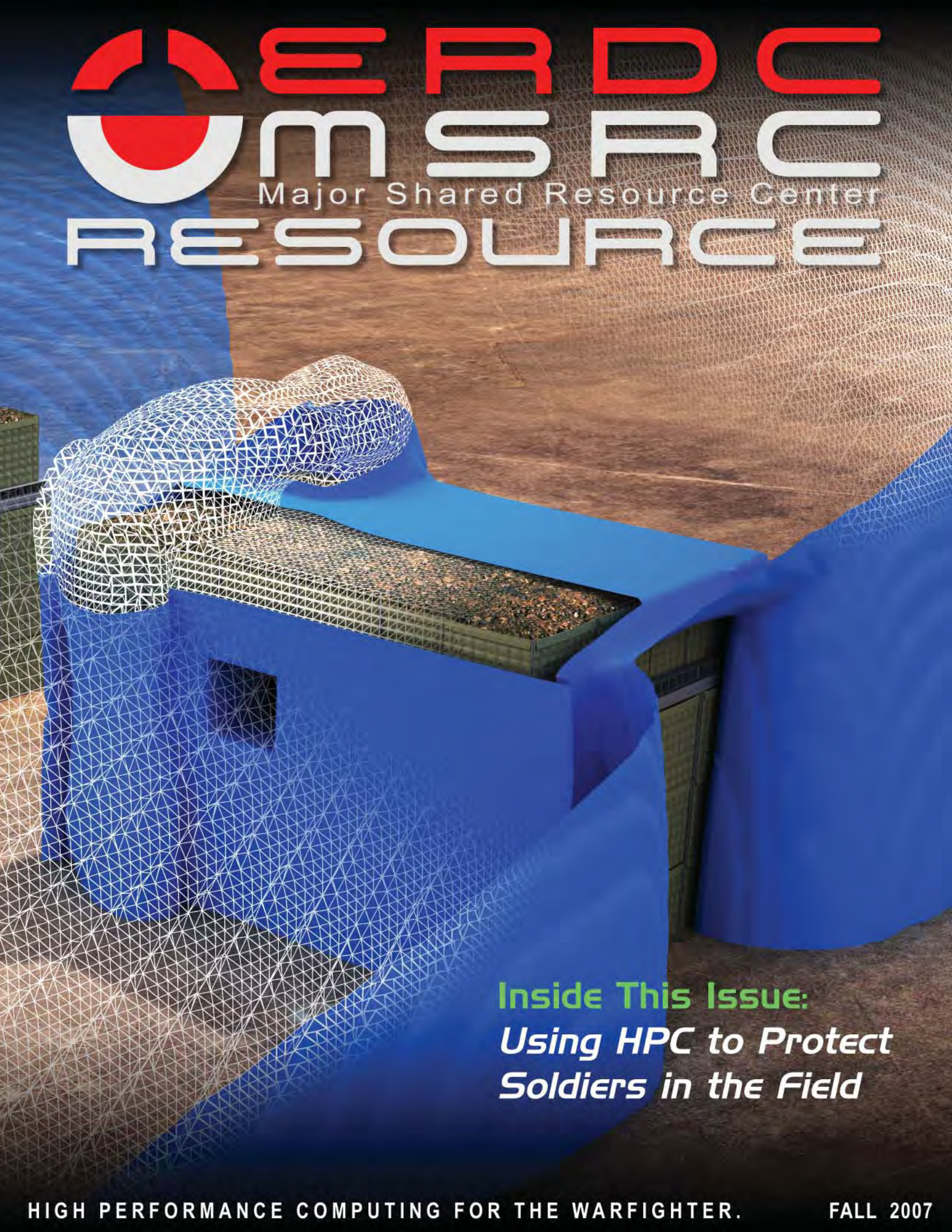
ERDC MSRC RESOURCE

Major Shared Resource Center



Inside This Issue:

Using HPC to Protect Soldiers in the Field

HIGH PERFORMANCE COMPUTING FOR THE WARFIGHTER.

FALL 2007

from the director . . .

Even though I have been part of the U.S. Army Engineer Research and Development Center Major Shared Resource Center (ERDC MSRC) since serving on the source selection team for the original MSRC contracts in the mid-90s, I only recently have come to appreciate the challenges that the Director of one of these Centers faces daily. I have realized during the past 6 months that just as John West indicated in the last issue of the *Resource*, that as I serve as the ERDC MSRC Acting Director, I am "working alongside an incredibly talented team" and that "it is truly a humbling experience."

David Stinson
Acting Director, ERDC MSRC

Just as it must always have seemed to past ERDC MSRC Directors, I too am aware that even during my short time at the helm, much has been accomplished at this Center and is continuing to be done so to provide the support for solutions to Department of Defense (DoD) problems that are too complex, dangerous, and expensive to solve any other way than with the use of high performance computing (HPC). As you read through the articles in this edition of the *Resource*, you will see evidence of this Center's continual attempt to stay focused on this important goal.

The *Resource* articles in this issue help to validate that we here at ERDC, through the DoD High Performance Computing Modernization Program (HPCMP), are dedicated to providing the latest and greatest machines for HPC, with the latest upgrades we have made to the Cray XT3 and the purchase of the very powerful Cray XT4, along with the new Sun/StorageTek SL8500 tape library for improving our storage capacity.

However, providing these monstrously powerful machines would be for naught without having the capability to harness them and get what we need from them. This is where our awesomely talented workforce comes in—our Computational Science and Engineering group (see Multiple Cores article), our folks affiliated with universities for technology transfer (see CaseMan article), and our visualization resources that aid DoD scientists and engineers in getting the most from their research (see feature article on Field Fortifications).

We also consider it part of our duty to aid in the important effort of ensuring that we have an intelligent, highly educated, and trained future workforce. We are supportive of any effort of reaching out to and support of students, especially in the fields of science and mathematics (see Future Generation article).

The bottom line for us is accommodating our users of this HPC facility. We want to support them in any way we can and join them in the ultimate dedication of providing the best support for the warfighter.

About the Cover: The front and back covers show three time-steps of a shock wave impacting a field fortification (see article, page 2).

Contents

DYSMAS Benchmark Calculations of In-Air Explosive Effects on Expedient Field Fortifications

By Michael J. Roth, William F. Heard, and Ryan D. Stinson, ERDC Geotechnical and Structures Laboratory; and Paul Adams, Kevin George, and Miguel Valenciano, ERDC Data Analysis and Assessment Center

High Performance Computing (HPC) helps protect soldiers in the field by aiding in the investigation of earth-filled revetment structures that are typical of expedient construction methods used in a hostile field environment for building security check points and other protective structures.

In support of the U.S. Army Engineer Research and Development Center (ERDC) survivability and protective structures research, the Survivability Engineering Branch, Geotechnical and Structures Laboratory, has conducted extensive studies on the performance of expedient protective structures constructed and occupied by military forces operating in a contingency environment. Common characteristics of these expedient structures include construction with nontraditional materials and exposure to a wide range of direct fire, indirect fire, and blast threats. Because of this, an understanding of the expected protective performance requires thorough study of the structures in a variety of attack conditions.

Over the last decade, one of the most prevalent contingency environment protective construction materials has been HESCO Bastion® revetment walls. Consisting of geotextile-lined wire baskets, the HESCO Bastion® material is transported in a low-weight, low-volume configuration and is filled with soil once in place to create a protective revetment wall. The soil-filled baskets are also often used to construct protective structures such as observation posts, an example of which is shown in Figure 1. The observation post shown is considered to be representative of a typical contingency environment field fortification used by the U.S. military in Iraq or Afghanistan.

To investigate the effects of a large explosive detonation on this type of structure, ERDC has performed a combined experimental and numerical research effort. In two international experiments conducted by the Australian Defence Force (ADF), ERDC gathered data on the internal and external pressure environment resulting from high-yield explosive events, as well as gathered structural response data to validate assumptions made in a concurrent modeling and simulation

Figure 1. Observation post

effort. In the modeling and simulation component of the project, numeric models were built with a computation fluid dynamics (CFD) code to simulate the experimental conditions and benchmark the computational results against collected data. With the code results validated, the numerical models could be used as a "virtual test bed" to consider the influence of variations in (1) charge weight, (2) standoff, and (3) relative orientation of structure to the charge—which would be prohibitive to do through physical experimentation.

Gemini (Wardlaw et al. 2003), a first-principles CFD code developed and maintained by the U.S. Naval Surface Warfare Center (NSWC) at Indian Head, MD, was selected for use in the numerical efforts. Gemini is the Eulerian component of the DYSMAS code suite (McKeown et al. 2004) and is coupled with the Lagrangian code DYNA-N to perform fully coupled fluid-structure interaction calculations. Gemini, within the framework of DYSMAS, has been used in many instances by ERDC analysts to simulate in-air explosive events because of the tight coupling algorithm that is used between the Eulerian and Lagrangian solvers. However, because the code is maintained by the Navy, its primary application has been for below-water simulations; subsequently, little data exist to validate results for in-air explosions. Therefore, the combined experimental and numerical effort described here provided an added benefit of generating benchmark calculations for Gemini simulation of in-air explosions.

To simplify the computational effort and focus benchmarking on Gemini, the calculations were limited to purely Eulerian, and the observation post was modeled with Gemini's "blocked cells" option. In Gemini, blocked cells are treated as rigid material, and their surface is perfectly reflecting. Representation of the observation post in this manner was based on an assumption that because of the structure's significant mass, the controlling hazard to occupants would be driven by internal pressure conditions and not by structural collapse. This assumption was verified in the experimental work, in which the structure shown in Figure 2 experienced internal pressure conditions that correlated with a high probability of lethality based on published physiological response data (Cooper 1996). However, as seen, the structure did not collapse.

Figure 2. Heavily damaged but stable structure

In support of the experimental program, Gemini calculations were performed to simulate the ADF multiton explosive event, in which multiple field fortifications were exposed to the resulting blast effects. Gemini modeling was performed in two separate stages: two- or three-dimensional (3-D) free-field calculations that were subsequently mapped into a 3-D model of the flow field and structure.

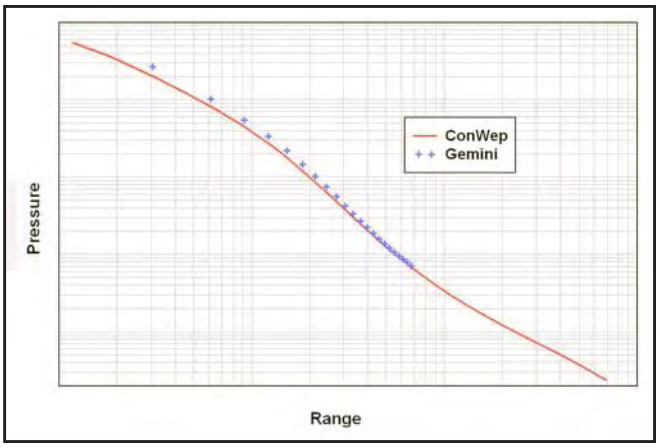


Figure 3. Empirical vs. DYSMAS (Gemini) comparison

Initial free-field calculations were performed in a 2-D, axisymmetric domain with the charge loaded into a quarter-circular region based at the domain's origin (thus representing a hemispherical surface detonation). The domain was discretized with a gradient mesh, and the total cell count was approximately 2.6×10^6.

After simulation of the hemispherical detonation, results were compared with experimental measurements. It was found that the simulation results did not match the measured pressure conditions. Peak pressure and maximum impulse differed by as much as 40 and 25 percent, respectively.

Simulation results were compared with empirically determined pressure-distance curves (Hyde 2004), which have long-been shown to be accurate for hemispherical charges. Figure 3 indicates good agreement was found.

Therefore, to determine the source of disagreement, the trial records were reviewed. It was found that—although the charge was initially planned to be a hemispherical charge—when built, it was actually in the shape of two stacked rectangles. Furthermore, instead of

Figure 4. Charge configuration

Figure 5. Simulation of nontypical charge shape and initiation mode

being center-point detonated as is commonly done, the charge was simultaneously surface-detonated at 36 initiation points. Figure 4 shows the charge being constructed.

With the recognition that the experimental charge was both built and initiated under nontypical conditions, the free-field calculations were repeated with the expectation of better capturing the overpressure conditions. In the repeat calculations, both 2- and 3-D domains were employed. A relatively small but finely discretized 3-D domain (cell size 3 cm, domain size 9 m by 9 m by 3.5 m) was used to simulate the detonation with a high level of fidelity and provide insight on the nonidealized conditions influence on the shock front formation. Figure 5 shows expansion of a shock front isosurface in the 3-D domain, and as seen, the nature of the front is noticeably different from that expected from an idealized hemispherical charge.

The free-field calculations were also performed in a 2-D domain to propagate the shock front to the appropriate standoff with less computational cost than that of the 3-D domain. In the 2-D domain, the charge was simulated as two stacked disks, with concentric initiation rings used to approximate the initiation points.

Figure 6 compares results from the 2-D domain with an experimental record in the free field. As seen, the calculated results reasonably match the experimental data, with differences in maximum pressure and impulse of approximately 14 and 7 percent, respectively. The

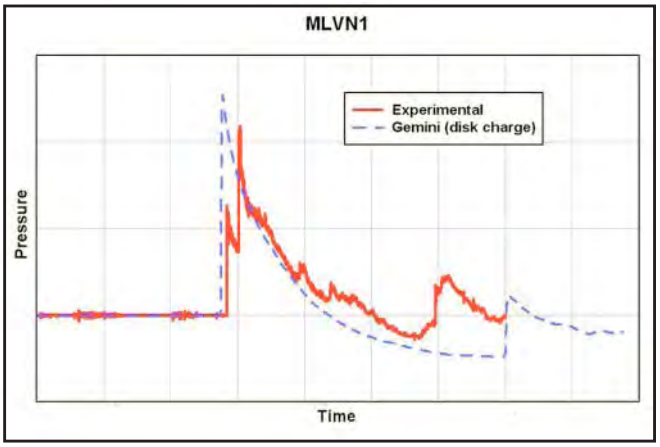


Figure 6. Experimental vs. DYSMAS (Gemini) comparison

primary difference noted in the records is the shape of the wave forms: the experimental record shows a second peak during the initial decay, whereas the numeric results do not show this second pressure rise.

To investigate the cause of the difference of wave forms and why the second rise was not seemingly captured in the numeric results, the shock front expansion prior to arrival at the gage was studied. On review of the numeric results, it was seen that, in fact, Gemini did calculate a nontypical shock expansion of similar nature to that measured in the experiment. As seen in Figure 7, because of the nontypical charge configuration

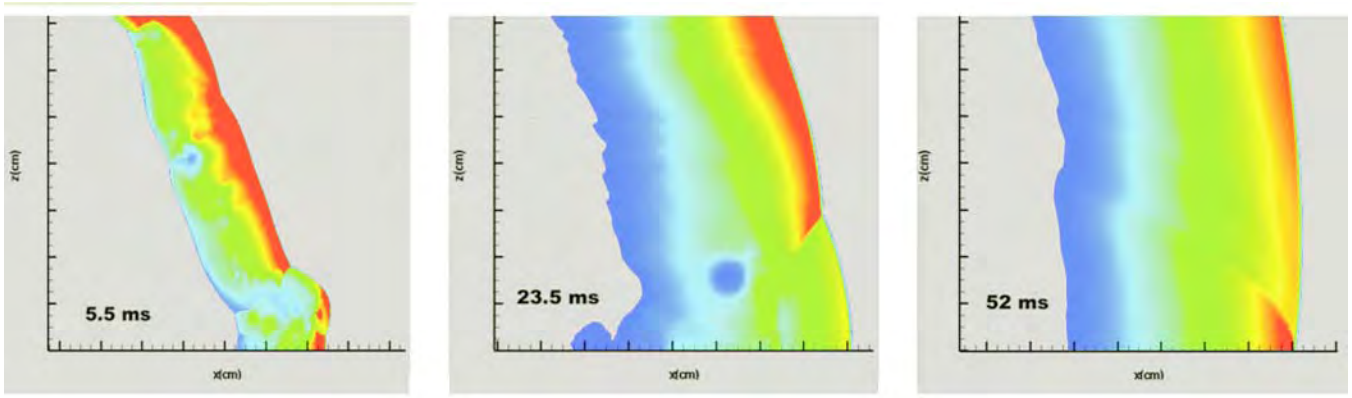


Figure 7. Pressure state plots, DYSMAS (Gemini) calculation of nontypical shock front expansion

and means of initiation, at 5.5 msec after detonation, several distinct wave fronts had formed. Near the ground surface, a small uniform front had formed beneath a faster moving, parabolic-shaped front. Above these, a larger, more uniform front had expanded and was more characteristic of a shock front that might be expected from a typical hemispherical charge. At 23.5 msec, the two lower fronts had converged into a single uniform wave, but the upper front still remained distinct and had begun to generate a downward moving wave into the lower, uniform zone. At 52 msec, with the ground wave front at a standoff approaching that of the experimental free-field gage, the downward moving front had reflected off the ground surface and generated a double pressure pulse, as seen in the experimental data. Based on the height of the gage and the timing of wave coalescence in Gemini, the output point corresponding to the experimental gage showed only the uniform wave front formed after the ground reflection. However, as seen from review of the entire pressure field, Gemini clearly captured the nontypical wave expansion measured in the experiment.

With the pressure environment in the free field accurately modeled, the 2-D pressure field was mapped into a 3-D Cartesian domain to calculate conditions in, and around, the structure. For the 3-D calculations, the domain was approximately 23.1 m (direction of shock flow) by 4.8 m (transverse to shock flow) by 3.6 m (height). Discretization of the domain resulted in a total cell count of approximately 1.9×10^6 cells. The structure (rendered view) during engulfment by a shock isosurface is shown in Figure 8.

In both of the ADF trials, ERDC constructed multiple structures with various standoffs from—and orientations to—the charge. Active instrumentation was used to measure the overpressure environment at specific locations within the structures, which was in turn used for comparison with Gemini results. Four pressure-time records from the experimental events are shown in Figure 9, along with time-domain shifted Gemini-calculated conditions at the same locations. Shifts of the Gemini data (in time-domain only) were minor, ranging from 5 to 8 percent, and were done to provide more direct comparison of the calculated versus experimental wave forms. As seen, Gemini closely matched all aspects of the pressure-time conditions. The difference (between experimental and computed) in peak pressure for these records ranged between 4 and 26 percent, and maximum impulse difference ranged between 2 and 18 percent. Furthermore, exceptionally close agreement was seen in the experimental and computed wave forms, showing that Gemini accurately captured the nature of shock flow into—and through—the structures.

Figure 9 shows that overpressure conditions computed by Gemini can be output at specific locations, providing detailed flow field information at discrete locations. However, state-plots (e.g., pressure, density) are also available in the Gemini output, providing more comprehensive insight into the conditions impinging on the structure and its occupants. Figure 10 shows pressure conditions during engulfment of an experimental structure. Shown is a cut-away view of the structure, with pressure contours plotted on the internal and

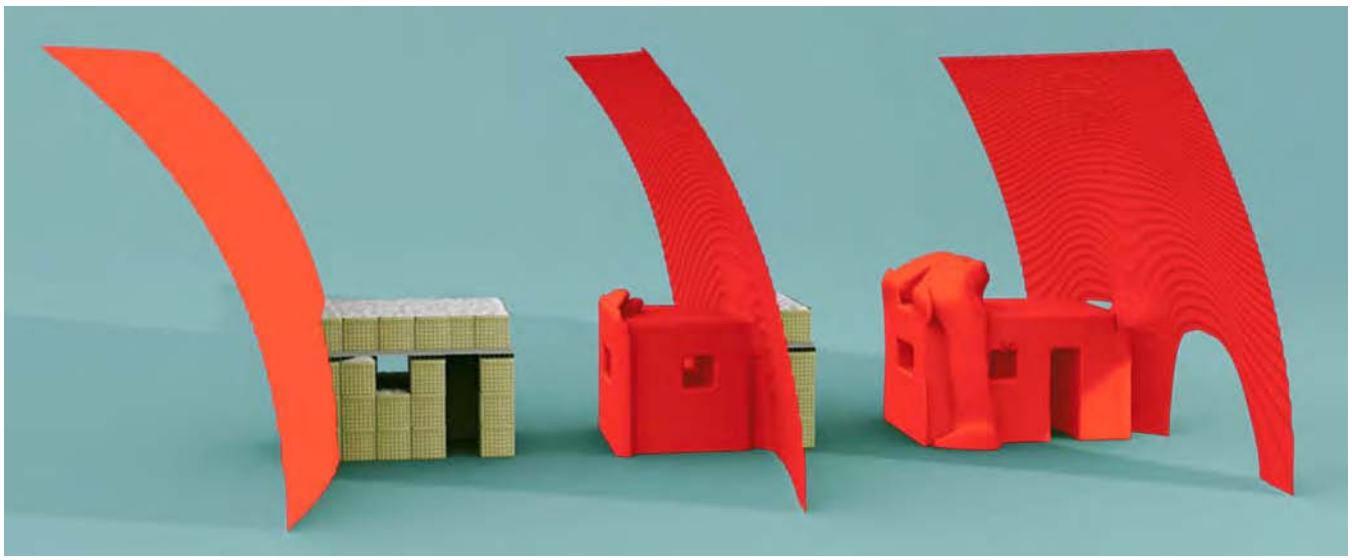


Figure 8. Shock isosurface engulfment of structure (with structure rendered)

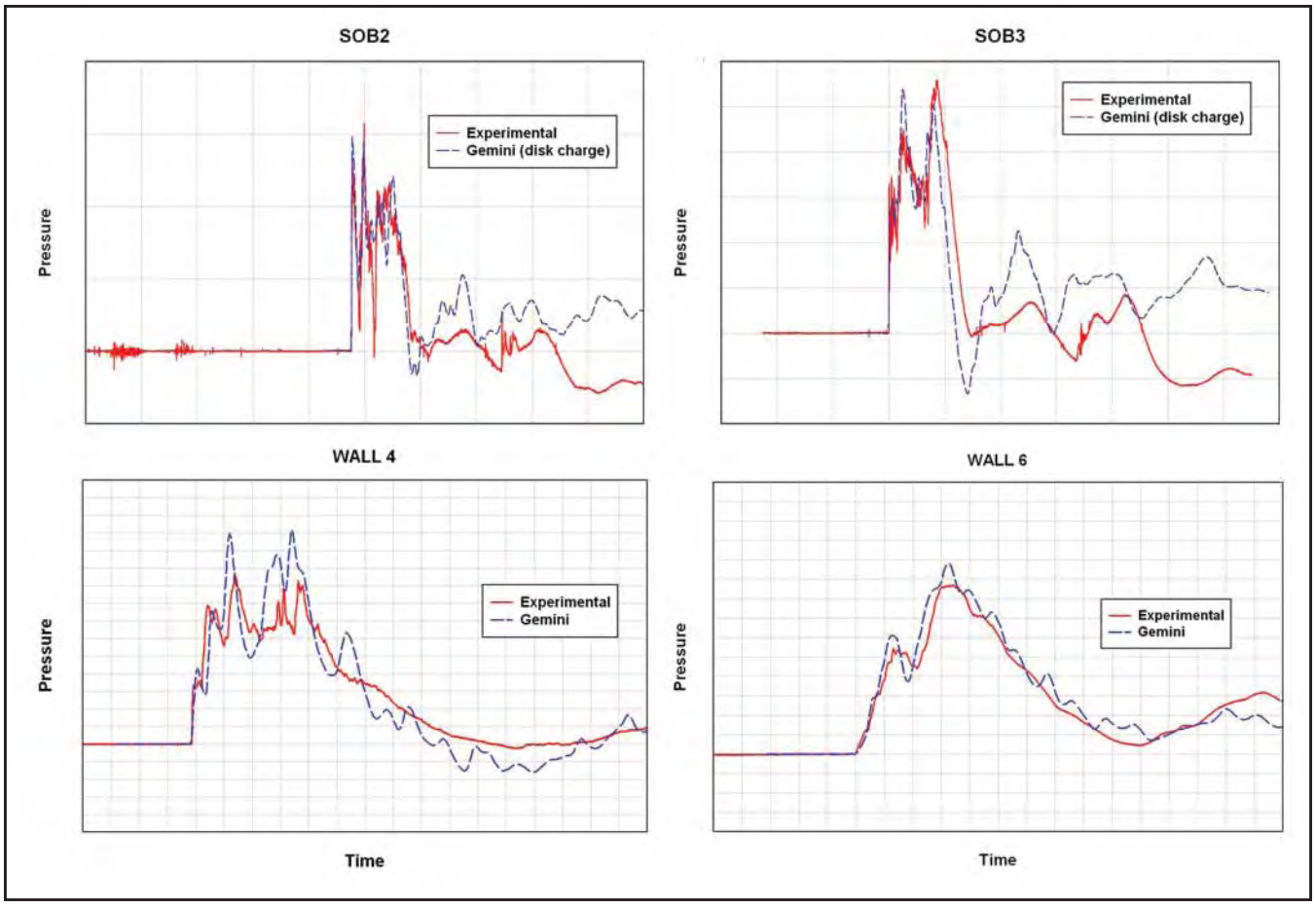


Figure 9. Internal conditions - experimental vs. DYSMAS (Gemini) results

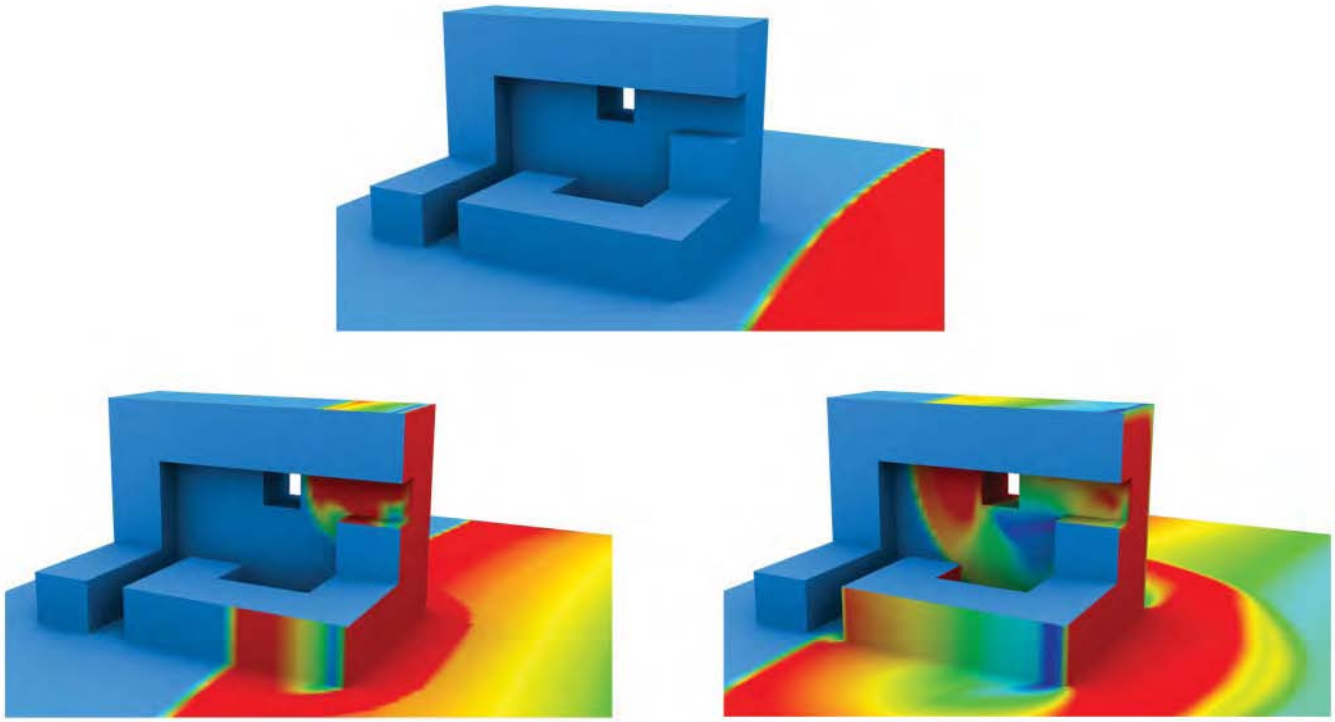


Figure 10. Visualization of structure engulfment, pressure contours

external structure faces. From this, the nature of the shock flow in, and around, the structure can be clearly seen. Furthermore, detailed studies of the flow conditions can be made, and if desired, the structure configurations could be modified in an attempt to improve survivability of position occupants.

This benchmarking effort shows that Gemini was capable of accurately modeling the in-air detonated free-field effects of both an idealized hemispherical charge and a nontypical charge configuration initiated with a multipoint surface detonation scheme. Mapping the free-field results into a 3-D domain containing the experimental structure further showed that Gemini accurately captured the internal pressure conditions—both in terms of pressure magnitude as well as nature of the wave forms. Only four internal pressure-time records were presented to evidence the accuracy of Gemini's calculations; however, additional data were available to researchers for more extensive comparison. Although in some cases the peak pressure magnitude agreement was not as close as that shown here, the computed wave forms closely agreed with the experimentally measured. Because of the wave form agreement, regardless of maximum pressure differences, it is believed that Gemini accurately computed the fundamental characteristics of the shock flow through the structures, thereby showing strong indication of its capability to accurately model complex shock flow resulting from in-air explosive events. Therefore, based on the results of this effort, it is recommended that additional opportunities be identified and exploited to make further experimental/numerical comparisons and extend the code's validation data set in applications of Army interest.

Acknowledgment

Permission to publish by the Director, Geotechnical and Structures Laboratory, is gratefully acknowledged.

References

Cooper, P. W. (1996). Explosives Engineering. Wiley-VCH. New York.

Hyde, D. W. (2004). "ConWep, Version 2.1.0.3" (computer program), IBM-PC, U.S. Army Engineer Research and Development Center, Vicksburg, MS.

McKeown, R., et al. (2004). "Development and Evaluation of DYSMAS Hydrocode for Predicting Underwater Explosion Effects," IHTR 2494, 13 February 2004.

Wardlaw, A. B., et al. (2003). "The Gemini Euler Solver for the Coupled Simulation on Underwater Explosions," IHTR 2500, 24 November 2003.

Our Newest Addition — The Cray XT4

By Jay Cliburn

ERDC is adding in 80,000,000,000,000 FLOPS for DoD users with the Cray XT4.

Pretty in camo: Conceptual rendering of Jade

The XT4 hostname will be Jade and will be housed in 24 equipment cabinets. In its final configuration, Jade will consist of 538 compute blades, each containing four quad-core 2.3 GHz Opterons, for a total of 8,608 compute cores. (The 2.3 GHz clock speed is an estimate and depends upon what's available from AMD at the time of Jade's delivery.) Each compute node will run Linux – unlike Sapphire, which runs Catamount on its compute nodes – and will be populated with 8 GB of memory, thus maintaining the 2 GB per core memory-to-CPU ratio found today on Sapphire. The system will contain over 370 terabytes of Lustre workspace disk storage. Jade also sports an improved internal node interconnect, the SeaStar2, which provides a sustained bandwidth of over 6 GB/sec. (By comparison, the older SeaStar on Sapphire provides 4 GB/sec of sustained bandwidth.)

Jade will be delivered with 76 dual-core service and I/O nodes, of which 32 will be configured for user interaction according to the table below.

The ERDC MSRC is excited about the computational capacity offered by this powerful new system and looks forward to bringing it into production service in the spring of 2008 to meet the needs of its users. Please don't hesitate to contact the Consolidated Customer Assistance Center (CCAC) if you have questions or need additional information. CCAC can be reached at help@CCAC.hpc.mil or telephone 1-877-222-2039.

Cray XT4 Anticipated Login Node Configuration

Node	*Quantity*	*Purpose*
jade01-06	6	user remote login
jade07	1	user remote login, compiler license server
jade08-13	6	batch interactive (user login through batch system)
jade14-19	6	utility (10 Gbit network connection to mass store)
jade20-31	12	batch job control and management (no user login)

Say *Goodbye* to Old Systems and *Hello* to New Storage

By Jay Cliburn

In the past few months, the ERDC MSRC has bid farewell to one very familiar system and significantly modified another not-so-well-known system.

SGI Origin 3900

Most noticeably, we've decommissioned the SGI Origin 3900 system known as Ruby. Ruby was actually comprised of multiple systems: two 512-processor nodes called Silicon and Sand, and a front-end login host that was the actual physical host called Ruby. These systems were kept in service longer than scheduled to accommodate user needs. (There were other smaller systems, too, but they had support roles with which users never knowingly interacted.) Silicon and Sand entered service at the ERDC MSRC in fall of 2003 and proved to be remarkably stable workhorses, routinely exceeding 90 percent monthly utilization and providing users with a vast quantity of shared memory.

In a configuration change much less visible to the user community, but significant nonetheless, the ERDC MSRC replaced its three StorageTek 9310 tape silos with a new, single Sun/StorageTek SL8500 tape library in May 2007. The old 9310 silos had been on the floor for well over a decade and contained a total of about 16,500 slots for tape cartridges. The new silo contains "just" 10,000 slots, but in a significantly smaller footprint. Each tape cartridge in the new silo holds over twice as much data as its predecessor (500 gigabytes versus 200 gigabytes).

The ERDC MSRC is pleased to provide new mass storage capacity to its users. If you have questions or comments, please contact the Consolidated Customer Assistance Center (CCAC) at help@CCAC.hpc.mil or telephone 1-877-222-2039.

StorageTek 9310 tape silos

Sun/StorageTek SL8500 tape library

Getting the Most from Multiple Cores on the XT3

By Tyler Simon

Sapphire has become a multicore system. Find out below how to improve your code performance.

Commodity multicore chips have become an integral part of high performance computing architectures. As processor vendors move towards concurrent process execution on a single chip, the software developers for these systems can no longer rely on increased processor frequency to lead to increased application performance. Rewriting an application to take advantage of multicore chip architecture is a good start at improving performance. However, users must also become more aware of the resource demands of the multicore computing system and runtime environment for best code performance. This article provides a brief overview of three areas where a developer or user can potentially improve application performance on the ERDC Cray XT3 (Sapphire) and other multicore systems. Additionally, by understanding some common areas of contention and performance bottlenecks in existing dual-core hardware, users may be better prepared to make more detailed improvements to their code and prepare for the upcoming quad-core Cray XT4 to be installed at ERDC. The following recommendations come from experiences running codes on the single- and dual-core Sapphire.

Sapphire Overview

Currently, Sapphire contains 4,160 processing nodes with each node running a 64-bit, 2.6 GHz dual-core Opteron processor with 4 GB dedicated memory. The nodes are connected to each other in a three dimensional (3-D) torus using a Hyper Transport link with a dedicated Cray SeaStar communications engine. Sapphire is rated at 42.6 TFLOPS and contains 374 TB of Fibre Channel RAID disk storage. Sapphire runs the UNICOS 1.5.39 operating system with the Catamount microkernel running on the compute nodes. Service nodes run a full SuSE Linux distribution with Cray XT3 extensions. The pre-upgrade system specifications included the 1.4.43 version of UNICOS and 4,096 nodes of 2.6 GHz AMD Opteron processors, with one core per node and 2 GB of user-accessible memory.

Areas of Contention and Solutions

Memory Contention at the Chip Level

In order to get a code to perform better on multicore processors, an understanding of Sapphire's dual-core memory hierarchy will be helpful. On Sapphire the L1 cache is divided into a 64K data cache and 64K instructions cache, with 1 MB of L2 cache. Each core has access to a single pool of main memory; thus the main memory bus becomes a main point of contention, especially for memory-intensive codes, as each core must be scheduled for individual memory access. Figure 1 shows main memory read+write access time on the single- and dual-core chips on Sapphire. The test increased the cache stride size in bytes (x axis) and calculated the read and write time. Figure 1 depicts the dual-core memory access time is greater than single core, thus quantifying the effect of memory contention on Sapphire's AMD Opteron chips.

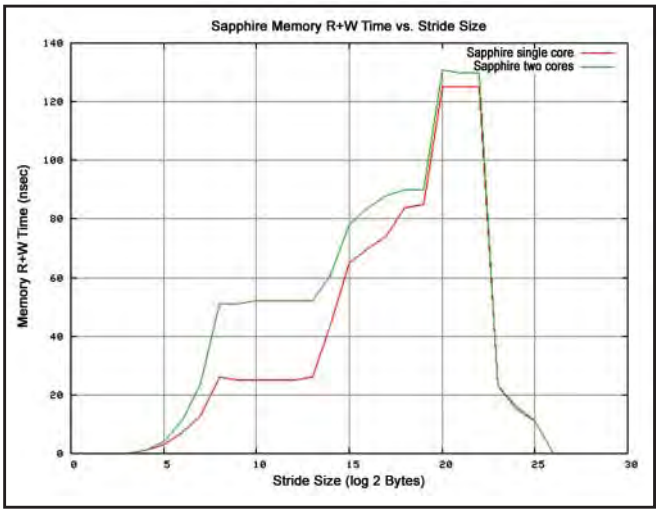


Figure 1. Single- and dual-core memory read and write access time for variable block size

As a developer, off-chip memory contention may be alleviated by fitting arrays primarily into cache. When this is not feasible or the array size is larger than cache, the users should reduce the number of cores that access such data. Thus a process scheduling solution may benefit code execution time by specifically limiting or interleaving core-to-memory access.

Process Affinity at the OS Level

Each dual-core node retains a single operating-system image; thus each execution thread must be assigned an execution core and scheduled by the OS. This process can be examined by looking at the current OS scheduling algorithm and by taking advantage of process affinity. Process affinity allows a user to map a process ID (pid) to a core for execution. A user can gather which SMP scheduler the current OS is using via the `sched_getscheduler(pid)` function. A user can view the core a process is intended on running by

viewing the process affinity mask using **`sched_getaffinity (pid, len, &mask)`** where **`&mask`** is returned as the core **`umask`** of the process ID and **`(sched_setaffinity (pid, len, &mask)`** will allow the user to set this value to the appropriate core. A more dynamic or adaptive approach to process affinity may provide more efficient use of the additional core.

MPI Process Placement at Runtime

One of the effects of running MPI over multicore nodes is that MPI ranks have the chance of being placed on the same node, thereby improving the bandwidth for those particular ranks. The difficulty then becomes how to properly map MPI ranks to execution cores for optimal throughput and reduced overall job runtime. As an example of this behavior, Figure 2 demonstrates the throughput between ranks in a persistent blocking all-to-all MPI on the 2.6 Dual Core AMD Opteron system at the Arctic Region Supercomputing Center (ARSC). Each node on this system contains 8 dual-core chips. The increased throughput is visible here in groups of 16, as expected. A user can exploit this behavior only at runtime, as the physical node location is often non-deterministic. I propose the following method, which can be used for taking advantage of increased internode bandwidth for the ERDC XT3.

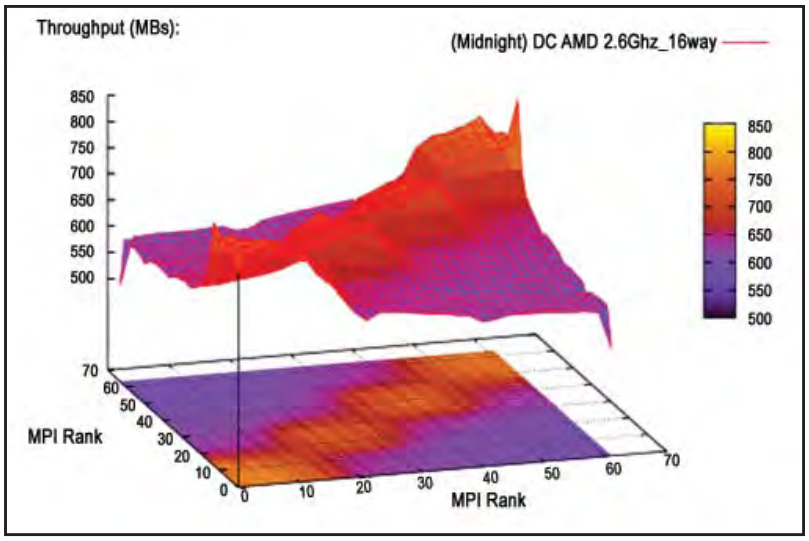


Figure 2. Examines the MPI bandwidth between ranks for a 64-node run on the ARSC Midnight Cluster

On Sapphire, once a user submits a job, it is generally run on any available processors, whether they are contiguous or not. Thus users have little control of their MPI rank to compute node placement. Users may specify a MPICH_RANK_REORDER_METHOD in their batch submission script to attempt different process placement strategies, but any benefits will depend on the communication patterns of the code. For example, Figure 3 shows GAMESS runtime is improved as a symmetric multiprocessor and folder rank reordering method is used, as opposed to the default round-robin placement strategy. Taking this idea even further, users can specify any rank ordering the setting MPICH_RANK_REORDER_METHOD=3 in the batch submission script. At job runtime the file "MPICH_RANK_ORDER will be read, and ranks will be placed in the order specified, such as "0,1,3,2" for a 4-node job.

For codes with specific data-locality needs, Sapphire allows for "yod" to be executed with an ordered list of nodes using the "yod –list" option. Thus the user can combine the MPI_RANK_ORDER file with a specified list of nodes to run on a more custom process topology. In the following batch script, the first yod runs a program that prints out the node id's and MPI ranks to a file. Users then can perform some selection criteria on how they want those nodes ordered as well as saving their associated ranks into the MPI_RANK_REORDER file. This example demonstrated just a sort routine based on the numerical value of the fourth column. The script then waits 5 seconds for the job to complete, creates the list of nodes, and submits another job with "myexecutable" for the modified topology.

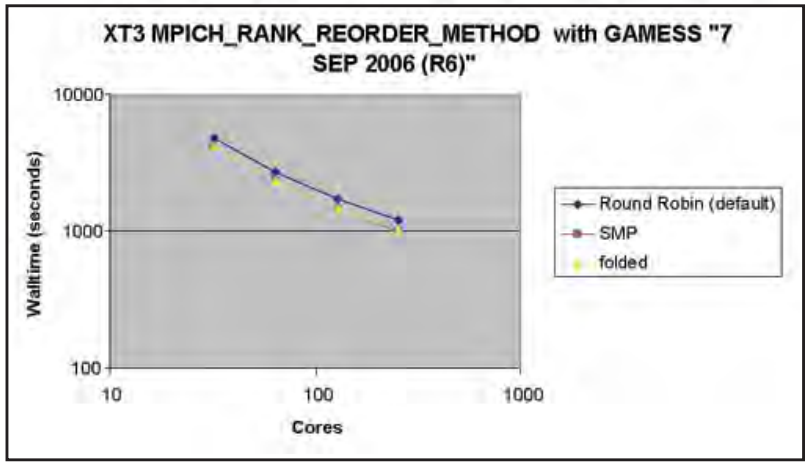


Figure 3. MPI process placement

```
###Pretest code to generate node id's
yod -VN -np 4 ./nodelist >>nodelist_4_$PBS_JOBID.in


###Reorder based on some metric!
cat nodelist_4_$PBS_JOBID.in | sort -nrk4 | head -4 | > MPICH_RANK_ORDER


###wait for yod to cleanup
sleep 5
nodelist=(`cat nodelist_4_$PBS_JOBID.in | sort -nrk4 | head
-4 | awk '{printf("%d,", $1);  }'`)


###Add your executable here!
export MPICH_RANK_REORDER_METHOD=3
yod -list ${nodelist[@]} -VN -np 4 ./myexecutable >reorder_$PBS_JOBID.out
```

Conclusions

Some codes tend to do better on multicore systems with little to no modification; these tend to be codes that have a little memory contention or have non-uniform process needs, such as in GAMESS. Figure 4 shows a comparison of runtimes on single- and dual-core Sapphire. LAMMPS is computationally intensive with little memory access. The codes used to evaluate the performance of the XT3 are a subset of the benchmarks that are used in the High Performance Computing Modernization Program (HPCMP) Technical Insertion (TI) procurement process and also represent the HPCMP computational technology areas. Each code was executed with a fixed problem size on the single- and dual-core Sapphire nodes, with the dependent variable being runtime. Each code was compiled with the PGI compilers with the default compiler optimization levels set "-02". The improvement seen in GAMESS is due to the processor upgrade as well as a code revision, from R4 to R6.

In conclusion, multicore architectures are becoming more common in high performance computing environments, and more traditional methods of code performance gain will not work as well for current HPC computing environments. Thus, developers need to educate themselves in these new processor and compiler technologies and customize their runtime and code development practices around them.

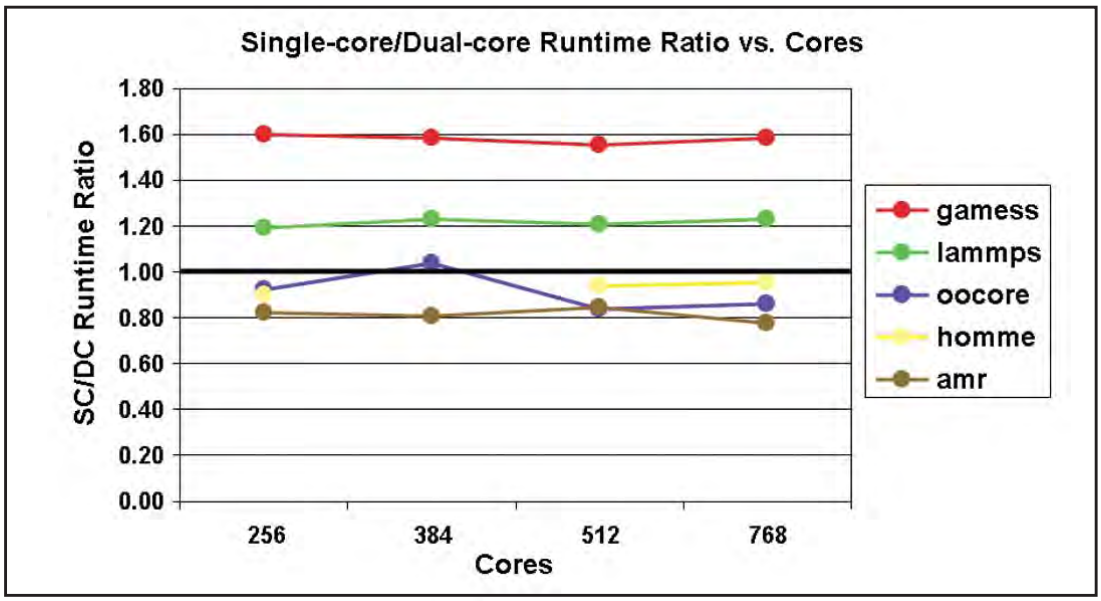


Figure 4. Ratio of single- and dual-core application runtimes

Lustre: Five Things That Can Make it Work Effectively

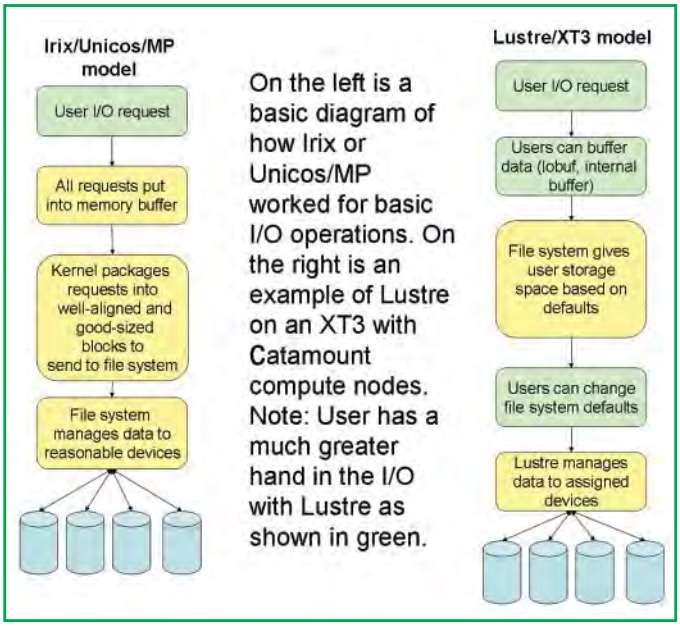
By John Salinas

The last time I heard John West give a talk, he spoke about how we need to make supercomputing easier for users. Since the room was full of bright people, I felt confident that they would go and make this happen while I went happily back to doing whatever I was doing. But as my week progressed, I kept receiving problems that dealt with how users were using the Lustre file systems on Sapphire (ERDC Cray XT3). Many aspects of what they were trying to do were ill-suited for the place they were trying to do them – the thought occurred to me that we might be able to simplify the life of some users if we gave them some basic guidelines on how to use the Lustre file systems to their advantage. The purpose of this article is to provide information to users that will help them use Luster file systems more effectively.

Basics of a Lustre File System

Lustre is designed to be a high performance scalable file system that runs over a wide variety of configuration. It is also designed to be easily configured by users to meet their specific needs. This is a change from many older file systems that were designed to hide all the complexity of input output (I/O) operations from the user (See graphic on the right).

The Lustre implementation on Sapphire has a client reading and writing data from a Catamount compute node over the high-speed network to object storage servers (OSSs) that contain object storage targets (OSTs). The metadata server (MDS) interacts with the OSTs and the client to keep track of files, directories, and file system information. A basic overview looks something like the following:



/work	221 TB on 111 OSTs on 27 nodes connected via 4 GB fiber channel host bus adapters to Data Direct Networks (DDN) 9550 fiber channel disk. This file system is configured for performance throughput. In a production environment a user can expect to be able to write about 300 GB a second per stripe (OST)	Default stripe count of 2 OSTs per file and a default stripe size of 1 MB.
/work2	92 TB on 47 OSTs on 11 nodes connected via 2 GB fiber channel host bus adapters to DDN 8500 fiber channel disk. This file system is configured for capacity. In a production environment, a user can expect to be able to write about 170 GB a second per stripe (OST).	Default stripe count of 6 OSTs per file and a default stripe size of 1 MB.
/u	User home file system of 540 GB home file system served via Network File System (NFS) (slow performance)	
/ptmp	User semipermanent 540 GB file system via NFS (slow performance)	
/tmp	Not for users to use on very small ram file system that is local to each login node. Please use *$tmpdir* to get a unique temporary directory.	

The client and the OSSs, which contain OSTs, take care of the data, file locking, and acknowledgment of packets being sent back and forth. The OSTs and the MDS take care of file creation, file status, and recovery. Communication between the MDS and the client ensures concurrency and directory metadata. For more details on what each component does and what it connects to, see the Lustre documentation: http://manual.lustre.org/manual/LustreManual16_HTML/DynamicHTML-01-1.html.

Choosing a File System

We have established that the Lustre file systems have defaults that are picked up each time a file or directory is created. These defaults are not as well suited for a wide range of I/O operations as many previous types of file systems were. It is in the users' best interest to consider what the defaults are so they know where to run their code. If this information is not published in a guide, it can be obtained by creating a file on the Lustre file system in question and then running the following:

```
lfs --verbose getstripe filename
```

Printing out information about the defaults would be helpful. The following shows how the file systems on Sapphire look:

/work	221 TB on 111 OSTs on 27 nodes connected via 4 GB fiber channel host bus adapters to Data Direct Networks (DDN) 9550 fiber channel disk. This file system is configured for performance throughput. In a production environment a user can expect to be able to write about 300 GB a second per stripe (OST)	Default stripe count of 2 OSTs per file and a default stripe size of 1 MB.
/work2	92 TB on 47 OSTs on 11 nodes connected via 2 GB fiber channel host bus adapters to DDN 8500 fiber channel disk. This file system is configured for capacity. In a production environment, a user can expect to be able to write about 170 GB a second per stripe (OST).	Default stripe count of 6 OSTs per file and a default stripe size of 1 MB.
/u	User home file system of 540 GB home file system served via Network File System (NFS) (slow performance)	
/ptmp	User semipermanent 540 GB file system via NFS (slow performance)	
/tmp	Not for users to use on very small ram file system that is local to each login node. Please use *$tmpdir* to get a unique temporary directory.	

With this information, users can refer to the five basic guidelines below to help decide where to run:

1. The basic principle is to use as few stripes as possible to accomplish good performance on the application. The more stripes that are used, the more overhead, contention, and risk are involved.
2. Small ASCII text files need to be buffered and put on one of the Network File Systems (NFSs). The more small transfers that are done, the more time is taken away from OSTs.
3. If one I/O client (one CPU) writes one large file of well-aligned, large I/O, */work2* should be used where the default stripe size is six. Because the stripe size is six, the pipe to disk is three times that of /work where the default stripe size is two. This means better performance for a small number of clients writing one file.
4. If multiple files are reading/writing I/O, then the default stripe of two on */work* will likely be the best option, as two OSTs are provided for each file.
5. If a large number of files are opened for reading/writing and they all need a high performance file system, */work2* is used if the number is less than 192, and */work* is used if the number of open files is between 192 and 443. The formula is to try and not use more than ~4 times the number of OSTs on the system. If there is a need to run over 444 files, turning off striping is suggested by using `lfs setstripe testfilename 0 0 1` (file name stripe size, start OST and stripe count).

For specific examples, see the Sapphire I/O User Guide, which is on the ERDC MSRC Web site at www.erdc.hpc.mil.

Configuring Lustre

Since most applications have many different I/O operations and files, it is unlikely that any one file system will have defaults that will work well for every file. Generally, users want to find their most I/O-intensive files and find the file system that best meets those needs. Then they can make files and directories that meet the needs of their other files. Users can select `lfs setstripe` to change the file system defaults to suit their needs:

```
lfs setstripe largefile 0 -1 4
lfs setstripe smallfile1 0 -1 2
...
lfs setstripe smallfile10 0 -1 2
```

In this example, `lfs setstripe` is used to create a file called `largefile` with a default stripe size (1 MB), a default start OST (rarely desirable to change this), and a default count of four OSTs. The second file would be created called `smallfile` with a default stripe size and start OSTs, but a default of two OSTs being used. Using this same process, users could create 10 files that are all like this, having 11 files, one with a stripe count of four, the rest with stripe counts of two. If users were running on */work* on Sapphire, they would only have to change the default for `largefile` to use four OSTs, and the rest could pick up the default of two OSTs. To the right is a basic chart to help give guidelines on how many OSTs to use per file.

File Size	Sustained I/O per Client	Stripe Count
Under 2 GB	100 MB/sec or below	1
2-12 GB	150 MB/sec or below	2
12-32 GB	225 MB/sec or below	4
32-128 GB	300 MB/sec or below	6
128 GB and over	Greater then 300 MB/sec	8-10

The basic rule of thumb is with multiple files that are all under ~12 GB, the default stripe count of two on */work* is a good default. The goal should be to set up each run with file system options that will be the best for each file. Remember that OSTs are a shared resource. If an application uses them poorly, multiple users can be affected.

Performance Information

Cray PAT Setup

Cray provides performance analysis tools that can help users better understand their application. However, they behave differently on a Lustre file system, such as */work*, from an NFS such as */u*. Multiple processor jobs require the ability to do record-locking. This means that if users are running any parallel applications, they need to run them for either */work* or */work2*. Also, the NFSs have a limit of 1,024 files that can be opened at the same time. To get around this, users should run from a Lustre file system (*/work* or */work2*). If it is not possible to run the code on a Lustre file system, then users can set PAT_RT_EXPFILE_DIR runtime environment variable to redirect CrayPat output to a target directory on the Lustre file system:

```
export PAT_RT_EXPFILE_DIR=/work/username/dirname
```

Using Pat Build and Pat Report

The first step is to instrument the code using the `-g` trace group option to select a relevant experiment. The options for I/O are as follows:

io	Includes the stdio and sysio groups
stdio	All library functions that accept or return the buffered I/O

`(FILE *) construct`

sysio	System I/O calls
system	System calls

```
module load craypat
(remember pat_build needs access to .o files)


pat_build -g io IOR

ls -lart IOR IOR+pat
-rwxr----  1 jsalinas erdcssta 12028665 Sep 10 11:57 IOR
-rwxr----  1 jsalinas erdcssta 12263601 Sep 10 11:57 IOR+pat
```

The second part is to run the program **qsub** to submit with `yod ./IOR.exe-instr`. After the job completes, run `pat_report`.

```
pat_report -O write_stats IOR.exe-instr+136
              (also a read_stats)
Data file 24/24: [....................]
CrayPat/X:  Version 3.0 Revision 210 (xf 73)
06/20/06 16:28:30

Experiment:  trace
 ...etc...
```

The report will give information on how long it took to write something, how much I/O was done, what the rate was, etc. This is often helpful if combined with other `pat_build -g tracegroup` options to find out what percentage of time is spent where.

If running with **Iobuf**, users can get it to tell them what it knows about the files it has been monitoring:

```
% setenv IOBUF_PARAMS '*:verbose'
% ftest2 input2
         I/O time                          15.32200
         Compute time                      53.26200
         Total time                        68.58400
         I/O time per iteration           0.1532200
         Compute time per iteration       0.5326200
         Total time per iteration         0.6858400
         Total I/O (bytes)                800000000
         I/O rate (MB/s)                   49.79373

        File "input2"

              Calls         Seconds       Megabytes   Megabytes/sec

        Open      1        0.000006
        Read    201       15.318910      762.940216       49.803818
        Close     1        0.000006
        Total   203       15.318923      762.940216       49.803778
        Sys Read 49       78.300950      762.940216        9.743690
```

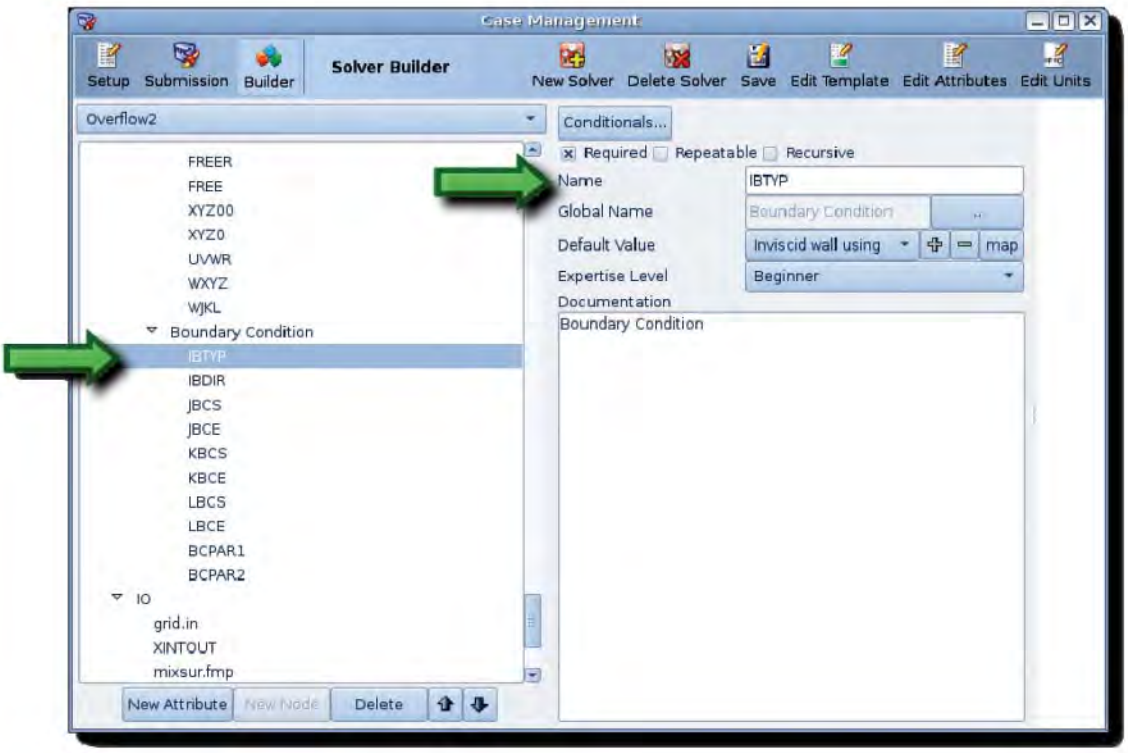
CFD Made Easy with CaseMan

By Dr. Alan Shih, Marcus Dillavou, Corey Shum, Fredric Dorothy, and Dr. Bharat Soni, University of Alabama, Birmingham

CaseMan takes care of all the details for the HPC user.

With the advent of computer hardware and numerical algorithms, computational fluid dynamics (CFD) has become a reliable and effective tool for performance prediction in the design selection process. It also holds great potential for design optimization for large-scale and complex designs. It can be used to acquire a large number of design points that traditionally relied upon expensive and tedious experiments. However, despite all the potential that CFD has for better designs in a more cost-effective manner, it is still hampered by the need for large amounts of central processing unit (CPU) time on a sophisticated high performance computing (HPC) environment to iteratively solve a set of nonlinear governing equations called the Navier-Stokes equations for a single-phase flow. When chemical reaction or other more sophisticated physics models are also needed for complex flow fields, such demand for CPU time increases further. CFD solvers also require a user to have fairly extensive experience and knowledge in order to use them correctly and effectively. This poses major challenges in terms of accuracy, throughput, and cost-efficiency when using a CFD tool to acquire important performance data. Compounded with these challenges are the less trivial Linux/Unix working environments and job queuing systems on the HPC systems on which CFD cases are usually calculated. This is especially challenging for most of the novice CFD users who are more familiar with the single-user, graphics-driven Windows environment instead of the command-based and script-based Linux/Unix operating systems.

Accuracy improvement, throughput increase, turnaround time, and cost reduction are the key challenges to CFD use in the design process. Solution throughput must be significantly improved in the generation of aerodynamics, propulsion, and fluid dynamics simulations that involve parametric and sensitivity design studies. Such parametric and sensitivity studies require a large number of CFD simulation runs. However,



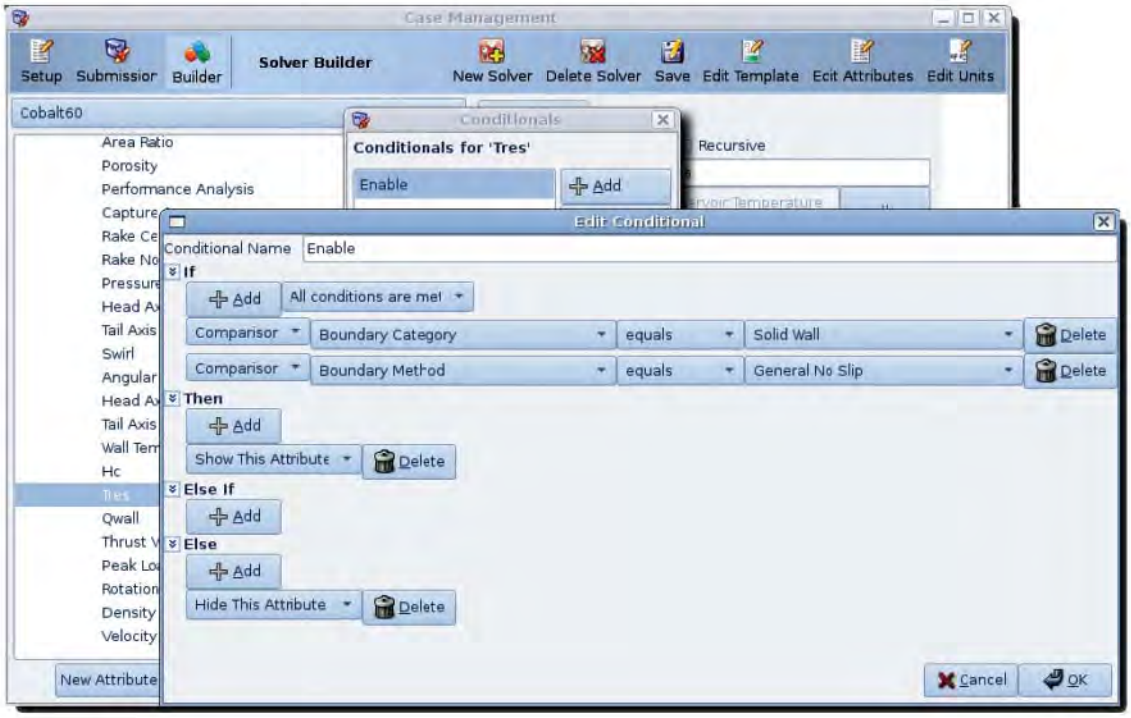
Flow Solver Builder module allows the user to specify the variable types and names that the flow solver will need. It can be used to specify the value ranges to prevent users unknowingly specifying invalid numbers

performing a CFD simulation is a challenging problem for both novice and experienced users, as they must deal with the preparation of the simulations by specifying different input parameters; the complexity of running these simulations on different environments of the DoD Major Shared Resource Center (MSRC) HPC systems; and managing the output produced by these simulations. Despite the fact that most of the CFD codes used by DoD users represent state-of-the-art technologies with excellent parallel performance, the current overall simulation process falls short of achieving the required throughput. A productivity enhancing toolkit called "CaseMan," supported under the DoD High Performance Computing Modernization Program (HPCMP) User Productivity Enhancement and Technology Transfer (PET) program (Project CFD-KY7-001), is currently under development with the alpha version available to the DoD user community. This framework allows the user to prepare, submit, monitor, and manage a large number of CFD simulations on the DoD MSRC HPC systems and other non-DoD cluster systems.

Overview of CaseMan

CaseMan is a tool designed to make setting up, submitting, and monitoring CFD jobs simple and easy. To achieve this, CaseMan abstracts out solvers and computing environments and only presents them to the user in a high-level, intuitive way. No longer do users need to edit complex input files, write submission scripts, or learn the intricacies of each solver and environment. Instead, CaseMan takes care of all these details for the user. The immediate impact is to allow the user to quickly prepare the simulation input files and make complex supercomputing tasks associated with CFD much more user friendly, as CaseMan shields the user from the complexity of utilizing HPC systems.

On startup, the user selects which solver to set up. A simple interface with all the necessary parameters the user needs to input is presented. The interface by default shows short descriptions of variables instead of the solver variable name (i.e., Mach Number instead of MACHNO), but it is possible to switch the view to show the solver variable names. Each input is fully documented and is also statically typed, which means that CaseMan knows what type of variable the user needs to enter (string, integer, floating point, etc.). Conditionals are also set up on variables that can cap values, warn the user about certain situations, or disable choices that are not valid in certain situations. CaseMan also has recommended values for each variable, which gives the user a good starting place. Users can also set their preferred skill level. If the user chooses a beginner or intermediate skill level, then



Flow Solver Builder module also allows the user to establish the conditionals relationship

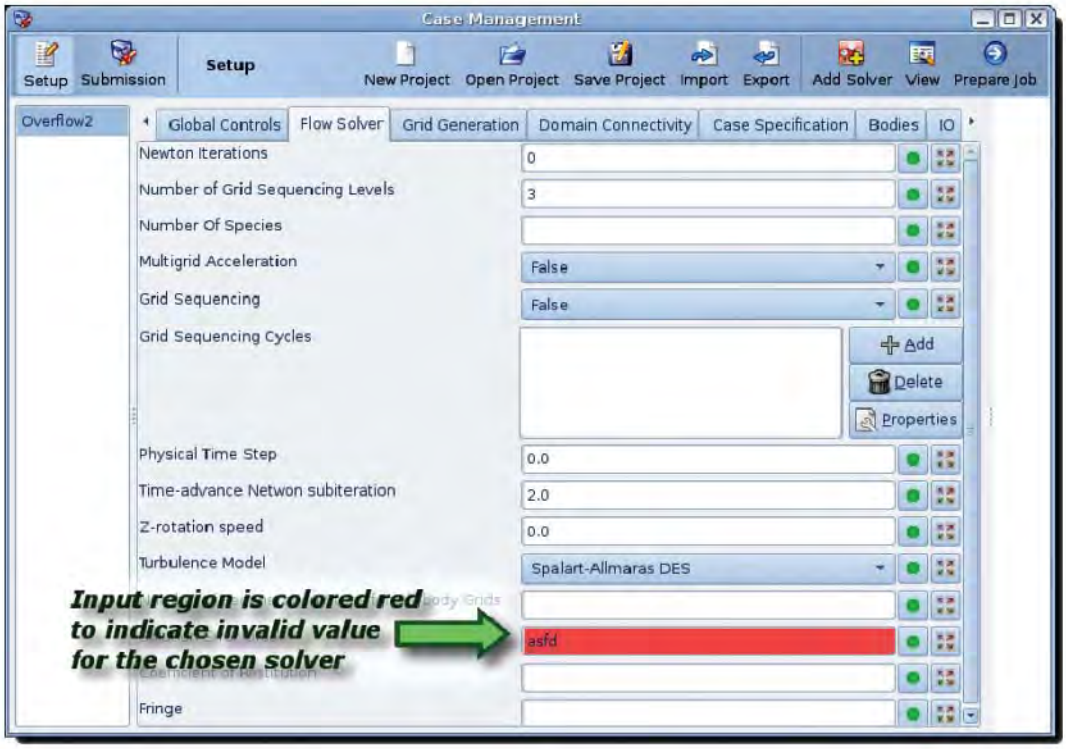
variables that are not required are hidden from the user to keep the interface simple. This feature enables casual CFD users to use CFD solvers to obtain important design performance data without a steep learning curve.

CaseMan also manages all the necessary files for a job. After setting up a job in CaseMan, the user-entered input is automatically converted to the input file(s) for the specified solver. On submission to an HPC system, all required files, including the input files, grid files, restart files, and any other file the user has specified, are automatically transferred to the HPC system. On the HPC side, the directory structure is automatically created; all files are automatically written to their correct location and filename; and submission scripts are automatically generated. CaseMan interacts with the queuing system to submit and monitor the jobs and also knows how to associate with MPI or other required libraries to run the solver. As the job runs, CaseMan constantly monitors the job status. This allows CaseMan to warn the user immediately if there is a failure in the simulation job. CaseMan also extracts lightweight data as the job runs. These data can be plotted and visualized in real time on the client machine, allowing the user to check the convergence history. These data could also be used to steer the job or stop the job if it is not converging.

Current Features of CaseMan

In its third year of development, CaseMan will bring many new changes including the support of more CFD solvers, more HPC systems, and process control. Currently, CaseMan supports Overflow2, NXAir, Hyb3D, Wind, Cobalt, FDNS, and NASCART, with several others in the testing phases. CaseMan has been tested on several MSRC HPC systems such as Sapphire and Ruby at ERDC, Falcon at the Aeronautical Systems Center, and other systems at the Army Research Laboratory and Maui High Performance Computing Center. Other untested MSRC systems should also work, but CaseMan has not been fully tested on them yet.

For authentication on MSRC HPC systems, CaseMan utilizes the User Interface Toolkit (UIT) that was developed and well-supported by a team of researchers at ERDC. As deployed last year, the UIT includes a library of method calls via a secure application programming interface that enables researchers to develop their own interfaces to access DoD HPC systems. More can be found on the UIT at https://www.uit.hpc.mil/UIT/. UIT handles the entire Kerberos authentication process and makes sure all transmissions are properly encrypted. This relieves CaseMan from this

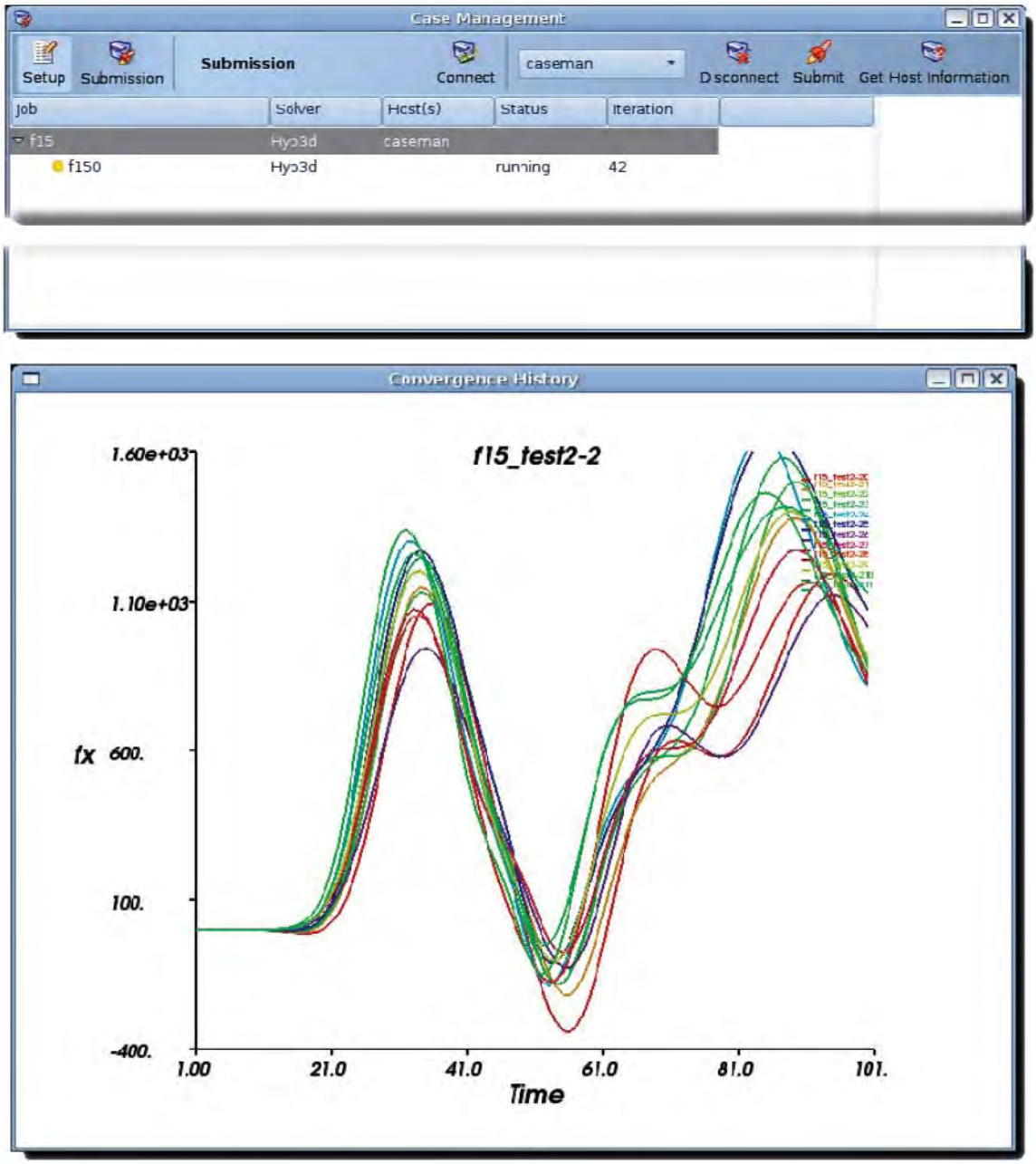


Job preparation is intuitive and well-labeled. If an invalid input value is provided by the user, CaseMan will raise the red flag to warn the user

sensitive security-related task to instead focus on its case management core features. The UIT will always guarantee that CaseMan can connect to the MSRC systems safely and securely. CaseMan can also utilize SSH for connecting to commodity clusters that UIT does not support.

Incorporating a new flow solver into the CaseMan framework has been greatly simplified using the built-in "Flow Solver Builder" module. This module, which is hidden by default since only expert users will work with it, allows expert users to set up the necessary input for a new solver. For each input value the solver needs, the expert user can pull from a list of globally shared attributes. Having a list of globally shared attributes allows CaseMan to identify shared attributes between solvers and, in the future, will allow users to easily migrate from one solver to another similar solver.

As expert users add inputs, they also input documentation, recommended values, and the variable name. The expert user can also mark inputs as required or not required. If the value requires a unit, the expert user specifies the unit in which the solver expects the input



Lightweight data from multiple jobs can be visualized together on a local client to monitor the progress on the HPC systems

to be. CaseMan will automatically convert the value the user specifies into the unit the solver requires. If the input has specific requirements, conditionals can be added to it. Conditionals allow for capping values, setting the value in specific cases, disabling or enabling based on other inputs, or notifying the user of a problem. All of this is set up through a simple graphical interface.

After all the inputs are set up, it is still necessary to convert CaseMan's input into an input file that the solver uses. To simplify this task, CaseMan has a template system incorporated into it. The template system at the simplest level allows for variable replacement. For example, any variable starting with a '$' is automatically replaced with the value the user has entered:

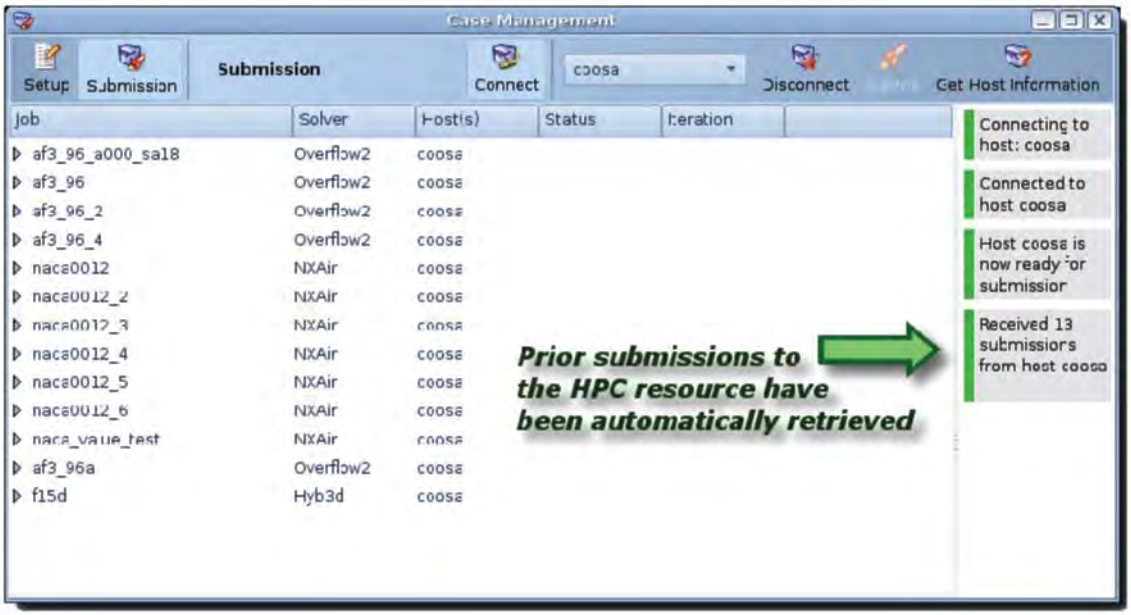
```
&FlowConditions
     FSMACH = $Mach_Number,
     ALPHA = $Angle_of_Attack,
       ...
/
```

The template system also incorporates a python-like language to allow for looping and conditionals. This is useful for writing out boundary conditions where an unknown number may be set up by the user. Unfortunately, to import a solver input file into CaseMan still requires writing a small python script to parse the file although this feature is not required to add a new solver to CaseMan.

Short Tutorial Using CaseMan

What follows is a short introduction of how a typical user would interact with CaseMan. Upon opening CaseMan on the user's workstation, the user is presented with two options: setting up a solver from scratch or importing an existing solver input file. Often times, users already have input files for a specific solver that they use as a template. CaseMan can import these files, saving the user from having to set up variables. After importing a file or setting up a solver from scratch, an interface specific to that solver is generated. Each input is given a short descriptive name along with full documentation. Inputs are also grouped together logically, helping the user quickly identify attributes.

As the user enters values, CaseMan will validate the input, notifying the user if the value is invalid. CaseMan will also disable or enable other choices based on the values entered. If the user is unsure about what value an input should use, the user can leave it blank or press the recommended value button to use the solvers default or recommended value. It is also possible to enter multiple values for a single input using the multiplex feature. A multiplex button next to each input allows the user to enter multiple values or a range of values. This is useful for doing a parametric study. When generating jobs, CaseMan will create a new job for every possible combination. It can submit all the jobs as a group and monitor each job separately.



All submitted jobs through CaseMan will be stored in the database so that a user can manage them easily

Once the user has finished inputting all the values, if every input has been validated, the job can be prepared. In this step, the job information is generated. If the user has set up multiplexing, then multiple jobs will be generated all under the same case. During this step, the input is converted into the solver's native input file system from CaseMan's template system.

After preparing the job, the user needs to specify which HPC system to connect to. If the system is at an MSRC, then UIT is used for authentication; otherwise, SSH is used. If the HPC system is a supported system, then all the default configurations are created and the job should be ready for submission. If the system is not supported, then on the first time connecting, the user will be required to enter some initial values about the queuing system, available solvers, and information on MPI. After connecting, the user can submit the job.

Upon submission, all specified input files, such as grid and restart files, are transferred to the HPC system. CaseMan keeps a database of all the files ever used by jobs; if any of the files have been used before, they are not transferred. This keeps from having duplicate files on the file system. CaseMan also sets up a directory structure for all the jobs, writes out all the files with the correct names, generates a submission script specific to that solver and HPC system, and submits the job to the queuing system. As the job runs, CaseMan will monitor the job for failures and will also extract data. The extracted data can be plotted and viewed on the local workstation as the job is running. This lets the user check for convergence or other useful information. When the job finishes, the user can download the solution file through CaseMan's interface to the local workstation for postprocessing and visualization.

Future Plans

CaseMan is currently in its third year of development. Besides adding more solvers and support for more computer systems, several major features are planned. The biggest and most important feature planned is "process control." Process control allows conditions to be set up and to also link multiple solvers or tools together so that they can be executed in a user-specified process or sequence. With this feature, very large complex simulations can be performed. This feature can be used to start up a simulation with one solver and, then for those jobs that succeed, finish the solution with a second solver or the same solver with different control parameters such as the time-step. Or this feature could be used to detect when a solution converges and to stop the simulation. Multiple tools can be chained together as well in the process control step. It could allow for a complex geometry optimization loop between a geometry/grid generator, a flow solver, and an optimization code. Process control will have a simple graphical setup to easily allow users to set up the processes. With the success in CFD applications, CaseMan also shows potential to be migrated into other computational technology areas for their case preparation, submission, monitoring, and management. To request a current version of CaseMan, please visit http://me.eng.uab.edu/etlab/content/view/17/40/ or contact Dr. Alan Shih (ashih@uab.edu) at the University of Alabama at Birmingham.

Diesel Fuel and High Performance Computing

By Mike Gough

What does diesel fuel have to do with high performance computing? A backup power production facility will sustain the ERDC MSRC high performance computing (HPC) center indefinitely in the event of a utility power failure. Equipment failure not withstanding, clean fuel is the limiting factor in the ability to provide backup power to the MSRC. This article introduces you to our diesel fuel challenge. Since the installation of the electrical generation capability, ERDC has sustained numerous commercial power outages. Most are short-lived, but in fiscal years (FYs) 1998, 2002, and 2003, unplanned outages lasting 5 to 9 days occurred. Amazingly, the ERDC MSRC did not lose power during Hurricane Katrina.

Electrical Power Generation at ERDC

Three Caterpillar diesel generator sets power the ERDC MSRC backup system. Presently, only two generators are required to carry the power load. However, upon the arrival of the Cray XT4 in the first quarter of FY08, all three generators will be mandatory. A 24,000 gallon, in-ground fuel tank provides diesel fuel for the generators. The fuel from this tank is pumped into smaller "day" tanks located next to the generator sets. The engine fuel pumps obtain their immediate fuel from these smaller day tanks.

Fuel Challenge

Diesel engine manufacturers recommend that fuel be stored for no more than 1 year. Historically, the MSRC turns its fuel about every 2 years. Since fuel turnover is never complete, vestiges of old fuel from every fueling remain in the tank. This incomplete turnover is a major contributor to stale and contaminated fuel. Water, environmental pollutants, and problems within the distribution system also contribute to the problem.

Diesel fuel begins to deteriorate as soon as it is produced. Within 30 days of refining, all diesel fuel begins a natural process called repolymerization and oxidation. This process forms varnishes and insoluble gums in the fuel when the molecules of the fuel lengthen and bond together. These heavier components drop to the bottom of the fuel tank and form diesel "sludge" (asphaltene). The fuel begins to darken in color, smell, and causes engines to smoke. As these clusters increase in size, only part of the molecule is burned. The remainder goes out the exhaust as unburned fuel and smoke. The increased cluster sizes begin to reduce the flow of fuel by clogging filters. Fuel filters address the symptom and not the cause.

Sample results before (left) and after (right) fuel cleaning

Most fuel contains some water from either condensation or vents. The water threat requires the understanding of the added burden placed upon diesel fuel as opposed to gasoline. Gasoline is only fuel, while diesel fuel cools and lubricates the injection system. Water contamination increases engine wear. Water can cause damage that is more serious when it enters the combustion chamber. When it is exposed to the heat of the combustion chamber (in excess of 2,000 degrees F), it immediately turns to steam and often explodes the tip of the injector. It also corrodes tanks, lines, and injectors and greatly reduces combustibility.

Fungus and bacteria are also a serious problem. Bacteria exist at the water and fuel threshold and feed on nitrogen, sulfur, and iron that may be present in the fuel. Byproducts of fungus and bacteria contribute to the diesel sludge in the bottom of the fuel tank. Natural chemical changes, water accumulation, biological growth, and accumulation of other pollutants contribute to the degradation of stored fuel.

Fuel Solution

Until a fuel tank is drained and cleaned, it retains a vestige of its first gallon of fuel. Therefore, fresh fuel is contaminated by the old fuel in the tank. Diluting the good with bad is a losing battle since the fuel will always be bad until the core problem is addressed. Policies and procedures must be in place to prevent, minimize, and remove contaminants from the fuel. The order of treatment for fuel-related problems begins with determining the type and amount of contaminants in the fuel. Water paste and laboratory fuel testing is used for this stage.

Next, active remedial measures are instituted. Laboratory test results determine the exact treatment option. Water contamination is remedied by using fuel water separators. If microbes are detected, then the use of biocides is needed. Biocides are similar to "antibiotics" that kill fuel bacteria. Like human antibiotics, the biocide must be administered correctly to remove the contaminants and avoid rebound. If successful, the effect of the additives, without other measures, is temporary and will not eliminate the sludge problem. Next, chemical additives dissolve diesel sludge, gums, and varnishes that clog filters and injectors. For long-term prevention, an inline fuel purifier continually cleans the fuel on demand and reduces the need for ongoing additive use.

A stand-alone, closed-looped fuel polishing system, the Diesel Dialysis Solution, removes inorganic contamination. It circulates the fuel through a cleanser that performs fuel particulate filtration, water separation, and fuel recirculation. Polishing cleans the fuel, but does not refurbish stale fuel.

An inline fuel purifier performs similar functions to the polisher. In real time, it continuously cleans the fuel before it reaches the engine filters. The native OEM (original equipment manufacturer) filtration on the engine is not adequate to process hundreds of thousands of gallons of fuel. The fuel purifier is installed between the main tank and the day tanks. It removes 100 percent of the visible water and up to 98 percent of dust, dirt, and other normal and natural contaminations found in the diesel fuel. By removing contamination immediately before the fuel enters the engine filter system, the purifier delivers cleaner fuel (virtually eliminating filter clogging) and therefore greatly extending maintenance intervals.

A modern fuel purifier generally uses a three-stage purification process employing two well-known fuel separation principles, centrifugal and coalescence, to remove water and contaminates down to 10 microns. By using these two principles, water and other contaminants are separated from the fuel.

Finally, the most severe situation warrants a complete fuel removal and replacement. The existing fuel is “traded” for clean fuel. This option also allows for the cleaning and inspection of the fuel tank. The replaced fuel can subsequently be polished and used in other less sensitive applications.

ERDC MSRC Solution

The ERDC MSRC instituted an aggressive plan to guarantee clean fuel. This approach begins with the main tanks and ends at the Caterpillar generators. To minimize the effect of stale fuel, the MSRC purchased a state-of-the-art fuel-sampling receptacle that allows sampling from varying depths. It installed a fuel purifier system and redundant switchable fuel filters for the diesel generator sets.

Additionally, the MSRC is putting a fuel management contract in place that will remove the existing fuel, and clean and inspect the tank and piping. After the inspection, the tank will be filled with fresh fuel. Finally, the fuel will be checked on a regular basis and treated accordingly.

The MSRC is strengthening its fuel quality management program by instituting a program of weekly water checks with a probe and paste, monthly fuel sampling from multiple depths in the tank, and sampling of all incoming fuel before it enters the main tank.

Reliable HPC cycles are the core product of the ERDC MSRC HPC Center. Although mundane, diesel fuel management is one important way that to ensure that the Center continues to provide computer cycles to the HPC customers. Through an ongoing process of constant policy and procedure improvements, the ERDC MSRC continues to provide cycles to its important HPC customers.

UGC 2007—"A Bridge to Future Defense"

By Rose J. Dykes

The ERDC MSRC participated with high visibility in the 17th annual DoD HPCMP Users Group Conference, held in Pittsburgh, Pennsylvania, June 18-22. The HPCMP presented two ERDC team members with Hero Awards. Randall Hand won the "Technical Excellence" category for his leading role in launching the new HPCMP Data Analysis and Assessment Center Web site and also for developing the ezVIZ batch visualization scripting tool. Scotty Swillie won the "Innovative Management" category for his effort in developing the User Interface Toolkit (UIT) and the ezHPC projects.

Five ERDC Team Members made technical presentations: Dr. Paul Bennett – "Sustained Systems Performance Test on HPCMP Systems" and "Targeting CTA-Based Computing to Specific Architectures Based upon HPCMP Systems Assessment"; Dr. Fred Tracy – "Testing Parallel Linear Iterative Solvers for Finite Element Groundwater Flow Problems"; Dr. Gerald Morris – "Floating-Point Computations on Reconfigurable Computers"; Dr. Ruth Cheng – "Software Development and Applications of Consistent/Inconsistent-Conservative Flux Computation" and "Coupled Watershed-Nearshore Modeling—Phase II"; and Tyler Simon – "Application Scalability and Performance on Multicore Architectures."

The Conference Poster Session included three ERDC posters: Paul Adams – "HPCMP Data Analysis and Assessment Centers"; Scotty Swillie and Glen Browning – "ezHPC: Incorporating a Program-wide, User-Centered Design Approach into the ezHPC User Interface"; and Dean Hampton and John Mason – "Do You Know What Resources Are Offered by the OKC?"

DoD HPCMP presents Innovative Management Award to Scotty Swillie (left) and Technical Excellence Award to Randall Hand (right) (Photograph courtesy of HPCMPO)

Tyler Simon and Dr. Tom Oppe conducted a tutorial entitled "Performance Programming on HPC Platforms Utilizing Multicore Processors." In another tutorial, Paul Adams presented a hands-on demonstration of software tools for remote visualization and explained the features of the Data Analysis and Assessment Center Web site.

With the theme "A Bridge to Future Defense," the conference brought together personnel from all of the HPCMP computing centers and the users of their resources, providing a forum for communication, training, and discussion of HPC and its impact on science and technology.

(From left) Amanda Hines, Chris Merrill, Owen Eslinger, and Dean Hampton, all from ERDC, at Hero Awards Celebration

By Rose J. Dykes

ERDC MSRC Team Members Mentor JSU Graduate Students

Drs. Gerald R. (Jerry) Morris and Robert S. (Bob) Maier mentored four Jackson State University (JSU) graduate students this past summer. All of the students received funding from the National Science Foundation Louis Stokes Mississippi Alliance for Minority Participation Bridge to the Doctorate Fellowship Program.

Dr. Morris, a computer scientist at the ERDC MSRC, served as a mentor for Kevin Pace, Miguel Gates, and Justin Rice, all receiving master's degrees in computer engineering from JSU. Their research focuses on mapping computational kernels onto reconfigurable computers.

Tomekia Simeon, who is presently completing final requirements for her doctoral degree in computational chemistry from JSU, was mentored by Dr. Maier, Assistant Director for the ERDC MSRC. Simeon graduated in 2005 from the Computational Center for Molecular Structure and Interactions at JSU with a master's degree in theoretical chemistry. She has been published in peer-reviewed international journals and has made over 30 presentations in the United States and international conferences.

All four students made presentations at a seminar held in the ERDC Information Technology Laboratory on August 22.

Tomekia Simeon presents "Computational Insight into the Chemical and Electronic Properties of Doped C70 Fullerene and Nanoclusters"

Kevin Pace presents "Introduction/Overview of VHDL, Coding, and Necessary ToolSets"

Miguel Gates presents "Step-by-Step Derivation Process Used for the Formation of High-Speed Computations"

Justin Rice presents "Role of Reconfigurable Computers in the Hardware Development of High-Speed Computations"

ERDC MSRC Participates in SAME/Army Engineering and Construction Camp

David Stinson and Paul Adams participated in a 1-week program of the Society of American Military Engineers (SAME)/Army Engineering and Construction Camp held in Vicksburg, Mississippi, June 10-16. Stinson discussed high performance computing and its use in supporting the U.S warfighter. He then conducted tours of the ERDC MSRC DoD High Performance Computing Center, talking about each of its computing resources and their respective computing capabilities. Adams discussed scientific visualization and its use in enabling DoD scientists and engineers in communicating all aspects of their research. He also presented several visualization demonstrations of DoD research projects.

Forty high school juniors and seniors comprised this year's campers, who were competitively selected to attend the camp. Some of the criteria for selection are being on a high school track that will provide a basis for attendance at an accredited college or university (i.e., taking appropriate mathematics and science courses); expressing intent to pursue a degree in engineering or associated field; and having demonstrated leadership characteristics through participation in extracurricular, sports, and community activities.

According to SAME/Vicksburg Web site, "the Engineering and Construction Camp is designed to provide high school students with an excellent opportunity to gain hand-on experience in engineering and construction skills in Vicksburg's wide-ranging engineering community. This one-week program is supervised by professional engineers and volunteers from the local engineering organizations. The campers will gain a wealth of knowledge about the various career choices in the fields of engineering and construction."

David Stinson, ERDC MSRC Acting Director, talks to a few of the 40 SAME/Army Engineering and Construction campers

(From left) Dr. Bob Maier, ERDC MSRC Assistant Director; Felicia Thompson, ERDC Public Affairs Office; and Frank Ellis, Engineer Inspector General, U.S. Army Corps of Engineers (USACE) Inspector General's Office, August 15

(From left) David Stinson, ERDC MSRC Acting Director; Dr. Guillermo Riveros, ERDC Information Technology Laboratory (ITL); and Dr. Felipe Acosta and Professor Ismael Pagan, University of Puerto Rico-Mayaguez, August 10

(From left) Dr. James Houston, ERDC Director; Dr. Alexander MacLachlan, Army Laboratory Assessment Group; Tony Mancini, USACE Liaison to Assistant Secretary of the Army for Acquisition, Logistics, and Technology; and Greg Rottman, ITL Acting Deputy Director, July 25

Dr. Mike Stephens (left), ERDC Data Analysis and Assessment Center (DAAC) Lead, and Ed Gough (right), Deputy Commander and Technical Director, Naval Meteorology and Oceanography Command, Stennis Space Center, July 17

(From left) Dr. Mitch Erickson, Science and Technology Directorate, Department of Homeland Security; Dr. Mike Stephens; Dr. Stan Woodson, ERDC Geotechnical and Structures Laboratory (GSL); Georgette Hlepas, Naval Postgraduate School, Science, Mathematics and Research for Transformation program intern, GSL; Mitch Erickson, Department of Homeland Security; Dr. Mary Ellen Hynes, USACE Headquarters; Dr. Robert Hall, GSL, July 11

(From left) MG Steve Abt, Deputy Chief of Engineers--Reserve Component; SGM McClinton Brown, USACE, Washington, D.C.; and David Stinson

(From left) Tom Biddlecome, DAAC; William Laska, Science and Technology Directorate, Department of Homeland Security; and Dr. Mike Sharp, ERDC GSL, June 20

David Stinson with students from University of Puerto Rico-Mayaguez, June 19

(From left) David Stinson; BG Todd Semonite, Commander, North Atlantic Division, New York; Greg Rottman; Dr. James Houston; COL Rick Jenkins, ERDC Commander

Paula Lindsey, ERDC MSRC, and Dr. Bob Maier with 20th Engineer Brigade, Fort Bragg, North Carolina, May 9

acronyms

Below is a list of acronyms commonly used among the DoD HPC community. These acronyms are used throughout the articles in this newsletter.

ADF	Australian Defence Force
AMD	Advanced Micro Devices, Inc.
ARSC	Arctic Region Supercomputing Center
CCAC	Consolidated Customer Assistance Center
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CTA	Computational Technology Area
DAAC	Data Analysis and Assessment Center
DoD	Department of Defense
ERDC	Engineer Research and Development Center
FY	Fiscal Year
GB	Gigabyte
GHz	Gigahertz
GSL	Geotechnical and Structures Laboratory
HPC	High Performance Computing
HPCMP	HPC Modernization Program
HPCMPO	HPCMP Office
I/O	Input/Output
ITL	Information Technology Laboratory
JSU	Jackson State University
MDS	Metadata Server
MPI	Message Passing Interface
MSRC	Major Shared Resource Center
NFS	Network File System
NSWC	Naval Surface Warfare Center
OS	Operating System
OSS	Object Storage Servers
OST	Object Storage Target
PET	User Productivity Enhancement and Technology Transfer
SAME	Society of American Military Engineers
TB	Terabyte
TFLOPS	Trillion Floating-Point Operations per Second
TI	Technology Insertion
UGC	Users Group Conference
UIT	User Interface Toolkit
USACE	U.S. Army Corps of Engineers

training schedule

For the latest on training and on-line registration, one can go to the User Productivity Enhancement and Technology Transfer (PET) Online Knowledge Center Web site:

https://okc.erdc.hpc.mil

Questions and comments may be directed to PET at (601) 634-3131, (601) 634-4024, or *PET-Training@erdc.usace.army.mil*

ERDC MSRC *Resource* Editorial Staff

Chief Editor/Technology Transfer Specialist
Rose J. Dykes

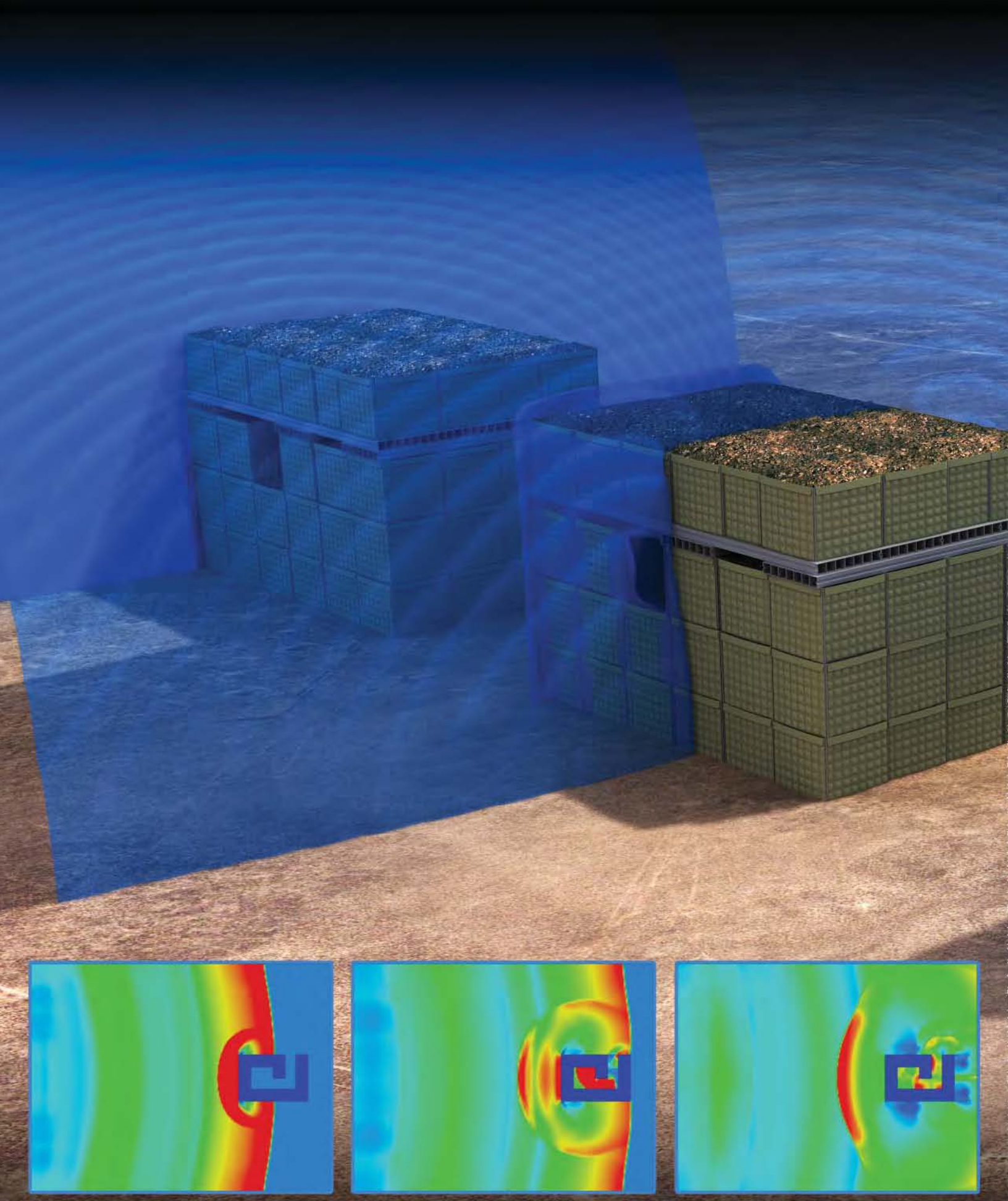
Visual Information Specialist
Betty Watson

ERDC MSRC Web site: *www.erdc.hpc.mil*
Consolidated Customer Assistance Center (CCAC)
E-mail: help@CCAC.hpc.mil
Telephone: 1-877-222-2039

The ERDC MSRC welcomes comments and suggestions regarding the *Resource* and invites article submissions. Please send submissions to the above e-mail address.

Design and layout provided by the Visual Production Center, Information Technology Laboratory, U.S. Army Engineer Research and Development Center.

Approved for public release; distribution is unlimited.



U.S. ARMY ENGINEER RESEARCH
AND DEVELOPMENT CENTER
INFORMATION TECHNOLOGY LABORATORY