NASA/TM—2000-209801 AIAA 2000–0336

465397
16p.

National Combustion Code: Parallel Implementation and Performance

A. Quealy
Dynacs Engineering Company, Inc., Brook Park, Ohio

R. Ryder
Flow Parametrics, Bear, Delaware

A. Norris
Institute for Computational Mechanics in Propulsion, Cleveland, Ohio

N-S. Liu
Glenn Research Center, Cleveland, Ohio

May 2000

NASA/TM—2000-209801

AIAA 2000–0336

National Combustion Code: Parallel Implementation and Performance

A. Quealy
Dynacs Engineering Company, Inc., Brook Park, Ohio

R. Ryder
Flow Parametrics, Bear, Delaware

A. Norris
Institute for Computational Mechanics in Propulsion, Cleveland, Ohio

N-S. Liu
Glenn Research Center, Cleveland, Ohio

Prepared for the
38th Aerospace Sciences Meeting and Exhibit
sponsored by the American Institute of Aeronautics and Astronautics
Reno, Nevada, January 10–13, 2000

National Aeronautics and
Space Administration

Glenn Research Center

May 2000

Acknowledgments

This work was supported by the NASA Glenn Research Center under contract NAS3–98008 with Gary Cole as monitor. The use of IBM SP-2 and SGI Origin 2000 via the NASA IT/HPCC program is gratefully acknowledged.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

NA TIONAL COMBUSTION CODE: PARALLEL IMPLEMENTATION AND PERFORMANCE

A. Quealy
Dynacs Engineering Company, Inc.
Brook Park, Ohio

R. Ryder*
Flow Parametrics, LLC
Bear, Delaware

A. Norris*
Institute for Computational Mechanics in Propulsion
Cleveland, Ohio

N-S. Liu**
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio

Abstract

The National Combustion Code (NCC) is being developed by an industry-government team for the design and analysis of combustion systems. CORSAIR-CCD is the current baseline reacting flow solver for NCC. This is a parallel, unstructured grid code which uses a distributed memory, message passing model for its parallel implementation. The focus of the present effort has been to improve the performance of the NCC flow solver to meet combustor designer requirements for model accuracy and analysis turnaround time. Improving the performance of this code contributes significantly to the overall reduction in time and cost of the combustor design cycle. This paper describes the parallel implementation of the NCC flow solver and summarizes its current parallel performance on an SGI Origin 2000. Earlier parallel performance results on an IBM SP-2 are also included. The performance improvements which have enabled a turnaround of less than 15 hours for a 1.3 million element fully reacting combustion simulation are described.

*Member AIAA.
**Aerospace engineer, Senior member AIAA.

Introduction

The National Combustion Code (NCC) is an integrated system of computer codes being developed by an industry-government team for the design and analysis of combustion systems. The objective of this effort is to develop a multidisciplinary combustion simulation capability that will provide detailed analyses during the design process of combustors for gas turbine engines. NCC will enable the analysis of a full combustor from compressor exit to turbine inlet. Such a system is critical for optimizing the combustor design process.

The primary flow solver for NCC is CORSAIR-CCD. This is a Navier-Stokes flow solver based on an explicit four stage Runge-Kutta scheme. The original code (CORSAIR) was developed by Pratt & Whitney[1] and was designed from the beginning to use unstructured grids and parallel processing. The code has since been upgraded by NASA Glenn with new models (chemistry, spray, turbulence) and enhanced parallel processing[2].

The Numerical Propulsion System Simulation (NPSS) project at NASA Glenn has supported the NCC flow solver performance enhancement effort. An NPSS milestone to use NCC to run a large scale, fully reacting combustor simulation within an overnight turnaround time of 15 hours was met September 1998. The effort to meet this milestone along with subsequent performance improvements will be described in the Performance Improvements section.

Parallel Implementation

The memory requirements of large scale, fully reacting simulations of real world combustors was one of the primary motivators for originally parallelizing the CORSAIR code. Problems of this size and complexity would not fit within the memory of traditional supercomputers. Running CORSAIR in parallel across a cluster of workstations allowed solving problems which could not be solved on traditional supercomputers.

Developing a parallel application requires addressing a number of issues including domain decomposition, process organization, message passing requirements and portability.

Domain Decomposition

The large memory requirements for the simulation of real world combustors dictated that a parallel code be developed using domain decomposition. A distributed memory, message passing model was used. The current domain decomposition method for NCC is relatively simple. It is based on the number of computational elements in the simulation geometry and on the number of available processors. During a pre-processing stage the cells of the unstructured grid are re-ordered to run consecutively along the longest axis of the grid. The number of cells is then evenly divided among the number of available processors. The last processor takes on any 'extra' cells if the division between processes is not even. These extra cells are typically not a significant factor with the overall load balance.

This "on-the-fly" domain decomposition allows the user to select the number of processors used by the simulation at startup, based on processor availability. The load is well balanced across all processors rather than being statically determined during grid generation. However, no effort is made to minimize the size of messages exchanged between processes by minimizing the number of cells along the process interface boundaries. This will be addressed in the future.

Process Organization

A Single Program Multiple Data (SPMD) strategy was used with CORSAIR-CCD. All processes are computational processes consisting of a copy of CORSAIR-CCD operating on its own local domain and exchanging information with neighboring processes which share common cell faces.

Depending on the geometry, each process typically communicates with at most two neighboring processes. As the number of processors increases and the domain is more finely divided however the number of communication partners per process can increase.

Message Passing Requirements

The CORSAIR-CCD code solves 19 partial differential equations in the benchmark configuration used in this study. The chemistry is modelled by a 12-species, 10-step reduced kinetics mechanism. The message passing required to handle these computational requirements in the original code was significant. Depending on the given simulation each process exchanged as many as 563 messages with a neighboring process each iteration. The amount of message passing in the code has been reduced significantly through various code enhancements which will be described in the Performance Improvements section.

Portability

NCC is intended to be used by a wide audience and therefore should be portable to a variety of platforms and parallel environments. A message interface layer was created to allow the use of either MPI or PVM message passing libraries with no modification to the code. The code can also be compiled to run in serial mode if desired.

Performance metrics were added to the message interface layer, which allows the computation of process and message passing statistics as desired. These metrics assist with tuning the CORSAIR-CCD code to a particular architecture.

A makefile structure was designed to simplify building CORSAIR-CCD on various platforms using either message passing library. The platforms on which CORSAIR-CCD has run include the the SGI Origin 2000, IBM SP-2 and HP Exemplar, along with clusters of workstations (IBM, SGI and SUN) and PCs (LINUX). CORSAIR-CCD has also been ported to the Windows NT environment. The NT version of the code is maintained separately from the UNIX based version and has not been used extensively.

Performance Measurement

Benchmark Test Cases

The test case used for this performance evaluation was the Lean Direct Injection/Multiple Venturi Swirler (LDI-MVS) combustor which is a full 3-D planar periodic sector rig[3]. This combustor has been proposed for use in the evaluation of advanced low emission combustor concepts. The computational domain consists of 443,926 tetrahedral elements. The problem size of interest for the NPSS overnight turnaround milestone however is approximately 1.3 million elements. Until a geometry of this size becomes available the execution time of the smaller 444k element test case is being scaled up by a factor of three to estimate the performance of the larger problem. Recently a 971,054 tetrahedral element version of the LDI-MVS combustor geometry became available. The results for this test case are being scaled by a factor of 1.34 to also estimate the execution time of the larger (1.3 million element) problem.

A 12 species, 10 step reduced kinetics mechanism is being used to account for the amount of computational resources required for the chemistry simulation. Unless otherwise stated, all turbulence, species and enthalpy equations are turned on during benchmarking. Convergence for a fully reacting solution is estimated to require 10000 iterations.

Benchmark Hardware Platforms

The CORSAIR-CCD parallel enhancement effort initially began in 1995 using an IBM SP-2. This machine consisted of 144 IBM RS6000/590 processors interconnected by a high speed switch. All performance metrics were recorded in a dedicated environment where processor usage was restricted to one user job per node. Network activity on the SP-2 may have competed with other active users. However, the effect appeared to be minimal as benchmark results were highly repeatable from one run to the next. IBM's version of MPI, which was tuned to use the IBM SP-2 high speed switch, was used for benchmarks on this platform.

In 1998 an SGI Origin 2000 replaced the IBM SP-2 as the benchmark hardware for this performance evaluation. The SGI Origin 2000 is a cache coherent non-uniform memory access (ccNUMA) architecture. The machine appears to the user as a shared memory machine however memory is physically distributed among the processing nodes. Initial benchmark results on this platform were averaged over multiple runs in a lightly loaded environment. Recent benchmarks on this platform were run on a dedicated 64 processor system to ensure repeatability of the benchmark measurements. SGI's version of MPI was used for all benchmarks on this platform.

Metrics

The metric of most interest in this performance improvement effort has been the time required to reach a solution for a 1.3 million element problem. This "estimated time to solution" is calculated by timing the main iteration loop of CORSAIR-CCD after one full iteration has completed. This allows systems which load code incrementally from disk to complete an entire cycle before benchmark timing begins. The main iteration loop is timed for a fixed number of iterations so an average time per iteration can be calculated. The number of iterations timed varies depending on the size of the problem and the number of processors used. With recent benchmarks on the SGI Origin 2000 typically 200 iterations are timed. The estimated time required to complete 10000 iterations for the 1.3 million element problem is then estimated using the appropriate scaling factor for the test case. The goal of this effort was to achieve a 15 hour estimated time to solution for a 1.3 million element problem.

The initialization and termination sections of the code are excluded from benchmark timing since these sections of code consume very little time relative to the time required to reach a solution. This time segment would have been enough to skew the results of a 200 iteration benchmark result.

Parallel speedup and efficiency are calculated to indicate how well CORSAIR-CCD uses the available parallel resources. The parallel speedup metric is calculated by taking the ratio of the time per iteration for the serial case vs. the time per iteration for the parallel case. The parallel efficiency is the ratio of the parallel speedup to the number of processors used in the calculation. A desire to keep the parallel efficiency above 80% determined the maximum number of processors used at any point during this performance improvement effort. These metrics may indicate potential for further performance improvement.

The benchmark test cases were too large to run within a single node of the IBM SP-2. Since speedup and parallel efficiency could not be calculated traditionally, an estimated speedup was calculated using the time per iteration for the code running on the smallest possible number of processors. This time was assumed to be linear, and it was used to extrapolate the single processor time per iteration from which the speedup curve could be estimated.

Performance Improvements

Efforts to achieve a 15 hour turnaround with CORSAIR-CCD focused on the steady state problem only. The baseline performance of the original code is described below, along with the code enhancements and hardware upgrades listed in roughly chronological order which have significantly contributed to the improved performance of CORSAIR-CCD.

Baseline Performance

CORSAIR-CCD was initially ported to an IBM SP-2. The 444k element benchmark test case consumed 61.4 seconds per iteration when running on 64 processors of the IBM SP-2. It was estimated that a solution for a 1.3 million element simulation could be

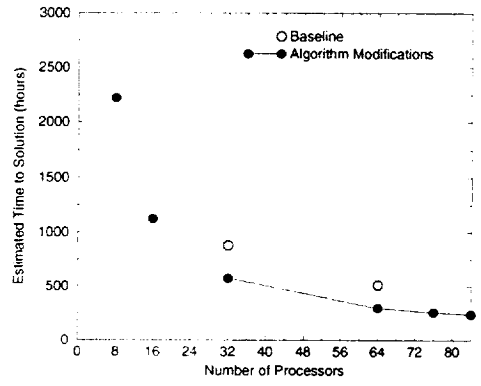

Figure 1: Performance improvement for the LDI-MVS 444k element test case due to algorithm modifications.

reached within 512 hours using this baseline code. This was the starting point for the performance improvement effort.

Algorithm Modifications

CORSAIR-CCD uses a four stage Runge Kutta algorithm. Originally the convective, viscous and artificial dissipation terms were computed at each stage. This algorithm was modified so that the viscous and artificial dissipation terms are now computed at the first stage and then held constant for the remaining stages. The convective terms continue to be computed at every stage. This modification eliminated substantial computation and cut the required message passing in half, from as many as 563 messages being exchanged between communication partners per iteration to at most 286 messages exchanged per iteration. Arrays being exchanged between processes were being transmitted as individual messages. Some of these arrays were packed together into fewer, larger messages, further reducing the number of messages exchanged per iteration to 190. The estimated time to reach a solution for a 1.3 million element problem using 64 processors was reduced from 512 hours to 299 hours (Figure 1).

The 444k element benchmark test case was too large to run within a single node of the IBM SP-2 so an estimated speedup curve was calculated based on the eight processor time per iteration (Figure 2). An estimated speedup appears to be reasonable in this sit-

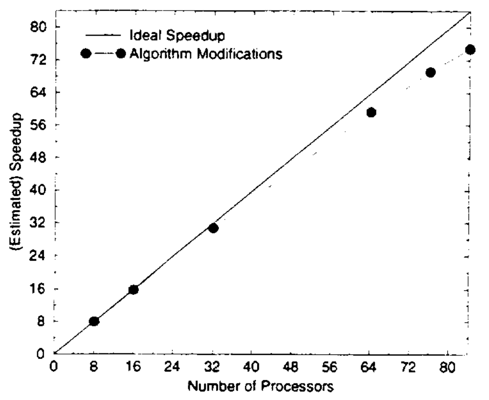

Figure 2: Estimated speedup for the LDI-MVS 444k element test case following algorithm modifications on the IBM SP-2.

uation since it was observed that the time required per iteration for the 16 processor simulation was half the time required for the 8 processor simulation. This 16 processor simulation scaled well enough to assume the eight processor result was near linear and could be used to extrapolate a single processor time per iteration.

The best case solution following the algorithm modifications used 84 processors. The estimated time to reach a solution for a 1.3 million element problem was 238 hours, with an estimated speedup of 74.8 and an estimated parallel efficiency of 89%.

Code Streamlining

The gprof profiling tool was used on the IBM SP-2 to determine which routines were consuming the most time during a typical reacting flow simulation. It was determined that two finite rate chemistry subroutines called four times per iteration for each computational element required 54% of the code's execution time. In one of these routines a statement which executed an exponentiation operation (a**0.25) was replaced by square root intrinsics (sqrt(sqrt(a)), yielding a 21% improvement in performance when using 84 processors.

These two chemistry subroutines were originally written to operate over a variable number of computational elements. Since CORSAIR-CCD called these routines once per element the indexing of temporary variables was unnecessary and could be eliminated. This resulted in a 10% performance savings. It is likely that the compiler could not optimize these routines properly due to the elaborate indexing and temporary variables in the original routine.

Some calculations which were constant for all iterations were relocated to the initialization section of the code rather than being recomputed on each call to these subroutines. This resulted in a 13.4% improvement in performance. Finally, several division operations were replaced by their multiplicative inverse, resulting in another 8.6% improvement in performance.

With these modifications the 444k element test case required 14.8 seconds per iteration when using 84 processors. The estimated time to solution for a 1.3 million element problem was 123 hours. Following these modifications gprof indicated that the two modified chemistry subroutines still consumed approximately 32% of the code's execution time on the IBM SP-2 indicating the finite rate chemistry routines continued to be dominant.

Deadlock Elimination

The original communication scheme in CORSAIR-CCD involved all processes sending to and then receiving from their neighbors. When CORSAIR-CCD was ported to the IBM SP-2, deadlock problems were encountered with this scheme due to limited message buffering capability on that platform. To resolve this situation an odd/even communication scheme was implemented. This scheme assumed all processes could be mapped to a ring topology, with each process communicating with at most two neighbors. All even processes performed send and then receive operations, while all odd processes performed corresponding receive and then send operations. This solution resolved the deadlock conflict initially, however this scheme fails when the process topology becomes more complex. This failure was encountered with the 444k element benchmark test case when the number of processors increased past 84. In this situation the number of communication partners for some processes increased from two to three. A deadlock condition resulted once again because the communication pattern could no longer be mapped to a ring topology.

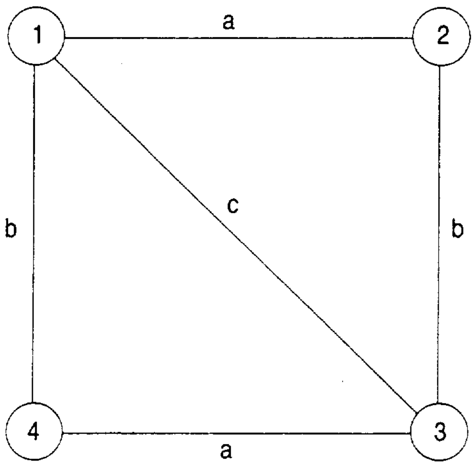

Figure 3: Example graph with three communication stages.

To resolve this deadlock situation, a graph coloring algorithm[4] was implemented to determine a deadlock free communication pattern between an arbitrary configuration of processes. A graph represents the communication requirements between processes. The processes are represented by vertices in the graph. An edge between two vertices indicates that the corresponding processes exchange messages. The edges are "colored" so that each edge from any one vertex is a unique "color". The communication pattern is dictated by the resulting colors. The number of colors represents the number of communication stages. In the example in Figure 3 there are three communication stages. For each communication stage, all process pairs exchange messages.

This new algorithm eliminated the deadlock problem and allowed increasing the number of processors used on the IBM SP-2 from 84 to 96. It was discovered that the new communication pattern was slightly more efficient because it took advantage of the underlying parallelism of the IBM SP-2 interconnection network. The performance when using the new communication pattern with 84 processors improved by 3%. The time per iteration for the 444k element test case when using 96 processors was 13.0 seconds per iteration. Figure 4 illustrates the estimated speedup curve for CORSAIR-CCD on the IBM SP-2. The estimated speedup using 96 processors was 80.4 with an estimated parallel efficiency of 84%. It was estimated to require 108 hours to reach a solution for the 1.3 million element problem.

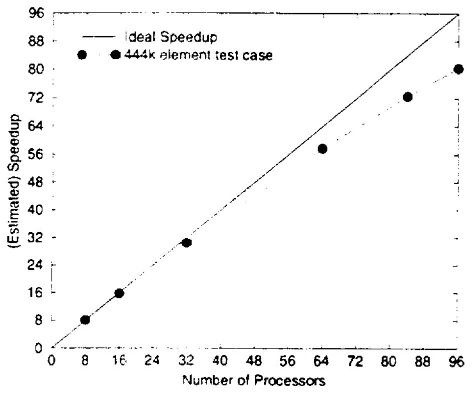

Figure 4: Final estimated speedup on the IBM SP-2.

Hardware Upgrade

CORSAIR-CCD was ported to an SGI Origin 2000 with 195 MHz CPUs in 1998, replacing the IBM SP-2 as the benchmark hardware platform. Initially 32 processors were used for benchmarking on the SGI Origin. When comparing the 32 processor performance of the 444k element test case between the IBM and SGI platforms, the Origin 2000 proved to be a factor of 3.4 faster than the IBM SP-2. Of more interest however is the best case performance of the 444k element test case on each platform. The best case performance on the IBM SP-2 used 96 processors and required 13.0 seconds per iteration. The initial best case performance on the Origin used 32 processors and required 10.1 seconds per iteration. Therefore a 1.3x improvement in performance for the 444k element test case was realized by switching from the IBM SP-2 to the Origin 2000 platform. The Origin 2000 processors were later upgraded to 250 MHz resulting in an additional 1.1x improvement.

ILDM Kinetics Module

A new Intrinsic Low Dimensional Manifold (ILDM) Kinetics module[5,6] was integrated into CORSAIR-CCD to be used in place of the existing finite rate chemistry module. The original chemistry module required solving 12 species equations for the current benchmark configuration while the ILDM Kinetics module requires solving only two. The remaining species are obtained from an ILDM table which is generated during a preprocessing stage. The reduction of species does not give any appreciable reduction in the fidelity of the results. The advantage to this technique is that computation is significantly reduced. The CORSAIR-CCD code solved 19 equations using the original finite rate chemistry module whereas only nine are solved when using the ILDM Kinetics module. Slightly more memory is required (10 MB/process) to store the ILDM tables, however this can be offset completely by eliminating storage for the 10 species which no longer need to be calculated.

Properties such as density, viscosity and temperature can also be obtained from the ILDM tables, further reducing the amount of computation required. Also for some cases the enthalpy calculations can be turned off, reducing the number of equations to be solved to seven. The enthalpy calculations could be turned off for the benchmark test case.

When using this module, message passing costs are completely eliminated for 10 species, along with their associated derivative and flux terms. Also if enthalpy can be turned off the message passing of enthalpy variables can be eliminated. For the benchmark test case the number of messages exchanged between communication partners dropped from 190 to 58 per iteration due to the addition of the ILDM Kinetics module.

Initial benchmarks on the SGI Origin 2000 indicate using the ILDM Kinetics module in place of the original finite rate chemistry module resulted in a 4.8x improvement in performance for the benchmark test case. The 444k element test case required 10.1 seconds per iteration when using 32 processors with the original finite rate chemistry module. The corresponding test case using the ILDM kinetics module required 2.1 seconds per iteration. With the ILDM kinetics module the time required to reach a solution for a 1.3 million element problem was estimated at 18 hours.

FORTRAN I/O Library

Performance began to taper off on the SGI Origin 2000 when increasing the number of processors from 32 to 48, with the parallel efficiency dropping to 80%. Performance dropped off even more significantly above 56 processors. Scaling improved by switching from SGI's f77 compiler to the f90 compiler (Figure 5). It was discovered that the initialization time, where all processes read the same geometry file, was cut considerably when using f90. This pointed to the I/O library as the source for the improved performance. During the main iteration loop all processes printed their residual to the standard output file. This had been added to monitor the progress of the solution and inadvertently had not been removed for benchmarking. The f90 I/O library handled this much more efficiently than the f77 I/O library and a 9.4% improvement in performance was realized when using f90 with 56 processors. Once this extraneous I/O was eliminated, the f77 performance for the main iteration loop matched the f90 performance.

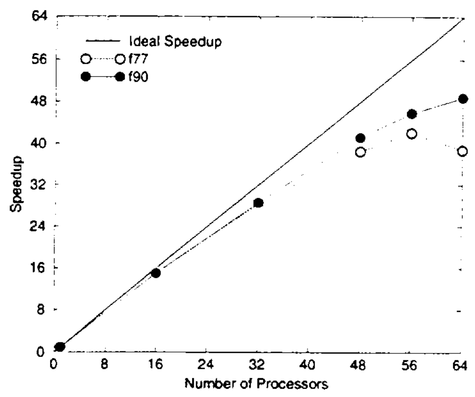

Figure 5: Performance differences for the 444k element test case due to the I/O library on the Origin 2000.

Performance Summary

Current benchmarks with the 444k element test case on the Origin 2000 using 56 processors require 1.08 seconds per iteration. This includes performance improvements due to the use of new compiler optimization flags as well as a recent change to the ILDM kinetics module algorithm which resulted in a 6% improvement in performance. The packing

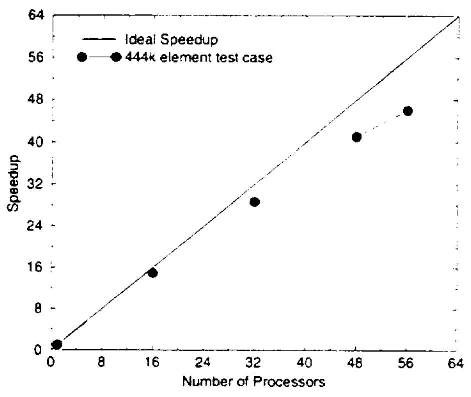

Figure 6: Speedup for the LDI-MVS 444k element test case on the SGI Origin 2000.

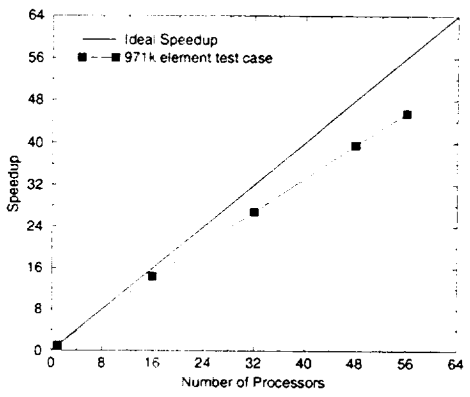

Figure 7: Speedup for the LDI-MVS 971k element test case on the SGI Origin 2000.

of multiple arrays into fewer, larger messages was also eliminated, increasing the number of messages exchanged between processes from 58 to 127. The smaller message sizes were originally believed to improve performance on the Origin however recent benchmarks indicate this has little effect. Further investigation is required. It is now estimated to require 9 hours to reach a solution for a 1.3 million element problem. The speedup is 46.0 and the parallel efficiency is 82%. Figure 6 illustrates the speedup curve for the current code.

The performance results for the 971k element benchmark test case supports the conclusion that the 1.3 million problem could be solved in less than 15 hours to meet the NPSS overnight turnaround milestone. Using 56 processors on the Origin 2000 the 971k element test case requires 2.25 seconds per iteration. It is estimated to require 8.4 hours to reach a solution for a 1.3 million element problem. The speedup is 45.5 and the parallel efficiency is 81%. The speedup curve is illustrated in Figure 7. It was noted that the speedup and efficiency for the larger 971k element test case is slightly less than the 444k element test case. This is due to the larger messages exchanged with the 971k element test case.

The overall results since the CORSAIR-CCD performance improvement effort began in 1995 are illustrated in Figure 8. At times some modifications to improve numerical accuracy have negatively impacted performance. The original CORSAIR-CCD code exchanged 563 messages per iteration between communication partners. Algorithm

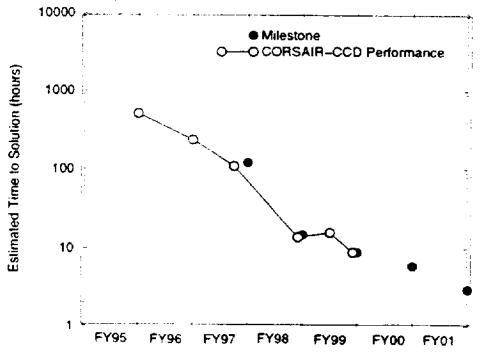

Figure 8: Overview of performance improvement since 1995.

modifications and the ILDM Kinetics module are primarily responsible for the reduction in message passing by a factor of 4.4 to 127 messages per iteration. Attempts to further reduce message passing are on-going.

Future Work

Current effort is focused on further reducing the time to reach a solution to a large scale, fully reacting combustion simulation to three hours. A more sophisticated domain decomposition algorithm is being investigated. This has become a more critical issue as the amount of computational work has decreased with the use of the ILDM Kinetics module. In addition the mixing of message passing and shared memory programming via OpenMP will be investigated to enable the efficient use of more processors. MPI will continue to be used for the existing domain level, coarse grained parallelism, and OpenMP will be investigated for use with loop level parallelism.

Concluding Remarks

The performance of the NCC flow solver, CORSAIR-CCD, has been enhanced significantly over the past several years. Performance enhancements have included algorithm modifications, streamlining of computationally intensive code, restructuring the communication pattern to eliminate deadlock, and the addition of the ILDM Kinetics module which greatly reduced the computational requirements of the code. Additional improvements can be attributed to hardware upgrades over the past few years. It was estimated that the baseline code would require more than 500 hours to reach a solution for a 1.3 million element problem in 1995. The current code is estimated to achieve a solution to the same problem within 9 hours.

References

1. Ryder, R., "CORSAIR USER's Manual: Version 1.0," SID: Y965, Pratt and Whitney Engineering, United Technologies Corporation, January 1993.

2. Liu, N.S. and Quealy, A., "NCC - A Multidisciplinary Design/Analysis Tool for Combustion Systems," NASA/CP-1999-208757, January 1999.

3. Stubbs, R.M., and Liu, N.-S., "Preview of National Combustion Code," AIAA Paper 97-3114, July 1997.

4. Tomaich, T., "A Genuinely Multi-Dimensional Upwinding Algorithm for the Navier-Stokes Equations on Unstructured Grids using a Compact, Highly-Parallelizable Spatial Discretization," Technical Report, University of Michigan Department of Aerospace Engineering, 1995. Ph.D. Thesis.

5. Norris, A., "Automated Simplification of Full Chemical Mechanisms," 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Seattle, Washington, AIAA-97-3115, 1997.

6. Norris, A., "Automated Simplification of Full Chemical Mechanisms: Implementation in National Combustion Code," 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, Ohio, AIAA-98-3987, 1998.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE April 2000	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE National Combustion Code: Parallel Implementation and Performance	5. FUNDING NUMBERS WU-509-10-24-00
6. AUTHOR(S) A. Quealy, R. Ryder, A. Norris, and N-S. Liu	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration John H. Glenn Research Center at Lewis Field Cleveland, Ohio 44135-3191	8. PERFORMING ORGANIZATION REPORT NUMBER E-12106
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM—2000-209801 AIAA-2000-0336

11. SUPPLEMENTARY NOTES

Prepared for the 38th Aerospace Sciences Meeting and Exhibit sponsored by the American Institute of Aeronautics and Astronautics, Reno, Nevada, January 10–13, 2000. A. Quealy, Dynacs Engineering Company, Inc., 2001 Aerospace Parkway, Brook Park, Ohio 44142; R. Ryder, Flow Parametrics, LLC, 15 Debra Drive, Bear, Delaware 19701; and A. Norris, Institute for Computational Mechanics in Propulsion, Cleveland, Ohio 44135; N-S. Liu, NASA Glenn Research Center. Responsible person, N-S. Liu, organization code 5830, (216) 433–8722.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category: 61 Distribution: Nonstandard This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The National Combustion Code (NCC) is being developed by an industry-government team for the design and analysis of combustion systems. CORSAIR-CCD is the current baseline reacting flow solver for NCC. This is a parallel, unstructured grid code which uses a distributed memory, message passing model for its parallel implementation. The focus of the present effort has been to improve the performance of the NCC flow solver to meet combustor designer requirements for model accuracy and analysis turnaround time. Improving the performance of this code contributes significantly to the overall reduction in time and cost of the combustor design cycle. This paper describes the parallel implementation of the NCC flow solver and summarizes its current parallel performance on an SGI Origin 2000. Earlier parallel performance results on an IBM SP-2 are also included. The performance improvements which have enabled a turnaround of less than 15 hours for a 1.3 million element fully reacting combustion simulation are described.

14. SUBJECT TERMS Parallel processing; Combustion; CFD applications	15. NUMBER OF PAGES 15
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT