PARALLELIZATION OF A MULTIREGION FLOW AND TRANSPORT CODE USING SOFTWARE EMULATED GLOBAL SHARED MEMORY AND HIGH PERFORMANCE FORTRAN

Eduardo F. D'Azevedo
Computer Science and Mathematics Division
Oak Ridge National Laboratory[1]
P.O. Box 2008, MS6367
Oak Ridge, TN 37831-6367
email: efdazedo@msr.epm.ornl.gov

Jin-Ping Gwo
Center for Computational Sciences
Oak Ridge National Laboratory
P.O. Box 2008, MS6203
Oak Ridge, TN 37831-6203
email: g4p@ornl.gov

KEY WORDS

Global shared memory emulation, High performance FORTRAN, Preconditioned conjugate gradient matrix solver, Groundwater, Finite element method.

ABSTRACT

The objectives of this research are (1) to parallelize a suite of multiregion groundwater flow and solute transport codes that use Galerkin and Lagrangian- Eulerian finite element methods, (2) to test the compatibility of a global shared memory emulation software with a High Performance FORTRAN (HPF) compiler, and (3) to obtain performance characteristics and scalability of the parallel codes. The suite of multiregion flow and transport codes, 3DMURF and 3DMURT, were parallelized using the DOLIB (Distributed Object LIBrary, D'Azevedo and Romine, 1994) shared memory emulation, in conjunction with the PGI HPF compiler, to run on the Intel Paragons at the Oak Ridge National Laboratory (ORNL) and a network of workstations. The novelty of this effort is first in the use of HPF and global shared memory emulation concurrently to facilitate the conversion of a serial code to a parallel code, and secondly the shared memory library enables efficient implementation of Lagrangian particle tracking along flow characteristics. The latter allows long-time-step-size simulation with particle tracking and dynamic particle redistribution for load balancing, thereby reducing the number of time steps needed for most transient problems. The parallel codes were applied to a pumping well problem to test the efficiency of the domain decomposition and particle tracking algorithms. The full problem domain consists of over 200,000 degrees of freedom with highly nonlinear soil property functions. Relatively good scalability was obtained for a preliminary test run on the Intel Paragons at the Center for Computational Sciences (CCS), ORNL. However, due to the difficulties we encountered in the PGI HPF compiler, as of the writing of this manuscript we are able to report results from 3DMURF only. This work is part of an on-going research project, and the most recent development and performance information of 3DMURF and 3DMURT will be posted at www.epm.ornl.gov/~efdazedo/ and www.ccs.ornl.gov/staff/gwo/gwo.html as soon as they are available.

INTRODUCTION

For science- and engineering-oriented computational minds, ease of porting serial codes to massively parallel supercomputers has always been a prime topic well looked into. However, efforts in helping scientists and engineers facilitate their code conversion and thereby increase their productivity on high performance parallel computers were not available until the very recent. Especially on the distributed memory machines such as Intel Paragons and

[1] managed by Lockheed Martin Energy Research, Corp. for the U.S. Department of Energy under contract DE-AC05-96OR22464

IBM SP2, on which the vendors do not provide share memory libraries, parallelizing serial codes was one of the major efforts before production runs can be resumed. The DOLIB shared memory emulation library was developed to simplify and ease the burden of porting serial codes to distributed memory environments, especially particle-based applications such as molecular dynamics simulations (D'Azevedo et al., 1996) or contaminant transport modeling by Lagrangian particle tracking (D'Azevedo and Romine, 1995) where the communication pattern depends on the particle distribution at run-time. The shared memory library was also successfully used in DONIO (Distributed Object Network I/O Library, Semeraro et al., 1995) as a large disk cache to increase I/O performance. Another approach is the use of an HPF compiler with automatic parallelization. While the message passing and I/O performance of HPF codes remains to be improved (Gwo and Yeh, this issue), this research emphatically addresses the compatibility of DOLIB and HPF, especially on Intel distributed memory machines. The possibility of using DOLIB to improve load balancing when particle tracking is needed to perform Lagrangian-Eulerian finite element simulations is also examined.

DOLIB is a collection of Fortran and C callable routines that emulate global or virtual shared memory on distributed multiprocessor environments such as Intel Paragons or a network cluster of workstations (D'Azevedo and Romine, 1994). The library is portable and does not require any language extension, operating system kernel modifications, nor special compiler support. It allows the programmer to dynamically allocate and destroy large (Gigabyte) global arrays. Access to the distributed array is through explicit calls to perform gather/scatter. DOLIB implements atomic scatter accumulate operations without locks for efficient implementation of global finite element matrix assembly. The library also supports automatic caching of remote data to reduce communication volume. One additional advantage of DOLIB is that debugging can be simplified by using a serial version of DOLIB where gather/scatter calls are simply memory copies. Once a serial code is modified to use DOLIB primitives, parallel debugging is required mainly to find bugs related to race conditions.

3DMURF and 3DMURT are a suite of finite element groundwater flow and solute transport codes that simulate fate and transport of contaminants in the subsurface (Gwo et al., 1994; Gwo et al., 1995a). More specifically, they focus on the movement of water and chemicals in soils with microscopic structures and rock formations with discontinuities. Galerkin and a hybrid Lagrangian-Eulerian finite element methods are used in 3DMURF and 3DMURT, respectively, to solve the governing mass balance equations. Lagrangian concentrations of individual finite element nodes are computed by tracking along flow characteristics the concentrations of fictious particles. Preconditioned conjugate gradient solvers are used to solve the linearized matrix equations. 3DMURF and 3DMURT use a spatial domain decomposition approach to implement Lagrangian particle-tracking, matrix equation assembling, and matrix equation solving. Each processor is responsible for all the elements and nodes in a (usually static) spatial subdomain. We adopt a particle-based decomposition, in which computational work associated with particles is assigned to processors. This approach is designed to avoid load imbalance and to minimize communications when dealing with advection dominated problems.

The parallel codes were applied to a pumping well problem that is designed to test the efficiency of the domain decomposition and particle tracking algorithms. The full problem domain consists of over 200,000 degrees of freedom with highly nonlinear soil property functions. Relatively good scalability was obtained for a preliminary fun on the Intel Paragons at CCS, ORNL. However, due to the difficulties we encountered in the PGI HPF compiler, as of the writing of this manuscript we are able to report results from 3DMURF only. This work is part of an on-going research project, and the most recent development and performance information of 3DMURF and 3DMURT will be posted at www.epm.ornl.gov/~efdazedo/ and www.ccs.ornl.gov/staff/gwo/gwo.html as soon as they are available.

THEORETICAL BACKGROUND OF SUBSURFACE MULTIREGION FLOW AND SOLUTE TRANSPORT

Subsurface porous media are usually composed of structures such as fractures, matrix blocks, and aggregates at a scale smaller than grid blocks in a numerical model but larger than the effective measurement volume of a measurement device. This difference usually results in closure problems and unresolved physics (Beckie, 1996; Dykaar and Kitanids, 1996). This in turn translates into uncertainties in model parameters and stochastic approaches have long been employed to address this problem (Dagan, 1989). On the other hand, multiregion and multiple-permeability-multiple-porosity approaches have been used by previous investigators to resolve mass transfer problems and incorporate parameter variations at scales smaller than grid blocks (Skopp et al., 1981; Nerethieks and Rasmuson, 1984; Gwo et al., 1995b). Conceptual models thus obtained can be collectively described as the multiple-flow-region models (e.g., Gwo et al., 1994, 1995a,b; Haggerty and Gorelick, 1995; Skopp et al., 1981). The governing equation for variably saturated multiregion fluid flow in the pore region α can be described as (Gwo et al., 1994):

$$\frac{\partial \theta_\alpha}{\partial t} = \nabla \cdot \left(\mathbf{K}_\alpha \cdot \nabla(h_\alpha + z)\right) + q_\alpha - \sum_{i=1, i \neq \alpha}^{N} \varepsilon_{\alpha i}\left(h_\alpha - h_i\right), \tag{1}$$

$$\mathbf{v}_\alpha = -\mathbf{K}_\alpha \nabla(h_\alpha + z), \tag{2}$$

where t, θ, h, z, $\mathbf{v}$, q, and $\mathbf{K}$ are time (T), water content, pressure head (L), elevation (L), fluid velocity vector (L/T), external fluid sources ($L^3/L^3/T$), and hydraulic conductivity tensor (L/T), respectively. The last term in eq.(1) is the net mass transfer between the pore region α and the others, where $\varepsilon_{\alpha i}$ is the advective mass transfer coefficient (1/LT) between pore regions α and i. The solute transport equation in a pore region α can be represented by the following equations (Gwo et al., 1995a):

$$\begin{aligned}
\left(\theta_\alpha + \rho_{b\alpha}\frac{ds_\alpha}{dc_\alpha}\right)\frac{Dc_\alpha}{Dt} &= \theta_\alpha \nabla \cdot \left(\mathbf{D}_\alpha \cdot \nabla c_\alpha\right) \\
&-\lambda_\alpha\left(\theta_\alpha + \rho_{b\alpha}\frac{ds_\alpha}{dc_\alpha}\right)c_\alpha - \left(K_{w\alpha} + K_{s\alpha}\rho_{b\alpha}\frac{ds_\alpha}{dc_\alpha}\right)c_\alpha \\
&+\theta_\alpha q_\alpha c^*_{q\alpha} - \theta_\alpha q_\alpha c_\alpha - \sum_{j=1, j\neq\alpha}^{N} \varepsilon_{\alpha j}\left(h_\alpha - h_j\right)c_\alpha \\
&- \sum_{j=1, j\neq\alpha}^{N} \kappa_{\alpha j}\left(c_\alpha - c_j\right) + \sum_{j=1, j\neq\alpha}^{N} \varepsilon_{\alpha j}\left(h_\alpha - h_j\right)c_\alpha \\
&-\rho_{b\alpha}\left(\lambda_\alpha + K_{s\alpha}\right)\left(s_{\alpha,0} - \frac{ds_\alpha}{dc_\alpha}c_{\alpha,0}\right),
\end{aligned} \tag{3}$$

$$\mathbf{D}_\alpha = a_{T\alpha}\mathbf{v}_\alpha\delta + a_{L\alpha} - a_{T\alpha}\frac{\mathbf{v}_\alpha\mathbf{v}_\alpha}{\mathbf{v}_\alpha} + a_m\tau_\alpha\delta, \tag{4}$$

where c and s are the concentrations (M/L^3 and M/M) in the solution and solid phases, respectively; ρ_b, λ, K_w, K_s are bulk density (M/L^3), radioactive decay rate (1/T), first-order degradation rate in the aqueous phase (1/T), and first-order degradation rate in the solid phase (1/T), respectively; q is source/sink rate ($L^3/L^3/T$); c_q^* is the concentration of external source (M/L^3) if q is a source (> 0), and c (M/L^3) if q is a sink (< 0); $\kappa_{\alpha j}$ is the diffusive mass transfer coefficient (1/T) between pore regions α and j; $s_{\alpha,0}$ and $c_{\alpha,0}$ are solid and aqueous phase concentrations of the previous time step, respectively. In eq.(4), $\mathbf{D}_\alpha$ is the hydrodynamic dispersion coefficient tensor (L^2/T); $a_{T\alpha}$, $a_{L\alpha}$, a_m are the transverse dispersivity (L), longitudinal dispersivity (L), and molecular diffusion coefficient (L^2/T), respectively; τ_α is tortuosity and δ is Kronecker delta tensor. The relationships between solid phase and aqueous phase concentrations can be expressed as (Gwo et al., 1995a):

Linear isotherm: $$s_\alpha = K_d c_\alpha \tag{5a}$$

Freudlich isotherm: $$s_\alpha = K c_\alpha^n \tag{5b}$$

Langmuir isotherm: $$s_\alpha = \frac{s_{\alpha,m} K c_\alpha}{1 + K c_\alpha} \tag{5c}$$

where K_d is the distribution coefficient (L^3/M), $s_{\alpha,m}$ is the maximum concentration on the solid phase (M/M), n is the order of Freudlich isotherms, and K is a functional coefficient for Freudlich or Langmuir isotherms.

With appropriate boundary and initial conditions, eqs.(1) and (2) completely define a multiregion fluid flow system in the subsurface. The flow equation (1) governs the movement of a fluid in a variably saturated porous medium and is widely know as the Richard's equation. The equations were solved by a Galerkin finite element method in 3DMURF. The system is nonlinear because the hydraulic conductivity tensor **K** is dependent on the primary variable h; however, the matrix equation is symmetric and is solved by a preconditioned conjugate gradient method. Similarly, eqs.(3) - (5) completely define a multiregion solute transport system in the subsurface, which was solved by a hybrid Lagrangian-Eulerian finite element method in 3DMURT. Note that in eq.(3), the left hand side is in material derivative form, indicating that Lagrangian method is used to solve the time derivative term. Because of this choice of numerical method, the resultant coefficient matrix of a transient solute transport system becomes symmetric and preconditioned conjugated gradient solvers can be similarly used in 3DMURT. Note, however, that a steady state solute transport system with upstream weighting of the advection term results in asymmetric coefficient matrix and other methods, e.g., bi-conjugate gradient methods, need to be used. A bi-conjugate gradient stabilized method (Barrett et al., 1994) was used in 3DMURT for this occasion. Detailed implementation of Lagrangian method in 3DMURT can be found elsewhere (Gwo et al., 1995a), but note that the particle tracking method used in 3DMURT may result in fictious points residing outside a subdomain. If the subdomain does not belong to the responsible processor, expensive scalar message passing results. This problem is especially severe and may cause load imbalance if high velocity difference exists in the problem domain, e.g., those resulting from an extracting or a pumping well. We resorted to the global shared-memory emulation library DOLIB and HPF for solutions.

IMPLEMENTATION OF DOLIB AND HPF IN 3DMURF AND 3DMURT AND APPLICATION TO A PUMPING WELL PROBLEM

The approach here is to use the PGI HPF compiler to perform automatic parallelization and to handle most of the data distribution and message passing. For performance-critical routines or cases where the computation cannot be expressed efficiently in a SIMD (Single Instruction Multiple Data) framework (such as particle tracking), we use extrinsic F77_LOCAL or HPF_LOCAL coupled with DOLIB shared memory calls.

An extrinsic local routine is limited to having available only strictly local data. The programmer has to rely on explicit message communication (e.g., MPI) to exchange global data with other tasks. Moreover, the mapping between local and global indices adds another layer of complexity. Although the HPF standard describes a set of translation intrinsics, such as GLOBAL_TO_LOCAL and LOCAL_TO_GLOBAL routines in the HPF_LOCAL_LIBRARY, these functions are not yet fully supported by most compilers.

A novelty in our approach is to interface the DOLIB shared memory library with HPF global arrays within extrinsic local routines so that HPF and DOLIB routines manipulate the SAME physical arrays. Such a capability would greatly simplify the mapping between local and global indices and the scheduling of message communication.

The test example, a three-dimensional pumping well problem, was used in Yeh (1987) to test a variably saturated groundwater flow code 3DFEMWATER. The problem involves the steady state flow to a pumping well. The problem domain is bounded by two rivers on the left and right, impervious aquifuges on the front, back, and bottom, and by an air-soil interface on the top. A pumping well is located 460 m away from the river on the right boundary and 400 m away from the front and back boundaries. The dimension of the domain is 1000 x 800 x 72 m^3. Water tables in the rivers are assumed to be constant, at 60 m, and a water table at 30 m is maintained in the well. The porous medium is assumed to be homogeneous but anisotropic, with saturated hydraulic conductivity in the x-direction Kxx = 5 m/d, y-direction Kyy = 0.5 m/d, and z-direction Kzz = 2 m/d. Because of symmetry, only half of the problem domain is simulated. The soil property functions and other model parameters can be found in Yeh (1987). The problem domain is discretized into 1,600 elements, consisting of 2,079 nodes. Three pore regions will be simulated using 3DMURF and 3DMURT for the preliminary test of the codes. An 100 pore region run is currently underway, and the results will be posted on the aforementioned web sites as soon as they are available.

Preliminary runs of 3DMURF for the pumping well problem indicated that finite element assembly and linear solution by a preconditioned conjugate gradient method are the most time consuming computational kernels. Moreover, the largest matrices are associated with these computational kernels and are prime candidates for distribution across processors. An HPF version of 3DMURF was produced by adding the appropriate subroutine interface declarations and compiler directives for alignment or distribution of the largest arrays. Other smaller arrays such as those associated with boundary conditions are replicated for simplicity. Initial performance of the HPF code was very disappointing since without the insertion of compiler directives, the compiler was very conservative in its parallelization.

To examine the performance of scatter-accumulation using either HPF intrinsic or F77 extrinsic calls, we reorganized the finite element assembly routines to construct N (say N = 64, distributed with the CYCLIC format) dense element matrices at a time. Another part of the computation searches the connectivity (sparsity) pattern and perform accumulation or "assembly" into the global (sparse) matrix. The following four strategies were tested, their corresponding results are shown in Tables 1 - 4. These test runs were conducted on the Intel Paragon XPS5, housed by CCS at ORNL, using the GP nodes.

(1) Interface DOLIB with HPF. The scatter-summation operations were performed by DOLIB primitives.

(2) Broadcast (replicate) all N dense matrices to all processors. Each processor perform identical (redundant) searches through the data structure. Each processor performs a test whether it is the unique "owner" of the entry and performs the sum update. No sum_scatter operation was needed.

(3) The N element matrices are distributed across all processors which conduct concurrent searches through the

data structures. The target positions are saved in auxiliary arrays. The actual summation is performed by a FORALL construct.

(4) This approach is similar to (2) except HPF intrinsic library function "sum_scatter" is used in performing the assembly into the global matrix.

Table 1. Performance statistics using strategy (1), using DOLIB primitives.

np	asembl	q8	scatter	sum	pcg	applya
1	431	157	162	66.4	315	115
2	114	44.1	62.6	48.7	86.4	72.8
4	73.3	22.4	47.0	40.9	48.9	43.4
8	72.2	13.9	54.6	50.0	37.9	31.2
16	68.1	6.21	59.0	56.2	41.8	35.6

Table 2. Performance statistics using strategy (2), broadcasting dense matrix.

np	asembl	q8	scatter	sum	pcg	applya
1	187	99.6	60.7	0.00	240	121
2	90.4	44.0	41.1	0.00	82.0	73.0
4	66.6	22.4	40.7	0.00	48.2	42.9
8	59.2	11.6	44.6	0.00	37.1	32.0
16	64.9	6.19	55.8	0.00	40.6	35.2

Table 3. Performance statistics using strategy (3), using FORALL construct.

np	asembl	q8	scatter	sum	pcg	applya
1	365	101	155	34.4	434	160
2	146	44.4	89.1	70.1	111	74.0
4	103	22.5	77.1	71.0	48.8	43.2
8	106	11.7	90.9	87.1	39.3	32.3
16	135	6.20	126	123	46.5	35.6

Table 4. Performance statistics using strategy (4), using HPF sum_scatter intrinsic.

np	asembl	q8	scatter	sum	pcg	applya
1	816	106	592	486	354	116
2	395	44.3	343	332	95.0	73.4
4	241	22.4	215	209	48.9	43.2
8	177	11.6	163	159	37.5	32.4
16	172	6.19	163	160	46.9	35.8

In Tables 1 - 4, *asembl* is the total time spent in matrix assembly; *q8* is the total time spend in only construction of dense element matrices; *scatter* is the total time spent in search sparsity pattern and perform summation into global matrix; *sum* is the total time spent in the performing the sum_scatter operation; *pcg* is the total time spent in the preconditioned conjugate gradient linear solver; and *applya* is the total time spent in performing matrix vector multiplies. All the times are in seconds.

Overall, strategy (4) appeared to result in better scalability than the other two HPF strategies, but it also had the poorest performance for a problem of this size (8,127 degrees of freedom). Sparse matrix vector multiplication, performed entirely in HPF, showed a reduction of 3.5 - 5 times on 8 processors. This computational kernel is difficult to parallelize efficiently since it has poor data reuse and a high ratio of communication to computation. The performance of DOLIB is slightly better than HPF. The same set of strategies, with BLOCK data distribution format, were also tested on the XPS5. The results (not shown) were slightly better than those using CYCLIC format.

3DMURT, like 3DMURF, is based on three-dimensional hexahedral finite element formulation and shared many commonality. The key difference in code structure lies in the modules for Lagrangian particle tracking along flow characteristics. DOLIB has been used in the parallelization of the SLT (Semi-Lagrangian Transport) for modeling the advection of moisture in global climate simulation. The velocity/flow field is first stored in global shared arrays, then particles are assigned to processors for tracking. The distribution of particles to processors may be influenced by spatial locality and the amount of work required in the previous time step. Similar approach will be used for the parallelization of 3DMURT.

CONCLUSION

High Performance FORTRAN is a developing technology and language standard. The purpose of this paper is to communicate with the scientific and engineering application communities the lessons we have learned from porting a serial code to a distributed memory machine. The compatibility of DOLIB with HPF has in principle been proved excellent, which may indicate that compatibility of similar previously developed libraries with HPF will not be an issue if these libraries are to be used in conjunction with HPF. This also suggests that numerous ways are available to implement a specific numerical method, e.g., matrix assembling in finite element codes. It is to the code developer’s discretion regarding the best way of achieving expected performance. For our specific application here, matrix multiplication in preconditioned conjugate gradient solver appears to scale reasonable well within the context of HPF alone. Nonetheless, like many newly developed technologies, currently available HPF compilers may still have lots of room for improvement.

ACKNOWLEDGMENT

This work was funded by the Environmental Technology Partnership program of the US Department of Energy. This work was also partially supported by the ORNL Partnership in Computational Sciences (PICS) program, supported by the Department of Energy's Mathematical, Information, and Computational Sciences (MICS) Division of the Office of Computational and Technology Research.

REFERENCES

Barrett, R., m. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozp, C. Romine, H. van der Vorst, 1994. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM, Philadelphia.

Beckie, R., 1996. Measurement scale, network sampling scale, and groundwater model parameters, Water Resour. Res., 32: 65-76.

Dagan, G., 1989. Flow and transport in porous formations, 465pp., Springer-Verlag.

D'Azevedo, E. F., and C. H. Romine, 1994. *DOLIB: Distributed Object Library*. ORNL/TM-12744. Oak Ridge National Laboratory.

D'Azevedo, E. F., and C. H. Romine, 1995. DOLIB: Distributed Object Library. In Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, SIAM, 1995.

D'Azevedo, E. F., C. H. Romine, and D. W. Walker, 1996. Shared-memory emulation enables billion-atom molecular dynamics simulation. In G. Astfalk (ed.*), Applications on Advanced Architecture Computers*, SIAM, 1996, p203-212.

Dykaar, B. B., and P. K. Kitanidis, 1996. Macrotransport of a biological reacting solute through porous media, Water Resour. Res., 32: 307-320.

Gwo, J. P., P. M. Jardine, G. T. Yeh, and G. V. Wilson, 1994. *MURF User's Guide: A Finite Element Model of Multiple-Pore-Region Flow Through Variably Saturated Subsurface Media*. ORNL/GWPO-011, Oak Ridge National Laboratory.

Gwo, J. P., P. M. Jardine, G. T. Yeh, and G. V. Wilson, 1995a. *MURT User's Guide: A Hybrid Lagrangian-Eulerian Finite Element Model Of Multiple-Pore-Region Solute Transport Through Subsurface Media*. ORNL/GWPO-015, Oak Ridge National Laboratory.

Gwo, J. P., P. M. Jardine, G. V. Wilson, and G. T. Yeh, 1995b. A multiple-pore-region concept to modeling mass transfer in subsurface media, J. Hydrol., 164, 217-237.

Gwo, J. P., and G.-T. Yeh, 1997. Application of a parallel 3-dimensional hydrogeochemistry hpf code to a proposed waste disposal site at the oak ridge national laboratory. this issue.

Haggerty, R., and S. M. Gorelick, 1995. Multiple-rate mass transfer for modeling diffusion and surface reactions in media with pore-scale heterogeneity, Water Resour. Res., 31:2383-2400.

Nerethieks, I, and A. Rasmuson, 1984. An approach to modelling radionuclide migration in a medium with strongly varying velocity and block sizes along the flow path, WRR, 20: 1823-1836.

Semeraro, B. D., E. F. D'Azevedo, and C. H. Romine, 1995. DONIO: Distributed Object Network I/O Library. Intel Supercomputer User's Group Meeting in Albuquerque, NM. Conference proceedings made available at www.cs.sandia.gov/ISUG.

Yeh, G.-T., 1987. *3DFEMWATER: A Three-Dimensional Finite Element Model of WATER Flow Through Saturated-Unsaturated Media*. ORNL-6386. Oak Ridge National Laboratory.