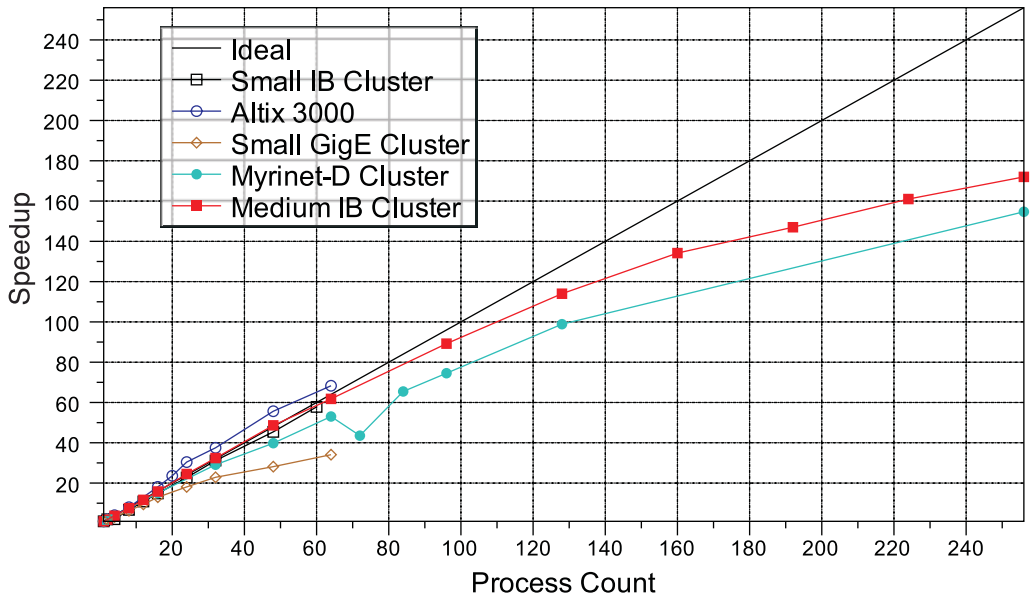

Fig. 10 Speedup for FVCOM-GoM on the test machines from Table 1.

Table 2
Runtimes for the FVCOM benchmark test case.

Machine	Cores	Secs/It	*Wall Time* / *Year*
Desktop	1	6.6	65 days
Small GigE cluster	64	.22	46 hours
Medium IB cluster	256	.034	8 hours

6 FVCOM Performance Model

Performance models are powerful tools for critical analyses of parallel programs (Kerbyson et al. 2001; Kerbyson and Jones 2005). They are helpful for locating bottlenecks in parallel computation, they can provide support in purchasing the most suitable computer equipment for a given code, and they can be used to perform virtual benchmarking and application tuning without wasting valuable CPU time. A performance model is applied to the parallelization implementation in FVCOM to examine more closely the factors that control the scalability of the algorithm and to extrapolate performance to the massive (> 1000 processors) parallel machines that are becoming increasingly available to investigators outside of traditional supercomputing disciplines.

6.1 Parameterization

The performance model parameterization of FVCOM uses an execution time approach:

$$T_{total} = T_{work} + T_{comm}, \tag{1}$$

where T_{total} is the total time required to perform an iteration of FVCOM using P processors, T_{work} is the time that a task spends performing the computation and T_{comm} is the time a task spends performing interprocessor communications. Idle time in FVCOM derives primarily from a non-uniform load balance and delays caused by asynchronous communication and is thus folded into T_{comm} and T_{work}. Work is parameterized using only problem size and hardware-specific rates and is computed using separate terms representing the integration of external and internal modes:

$$T^{e}_{work} = \alpha(P) t^{e}_{w} \frac{N_{\Omega}}{P} R_{mode} \tag{2}$$

$$T^{i}_{work} = \alpha(P) t^{i}_{w} \frac{N_{\Omega}}{P} N_{\sigma}. \tag{3}$$

Here, the time to integrate the external mode (T^{e}_{work}) is proportional to the maximum number of elements in the local surface grid in any partition $\left(\alpha \frac{N_{\Omega}}{P}\right)$ and work to inte-

grate the internal mode is proportional to the maximum number of three-dimensional prismatic cells in any partition $\left(\alpha\frac{N_\Omega}{P}N_\sigma\right)$ where N_Ω is number of elements in the horizontal grid, N_σ, the number of vertical layers in the domain, α, a load balance inefficiency factor ($\alpha \geq 1$) and P, the process count. The hardware parameters t_w^e and t_w^i are constant and are calibrated on a given platform using a least squares fit of timings from three or more model grids. A more complex relationship with problem size had been sought by incorporating cache effects, but no clear step function in T_{work} was discernible across a wide range of problem sizes. Due to the wide variety of data locations used in the discretization (Figure 2), without an associated programming effort to increase data locality, FVCOM does not make efficient use of cache. The load balance inefficiency factor (α) cannot be determined empirically but has been observed to be in the range $1 < \alpha < 1.05$ when using METIS. R_{mode} is the ratio of internal mode time step to external mode time step and is equal to the number of external mode subiterations in a time step. For most FVCOM applications: $5 \leq R_{mode} \leq 10$.

The communication time T_{comm} is more explicitly parameterized than T_{work} as the goal of this exercise is to examine aspects of the parallelization. For unstructured grids, an accurate representation of T_{comm} is difficult to achieve since directional sweeps and synchronous communication are not normally used and thus message sizes can vary substantially and the number of adjacent subdomains is not fixed. A switch between eager and rendezvous protocols in the non-blocking MPI sends/receives introduces nonlinearity in the relationship between message size and T_{comm} making an estimation based on total message size for a given processor less accurate. For this work, an approximate formulation for T_{comm} is developed using values which can be determined from empirical relations and network parameters. It includes important contribution from small message latency with N_{ney} neighbors as well as a contribution from the total message length from any given subdomain and is implemented as follows:

$$\begin{aligned} T^x_{comm} &= N_{ney}L_e && \text{if } M < M_{prot} \\ T^x_{comm} &= N_{ney}L_r + M\frac{P_{node}}{B} && \text{if } M \geq M_{prot}. \end{aligned} \quad (4)$$

Here L_e and L_r are the eager and rendezvous protocol latencies for a given network, M_{prot} is the message size (Bytes) at which a given MPI implementation switches protocols. Only the intranode network contention is parameterized. P_{node} is the number of cores utilizing a single network card. B is the bidirectional point to point bandwidth. Total communication cost is the sum over all exchanges, T^x_{comm}.

The message length M is dependent on the variable type and number of variables to be exchanged among processors.

$$\begin{aligned} M &= 2N_{\text{vars}}N_b && \textit{external mode element exchanges} \\ M &= N_{\text{vars}}N_b && \textit{external mode node exchange and flux summations} \\ M &= 2N_{\text{vars}}N_bN && \textit{internal mode element exchanges} \\ M &= N_{\text{vars}}N_bN && \textit{internal mode node exchange and flux summations} \end{aligned} \quad (5)$$

The message length is dependent on the maximum number of boundary elements for any partitions (N_b) which is parameterized using the following empirical formula:

$$\begin{aligned} N_b &= \sqrt{N_\Omega} && P \leq 8 \\ N_b &= 4\frac{\sqrt{N_\Omega}}{\sqrt{P}} && P > 8. \end{aligned} \quad (6)$$

A comparison between parameterized and actual maximum N_b for the Gulf of Maine domain (Figure 8) is shown in Figure 11. At lower processor counts, the edge length is highly dependent on the aspect ratio of the mesh graph and thus the empirical formulation can be quite inaccurate. For larger processor counts (> 32), the relation gives a good approximation and precludes the need to utilize explicit boundary length observations. There is a small additional cost associated with the packing and unpacking of data in the interprocessor exchange. The storage of three-dimensional variables in FVCOM is made using two dimensional arrays of size $[N_\Omega + N_{halo}, N_\sigma]$ and thus the halos are not stored contiguously in memory. This additional cost is not included in the parameterization.

6.2 Calibration and Comparison with Model Results

The performance model is first calibrated using an exact decomposition of a triangulated Cartesian domain into $\sqrt{P} \times \sqrt{P}$ blocks with N_b elements along each boundary edge. For this case the load balance is uniform ($\alpha = 1$) and the communication size and number of neighbors for each subdomain are known and are directly input into the performance model. Comparison of modeled and measured runtimes for three refinements (32 K, 131 K, and 524 K cells) of this idealized grid on the medium size Infiniband cluster are shown in Figure 12. Hardware parameter values used for the model are provided in Table 3. The mode-splitting ratio is set to a typical value of $R_{mode} = 10$. For this range of model size and processor

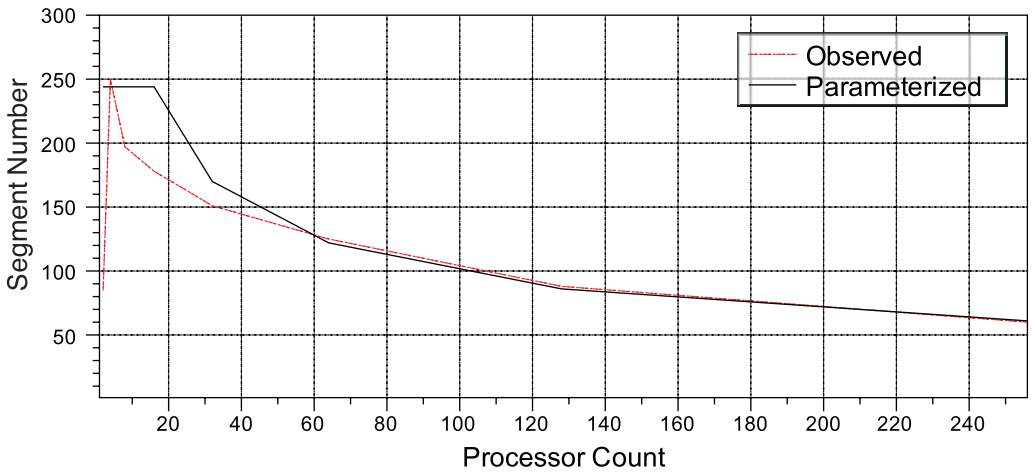

Fig. 11 Comparison of parameterized (equation 6) and actual maximum subdomain edge counts (N_b) from partitions of the FVCOM benchmark domain (Figure 8).

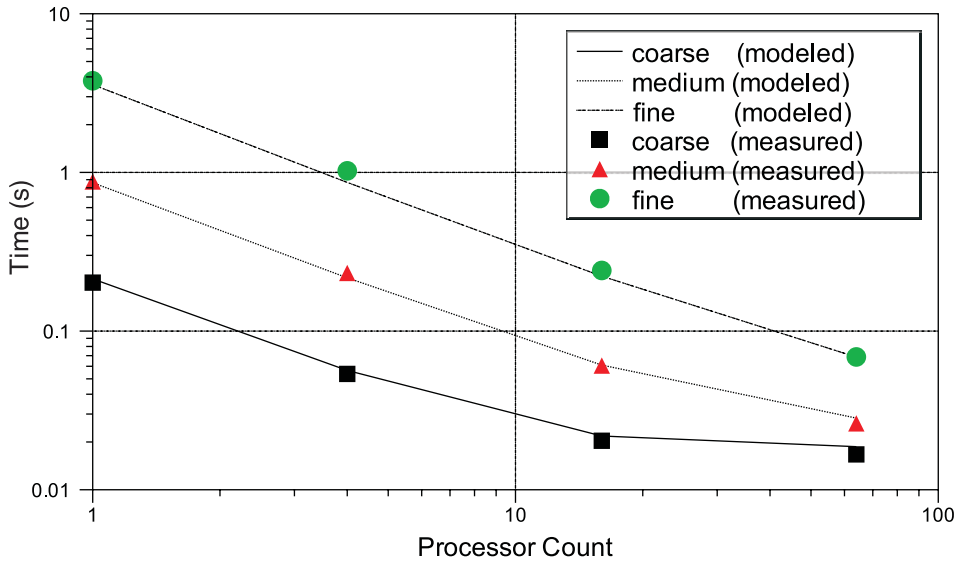

Fig. 12 Measured and modeled iteration time for idealized grid.

Table 3
Performance model hardware parameters.

Machine	t_w^e $\frac{\mu s}{\Omega}$	t_w^i $\frac{\mu s}{cell}$	B $\frac{Bytes}{\mu s}$	P_{proc} –	L_e μs	L_r μs	M_{prot} KB
Medium IB	1.35	3.31	900	2	9	11.5	2
Small GigE cluster	1.49	3.37	100	4	70	70	2

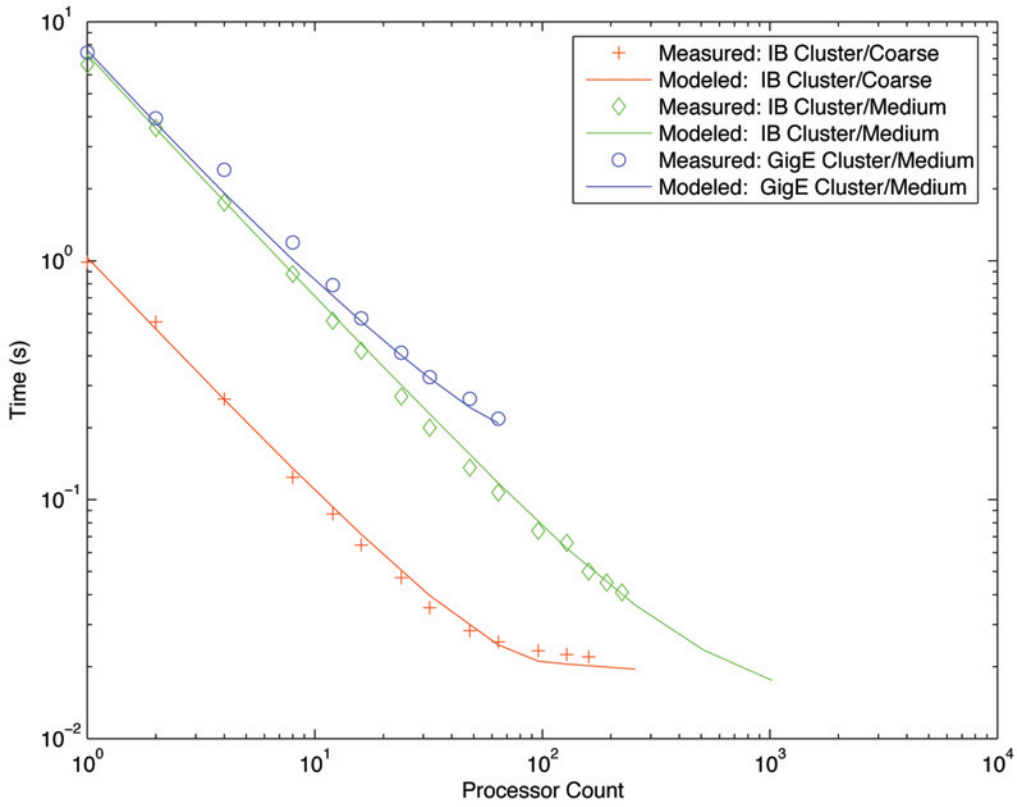

Fig. 13 Measured and modeled iteration time for GoM benchmark case and coarse case on two clusters.

count, the execution times from the performance model compare well with those observed.

The performance model is also applied to realistic model applications. Both the benchmark test case and a coarsened version of this grid are run on available medium Infiniband and small GigE clusters. The problem size for the coarse mesh (225 K cells) is considerably smaller than the benchmark problem size (1.8 M cells) and is selected in order to include a model with significantly poorer observed scalability. Perfect load balance ($\alpha = 1$) is assumed and the edge length is estimated from equation (6). Hardware parameters are shown in Table 3. Measured and modeled execution times (Figure 13) compare well. The sharp reduction in efficiency in the coarse case is captured, although with an apparent lag in process count. It was discovered that better agreement could be obtained if measured neighbor counts (N_{ney}) were used. However, in the desire to eliminate any model-specific information, beyond problem size, the explicit neighbor count was not included in the final performance model.

6.3 Implications

Having demonstrated the capability of generating accurate execution times for several problem sizes on two different clusters, the performance model can be used to examine what the scalability of the parallel FVCOM code might be for a large cluster. The prospective machine is a 1024 processor version of the medium Infiniband cluster from Table 1. Fully non-blocking communication is assumed. Performance for both the current GoM benchmark (1.8 M cells) and a larger model (30 M cells) are calculated (Figure 14). The larger model is representative of the next generation model for the Gulf of Maine region and thus the prospective performance is of direct interest to the FVCOM research group. The benchmark test case achieves a speedup of 400 on 1024 processors, while the next generation problem size scales much better to 900 on 1024 processors.

The cause of the reduction of efficiency of the benchmark case at high processor counts can be deduced from the performance model. The model results elucidate a

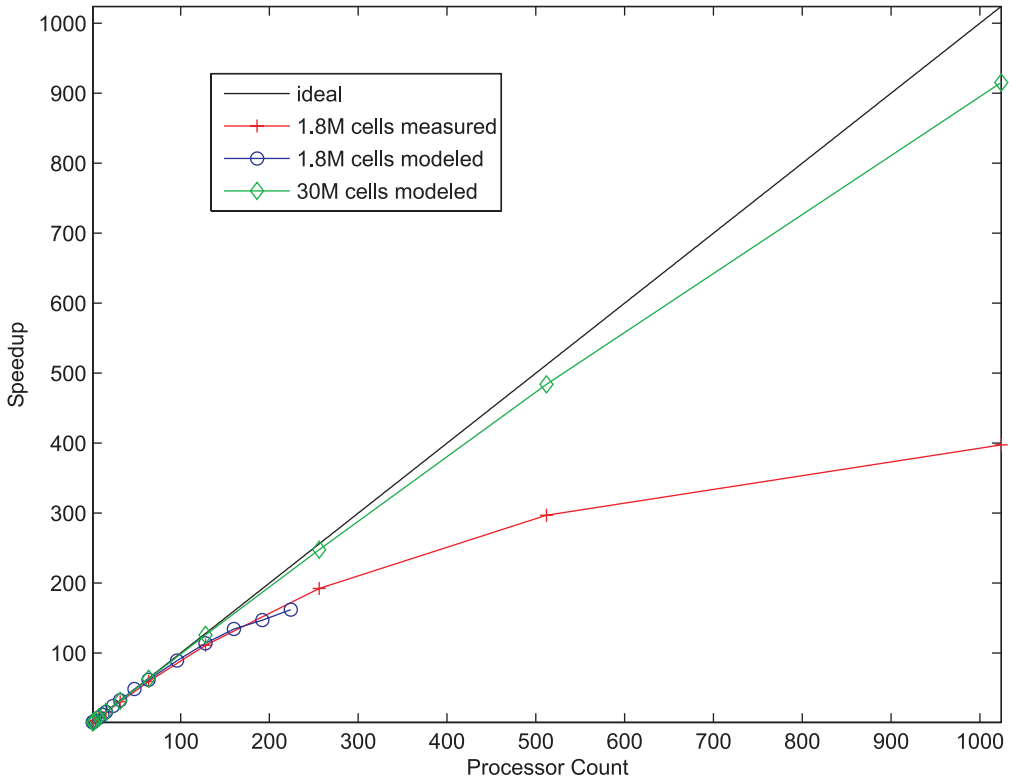

Fig. 14 Measured and modeled Speedup for GoM grid.

rather strong dependence on latency. Whereas much of the parallelization effort was focused on minimizing total message size by using graph-based partitioning strategies (Section 3) and stencil-specific communication (Section 4), the minimization of message counts was never stressed. The external mode communication begins to dominate the execution time because of the large number of data exchanges (~ 120) which must be performed during each FVCOM iteration (Figure 15). For a serial run, the execution time required for the external mode is nearly negligible (~ 10%).

The observed increase in execution time dedicated to the external mode at large processor counts represents a loss of efficiency in the integration scheme. This raises the question or whether this might make other formulations, such as those based on implicit schemes, more competitive. In typical FVCOM applications, the explicit split-mode provides an effective ten-fold boost in maximum stable time step over a nominal explicit approach to stepping the HPE system as the cost of integrating the barotropic mode is negligible. This enables the explicit scheme to be competitive with implicit schemes that are not constrained by a cell-based CFL criteria. For a given application on a serial computer, time to execute the explicit scheme will scale as $1/\delta x^4$ where δx represents the nominal mesh spacing. Implicit solvers can scale as $1/\delta x^3$ although this is only possible using idealized convergence schemes. The differing functional dependencies on problem size for these two classes of schemes imply a crossover point for a given mesh spacing in a given application at which the implicit scheme is less expensive. The mode-splitting acts to move this crossover point to higher resolution.

As an example, the benchmark test case discussed in this paper has a maximal external mode time step of 30 seconds computed from

$$\Delta t_{ext} = \min_i \left(\frac{\Delta x_i}{\sqrt{g H_i} + u_i} \right) \tag{7}$$

where H_i is the water depth and u_i is the local vertically averaged velocity in element Ω_i. The internal time step for this case is limited by mode-splitting errors and is

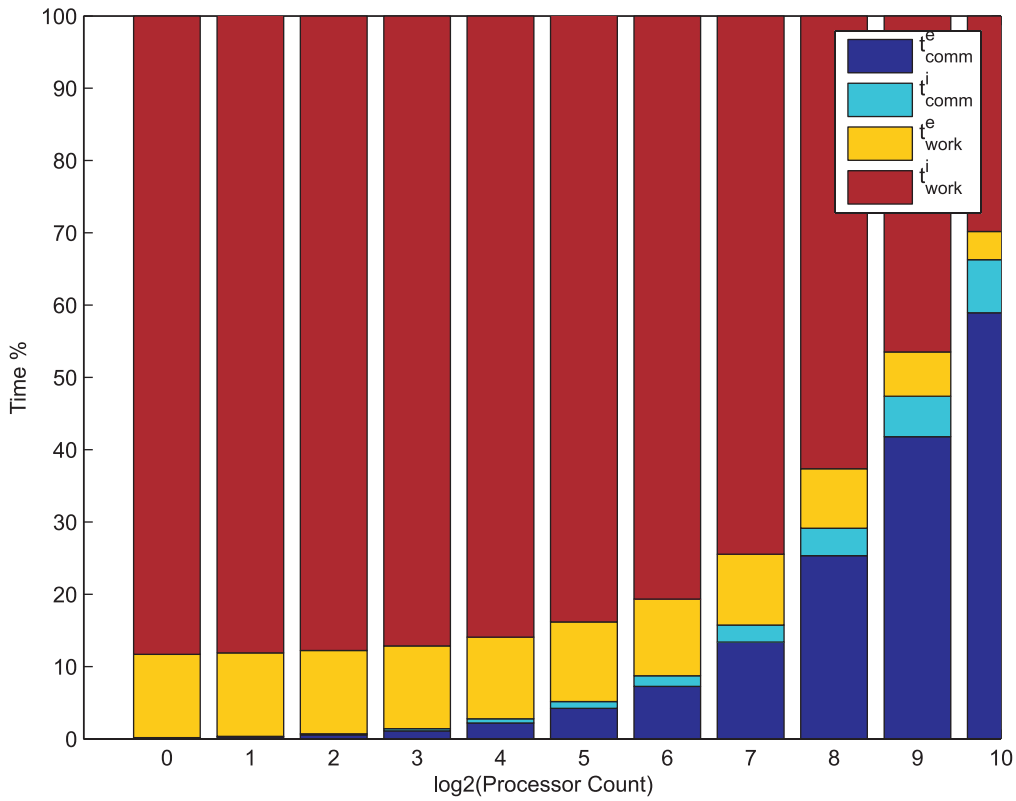

Fig. 15 Execution time fractions.

normally set to a conservative value of $10\Delta t_{ext}$ = 300 s. The minimum period of motion for the forcing in this simulation is the M_2 or twice-daily tide which has a period of T_{M_2} = 124.2 hours. A reasonable maximum time step for an implicit simulation would be roughly $T_{M_2}/48$ ~ 900 s. Thus the ratio of implicit to explicit time step $R_{i/e}$ would be approximately 3. It is doubtful that an implicit solver can integrate one time step at the rough cost of three iterations of the split-explicit scheme (~ 6 three-dimensional residual evaluations) and thus unlikely to be cost-competitive for this particular application. One implicit scheme, dual-time stepping, advances hyperbolic systems with a conservative cost of 150 residual evaluations per time step using preconditioning and multigrid convergence acceleration (Peyret 1976; Alonso 1997; Cowles 2001). Thus the transition in solver efficiency from the split-explicit method in FVCOM to this example-implicit scheme occurs when $R_{i/e} \geq 75$. For bay- and estuary-scale applications, due to the O(10 m) grid scale, the time step can be quite small for split-explicit schemes. In the high-resolution FVCOM model of Narragansett Bay, Rhode Island, the internal time step is five seconds (Zhao, Chen and Cowles 2006). The ratio of implicit to explicit time step in this case would be $R_{i/e}$ ~ 150, giving an implicit scheme such as dual time-stepping the likely advantage. Thus, an estimate of the crossover point where implicit integration schemes are likely to be more cost effective than explicit split-mode techniques occurs between gulf-scale (O(1000 m) resolution at 100 m depth) and estuary-scale (O(10 m) resolution at 10 m depth) applications.

For multiprocessor runs, the position of the transition point is affected by the relative scalability of the two methods. As discussed above, T_{work} is equitable when the number of three-dimensional flux evaluations is the same, or when $R_{i/e}$ ~ 75. As elucidated by the FVCOM performance model, T_{comm} is dominated by the latency required for the barotropic mode data exchanges. Even if FVCOM is modified to minimize the number of data exchanges, the ratio of barotropic mode exchanges to three-dimensional flux evaluations will remain fixed at twenty. In the dual-time stepping scheme, there is a one to one correspondence between data exchanges and three-

dimensional flux evaluations. Thus for an application where T_{work} is equitable, the fraction of T_{comm} derived from network latency will be a factor of twenty larger in the split-explicit scheme than in the hypothetical dual time-stepping scheme, indicating that at high processor count, there is a reduced advantage in using the split-mode explicit integration scheme.

7 Example Application

Hindcasts of water circulation in the Gulf of Maine for the period from 1995 to the present have been completed using the parallelized FVCOM model. Sea surface temperature and surface current vectors for August 5, 1996 are shown in Figure 16. The mesh for this model is based on the benchmark mesh (Figure 8) with reduced resolution near the coast. The mesh consists of 30 K elements in the surface grid and 30 vertical layers and ranges from 2 km resolution near the coast to 50 km at the open boundary. The model is driven using surface winds and heat flux from a configuration of the 5th generation Mesoscale Meteorological model (MM5) for the Gulf of Maine (Dudhia et al. 2003). Fresh water input from seven major rivers is also included. Nudging-based

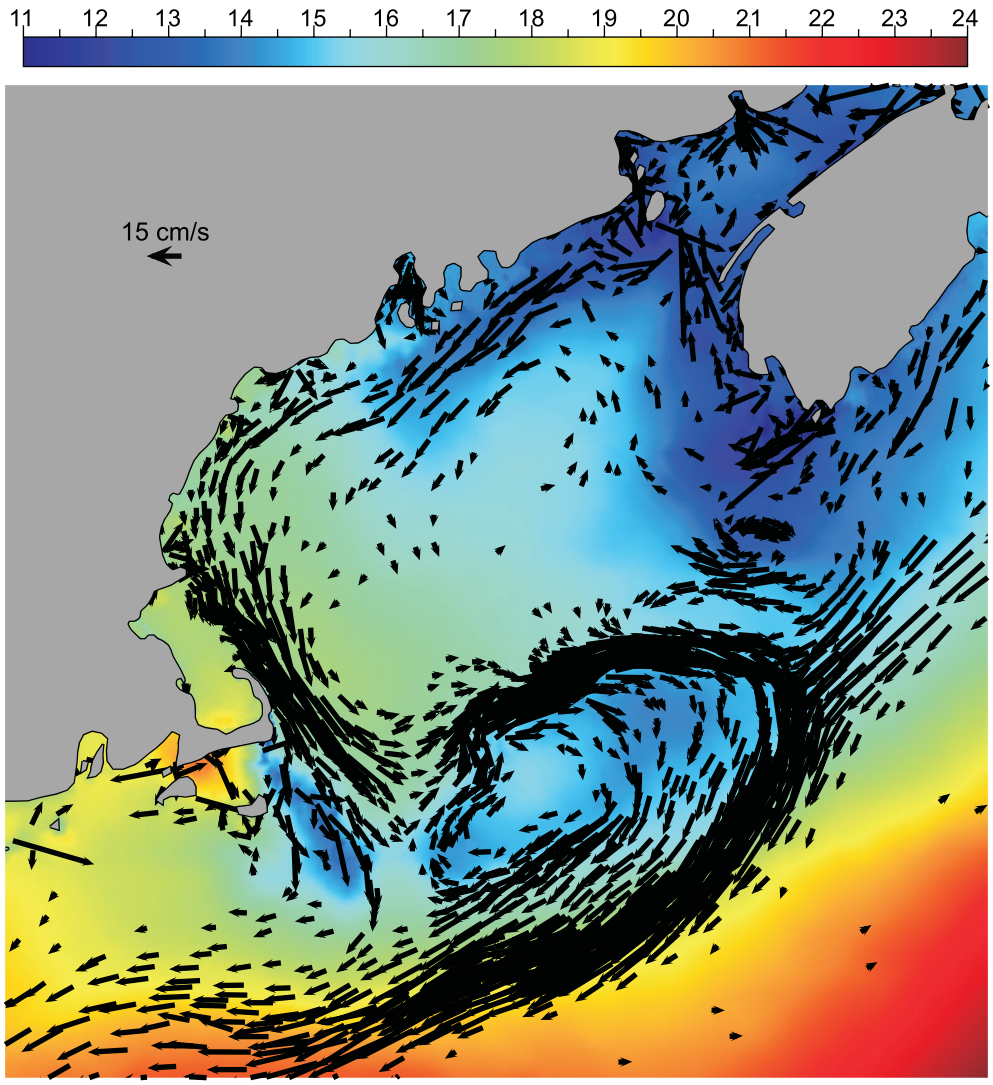

Fig. 16 FVCOM-computed surface currents and sea surface temperature (°C) for August 5, 1996 in the Gulf of Maine and Georges Bank.

data assimilation techniques are used to incorporate observations of ocean currents and ocean surface temperature to improve the accuracy of the model state. The results will be used in investigations of the effects of long term variability in external forcing on the water mass structure and the marine ecosystem in the Gulf of Maine. Future model improvements include the employment of more advanced data assimilation techniques based on the Ensemble Kalman Filter (Evensen 1994). Such modifications will require additional parallelization efforts.

The hindcast computations were made feasible by the success of the code parallelization effort. With a nominal use of 32 processors of the available cluster (medium IB, Table 1), a one year hindcast of this model requires around a week of wall clock time. Other applications enabled by the parallelized FVCOM code include a study of water exchange and tidal eddy formation in Narragansett Bay (Zhao, Chen and Cowles 2006), simulation of tides in the Arctic Ocean (Chen et al. 2006a), and an examination of detachment of the Changjiang River plume in the East China Sea (Chen et al. 2006b).

8 Conclusions

An efficient parallelization methodology was designed and implemented for the Finite Volume Coastal Ocean Model (FVCOM). The efficiency as measured on several modern multiprocessor platforms shows good scalability for a representative application. The critical data exchange between processors is programmed using non-blocking sends and receives. All message passing is coded in the MPI standard interface which is portable to a large variety of parallel machines. Interprocessor data exchange is reduced by using a halo approach for update of momentum quantities on the domain boundaries and a flux summation approach for update of scalar quantities on the domain boundaries. The efficiency as measured on several modern multiprocessor platforms shows good scalability for a representative application. An execution time model is developed to elucidate bottlenecks and extrapolate performance of future FVCOM applications with larger problem sizes. Reduction of efficiency in the benchmark case is primarily due to latency, indicating future effort to improve scalability should focus on reduction of message counts.

The parallelized code enables a significant increase in model grid density while maintaining practical compute times. This extends the range of application of the model to include accurate forecasting of large coastal regions and long term studies of regions where small scale processes are of interest. By exploiting the multi-teraflop potential of modern computing platforms, the parallelized FVCOM code can be utilized for a study of the variability of circulation and biological structure on decadal timescales in the Gulf of Maine region.

Acknowledgments

The author would like to thank the reviewers for their helpful suggestions that have done much to improve the paper. The FVCOM model is developed under the guidance of Dr. Changsheng Chen. Initial testing of the parallelization implementation was performed on the IBM P690 SMP system at the Arctic Region Supercomputing Center (ARSC, http://www.arsc.edu) in Fairbanks, Alaska. This research was supported by the National Aeronautics and Space Administration under Grants NAG13-02042 and NAG13-03021 and through the Massachusetts Marine Fisheries Institute under NOAA grants DOC/NOAA/NA04NMF4720332 and DOC/NOAA/NA05NMF4721131.

Author Biography

Geoff Cowles is a research scientist in the School for Marine Science and Technology at the University of Massachusetts-Dartmouth. Before joining SMAST in 2003, he was a postdoctoral fellow at the Ecole Polytechnique Fédérale de Lausanne in Switzerland, where he performed hydrodynamic analyses of America's Cup yachts for the Swiss America's Cup Challenge Alinghi using software developed during his doctoral work. He obtained his Ph.D. in 2001 from Princeton University in computational ship hydrodynamics. His current interests include large scale simulations of geophysical flows, data assimilation methods, and the use of multigrid algorithms to solve initial value problems in coastal oceanography.

References

Aikman, F. III, Mellor, G. L., Ezer, T., Sheinin., D., Chen, P., Breaker, L., Bosley, K., and Rao, D. B. (1996). Toward an operation nowcast/forecast system for the U.S. East Coast, *Modern Approach to Data Assimilation in Ocean Modeling, Elsevier Oceanography Series*, P. Malanotte-Rizzoli, (ed.), **62**: 347–376.

Alonso, J. J. (1997). *Parallel Computation of Unsteady and Aeroelastic Flows Using an Implicit Multigrid Driven Algorithm*, Ph.D. Thesis, Dept. of Mechanical and Aerospace Engineering, Princeton University.

Arakawa, A. and Lamb, V. R. (1977). Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics*, **17**: 173–265.

Beare, M. I. and Stevens, D. P. (1997). Optimisation of a parallel ocean general circulation model, *Annales Geophysicae*, **15**: 1369–1377.

Bleck, R., Dean, S., O'Keefe, M., and Sawdey, A. (1995). A comparison of data-parallel and message-passing versions of the Miami isopycnic coordinate ocean model, *Parallel Computing*, **21**(10): 1695–1720.

Blumberg, A. F., Khan, L. A., and St. John, J. P. (1999). Three-dimensional hydrodynamic model of New York harbor

region, *Journal of Hydraulic Engineering*, 125: 799–816.

Boukas, L. A., Mimikou, N., Missirlis, N., Mellor, G., Lascaratos, A., and Korres, G. (1999). The parallelization of the Princeton ocean model, *Proceedings of the 5th Intnl. Euro-Par Conf. on Parallel Processing, '99* Toulouse, France, Aug 31–Sep 03.

Chen, C., Liu, H., and Beardsley, R. C. (2003). An unstructured, finite-volume, three-dimensional primitive equation ocean model: Application to coastal ocean and estuaries, *Journal of Atmospheric and Oceanic Technology*, **20**: 159–186.

Chen, C., Gao, G., Qi, J., Proshutinsky, A., Beardsley, R. C., Lin, H., Cowles, G., and Huang, H. (2006a). A new high resolution unstructured grid finite-volume Arctic Ocean model (AO-FVCOM): an application for tidal simulation, *Journal of Geophysical Research*, in revision.

Chen, C., Xue, P., Ding, P., Beardsley, R. C., Xue, Q., Mao, X., Gao, G., Qi, J., Li, C., Lin, H., Cowles, G., and Shi, M. (2006b). Physical mechanism for offshore detachment of the Changjiang diluted water in the East China Sea, *Journal of Geophysical Research*, accepted for publication.

Cowles, G. C. (2001). *A Parallel Viscous Multiblock Flow solver for Free Surface Flows past Complex Geometries*, Ph.D. Thesis, Dept. of Mechanical and Aerospace Engineering, Princeton University.

Dudhia, J., Gill, D., Manning, K., Wang, W., Bruyere, C., Wilson, J., and Kelly, S. (2003). *PSU/NCAR Mesoscale Modeling System Tutorial Class Notes and User's Guide, MM5 Modeling System Version 3*, Mesoscale and Microscale Meteorological Division, NCAR, Boulder, Colorado.

Evensen, G. (1994). Sequential data assimilation with nonlinear quasi-geostrophic models using Monte Carlo methods to forecast error statistics, *Journal of Geophysical Research*, **99**: 143–162.

Fringer, O. B., Gerritsen, M., and Street R. L. (2006). An unstructured-grid, finite-volume, nonhydrostatic parallel coastal ocean simulator, *Ocean Modelling*, **14**: 139–173.

Karypis, G. and Kumar, V. (1998a). A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing*, **20**: 359–392.

Karypis, G. and Kumar, V. (1998b). METIS* A Software Package for Partitioning Unstructured Graphs, partitioning Meshes, and Computing Fill-Reducing Ordering of Sparse Matrices, Version 4.0, *User Manual*, Technical Report, Dept. of Computer Science and Eng., University of Minnesota, Minneapolis, MN.

Kerbyson, D. J. and Jones, P. W. (2005). A performance model of the parallel ocean program, *International Journal of High Performance Computing Applications*, **19**: 261–276.

Kerbyson, D. J., Alme, H. J., Hoisie, A., Petrini, F., Wasserman, H. J., and Gittings, M. (2001). Predictive performance and scalability modeling of a large-scale application, *Proceedings of the IEEE/ACM Supercomputing Conference*, Denver, CO.

Kobayashi, M. H., Pereira, J. M. C., and Pereira, J. C. F. (1999). A conservative finite-volume second order-accurate projection method on hybrid unstructured grids, *Journal of Computational Physics*, **150**: 40–45.

Madala, R. V. and Piacsek, S. A. (1977). A semi-implicit numerical mode for baroclinic oceans, *Journal of Computational Physics*, **23**: 167–178.

MPI Forum. (1993). A message-passing interface, *Proceedings of Supercomputing '93*, Portland, OR, Nov. 15–19.

Peyret, R. J. (1976). Unsteady evolution of a horizontal jet in a stratified fluid, *Journal of Fluid Mechanics*, **78**: 49–63.

Phillips, N. A. (1957). A coordinate system having some special advantages for numerical forecasting, *Journal of Meteorology*, **14**: 184–185.

Simons, T. J. (1974). Verification of numerical models of Lake Ontario, Part I. Circulation in spring and early summer, *Journal of Physical Oceanography*, **4**: 507–523.

Wang, P., Song, Y., Chao, Y., and Zhang, H. (2005). Parallel computation of the regional ocean modeling system, *Intl. J. of High Performance Computing Applications*, **19**(4): 375–385.

Xue, H. J., Shi, L., Cousins, S., and Pettigrew, N. R. (2005). The GoMOOS nowcast/forecast system, *Continental Shelf Research*, **25**: 2111–2146.

Zhao, L., Chen, C., and Cowles, G. (2006). Tidal flushing and eddy formation in Mount Hope Bay and Narragansett Bay: an application of FVCOM, *Journal of Geophysical Research*, **111**: C10015.