AIAA 2003-4229

AUTOMATED CFD PARAMETER STUDIES ON DISTRIBUTED PARALLEL COMPUTERS

Stuart E. Rogers, **Michael J. Aftosmis**,
Shishir A. Pandya and **Neal M. Chaderjian**
NASA Ames Research Center
Moffett Field, California

Edward T. Tejnil and **Jasim U. Ahmad**
Eloret Institute
Moffett Field, California

16th AIAA Computational Fluid Dynamics Conference

23-26 June, 2003 / Orlando, Florida

AIAA 2003-4229

AUTOMATED CFD PARAMETER STUDIES ON DISTRIBUTED PARALLEL COMPUTERS

Stuart E. Rogers[*] **Michael J. Aftosmis**[†]
Shishir A. Pandya[‡] and **Neal M. Chaderjian**[§]
NASA Ames Research Center
Moffett Field, California

Edward T. Tejnil[¶] and **Jasim U. Ahmad**[||]
Eloret Institute
Moffett Field, California

Abstract

A software script system known as AeroDB has been developed to automate the process of running thousands of CFD cases on multiple distributed, heterogeneous, parallel computers. The software utilizes a grid-computing infrastructure for a uniform interface for compute-job submission and user authentication to different compute platforms. The software consists of several scripts written in the Perl programming language, and a database server where all information about each job is stored. In a seven-day period AeroDB was used to run over 3000 cases in a parameter study of the flow over a Langley-glide-back-booster vehicle, using both an Euler and a Navier-Stokes flow solver. Comparison of computed force coefficients with experimental wind-tunnel data shows good agreement.

Introduction

As Computational Fluid Dynamics (CFD) technologies and software mature, and as computational-resource costs continue to drop, it becomes possible to use CFD methods to run large parameter studies. Computational requirements for inviscid Euler CFD solvers for a complete aerospace vehicle are on the order of 5 to 20 hours on a high-end PC or workstation, depending on the solver and the complexity of the vehicle. Computational requirements for viscous CFD solvers are typically 10 to 20 times that of an Euler solution. By taking advantage of parallel computing platforms, it becomes possible to reduce the wall-clock time per Euler solution to less than an hour. If enough parallel computing platforms are available to a user, it becomes possible to perform a large parameter study. With this comes the possibility of using high-end CFD analysis in a number of unique ways, including trade studies and in building stability and control databases.

A typical large parameter study might involve examining 30 different angles of attack, 20 different Mach numbers, and 5 different side-slip angles, each for a number of different geometry configurations or control-surface deflections. This example results in 3000 different flow conditions for each geometric configuration; the number of CFD cases to be run could easily reach into the tens of thousands, and require at least 100,000 CPU hours. Such a parameter study could then be used for a number of things, including the development of stability and control derivatives for the vehicle, or to virtually "fly" the vehicle through the database. An example of the latter is given by Woodson and Bruner,[1] where they build a database of flow solutions for an aircraft store, and then examine its motion for a number of different release mechanisms from an aircraft.

A number of difficulties arise when attempting to run a large parameter study. Often a user must spend time performing a number of mundane tasks for each CFD job. These include pre-processing input files, logging into the compute system and transferring the input files, executing and monitoring the job, post-processing, and archiving the solver output. In the case of a viscous solver, time limitations in the queues of the job scheduler often require that the CFD run be resubmitted several times in order to accumulate enough CPU time to converge to a steady state. This can become a rather tedious full-time job when running more than a few jobs. The use of simple scripts to perform most of these tasks will help when running a few dozen jobs. But when one wants to run

[*]Aerospace Engineer. Associate Fellow AIAA.
[†]Research Scientist. Senior Member AIAA
[‡]Aerospace Engineer. Member AIAA.
[§]Research Scientist. Associate Fellow AIAA.
[¶]Research Scientist Engineer.
[||]Senior Research Scientist.

thousands of jobs, a more sophisticated processes is required. This is particularly true when the compute resources are spread out over a number of different heterogeneous compute platforms at more than one location.

One effort to facilitate the use of distributed heterogeneous compute systems (grid computing)[2] is currently a major focus under the NASA Computing, Information and Communication Technologies (CICT) Program, under the Computing, Networking, and Information Systems (CNIS) Project. This grid-computing effort is built upon the Globus[3] software. Grid computing is based on the concept that one could gain significant increases in computational throughput by accessing any number of remote compute nodes through a common job-submission mechanism. This would enable a group to meet its peak computing-rate needs without having to maintain peak-level computing resources locally, resources which might otherwise become idle during off-peak usage. The Globus software provides such a job-submission mechanism via a command called globusrun. In addition to job submission, the Globus software provides secure services for user authentication, remote shell execution, and file transfers, via a Grid Security Infrastructure (GSI). GSI versions of the secure shell and secure file copy programs ssh and scp[4] are known as gsissh and gsiscp. These commands enable secure, grid-based authentication and communication over an open network.

The objective of the current work is to build a prototype software system which will automate the process of running CFD jobs on grid resources. The goal of this system is to remove the need for user monitoring and intervention of every single CFD job. It should enable the use of many different computers to populate a massive run matrix in the shortest time possible. One previous effort in this area is the ILab software.[5] ILab provides a general purpose capability for creating and launching parameter studies. Because of its generality, it does require a significant amount of user input to customize it for the user's particular application. The current effort is an attempt to build a parameter study capability which is customized to run specific CFD flow solvers, such that the user does not have to provide any information about how to run the flow solver; they only have to provide the flow-solver inputs. Additional desired capabilities for the system are persistence and error recovery. For example, if a remote computer crashes while a job is being run, knowledge of this job will not disappear, and it can be resubmitted to run elsewhere. Such a software system has been developed in the current work, and is known as the AeroDB script system.

The approach taken for the development of AeroDB was to build several discrete modules. These include a database, a job-launcher module, a remote-execution module, a run-manager library, and a web-based user portal. The details of the design of AeroDB are presented in the following section. The subsequent sections present details on the Cart3D[6,7] and Overflow[8,9] flow solvers used in the current simulations, on the reusable launch vehicle (RLV) which is the used in the current calculations, and on the results of the parameter study which was performed using AeroDB. The paper concludes with a section on the lessons learned in this effort, and ideas for future work in this area.

AeroDB Design

The AeroDB system consists of a number of components, each with their own tasks. A flowchart of the AeroDB design is shown in Fig. 1. At the center of the flowchart is a database. The database is the communication hub and the repository of all known information about each job. All other modules only pass information to and from the database, not directly with each other. The other modules include a job-submission script, a job-launcher module, a remote-execution module, a run-manager library, and a web portal. All of the modules and scripts are written in the Perl[10] programming language using an object-oriented design.

All communication with the database uses Perl DBI[11], which provides a consistent database interface, independent of the actual database being used. The basic work-flow within AeroDB is as follows. The user inputs the specific values for the wind-vector parameters (angle of attack, side-slip angle, and Mach number) into the job-submission script and executes it. The job-submission script creates a new entry into the database for each case to be run, including the parameter values and location of input and output files. The job launcher, which is always running and monitoring the database, submits each individual job to execute on a specific computer. When a job begins execution on the computer, the remote-execution module is started. This module obtains all information it needs from the database and runs the flow solver. The run-manager monitors the solver output and instructs the flow solver to stop either when it determines that the case has converged, or when the job is about to exceed its alloted computing time. If the case does not converge during this run, it will be restarted once again by the job launcher. Further details of this process and each of the AeroDB components is presented in the following subsections.

Database

The database is the heart of the system; it is the communication hub and repository of all information about the job. The primary form of communication between the AeroDB components is via the database. In the current work, a MySql[12] database is used. All of the tables are initially built with a series of Perl

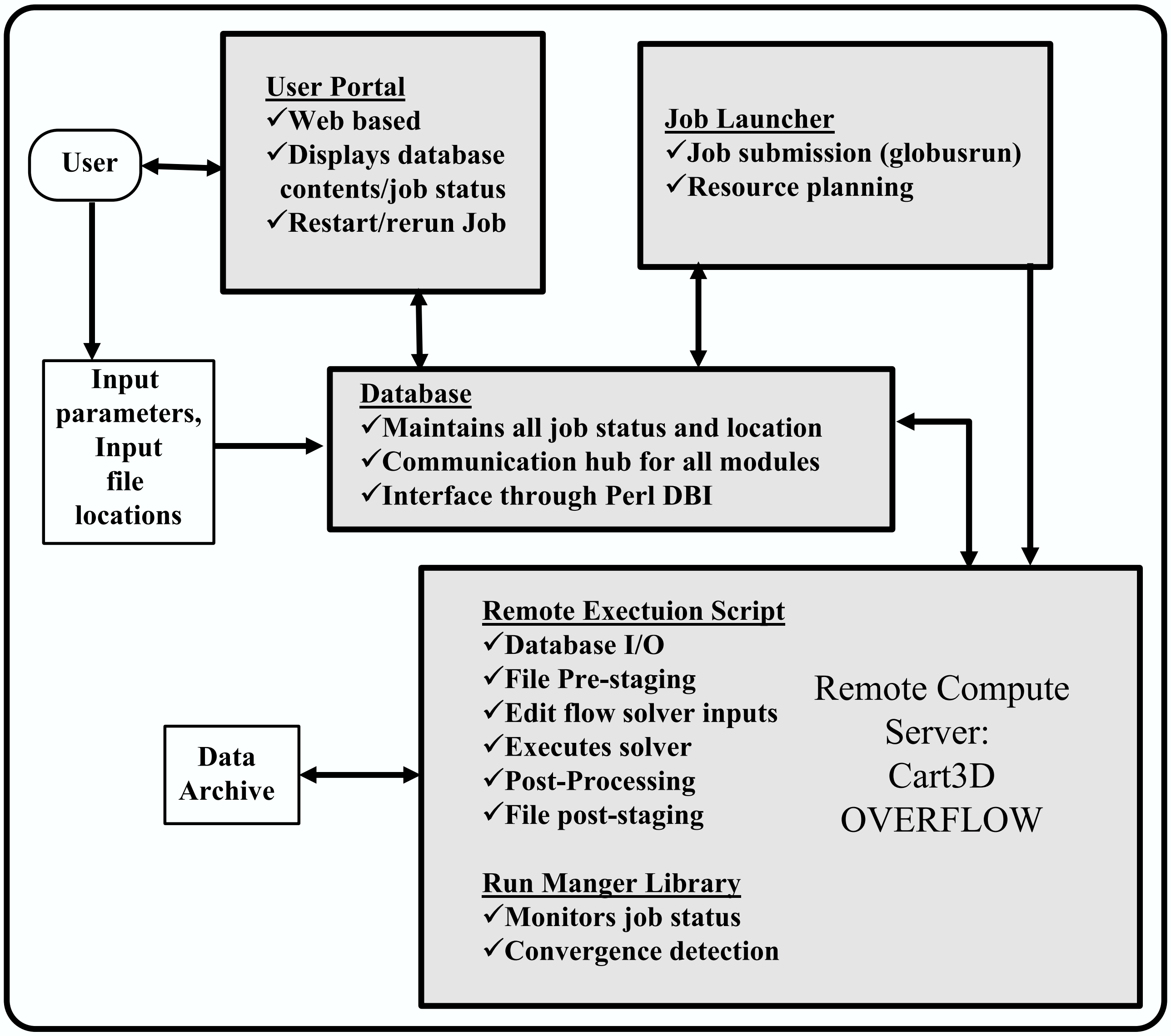


Fig. 1 AeroDB flowchart.

scripts. Two types of information are stored in the database tables: static and dynamic. The static tables contain information that does not change as jobs are created and run. This includes information about each compute host on which AeroDB can run jobs, the user accounts available on those hosts, and information about the flow solvers and their input and output files. The dynamic tables contain information about specific jobs, information that is created and updated while the jobs are running. This includes specific parametric input values for each job, the job's owner, the current job status and run host, information about the resource requirements for each job, the location of all input and output files, and a log of all events that have occurred for each job. There is also a dynamic table which contains integrated force and moment data from the final steady-state solution for each case.

Job Submission Script

The job-submission scripts are used to enter information about new jobs into the database. These simple scripts are used to enter the basic information about each job: the location of the input files, the size of the job, the flow solver to use, and flow parameters such as angle of attack, side-slip angle, and Mach number. The scripts enter many of the cases at once into the database, although not all of the cases in the entire parameter study need to be entered at once. The entries are typically staggered such that several hundred cases are added to the database at a time. Once the cases are entered in the database, the Job Launcher script automatically executes the jobs.

Job Launcher

The job launcher script (JL) is responsible for launching all jobs for execution on an appropriate remote compute resource. The JL runs as a background process which repeatedly searches the AeroDB database for any job whose status is “new” or “restart”. All attributes of such jobs are obtained from the database. The JL then determines the appropriate remote host on which to execute the job. This selection is made with the help of a broker service, which is part of the grid infrastructure. Input for the broker includes the resource requirements, such as memory, number of CPUs, and computer time; as output it provides the name of the host that is best suited to compute this job in the least amount of wall-clock time. The broker gathers information to do its job by periodically querying all available hosts and storing information about their load and the number of jobs each has queued for execution. The JL then passes certain job attributes (such as memory, time limits, number of processors, and remote host) as input to the globusrun command. The globusrun process then launches the job on the specified remote host via that host’s job scheduler. The JL also is responsible for cleaning up jobs once they have finished execution. This is merely the task of removing all remaining temporary files on a remote execution host with the use of a gsissh command.

The JL communicates back to the database and enters some information in the tables. It stores the name of the remote host and run directory on that host in the jobs table. It sends entries to the log table to record the launching and cleaning of each job.

Remote Execution Script

When a job on a remote host begins execution, it starts running an AeroDB script known as the Remote Execution Script (RES). The RES performs the following sequence of operations: transfer of input files to the remote host; pre-processing of the flow-solver input; executing and monitoring of the flow solver; post-processing the solver output files; and transfer of the output files to a permanent storage host. An entry for the database log table is sent to the database after the successful completion of each of these steps. Certain aspects of each of these steps are flow-solver dependent, and thus a separate Perl module was created for each flow solver. These modules provide the flow-solver specific methods for performing these operations. For example, the grid generation could be optionally performed on the remote host for the Cart3D jobs. The ability to use a sequence of different solvers and utilities, including different versions of the same program, was built into the logic of the RES.

RES solver modules were designed to execute both serial and parallel versions of the flow solvers. The run-time monitoring of the solver is accomplished by forking the execution of the solver. Thus the RES can continually examine the forked process and check for error conditions. The RES will also check for output from the Run-Manager library to determine the status of the flow solver, ie., whether it has converged, run out of time, or is diverging. Thus the RES determines whether the job has fully completed, or needs to be restarted, and it reports this back to the database. The solver post-processing includes the force and moment integration and storage of integrated data in the database. RES error checking, run-time exception handling, and job-status reporting to the database makes it possible for the user to easily monitor each job.

Run-Manager Library

The AeroDB effort included the development and use of a Run-Manager (RM) library. This is a library of routines for monitoring the progress of the flow-solver. This library is called by the flow solver through an Application Programming Interface (API). The library contains various utility tools that monitor quantities such as wall-clock time, residuals, forces, and moments. The flow solver can send these monitoring variables, or signals, through calls to the API. The flow solver also specifies the desired convergence criteria for each signal. These signals are analyzed in the RM library and a status variable is returned. This status variable indicates whether or not all signals have met the convergence criteria, and whether or not the flow solver has enough time to continue running. Based on this status information, the solver can be programmed to automatically monitor and stop itself.

The advantage of such tools for the flow solver is that it can utilize these monitoring capabilities in an automatic fashion with little or no user intervention. This general purpose and extensible library provides a much needed capability for large-scale and multiparametric design-space computations. The library is flexible in that the utilities can be invoked at any time, or at an interval of the flow-solver’s choice. It can even run continuously in the background in a separate compute thread or as a child process. When not monitoring a quantity, this separate process can sleep until the flow solver wakes it up to invoke any of its monitoring functions. The objective of this development is to provide flow solvers with a set of general purpose tools. In this initial implementation two types of convergence monitoring have been implemented: residual monitoring, and force monitoring.

Residual monitoring is the monitoring of signals that are expected to approach zero as the flow solver converges. Any number of residual norms $||R_i||$ are computed in the flow solver and supplied to the library at regular intervals. A residual signal is considered converged when $||R_i|| \leq \epsilon_i$, where ϵ_i is the user specified tolerance for the i^{th} residual signal.

Force monitoring is the monitoring of force and moment signals which are expected to asymptotically approach a constant value as the flow solver converges to a steady state. Any number of force or moment signals P_i are computed by the flow solver and sent to the library at regular intervals. A force signal is considered converged when it satisfies: $\Delta P_i \leq \epsilon_i$ over a statistically adequate sample size.

Web Portal

The web portal provides an interface for the user to see the status of each job in the database, and provides a mechanism for re-running and restarting jobs. The portal is password protected. Once logged on, a user is presented with a summary of all the jobs in the parameter study. The user can then access a page with a summary of all of the jobs run under their user login. A snapshot of one of these jobs pages is shown in Fig. 2. Several operations can be performed from this page. By clicking on a specific job ID, a page with detailed information about that job is displayed. This includes all job-attribute and input parameter entries, all entries in the log table for that job, and the computed aerodynamic forces and moments for that job.

Four other operations can be performed from the page for one or more of the jobs: re-run, restart, stop, and delete. The re-run operation will change the database status variable to "new," which will cause the job launcher to execute this job from initial conditions, effectively throwing away any previously saved solution for this job. The stop operation will prevent the job launcher from resubmitting this job. The restart operation will change the database status variable to "restart," which will cause the job launcher to resubmit this job for further execution, restarting from the previously saved solution. The delete operation will remove all entries for the job from all tables in the database.

Flow Solvers

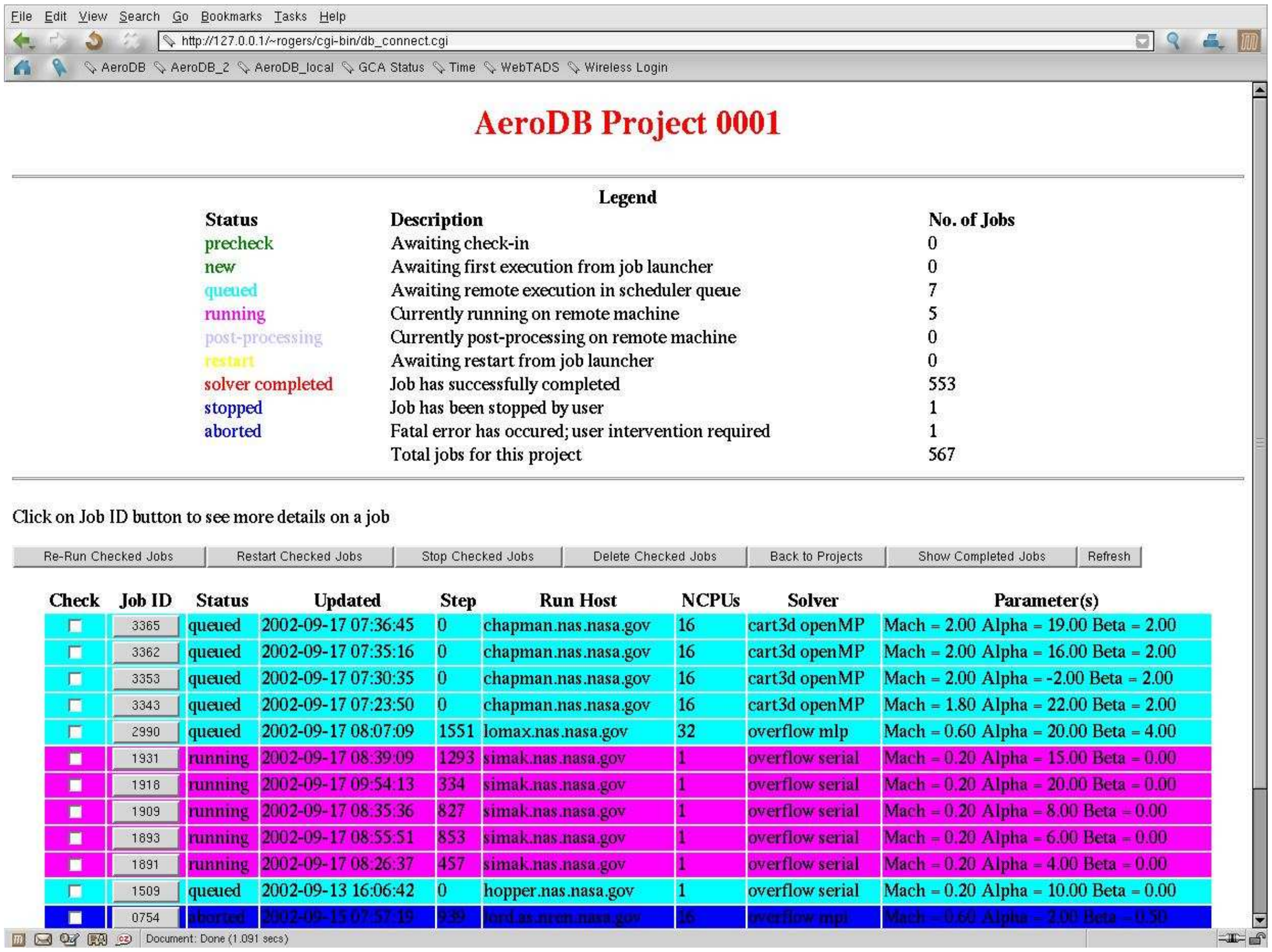
Two flow solvers were used in the AeroDB calculations, Cart3D[6,7] and Overflow[8,9]. The Cart3D code solves the Euler equations using unstructured Cartesian meshes. Cart3D takes as input the triangulated surface geometry and generates an unstructured Cartesian volume mesh by subdividing the computational domain based upon the geometry, and any pre-specified regions of mesh refinement. In this manner, the space near regions of high surface curvature contains highly refined cells, while areas away from the geometry contain coarser cells. The intersection of the solid geometry with the regular Cartesian hexahedra is computed, and polyhedral cells are formed which contain the swatch of surface geometry covered by the Cartesian hexahedra. Cells interior to the geometry are automatically removed. The solid-wall boundary conditions for the flow solver are then specified within these "cut-cell" polyhedra. The volume meshing procedure[6] is provably robust, and does not require user intervention. The meshing scheme is extremely fast (over 1 million cells-per-minute) and meshes are usually created on-demand in the run script and not stored after the computation has completed. Cart3D's solver is based on an explicit multi-stage procedure with strong multigrid acceleration. Convergence of this solver is comparable with the fastest multigrid solvers in the literature.[7] Cart3D has been parallelized to efficiently run on shared-memory computers using standard OpenMP directives.

The Overflow[8,9] code, which solves the Navier-Stokes equations using a finite-difference formulation, was the other flow solver used in the AeroDB calculations. These calculations were run using central differencing of the inviscid fluxes and a matrix dissipation scheme, and using a diagonalized, approximate-factorization, implicit solver. The Spalart-Allmaras turbulence model was selected. The code was run from initial conditions using full-multi-grid sequencing on three levels, and was run to steady-state convergence using three-level multi-grid acceleration. Two different parallel versions of the Overflow code were used. On machines with a shared-memory architecture a multi-level parallelism[13] (MLP) version was used. The MLP code uses native UNIX directives, and two levels of parallelism. The coarse-grained parallelism consists of splitting the problems up into groups of zones, such that each group contains nearly the same number of grid points. On the fine-grained level, each group is assigned a number of CPUs. These CPUs execute the code's DO loops in parallel. The performance of the MLP version of Overflow has been shown to scale linearly with increasing number of CPUs beyond 512 CPUs for large problems. Most of the current Overflow calculations used a total of 32 CPUs and 8 groups (4 CPUs per group).

On distributed memory machines, a Message-Passing Interface[14] (MPI) version of Overflow was used. Instead of relying on shared memory to pass inter-zonal boundary condition data between zones, this data is explicitly passed as a message between the CPUs, using the MPI standard. Load balancing is obtained by distributing the zones among all the CPUs. Since the zones can be significantly different in size, a CPU may be given just one zone, or multiple zones, or just part of a zone.

LGBB Vehicle

The particular vehicle used in the current simulations is a reusable launch vehicle known as the Langley-Glide-Back Booster (LGBB). This vehicle is being studied under NASA's Space Launch Initiative (SLI) program. Some experimental wind-tunnel data is available for this geometry, providing a capability to


File Edit View Search Go Bookmarks Tasks Help
http://127.0.0.1/~rogers/cgi-bin/db_connect.cgi
AeroDB AeroDB_2 AeroDB_local GCA Status Time WebTADS Wireless Login

AeroDB Project 0001

Legend

Status	Description	No. of Jobs
precheck	Awaiting check-in	0
new	Awaiting first execution from job launcher	0
queued	Awaiting remote execution in scheduler queue	7
running	Currently running on remote machine	5
post-processing	Currently post-processing on remote machine	0
restart	Awaiting restart from job launcher	0
solver completed	Job has successfully completed	553
stopped	Job has been stopped by user	1
aborted	Fatal error has occured; user intervention required	1
	Total jobs for this project	567

Click on Job ID button to see more details on a job

Check	Job ID	Status	Updated	Step	Run Host	NCPUs	Solver	Parameter(s)
☐	3365	queued	2002-09-17 07:36:45	0	chapman.nas.nasa.gov	16	cart3d openMP	Mach = 2.00 Alpha = 19.00 Beta = 2.00
☐	3362	queued	2002-09-17 07:35:16	0	chapman.nas.nasa.gov	16	cart3d openMP	Mach = 2.00 Alpha = 16.00 Beta = 2.00
☐	3353	queued	2002-09-17 07:30:35	0	chapman.nas.nasa.gov	16	cart3d openMP	Mach = 2.00 Alpha = -2.00 Beta = 2.00
☐	3343	queued	2002-09-17 07:23:50	0	chapman.nas.nasa.gov	16	cart3d openMP	Mach = 1.80 Alpha = 22.00 Beta = 2.00
☐	2990	queued	2002-09-17 08:07:09	1551	lomax.nas.nasa.gov	32	overflow mlp	Mach = 0.60 Alpha = 20.00 Beta = 4.00
☐	1931	running	2002-09-17 08:39:09	1293	simak.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 15.00 Beta = 0.00
☐	1918	running	2002-09-17 09:54:13	334	simak.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 20.00 Beta = 0.00
☐	1909	running	2002-09-17 08:35:36	827	simak.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 8.00 Beta = 0.00
☐	1893	running	2002-09-17 08:55:51	853	simak.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 6.00 Beta = 0.00
☐	1891	running	2002-09-17 08:26:37	457	simak.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 4.00 Beta = 0.00
☐	1509	queued	2002-09-13 16:06:42	0	hopper.nas.nasa.gov	1	overflow serial	Mach = 0.20 Alpha = 10.00 Beta = 0.00
☐	0754	aborted	2002-09-15 07:57:19	939	lord.as.nren.nasa.gov	16	overflow mpi	Mach = 0.60 Alpha = 2.00 Beta = 0.50

Document: Done (1.091 secs)

Fig. 2 Web portal jobs page.

validate the CFD results. Figure 3 shows the LGBB geometry used in the calculations, as well as the overset surface grids used by the Overflow solver. The overset grid system contains over 8 million grid points and 34 separate zones. Figure 4 shows some cutting planes through the Cart3D mesh used for the subsonic cases. This mesh contains 1.4 million cells. A similar mesh with 0.8 million cells was used for the supersonic cases. As can be seen in both of these figures, the LGBB geometry used here includes the wing and fuselage, vertical tail, and canards. The sting that was used to mount the experimental model is also included.

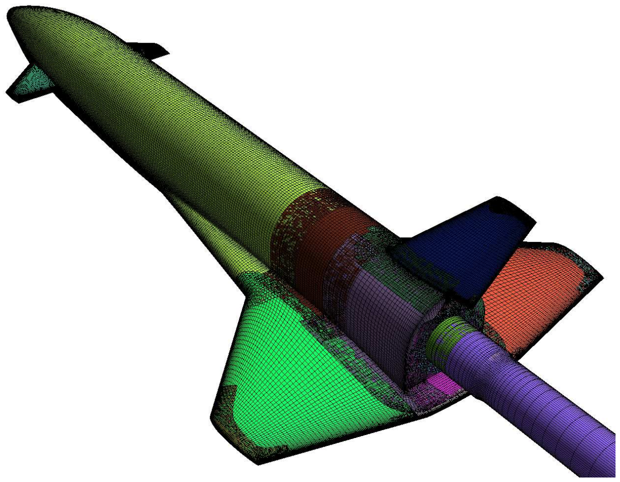


Fig. 3 Overset surface grids on LGBB geometry.

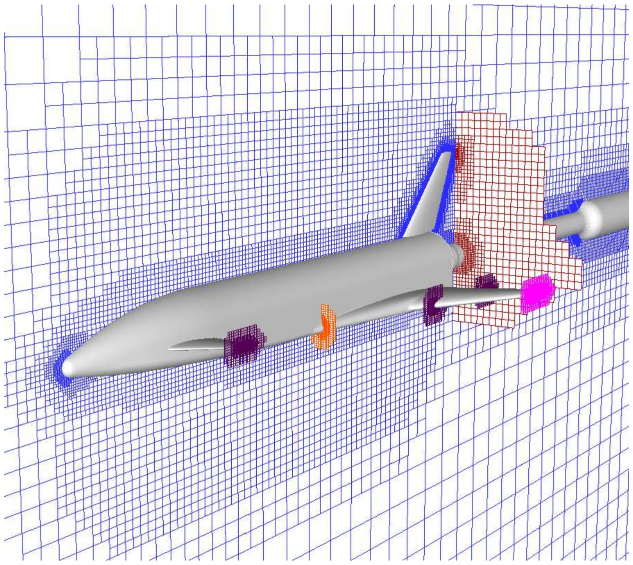


Fig. 4 Cart3D mesh used for subsonic cases.

The input files supplied for the Overflow runs include the entire pre-processed grid files and the standard input file. This grid system was generated using an automated script system from the Chimera Grid Tools[15,16] package and the Pegasus[17] grid-joining program. These input files totaled over 360 Mbytes of data, thus file-transfer time between remote machines is non-trivial. However, this was more efficient than re-computing the same overset grids on the remote compute nodes for each computational job. By contrast, the input files supplied for the Cart3D runs contained only a surface triangulation of the geometry. The Cart3d volume mesh generation was performed on the remote compute node for each computational job.

An important consideration when utilizing heterogeneous compute systems is the data-file format. The AeroDB system does not provide for any translation of different types of data-file formats. Instead it was required that each computer be able to read FORTRAN unformatted data in an IEEE 64-bit floating-point format using big-endian byte ordering. This is the standard on nearly all of the computers that were used in the current work, except for the Linux PC machines where little-endian byte ordering is the default. The appropriate format was obtained on Linux PCs with the use of a commercial compiler and an appropriate compiler option which resulted in big-endian byte ordering for input and output.

Results

AeroDB was used to run a large parameter study for the LGBB vehicle. For the Cart3d cases, 38 Mach numbers were run, ranging from 0.2 to 6.0. Five different side-slip angles were run with values from 0.0 to 4.0 degrees, however, not all side-slip angles were run for all Mach numbers. For each combination of Mach number and side-slip angle, 25 angles of attack were run from -5.0 to 30.0 degrees. In total, just over 3000 cases were targeted for Cart3D. Since the Overflow cases are over 10 times as expensive as the Cart3D runs, only 320 cases were planned, all at subsonic Mach numbers. Eight Mach numbers were planned for Overflow, ranging from 0.2 to 0.95, with five side-slip angles from 0.0 to 4.0 degrees. For each of these, 8 angles of attach were run from 0.0 to 20.0 degrees.

The cases were split up among seven different users. For each specific job, the name of the user who owned that job was stored in the AeroDB database. Each user was executing their own instance of the JL script. When a particular job was submitted to a remote system, it was run using the account owned by that user. However, not all seven users had accounts on all of the compute systems. In its database, AeroDB kept track of which hosts each user had an account on, and would only submit a user's job to run on a computer where that user had an account.

For the purposes of reporting results to the CICT/CNIS program, AeroDB recorded what it was able to accomplish after seven days of execution. The metric for the program was to demonstrate that AeroDB could execute 1000 Cart3D jobs and 100 Overflow jobs in seven days. Within 72 hours over 1000 Cart3D cases and over 100 Overflow cases were completed. At the end of seven days, 2863 Cart3D cases had completed, and 211 Overflow cases had completed successfully. Many of these cases required multiple job submissions in order to obtain enough computing time. A total of 5964 job submissions were successfully completed. These compute jobs utilized 13 different compute resources at four different locations. Table 1 shows the number of job submissions sent to each compute resource, and the approximate number of CPU hours used on each host. The four locations listed in Table 1 are: NASA Ames Research Center (ARC) at Moffett Field, California; NASA Glenn Research Center (GRC) at Cleveland, Ohio; the National Center for Supercomputing Applications (NCSA) at University of Illinois at Urbana-Champaign; and the Information Sciences Institute (ISI) at the University of Southern California. In addition to the computers listed in Table 1, three other compute resources were used at ARC: an SGI Origin front-end machine was used for all of the job launching, a linux PC was used as the database and web server, and an SGI Origin was used for mass storage of all input and output files. No special-priority queues were used on any of the computers.

The data in Table 1 shows that over 95% of the computing time came from the computers at ARC, and that very little speedup in the total elapsed time required for the parameter study was gained by running some of the jobs at the other centers. The computers at ARC are all part of the NASA Advanced Supercomputing (NAS) center at NASA Ames, which is the

Table 1 Job Distribution on Compute Hosts

Location	Host	Hardware/CPUs	# of Jobs	CPU Hrs
ARC	chapman.nas.nasa.gov	SGI O3K/1024	3489	25485
	lomax.nas.nasa.gov	SGI O3K/512	1074	15678
	steger.nas.nasa.gov	SGI O2K/256	477	8017
	hopper.nas.nasa.gov	SGI O2K/64	411	4702
	evelyn.nas.nasa.gov	SGI O2K/16	61	262
	simak.nas.nasa.gov	Sun Ultra/8	136	234
GRC	sharp.as.nren.nasa.gov	SGI O2K/24	126	1014
	aeroshark.as.nren.nasa.gov	Linux PC/128	70	976
NCSA	modi4.ncsa.uiuc.edu	SGI O2K/256	99	483
ISI	jupiter.isi.edu	SGI O2K/8	21	212
Total			5964	57065

biggest computing center within NASA. In order to make the grid-computing concept worthwhile to users, one needs to gain significantly more computing power than one could obtain by just getting an account at a single computing center. And although there was no advantage to relying on the grid-computing infrastructure in the current work, the goal of automating the CFD process was definitely met.

Despite the success of the AeroDB approach, there were a number of jobs which did not complete successfully, and some manual intervention was required. This happened in only a small percentage of the cases. The web portal greatly simplified the user intervention: all that was required was a few mouse clicks in order to restart the job when an error occurred. Typical failure modes include network timeouts during job submissions, and errors due to improper software installation on remote systems.

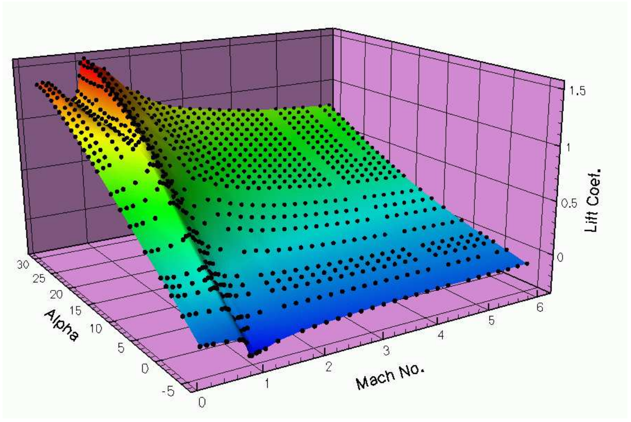


Fig. 5 Cart3D CL for zero side-slip.

In Fig. 5 a carpet plot of the lift coefficient (CL) is plotted versus Mach number and angle of attack for the Cart3D cases run with zero side-slip angle. This one plot represents about 20% of the cases that were run in seven days; each dot on the carpet plot represents the CL from one case. Figure 6 plots CL versus angle of attack from the Cart3d and Overflow calculations along with the experimental values for the case of zero slide-slip and a Mach number of 0.3. Good agreement is seen between the Overflow and experimental results, and that the Cart3D results are slightly higher. The results from the CFD calculations performed by AeroDB and more comparisons with experiment are presented in a companion paper by Chaderjian et al.[18]

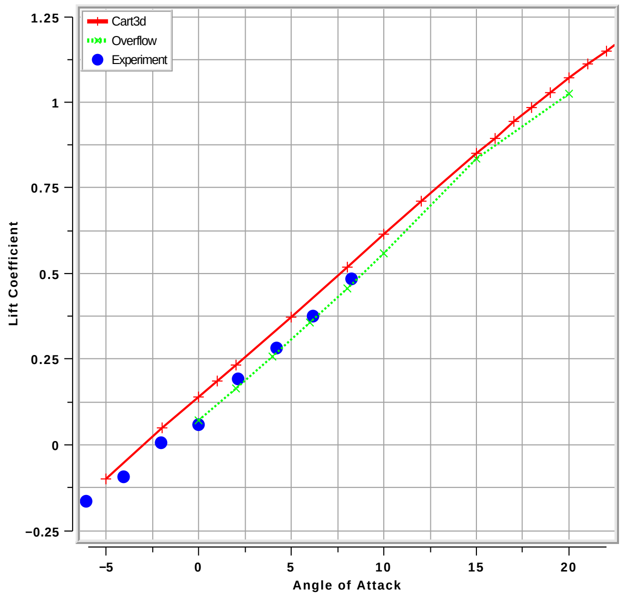


Fig. 6 CL versus angle of attack, Mach=0.3.

Conclusion and Future Work

AeroDB, an automated software system for running large CFD parameter studies on distributed parallel computers has been developed. AeroDB was successful in running over 3000 CFD cases in seven days, which required over 5900 job submissions to 13 computers distributed at four different computer centers across the country. The standardized security and user authentication services greatly simplified this process. A single standard method for job submission was also useful, although the lack of enforced standards in the local implementation of this software caused some problems. It was observed that the total time to perform this parameter study was not greatly improved due to the access to the remote compute centers; the parameter study could have been completed in nearly the same time using just the computers at the NAS facility at NASA Ames Research Center. This does not detract from the fact that the specific functionality of automatically running jobs at a number of distributed computing centers was demonstrated, and could lead to greatly enhanced throughput if more remote compute resources were available.

Several improvements for AeroDB are planned. These include the addition of another module which would continuously monitor jobs in the database for any error conditions. Certain errors that currently require user intervention could be automatically fixed by this module. For example, if a remote compute host crashed, or a network timed out while a job was being submitted, it could detect this and then restart or rerun the job. Other improvements are planned for the process of submitting new jobs into the database. The job-submission scripts will be replaced with a page on the web portal, allowing the user to specify the parameters for the runs through a web page. Further enhancements in this area could be gained with the use of a neural-net controller which would choose the specific parameter cases to be run within the a user specified range of parameters. The controller could adapt the cases to fill in the regions of higher gradients, and not run as many cases where the forces and moments are smoothly varying. This would have the potential to reproduce the carpet plot in Fig. 5 using far fewer cases. Finally, significant additions in the post-processing capability are needed to automate the process of plotting the force and moment results stored in the database.

Acknowledgments

The authors wish to thank Dr. Bandu N. Pamadi of NASA Langley Research Center for supplying the LGBB geometry and experimental data, and to Dr. Reynaldo J. Gomez of NASA Johnson Space Center for supplying some of the LGBB surface grids used in this work.

References

[1] Woodson S. H. and Bruner, C. W. S., "Analysis of Unstructured CFD Codes for the Accurate Prediction of Aircraft Store Trajectories", AIAA Paper 99-0123, Jan. 1999.

[2] Foster, I., "The Grid: A New Infrastructure for 21st Century Science," Physics Today, Vol. 55, No. 2, pp. 42–47, 2002.

[3] Foster, I., and Kesselman, C., "Globus: A Metacomputing Infrastructure Toolkit," *Int. J. Supercomputer Applications*, Vol. 11, No. 2, pp. 115–128, 1997.

[4] Barrett, J. D., and Silverman, R., SSH, The Secure Shell: The Definitive Guide, O'Reilly & Associates, Inc., Feb. 2001.

[5] Yarrow, M., McCann, K. M., DeVivo, A., and Tejnil, E., "Production-Level Distributed Parametric Study Capabilities for the Grid," NAS Technical Report NAS-01-009, NASA Ames Research Center, 2001.

[6] Aftosmis, M. M., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," AIAA Paper 97-0196, Jan. 1997; *AIAA Journal*, Vol. 36, No. 6, pp. 952–960, June 1998.

[7] Aftosmis, M.J, Berger M.J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000-0808, Jan. 2000.

[8] Kandula, M. and Buning, P. G., "Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in OVERFLOW Thin-Layer Navier-Stokes Code," AIAA Paper 94-2357, AIAA 25th Fluid Dynamics Conference, Colorado Springs, CO, June 1994.

[9] Jespersen, D. C., Pulliam, T. H., and Buning, P. G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.

[10] Wall, L., Christiansen T., and Schwartz, L. R, Programming Perl, O'Reilly & Associates, Inc., 2nd Edition, September 1996.

[11] Descartes, A., and Bunce, T., Programming the Perl DBI, O'Reilly & Associates, Inc., Feb. 2000.

[12] Reese, G., Yarger, J. R., and King, T., Managing and Using MySQL, O'Reilly & Associates, Inc., 2nd Edition, Apr. 2002.

[13] Taft, J. R., "Achieving 60 GFLOP/s on the Production CFD Code OVERFLOW-MLP," *Parallel Computing*, Vol. 27, No. 4, pp. 521-536, 2001.

[14] Jespersen, D. J., "Parallelism and OVERFLOW," NAS Technical Report NAS-98-013, NASA Ames Research Center, 1998.

[15] Rogers, S. E., Roth, K., Nash, S. M., Baker, M. D., Slotnick, J. P., Whitlock, M., and Cao, H. V., "Advances in Overset CFD Processes Applied to Subsonic High-Lift Aircraft," AIAA Paper 2000-4216, Aug., 2000.

[16] Chan, W. M., "The Overgrid Interface for Computational Simulations on Overset Grids," AIAA Paper 2002-3188, June 2002.

[17] Suhs, N. E., Rogers, S. E, and Dietz, W. E. "PEGASUS 5: An Automated Pre-processor for Overset-Grid CFD," AIAA Paper 2002-3186, June 2002.

[18] Chaderjian, N. M., Rogers, S. E., Aftosmis, M. J., Pandya, S. A., Ahmad, J. U., Tejnil, E. T., "Automated CFD Database Generation for a 2nd Generation Glide Back Booster," AIAA Paper 2003-3738, June 2003.