PCE-QUAL-ICM: A Parallel Water Quality Model Based on CE-QUAL-ICM

S. Chippada, C. Dawson, V.J. Parr, M.F. Wheeler, C. Cerco, B. Bunch, and M. Noel

CRPC-TR98766
February 1998

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Submitted September 1998; Also available as CEWES MSRC Technical Report 98-10

PCE-QUAL-ICM: A Parallel Water Quality Model Based on CE-QUAL-ICM

S. Chippada, C. Dawson, V. J. Parr, and M. F. Wheeler

Center for Subsurface Modeling (CSM)
Texas Institute for Computational & Applied Mathematics (TICAM)
University of Texas
Austin, TX 78712
and
C. Cerco, B. Bunch and M. Noel

Waterways Experiment Station
Vicksburg, MS

February 19, 1998

Abstract

CE-QUAL-ICM is a three-dimensional eutrophication model developed at the U.S. Army Corps of Engineers Waterways Experiment Station (CEWES), Vicksburg, MS. This water quality model is semi-explicit in time, and is based on an unstructured cell-centered finite volume numerical method. The hydrodynamics data such as velocities and turbulent diffusion are read in externally, and the model computes the advection-diffusion-reaction of a number of physical and state variables such as temperature, salinity, sediments, oxygen, algae, etc. This sequential FORTRAN 77 code was parallelized using data/domain decomposition strategy and a single program, multiple data (SPMD) paradigm. WQMPP, a pre/post processor for the water quality model which splits the global domain into a specified number of smaller subdomains and sets up the local data files and message passing tables, has been developed. WQMPP, when run in post-processor mode also combines the local subdomain output to produce global output in a format similar to that produced by the original CE-QUAL-ICM code. PCE-QUAL-ICM, the parallel water quality model enhances CE-QUAL-ICM with message passing. Inter-processor communication is done using MPI communication libraries and the parallel code has been ported onto the CRAY-T3E, IBM-SP2 and SGI O2000. This paper explains the domain decomposition and parallelization strategy employed in WQMPP and PCE-QUAL-ICM.

1 INTRODUCTION

CE-QUAL-ICM is a water quality model developed at the U.S. Army Corps of Engineers Waterways Experiment Station (CEWES), Vicksburg, MS, by Carl F. Cerco, Thomas Cole and others (Cerco & Cole 1994, 1995). This numerical code can model the transport and reaction of more than twenty state variables simultaneously. It also contains a sediment transport model, and can be run in one-, two-, or three- dimensional configurations. The numerical method is based on unstructured finite-volume method, and is explicit in time in the horizontal direction and implicit in the vertical columns. The fluid velocities are computed using

a hydrodynamics model such as CH3D-WES and are read in externally into CE-QUAL-ICM, and only the water quality modeling is done within CE-QUAL-ICM. There are sophisticated physics incorporated into the numerical model, and the code is written in ANSI Standard FORTRAN 77. The reader is referred to Cerco & Cole (1994, 1995) for a detailed description of this numerical model. In this paper only those features that are important from a parallelization point-of-view are discussed.

CE-QUAL-ICM has been used extensively in the eutrophication studies of Chesapeake Bay. Especially of interest are long term studies, typically comprising of tens of years. For these long term simulations, the current serial code requires hundreds of supercomputer (CRAY-YMP) hours. The motivation behind this project is to obtain at least an order of magnitude reduction in the simulation times. We hoped to achieve this by porting the serial code onto distributed memory parallel computing platforms. With most academic and government labs turning to parallel computers for their high performance computing needs, developing parallel algorithms seems to be a desirable way to speed up existing sequential simulators. PCE-QUAL-ICM, a parallel water quality model, is a product of this effort and has been developed at the Center for Subsurface Modeling (CSM) at the University of Texas at Austin in conjunction with CEWES. A data/domain decomposition strategy along with single program, multiple data (SPMD) paradigm is employed and inter-processor communcation is done through MPI message-passing libraries. The parallel code has been ported onto the IBM-SP2 and CRAY-T3E, which are distributed memory parallel computers, and onto the SGI O2000 which has some shared and some distributed memory. In the rest of this paper the parallel algorithm and the domain decomposition strategy are explained.

2 SOLUTION ALGORITHM OF CE-QUAL-ICM

The main component of CE-QUAL-ICM is the solution of the three-dimensional mass conservation equation of the following form for each state variable:

$$\frac{\delta\left(V_j C_j\right)}{\delta t} = \sum_{k=1}^{n} Q_k C_k + \sum_{k=1}^{n} A_k D_k \frac{\delta C}{\delta x_k} + \sum S_j \tag{1}$$

The above equation represents conservation of mass in the jth control volume, and n is the number of faces attached to control volume j. Q_k, C_k, D_k, and A_k are respectively, the volumetric flow rate, concentration, diffusion coefficient, and cross-sectional area at face k of control volume j. V_j is the volume of control volume j, and S_j are the external sources and sinks present in control volume j. $\frac{\delta C}{\delta x_k}$ is the spatial gradient of concentration in direction normal to face k, and $\frac{\delta V_j C_j}{\delta t}$ is the rate of change of the total concentration in control volume j.

CE-QUAL-ICM uses a simple time marching solution strategy. Within each time step itself, the solution update is broken into two separate steps. In the first step, an intermediate concentration is computed which takes into account the horizontal diffusion and advection along with all the external sources and sinks. This step is completely explicit and there is no need to solve any system of linear equations. In the next step, the vertical diffusion and advection are incorporated in an implicit manner and involves solving a tridiagonal system of equations for each vertical column of water. Solution methodology plays an important role in parallelization. Not all numerical techniques are readily parallelizable. CE-QUAL-ICM with its explicit treatment of horizontal diffusion and advection makes it conceptually easy to parallelize and can potentially

benefit a lot from parallelization. Implicit treatment of vertical transport and explicit treatment of horizontal transport implies that we can benefit by doing domain decomposition only in horizontal plane and assigning all of the vertical column to the same subdomain.

For computing the horizontal advective flux, the concentration C_k at face k is needed and CE-QUAL-ICM has two ways to compute this. One is the simple first-order accurate upwind differencing which sets C_k equal to the upstream value, with upstream direction determined by the sign of Q_k. Second is a higher-order accurate QUICKEST scheme which uses quadratic interpolation for computing C_k by taking two upstream cells and one downstream cell. Thus in a traditional domain decomposition sense we will need at least two layers of overlap. Note that upwinding or the QUICKEST scheme is used only in the horizontal direction and in the vertical direction a simple linear interpolation between the adjoining cells is used to compute C_k.

Either a fixed time step can be specified or an automatic time step selection based on stability criteria can be used. If the automatic time stepping option is chosen then the sub-domains need to communicate with each other to select a global minimum time step if the computations are to remain synchronous.

CE-QUAL-ICM has several types of boundary conditions. At inflow and outflow, first-order upwinding is used for advective fluxes, and the diffusion and dispersion fluxes are set to zero.

3 DATA STRUCTURES IN CE-QUAL-ICM

In addition to the solution algorithm, the data structures play an important role in parallel porting and these are discussed in this section.

CE-QUAL-ICM is an unstructured finite-volume code. Consequently, it does not have the traditional (i, j, k) indices common in structured finite-difference and finite-volume codes. The cells and the faces are numbered in a completely arbitrary manner and the cell connectivity is kept track of by having face-to-block lists. Concentrations are assigned to a cell, and the faces hold information such as flow rate, diffusion coefficient and cross-sectional area. As explained in the previous section, the concentration at the cell face for computing advective transport is computed using first-order upwinding or second-order QUICKEST scheme. Therefore, each face needs to know the two neighbouring blocks on either side. These are stored in arrays which are named as ILB, IB, JB, and JRB. For a given face, the definitions of each of these maps are stated in Table 1 and are also shown pictorially in Fig.1. Note that ILB, IB, JB, and JRB represent left-to-right, front-to-back and bottom-to-top in each of the co-ordinate directions. For each face there is an additional list called QD which takes a value of 1 for the x-direction, a value of 2 for the y-direction and a value of 3 for the z-direction. Also, for each vertical column, the surface block number is stored in SBN and the bottom block number is stored in BBN. Since the same code can be run in full 3-D or in a depth integrated mode, the SBN and BBN maps are very important. Another important map is the BU list, which specifies the block number above. There are certain phenomena such as sediment kinetics which take place along the water column with deposition of sediment particles along each water column. The BU list helps in computing these kinetics.

Even though the face and block numbering is arbitrary, there are some restrictions. Since the same code can be run in both depth-integrated form and fully three-dimensional form, CE-QUAL-ICM requires that

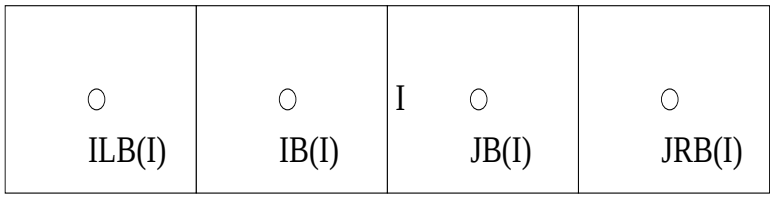

Figure 1: Face to block data structures: ILB, IB, JB, and JRB.

IB(I)	=	The block to the left of face I
ILB(I)	=	The second block to the left of face I
JB(I)	=	The block to the right of face I
JRB(I)	=	The second block to the right of face I
QD(I)	=	The axis perpendicular to face I, 1 = x, 2 = y, 3 = z
BL(1,I)	=	Box length in x-direction
BL(2,I)	=	Box length in y-direction
BL(3,I)	=	Box length in z-direction
SBN(I)	=	The surface block number in water column I
BBN(I)	=	The bottom block number in water column I
NVF(I)	=	Number of vertical faces in water column I
VFN(J,I)	=	Global face number of vertical face J in water column I
BU(I)	=	The block above block I

Table 1: Important data structures in CE-QUAL-ICM.

all the hoizontal faces be numbered before the vertical faces. No restriction on the numbering of blocks is mentioned in the user manual (Cerco & Cole 1995), but a similar restriction is implicit in the case of block numbers also, namely, that all the surface blocks be numbered first before the underlying lower-level blocks. No additional data structures are defined for specifying boundary conditions. The faces across which there is no flow such as the wall boundaries are not numbered at all, and consequently no transport calculations are done at these faces. The inflow and outflow faces are determined in the code by going through the IB and JB lists. Therefore, the order in which the boundary conditions are read in from disk files should be the same as the boundary faces order computed in the code. All these issues played a very important role in the development of our decomposition algorithm. The local face and block lists derived for each processor has to satisfy all of the above conditions, since the aim was to parallelize CE-QUAL-ICM with minimal modifications.

4 PARALLEL ALGORITHM

From the solution algorithm of CE-QUAL-ICM already outlined above, it is clear that it is readily parallelizable. It is an explicit code and is implicit only in the vertical direction. Even in this case a small tri-diagonal system of equations are solved locally within each water column and there is no implicitness in the horizontal direction. Further, a data parallel approach would be a natural way to parallelize this code. The original global domain is split into smaller subdomains, and each processor element (PE) works only on its local subdomain. Since the solution within a subdomain will depend on the solution in its neighbouring

subdomains, the PEs exchange information through message passing communication libraries. The explicit nature of the solution algorithm in CE-QUAL-ICM implies that it is enough to do message passing once every time step. Note that this type of parallel computation does not change the global solution algorithm. Conceptually all we are doing is splitting the work among processors. Thus the solution we would get through parallel computation will be identical to that we would obtain if we were to solve it sequentially up to machine precision.

CE-QUAL-ICM splits the mass-conservation solution into two parts, with horizontal transport being done explicitly and vertical transport being done implicitly. Therefore, it makes sense to do domain decomposition in the horizontal plane alone and assign all the underlying blocks in a vertical column to the same processor. This definitely minimizes inter-processor communication, since now the implicit step involving solving a system of tridiagonal linear equations is done locally within each PE. This is the approach we have taken in our parallel code. We first divide the surface cells among each processor, and then allocate all the underlying blocks under the surface cell to the same processor.

Domain decomposition itself is a non-trivial task and needs to take into consideration several issues such as load-balancing and locality. Carter Edwards (Sagan 1994, Edwards 1997) developed an effective decomposition strategy based on a Hilbert Space Filling Curve (HSFC), and this has been used with great success in parallelizing ADCIRC, an advanced coastal circulation model based on the shallow water equations (Chippada, Dawson, Martinez, and Wheeler 1996). A HSFC based domain decomposition is used in PCE-QUAL-ICM.

One of the main goals is to parallelize CE-QUAL-ICM with minimal modifications. This way, we hope to have a parallel code up and running quickly without introducing bugs into the code. This is especially important considering that the authors had no familiarity whatsoever with CE-QUAL-ICM before the project started. Also subsequent modifications and revisions to PCE-QUAL-ICM are expected to be done by the original developers of CE-QUAL-ICM. Hence, it is important that PCE-QUAL-ICM be as close as possible to CE-QUAL-ICM. What all this amounts to is that we would like most of the parallel specific work done to be outside the code. This led to the development of WQMPP, which is a pre/post-processor for PCE-QUAL-ICM. WQMPP splits the global domain into smaller sub-domains and sets up all the required message passing tables for inter-processor communication. Moreover, it decomposes each of the global files and creates local files for each PE. Thus the original I/O in CE-QUAL-ICM need not be changed and each PE sees only its local list of blocks and faces. This can result in significant savings in memory requirements. WQMPP can also be run in a post-processor mode which combines the local output files of each PE to produce global output files which look exactly same as the ones that would have been produced by CE-QUAL-ICM. The important details pertaining to WQMPP are described in greater detail in the next section.

5 WQMPP: PRE/POST PROCESSOR FOR PCE-QUAL-ICM

WQMPP is a pre/post processor for PCE-QUAL-ICM written in ANSI Standard FORTRAN 77. WQMPP can be run in both pre-processor mode and post-processor mode. When run in pre-processor mode, it splits the global domain, creates local input files for each PE, and sets up the communication table for inter-processor message passing. Below, we describe briefly the various steps involved

1. CE-QUAL-ICM has data structures that connects blocks to faces but not vice-versa. It is useful to have a list which specifies for each block what the global numbers of its six faces are. Therfore, we first create this list from the IB, JB, and QD lists. Note that, not all faces are numbered in CE-QUAL-ICM. The land boundaries through which there is no flux are not numbered for instance. Thus, a block can have a face numbered zero as one of its face which only implies that it is a face with zero flux.
2. CE-QUAL-ICM is a finite-volume code and all it needs are the box lengths in each direction, which it uses to calculate cross-sectional areas of the cell faces and the cell volumes. Thus, there are no global coordinates stored anywhere within CE-QUAL-ICM. However, we will need this for our domain decomposition, since we would like to maintain locality. We reconstruct the global co-ordinates from the box lengths and the face-to-block maps. Alternately, we could read the global coordinates of the vertices from a disk file. Both of these options have been implemented.
3. The next step is to do the actual domain decomposition. Since the solution algorithm is explicit in the horizontal direction and implicit in the vertical direction, it is advantageous to split the surface blocks among processors and assign all the underlying blocks in a vertical column to the same processor. The domain decomposition is done using the Hilber Space Filling Curve approach developed by Carter Edwards who also wrote the necessary C routines to do this. In this approach a continuous curve is passed through all the mid-points of the surface blocks with the purpose of maintaining locality. The number of vertical layers can vary in the domain and the HSFC routine partitions the domain so that we get good load balancing. In our case, it amounts to each processor getting approximately the same number of cell blocks, even though the number of surface blocks themselves may be widely different.
4. After the surface blocks are partitioned all the underlying blocks are extracted using the vertical face maps NVF and VFN and the face-to-block maps for each water column. We thus obtain a list of resident blocks for each PE. The QUICKEST numerical scheme for computing the advective flux requires that we know the two adjoining blocks on either side of each face. Using the face-to-box maps IB, ILB, JB, and JRB a padding of two cell layers is given for each PE. At the end of this we have the list of all the blocks that are assigned to each PE.
5. Using the block-to-face that has been computed previously all the faces that belong to a PE are computed. All other data structures such as SBN, BBN, NVF, VFN, and BU are computed for each processor.
6. After all the necessary local data structures are set up, each of the original global data files is split into smaller local files to be read separately by each individual processor.

6 PCE-QUAL-ICM: CE-QUAL-ICM WITH MESSAGE PASSING

In the previous section the pre-processor part of WQMPP was explained. At run time each PE needs to exchange interface data and this requires certain modifications to the CE-QUAL-ICM code. These modifications are briefly described in this section. CE-QUAL-ICM with the message passing related modifications is what we call PCE-QUAL-ICM.

In PCE-QUAL-ICM, variables specific to message passing are introduced and these are listed in Table 2. All of the resident, send, and receive block related lists are created by WQMPP and written into disk files. When PCE-QUAL-ICM is executed, each executable copy of PCE-QUAL-ICM is started on a different processor and they all read the files specific to them.

If the auto time step selection option is chosen, a global minimum needs to be computed. This is done through a call to the MPI_ALL_REDUCE subroutine available through the MPI communications library. Each PE computes a minimum for its domain, and then through a call to MPI_ALL_REDUCE a global minimum is computed from the local minimum which is then broadcast to all PEs. At the end of this call all PEs have the global minimum. Note that the time step calculation is dependent on the hydrodynamics data. So, a new time step is calculated only when new hydrodynamics data is read in.

CE-QUAL-ICM reads the hydrodynamics data such as flow rates and eddy visocity coefficients externally. Usually the hydrodynamics data is computed using CH3D-WES and written onto disk files. Since the data files are typically very large they are written in binary format. Once every few time steps, CE-QUAL-ICM reads the hydrodynamics data from the external disk files. After reading in the hydrodynamics it recomputes the volumes in the cells and the block lengths in the vertical direction. A recalculation of these quantities is necessary because the water level may change with time. The surface block depth is chosen in such a way that the water surface doesn't cross its boundaries. So, only the surface block's volume and box length in the vertical direction changes with time. By using MPI_SEND and MPI_RECEIVE calls, the volumes and box lengths in the overlap region are updated everytime new hydrodynamics data is read.

At the end of every time step, we have the right concentration values in the resident blocks. But the ghost blocks themselves will have wrong values. To be able to proceed to the next time step the ghost blocks also need to have the right values. This is obtained through MPI_SEND and MPI_RECEIVE calls. Thus, at the end of every time step we perform message passing so that all of the blocks in a PE, both resident and ghost, will have the right values.

To summarize, in PCE-QUAL-ICM we introduce some new variables to keep track of the resident, send and receive lists. We also do some communications when new hydrodynamics data is read, which is done only once every few time steps. At the end of every time step concentration values in the overlap region are exchanged among PEs at the end of which all blocks in a PE have the right value. From the above description it is clear that the changes to the original CE-QUAL-ICM are minimal.

7 POST-PROCESSING

Once PCE-QUAL-ICM is run successfully on a parallel computing platform, each processor will create output files pertaining to the domain it owns. To create output that looks indistinguishable from the original CE-QUAL-ICM's output,we will need to combine these local output files to create global output files. WQMPP described previously can also be run in post-processor mode. Using the decomposition tables it created previously while pre-processing, WQMPP post-processes the local output files to produce global output files.

NPES	=	Number of PEs
MYRANK	=	The local PE number (between 0 and NPES-1)
NUM_RES_BLOCKS	=	Number of resident blocks (i.e., the blocks the PE owns)
RES_BLOCK_NUM(I)	=	Local block number of resident block I
NUM_COMM_PE	=	Number of PEs with which the current PE needs to communicate
SEND_PE_NUM(I)	=	Target PE number for sending block data
NUM_BL_SEND(I)	=	Number of blocks to send to SEND_PE_NUM(I)
SEND_BL_NUM(J,I)	=	Send block list corresponding to SEND_PE_NUM(I)
RECV_PE_NUM(I)	=	Source PE Number for receiving data
NUM_BL_RECV(I)	=	Number of Blocks to receive from RECV_PE_NUM(I)
RECV_BL_NUM(J,I)	=	Receive-Block-List corresponding to RECV_PE_NUM(I)

Table 2: New variables/arrays introduced in PCE-QUAL-ICM.

8 APPLICATION TO CHESAPEAKE BAY

In the previous sections, the essential features of WQMPP and PCE-QUAL-ICM have been explained. These numerical tools have been first tested on a simple 30 box model. Later they were applied to Chesapeake Bay. The numerical model for Chesapeake Bay that was simulated consists of 8830 grid blocks, out of which 2100 were surface blocks. The two-dimensional numerical grid comprised of only the surface blocks is shown in Fig.2. The physical domain is highly irregular and extremely complicated. The number of blocks in a vertical water column varied between 1 and 18. The bathymetric depth of the physical domain is shown in Fig.3.

The original 8830 grid block data set is first decomposed into smaller local subdomain problems using WQMPP. A Hilbert space filling curve passing through the mid-points of all surface blocks is drawn as shown in Fig.4. Once the surface blocks have been numbered they are split among processors based on load balancing criteria. Note that some water columns have more underlying blocks than others and this needs is taken into account in actual partitioning. At the end of domain partitioning, we get subdomains that have nearly the same number of grid blocks, whereas the number of surface blocks may be widely different. Decompositions for 2, 4 and 8 processing elements (PE) are shown in Figs.5-7.

During initial testing, PCE-QUAL-ICM was run for a time period of 1 day and results obtained were post-processed to obtain global outputs. These global outputs obtained from PCE-QUAL-ICM were compared with those obtained from CE-QUAL-ICM, and were found to be identical.

A few speedup studies have been conducted to assess the efficiency of our parallelization scheme. A 30 day simulation has been run on Chesapeake Bay with 8830 blocks and the speedup obtained on 3 different parallel computer architectures are shown in Fig.8. The best speedups were obtained on SGI O2K which is a shared memory computer. CRAY T3E produced the least speedups. 8830 grid blocks is not a very big data set and we lose speedup efficiency beyond 16 PEs.

9 CONCLUDING REMARKS

CE-QUAL-ICM, a three-dimensional eutrophication model, has been successfully ported onto parallel computing environments. WQMPP is the pre/post processor that decomposes the global domain into smaller

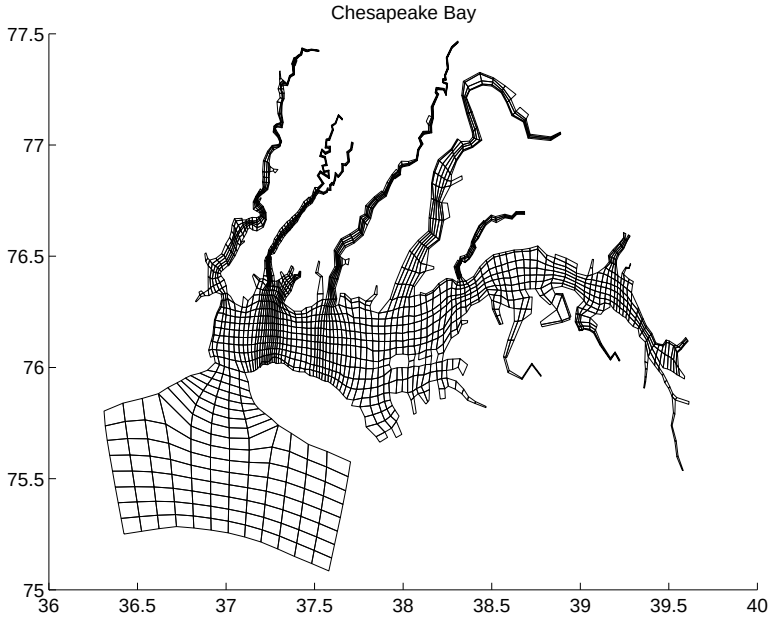

Figure 2: Chesapeake Bay: 2100 surface blocks and 8830 total grid blocks.

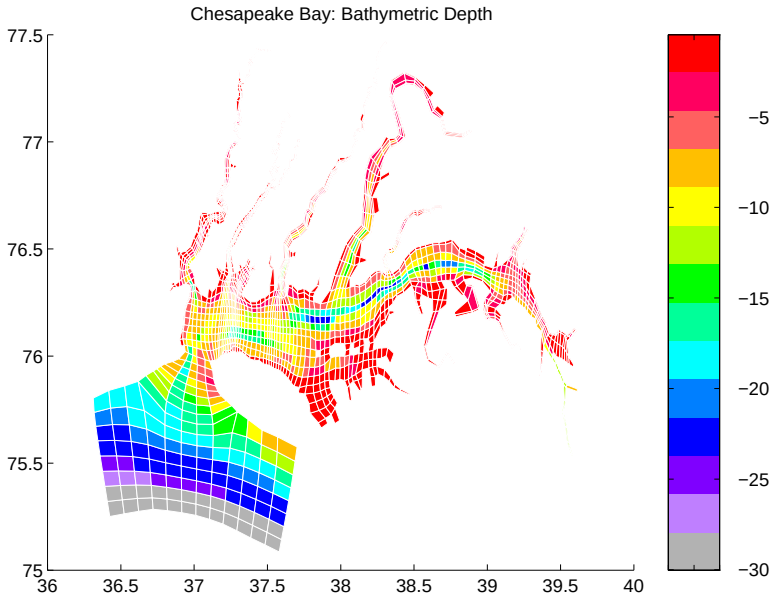

Figure 3: Chesapeake Bay: Bathymetric depth.

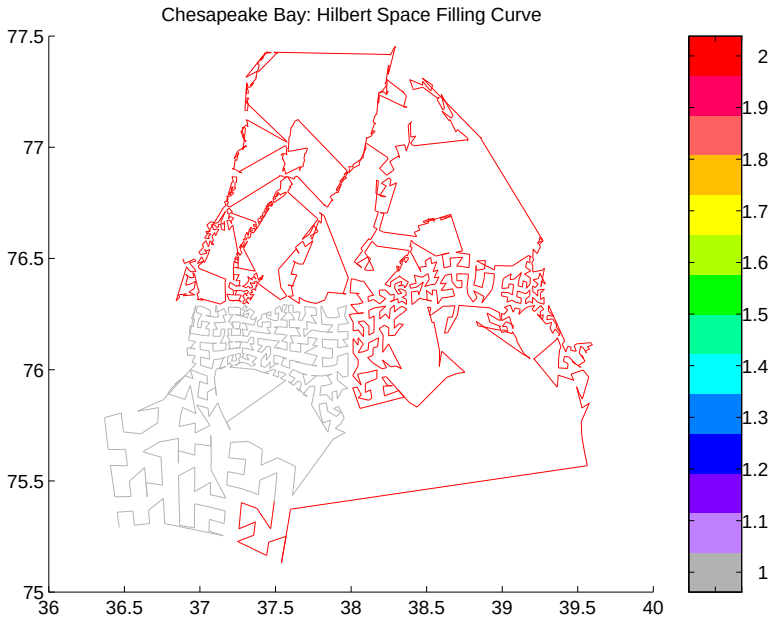

Figure 4: Hilbert Spave Filling Curve.

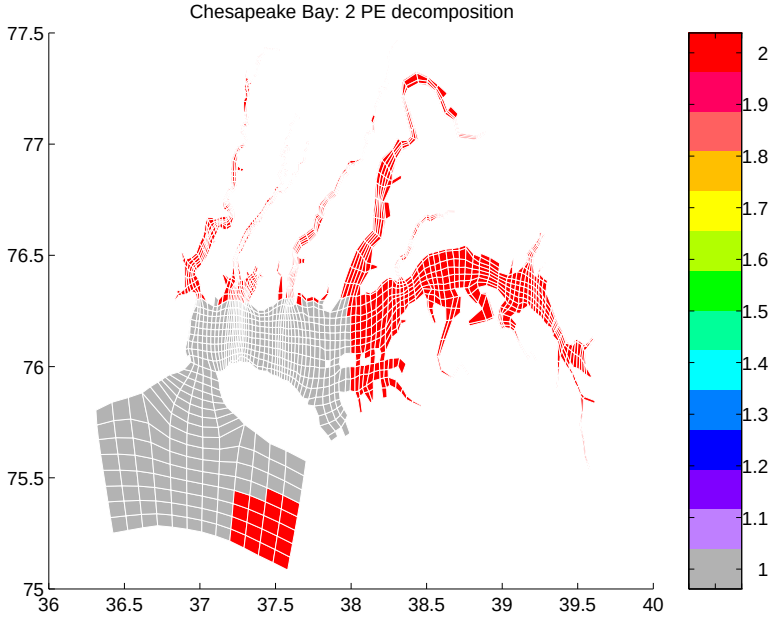

Figure 5: Domain decomposition for 2 processors.

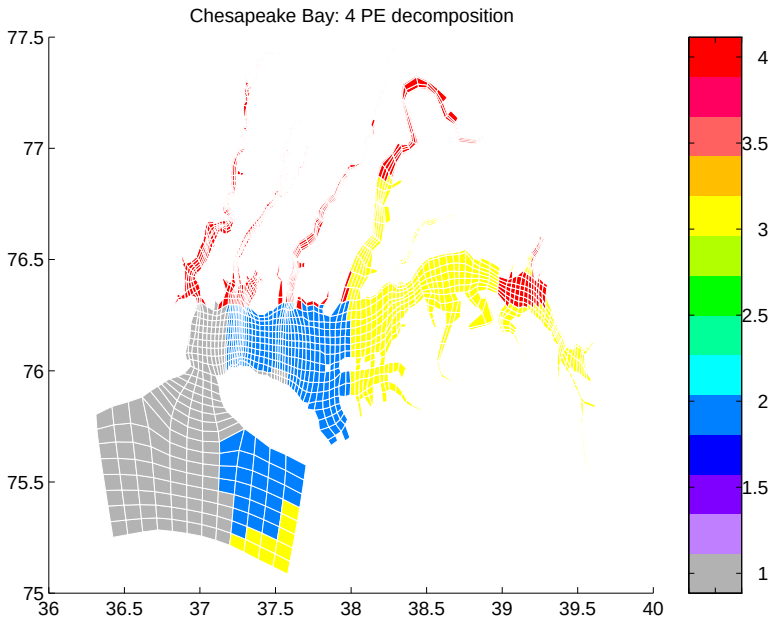

Figure 6: Domain decomposition for 4 processors.

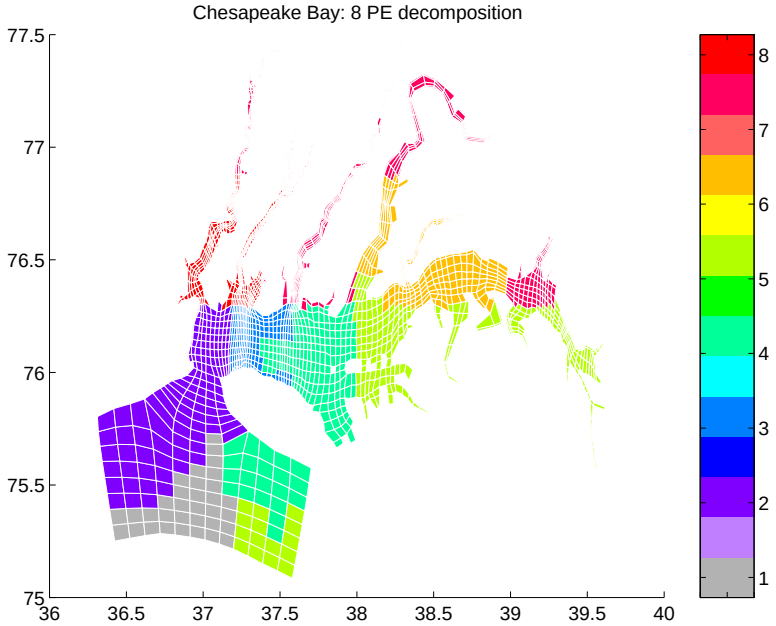

Figure 7: Domain decomposition for 8 processors.

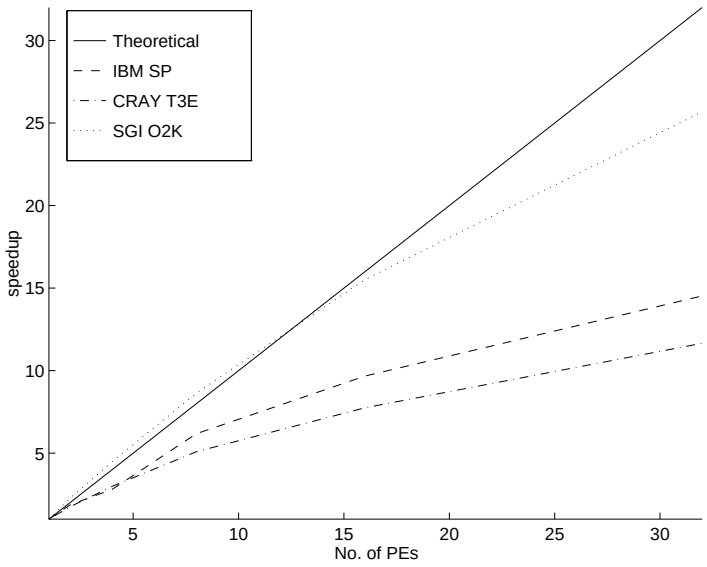

Figure 8: Domain decomposition for 8 processors.

local subdomains and sets up all the message passing tables required for inter-processor communication. WQMPP, when run in post-processor mode, also combines the local outputs to produce global output in the same format as that produced by the original CE-QUAL-ICM code. PCE-QUAL-ICM is the parallel water quality model, and is nothing but CE-QUAL-ICM with message passing.

The parallel porting of CE-QUAL-ICM was achieved in a relatively short amount of time with minimal changes to the code. PCE-QUAL-ICM can be run in both serial and parallel modes, and now we need maintain only one code instead of two separate ones. This makes future development and maintenance of codes easier. Since the modifications made to CE-QUAL-ICM are minimal, the original developers of CE-QUAL-ICM will find PCE-QUAL-ICM very familiar. This should help tremendously in transferring the technology back to the code developers.

The speed-ups obtained are only moderate. This could be due to many reasons. Most important being that the problem size is small (only 8830 blocks). The numbering conventions used in CE-QUAL-ICM has been at times frustrating. Especially, the way boundary conditions are read in. There is no boundary table as such, and the code goes through all its faces to see which doesn't have neighbouring blocks, and sets these faces to be boundary faces. The order in which the boundary values are to be read in is the same as the order in which they are computed in code. This placed unnecessary restrictions on the development of WQMPP, complicating it a lot. The same goes with the restriction that the surface blocks be numbered before the lower level blocks are numbered. This complicated WQMPP as well and made PCE-QUAL-ICM perform certain unnecessary calculations in the overlap region, which can normally be avoided. If some of these restrictions on numbering are removed, it may give some improvement is speed-ups as well.

Getting PCE-QUAL-ICM to work on various computing platforms has not exactly been easy. Especially hard has been the differences in wordlength used by CRAY-T3E and non-CRAY computers such as IBM-SP2 and SGI PowerChallenger. So, PCE-QUAL-ICM is essentially platform independent, but there are a few

things that need to be set by the user depending on the computing environent.

This is an on-going research project between the Center for Subsurface Modeling (CSM) at the University of Texas at Austin and the U.S. Army Corps of Engineers Waterways Experiment (CEWES) at Vicksburg. Future modifications and improvements to PCE-QUAL-ICM and WQMPP are dependent on the experiences of the end users such as researchers at CEWES.

10 ACKNOWLEDGEMENTS

This project was funded through the PET Program, managed by Nichols Research Corporation. The authors would like to acknowledge Carter Edwards for his contributions to the WQMPP code.

References

[1] Carl F. Cerco, and Thomas Cole, 1994, "Three-Dimensional Eutrophication Model of Chesapeake Bay," Technical Report EL-94-4, *US Army Corps of Engineers Water Experiment Station*, Vicksburg, MS.

[2] Carl F. Cerco, and Thomas Cole, 1995, "User's Guide to the CE-QUAL-ICM Three-Dimensional Eutrophication Model, Release Version 1.0," Technical Report EL-95-15, *US Army Corps of Engineers Water Experiment Station*, Vicksburg, MS.

[3] Srinivas Chippada, Clint N. Dawson, Monica L. Martinez, and Mary F. Wheeler, 1996, "Parallel Computing for Finite Element Models of Surface Water Flow", in Computational Methods in Water Resources XI, Vol. 2: Computational Methods in Surface Flow and Transport Problems, eds. A.A. Aldama, J. Aparicio, C.A. Brebbia, W. G. Gray, I. Herrera and G. F. Pinder. pp. 63-70. Computational Mechanics Publications, Boston, July 1996.

[4] H. C. Edwards, 1997, "A Parallel Infrastructure for Scalable Adaptive Finite Element Methods and its APplication to Least Squares C^∞ Collocation," *Ph.D. Thesis*, The University of Texas, Austin, TX, May 1997.

[5] Hans Sagan, 1994, *Space-Filling Curves*, Springer-Verlag.