V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

DEVELOPMENT OF THE CACTUS CFD TOOLKIT AND ITS UTILISATION ON LARGE-SCALE MULTI-BLOCK SIMULATIONS

Soon-Heum Ko[†], Prasad Kalghatgi[††], Erik Schnetter[†,†††], Sumanta Acharya[††], Gabrielle Allen[†,*], Shantenu Jha[†] and Mayank Tyagi[†]

[†]Center for Computation & Technology,
Louisiana State University, 216 Johnston Hall, Baton Rouge, LA 70803
e-mail: {sko,schnetter,gallen,sjha,mtyagi}@cct.lsu.edu
[††]Department of Mechanical Engineering,
Louisiana State University, Baton Rouge, LA 70803
e-mail: pkalgh1@tigers.lsu.edu, acharya@me.lsu.edu
[†††]Department of Physics & Astronomy,
Louisiana State University, Baton Rouge, LA 70803
[*]Contact Author

Key words: Cactus Framework, Cactus CFD Toolkit, Multi-block Driver, CGNS (CFD Grid Notation System), Automatic Parallelisation

Abstract. *Cactus is a general-purpose application framework that has been widely used in many application domains. However, it is yet to attract the attention of main stream CFD researchers. We assume that this is mainly due to a lack of concrete guidelines for CFD code porting and development. This motivated us to (1) design a standard CFD toolkit, (2) develop computational components for CFD applications, and (3) provide example CFD codes. We used our CFD toolkit for solving the supersonic flow field around a projectile with the base. Our new multi-block driver and example code enables the accurate flow analysis in multi-block geometries. Also, the parallel performance of our toolkit shows good scalability, which proves the capability of the Cactus framework as a high-performance CFD tool.*

1 INTRODUCTION

The Cactus framework[1, 2] is a general-purpose problem solving environment which has been used for scientific simulations in different disciplines since 1997. Cactus' modular structure enables collaborative code development between different groups, and the central core along with computational toolkits supports automatic parallelization, seamless development and deployment on modern computer architectures and easy access to many cutting-edge software technologies.

So far, Cactus has not been widely used for CFD simulations, mainly by a lack of standardization in existing CFD components, and by the lack of support for multi-block mesh systems. This led us to design a standard Cactus CFD toolkit and develop a number of components for CFD applications. The standard CFD toolkit is designed to fully utilize the benefits of the Cactus framework such as numerical library supports and automatic parallelization by drivers, and support additional drivers and modules specifically required for CFD applications. Following the standard design, standard CFD I/O routines are included, Cactus data structure and parallelization modules are improved to support multi-block structure. Also, compressible code on structured domain has been deployed.

In this paper, we present the design and development procedure of Cactus CFD toolkit, along with the current status. We begin with details on Cactus framework and its strength as a framework for scientific computations. The design and development process of Cactus CFD toolkit is expressed in the next Section. Compressible external flow simulations for the validation of the solver and its parallel performances are discussed in Section 4.

2 CACTUS FRAMEWORK

The Cactus framework is an open-source, modular, portable programming environment for high performance computing. It was designed and written specifically to enable scientists and engineers to develop and perform the large-scale simulations needed for modern scientific discovery across a broad range of disciplines. Cactus is well suited for use in large, international research collaborations. Cactus is today used by over two dozen numerical relativity groups for their cutting edge research.

As with most frameworks, the Cactus code base is structured as a central part, called the *flesh* that provides core routines, and components, called *thorns*. The flesh is independent of all thorns and provides the main programme which parses the parameters and activates the appropriate thorns, passing control to thorns as required. A thorn is the basic working component within Cactus. All user-supplied code goes into thorns, which are by and large independent of each other. Thorns communicate with each other via calls to the flesh API or, more rarely, via custom APIs of other thorns.

Thorns are generally stateless entities; they operate only on data which are passed to them. The data flow is managed by the flesh. This makes for a very robust model where thorns can be tested and validated independently, and can be combined at run-time.

Parallelism, communication, load balancing, memory management, and I/O are handled by a special component called *driver* which is not part of the flesh and which can be easily replaced. The flesh (and the driver) have complete knowledge about the state of the application, allowing inspection and introspection through generic APIs.

3 CACTUS CFD TOOLKIT DEVELOPMENT

3.1 Features of Baseline CFD Code

The governing equations are the three-dimensional compressible Euler equations in curvilinear coordinates. The flow solver uses the LU-SGS (Lower-Upper Symmetric Gauss-Seidel) scheme[3] for implicit time integration and local time stepping is used for the time iteration step. The AUSMPW+ (modified AUSM using Pressure-based Weight functions)[4] is applied as a numerical flux at a cell interface, with the use of MUSCL (Monotone Upstream-centered Schemes for Conservation Laws)[5] approach. Primitive variables are extrapolated at a cell interface and the differentiable limiter[6] is employed to suppress unphysical oscillations near physical discontinuities.

3.2 Design and Development of Cactus CFD Toolkit

The standard Cactus CFD toolkit is depicted in Fig. 1. Our toolkit design consists of three layers comprising the low-level computational toolkit, the CFD solver, and a high-level application layer. Computational toolkit consists of a number of components, including the driver, general solvers, mesh generator/reader and visualization models. The multi-block driver is integrated with the current Cactus driver systems, which integrates Cactus modules, manages memory, and controls the parallelism. CFD mesh reader which supports CGNS (CFD General Notation System)[7] mesh format is integrated with current mesh generators. This low-level computational layer is shared with the astrophysics community, which uses it for highly supersonic relativistic flows. CFD flow solvers discussed above constructs CFD solver layer. Components supporting CFD flux schemes, times stepping methods with time integration schemes, etc., are split to separate thorns in the solver layer. The simulation flow is managed by CFDBase module, where variables are declared and CFD components are scheduled. The modular feature with Cactus versioning system enables users to either use current implementations or replace any of supported module with their own implementation. Finally, the application layer combines these with initial and boundary conditions and other necessary elements to form complete simulations.

Two core components to support multi-block data structure are the mesh reader and parallel driver. In the mesh reader, we support the CGNS standard which stores grids, boundary conditions and auxiliary information. In a CGNS interface, boundary conditions and grid information (connectivity and coordinates) are read from a CGNS file into a native data structure. The interface is designed to handle 1-to-1 multi-block abutting connectivity and this connectivity information is referenced by a multi-block driver.

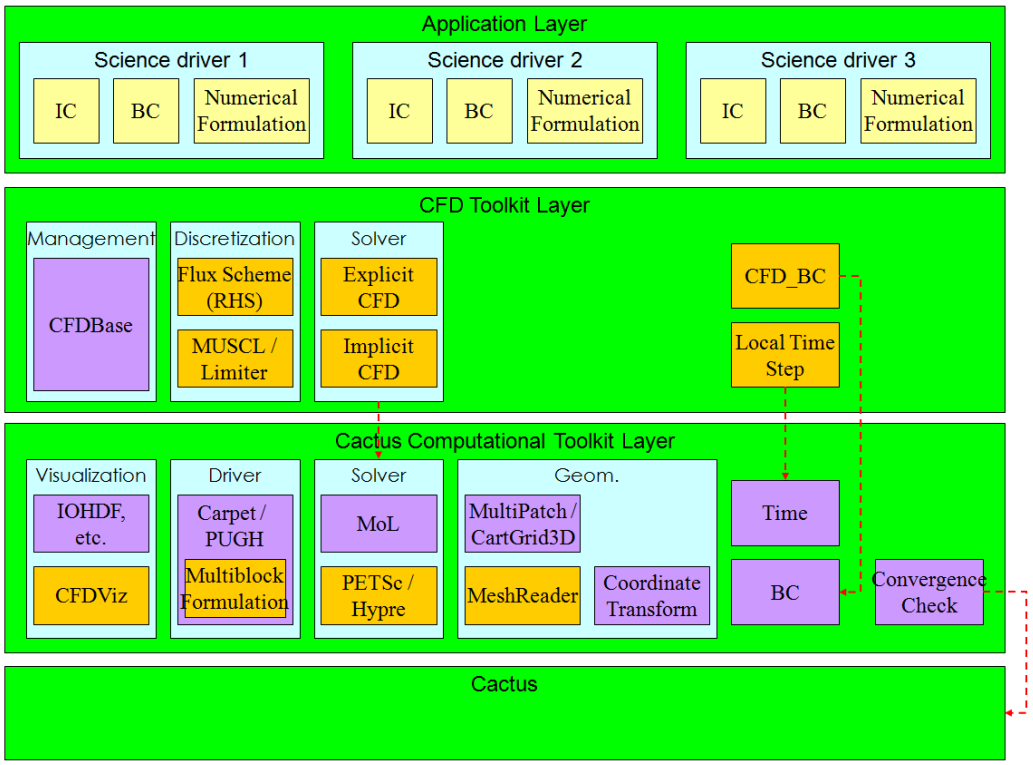

Figure 1: Structure of Cactus CFD toolkit

In Cactus, the multi-block system is implemented as a *driver* (see section sec:cactus above) which handles the different blocks, their connectivity, and their parallelisation. The local and global coordinate systems and their transformations are implemented in an independent layer above the driver. We base our driver on *Carpet*[8, 9, 10], which manages an arbitrary number of blocks, each of which can have a different size. There are no restrictions on the inter-block connectivities and boundary conditions, although we restrict ourselves to boundaries where cell faces between adjacent faces match exactly.

4 NUMERICAL RESULTS

A supersonic flow over an axisymmetric projectile[11] is simulated. The geometric feature with its multi-block mesh system is given in Fig. 2. It has a secant-ogive cylindrical configuration with the boattail. The length of the projectile is 3 diameters of the body. Forebody has a secant-ogive profile with the overall length of 3 diameters and ogive radius of 18.88 diameters. 1 diameter at the bottom of a projectile has a boattail shape with the angle of 7°. Diameter of the jet nozzle is 0.30 to the main body diameter. Considering its symmetric shape along the circumferential direction, a 6 block mesh each with $61 \times 19 \times 31$ is generated to cover the half of the body.

The pressure field around the body and surface pressure distribution compared with

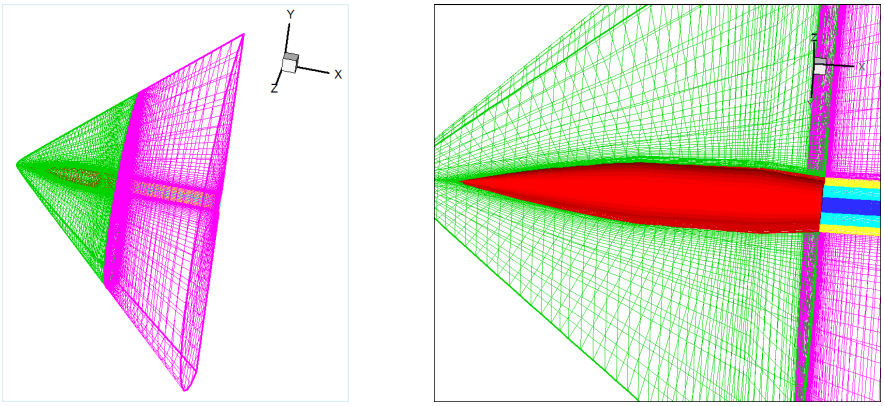

Figure 2: Mesh System around an Axisymmetric Projectile

experimental data[12] are presented in Fig. 3. The free stream Mach number is 3.0 and the jet plume condition is the same as the free stream flow. As can be seen from the left figure, a strong oblique shock and resultant high pressure around the forebody is recovered at the cylindrical main body and slightly expands in the boattail region. As flow goes to the base, high expansion wave is compensated by the plume shock. From the surface pressure distribution along the flow direction, we can validate that the Cactus CFD solver accurately predicts the compressible flowfield.

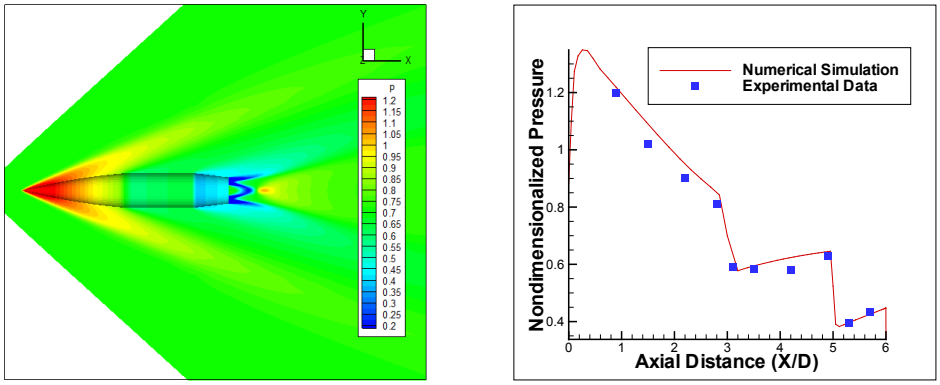

Figure 3: Flowfield around a Secant-ogive Cylinder with Boattail; Pressure Contour around the Body (Left) and Surface Pressure Distribution (Right)

Graphs in Fig. 4 show the parallel speed-up of a projectile simulation. In this case, we solve the six block mesh system with equal sizes. Left graph shows the parallel performance when 2^n CPU cores are used. With 64 cores, parallel performance was about 46.5, which is only 0.727 of parallel efficiency. We assumed that this is because number of cores mismatched the number of blocks. With more than 4 processors, blocks have to be partitioned in different topology and this causes the communication overhead. So, the next test is conducted using 6n cores. Compared with former measurement, this gives more reasonable parallel performance, the parallel efficiency of 0.842 with 48 processors.

This implies that parallel algorithm which is implemented in Cactus framework shows a good scalability on CFD simulations.

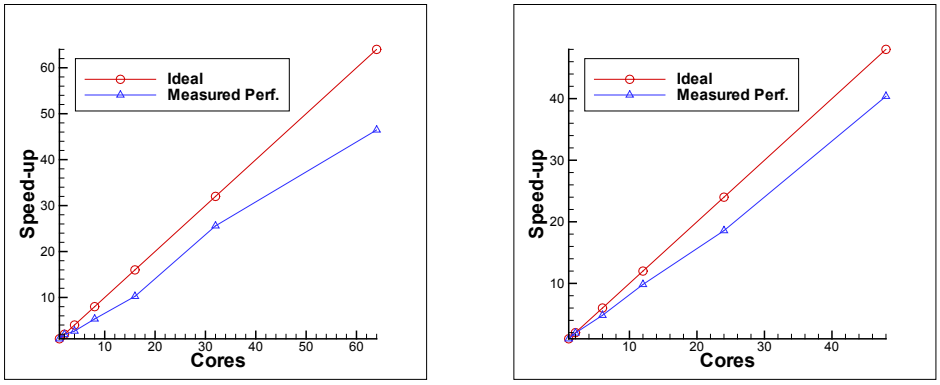

Figure 4: Parallel Performance of Cactus CFD Toolkit on Multi-block Projectile Simulation; Use of 2^n CPU Cores (Left) and 6n Cores (Right)

5 CONCLUSIONS

We have described the development of the Cactus CFD toolkit and illustrated its use for multi-block simulation. Our new developments are directed to provide capabilities for the Cactus framework to solve CFD problems without sacrificing any performance or functionality which Cactus currently provides. Computational infrastructure (multi-block driver with mesh reader) and application modules (compressible solver) are developed according to the standard of Cactus CFD toolkit. We validated the accuracy of our solver by applying it to multi-block simulation. Furthermore, the parallel performance observed from these simulations verifies that the high-level parallelism which Cactus provides can be also applied to CFD simulations. We emphasize that Cactus can be a great tool to any scientists who suffer from the frequent change in computer architecture, computing models, or scientific softwares.

ACKNOWLEDGEMENT

This work is part of the Cybertools (http://cybertools.loni.org/) project and primarily funded by NSF/LEQSF (2007-10)-CyberRII-01.

REFERENCES

[1] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.

[2] Cactus Computational Toolkit home page, http://www.cactuscode.org/.

[3] S. Yoon and A. Jameson, Lower-Upper Symmetric-Gauss-Seidel Method for the Euler and Navier-Stokes Equations, *AIAA J.*, **26**, 1025–1026 1988.

[4] K. H. Kim, C. Kim and O. H. Rho, Methods for the Accurate Computations of Hypersonic Flows, PART I: AUSMPW+ Scheme, *J. Comp. Phys.*, **174**, 38–80 2001.

[5] B. Van Leer, Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov's Methods, *J. Comp. Phys.*, **32**, 101–136 1979.

[6] W. K. Anderson, J. L. Thomas and B. Van Leer, Comparison of Finite Volume Flux Vector Splittings for the Euler Equations, *AIAA J.*, **24** 1453–1460 1986.

[7] CGNS Documentation - User Manual and Mid Level-Library Documentation, http://www.grc.nasa.gov/WWW/cgns/index.html/.

[8] E. Schnetter, S. H. Hawley, and I. Hawke, Evolutions in 3D numerical relativity using fixed mesh refinement, Class. Quantum Grav. **21**, 1465 2004, eprintgr-qc/0310042.

[9] E. Schnetter, P. Diener, N. Dorband and M. Tiglio, *A multi-block infrastructure for three-dimensional time-dependent numerical relativity*, Class. Quantum Grav. **23** (2006), S553–S578, eprint gr-qc/0602104.

[10] Mesh Refinement with Carpet, URL http://www.carpetcode.org/.

[11] M. Mirzaei and S. Arabi, Drag Optimization for Axisymmetric Afterbodies with Jet Plume, *ANZIAM J.*, **46**, C1069–C1085 2005.

[12] L. B. Schiff and W. B. Sturek, Numerical simulation of steady supersonic flow over an ogive-cylinder-boattail body, *AIAA Paper*, **80-0066** 1980