COMPUTING AERODYNAMICS ON PARALLEL COMPUTERS

S C Gupta
Institute of Armament Technology
Pune, India

Abstract

Current problems in computational aerodynamics (CA) involve very large calculations, necessitating fast computations. Algorithms in computational aerodynamics can be provided with extensive parallelism. Computations can, therefore, be carried on computers with parallel architecture. Provisioning of parallelism in various computational algorithms, associated problems that arise and the requirement of computer hardware parallelism are broughtout in this paper. Artificial intelligence (AI) in computational aerodynamics is described. Knowledge based expert system and symbolic manipulation make the characteristics of AI/CA system.

I. Introduction

Process of parallel operations into the computer architecture can be heavily depended upon for the flow field calculations in the computational aerodynamics. Principles of parallel operations are replication and pipelining. Replication requires parallelism in hardware in which individual array of processing elements (PEs) perform operations in their own individuals. Pipelining involves parallelism in time in which different operations are overlapped in time by placing them in a sequential pipeline manner. These methods are very effective in reducing the overall computational time. Time taken to run a process depends upon the parallelism of an algorithm and the computer paralleled architecture. A generic relationship measures the computer parallelism[1] (eqn. 1).

$$t_n = \frac{n_s + n_p}{r} \qquad (1)$$

Where r is the maximum performance in Mflops (million floating point operations in one second), t_n is the time required to perform n_s elemental operations and $n_p = 0$ for a serial computer and $n_p = \infty$ for a infinitely paralleled computer. Parametric pair (n_p, r) describes the computer performance for processing vectors. Alternatively, n_p/n_s can be conveniently used as an index to hardware parallelism in relation to the technological art of computer i.e. n_s. This paper brings out, $(n_s, n_p/n_s)$ pair requirement for the various computational tasks. Breaking down the sequential algorithm into a parallel one is aimed at carrying out simultaneous operations at a time. In certain cases, this results in slower convergence rate of the paralleled scheme. Recovery of the convergence rate is explained. Efficiency of the algorithm in relation to computer architecture can be expressed through the expression:

$$\text{Speedup ratio (SR)} = \frac{\text{computational time on a serial computer}}{\text{computational time on a parallel computer}}$$

In an ideal situation, speedup ratio is M for a computer that can do M simultaneous operations. Practically, speed up ratio is much different than the value of M and largely depends upon the computational technique. Value of SR for various algorithms can vary over a large range. SR can be expressed as a product of k and M, where k is always less than unity. Likely value of k for various algorithms is discussed. Along with speedup comes the decrease in efficiency, limiting the use of multiple processing.

Computational aerodynamics and experimentation are becoming complementary rather being competitive. CA-aided-experimentation is briefly broughtout. Application of AI to CA is described. Knowledge levels and symbolic manipulation for the AI base are discussed. Nature of parallelism in computer architecture in context to CA algorithm is drawn out.

II. Paralleling Aerodynamic Algorithms

Computational aerodynamics is concerned with flow field predictions (analysis) and generation of optimal configuration (design) tasks. This involves in numerical modelling, algorithm development, software write up and validation procedures. Computational design aerodynamics is multi-fold computer time costly than the computational analysis aerodynamics. Algorithm are the numerical steps involved in solving numerical models. Compatability exists between algorithm and computer architecture. An algorithm involves vector operations of specified lengths. Average parallelism of an algorithm ($\bar{p}$) may be written in the form:

$$\bar{p} = s/q, \quad \text{Where } s = \sum_{l=1}^{l_{max}} q_l \, p_l$$

$$\text{and } q = \sum_{l=1}^{l_{max}} q_l \qquad (2)$$

Here q are the total number of vector operations and s the total number of elemental operations. q_l operations with vector length of p_l at each stage of l, are summed to form s elemental operations. Typically, $\bar{p} = 1$ refers to nil parallelism. In a most ideal situation, n_p/n_s should be equal to $\bar{p}$. Time taken to run an algorithm is related in the form, $T \backsim f(q_l, p_l, n_p, 1/r)$.

Techniques of aerodynamic computations are classified under two heads, namely : 1) boundary integral methods, 2) numerical analysis based techniques which are classically referred to the word 'computational'. Both these techniques involve: 1) matrix formation and 2) matrix solution. Boundary integral methods involve surface discretization (paneling). The principle relationship in the boundary integral approach is written in the matrix form:

$$[A_{i,j}][D_j] = [W_i] \qquad (3)$$

Where D = source function distribution.
$A_{i,j}$ = the influence of jth panel element at ith control point
W = downwash velocity forming boundary conditions

Major portion of computation time is taken in formation of matrix [A]. $[D_j]$ is the function distribution, whose strength determines the flow field information subject to zero normal flow boundary conditions. Determining each of the influence coefficients involves several arithmatic expressions. For N number of panels which form the geometry divisions, the computational effort in determining [A] is of the order of N^2. If calculation of one value of $A_{i,j}$ is referred to as one operation, then N^2 operations need be performed and any number of these operations can be placed in parallel i.e. $n_p/n_s \backsim O(0 - N^2)$. Present day calculations involve flow field analysis over aircraft configuration with all stores mounted. Under such circumstances, the value of N may well exceed few thousands. [A] need be determined for each value of Mach number and angle of attack (flight attitude conditions). Under any such given conditions, [A] further needs be calculated several hundred times while estimating store trajectories or unsteady stability derivatives. For the problems involving one time estimation of [A], $n_p/n_s \backsim 0$ is adequate. Problems involving estimation of [A] several times, $n_p/n_s \backsim O(N)$ shall meet the requirement of quick computations. With the advancement in computer technology there is continuing increase in n_s. Keeping this into consideration, a state-of-the-art technology based computer could be aimed at, $n_p/n_s \backsim O$ (fractional value of N). Nevertheless, even a slow n_s computer could be aimed at higher n_p/n_s value for fast calculations in the integral approach. Value of k is approximately unity in the case of matrix preparation with the boundary integral approach. This is because the coefficients of matrix are individually determinable from some data record. Solution of this single matrix requires one time calculation. Gauss elimination and Gauss Seidel techniques are quite common for the solution of such matrix. With a serial Gauss elimination method computational time is $\backsim O(N^3)$, whereas serial Gauss Seidel results in computational time $\backsim O(N)$. Parallel Gauss algorithms are existent for paralleling the matrix solution algorithm. Since, there is only one matrix requiring one time solution, the parallel Gauss are not of concern.

Numerical analysis methodology involves space discretization (grid laying). While the boundary integral method require surface panelling of a 3-D complex configuration, the numerical analysis approach requires volume discretization of space surrounding the configuration. In a numerical analysis approach, solution of velocity potential function ϕ (i, j, k) is sought in the space domain. Here i, j, k are the vectors in the x, y, z directions respectively, in a cartesian coordinate axes system. Solution to the value of ϕ may be sought say, through some difference approximation. Finite difference approximation is applied herein. The solution ϕ^n using line relaxation in a row at any nth iteration in the subsonic domain is given as:

$$\phi^n (i, j, k) \backsim f [\phi^n (i-1, j, k), \phi^n (i+1, j, k), \phi^n (i, j-1, k), \phi^{n-1} (i, j+1, k), \phi^n (i, j, k-1), \phi^{n-1} (i, j, k+1)] \qquad (4)$$

In the supersonic domain, however, the ϕ^n (i, j, k) is dependent upon ϕ^n (i - 2, j, k) value instead of ϕ^n (i + 1, j, k) in eqn. (4). For N points in each of the directions, procedure involves estimation of ϕ^n at N^3 points, for each of the iterations n. Several thousand iterations need be performed before the convergence criteria can be met[2]. With increasing number N, there is exponential increase in the requirement of n. Some current progress is aimed at the improvement of the efficiency of numerical analysis techniques in a bid to reduce n. To mention, some of these are: 1) multi-grid techniques, 2) approximate factorization, and 3) use of acceleration parameters. Inspite of multifold increase in the efficiency of the numerical software, the solution of well known Navier-Stoke equations over a complex aircraft configuration still remains a much desired proposition.

Iteration can be placed successively at proper preceded spacings. How many of iterations can be pipelined at a time depends upon the number of points to be computed and as to how much of hardware parallelism is available for optimal pipe-lining (without wasting too much of start time and finish time). Start time is referred to as the time taken to occupy all the positions in pipe and finish time is referred to as the time taken to complete operations after all the pipes are not full towards finish (Figure 1).

An example of the 2-d finite difference scheme for the subsonic flow is illustrated here. 8 grid points are taken on x-axis and m points on z-axis at each of x-locations. Calculations in ϕ^n are made at any ith station on all m points. Solution to such a tridiagonal matrix is done through tangential flow boundary conditions. Calculation on the (i + 1)th station are preceded utilising the values of its station. Technique is well known as line relaxation. A total of 8 x m points mesh with say, 6 iterations requirement, requires 48 line calculations. Each line calculation to form a single tridiagonal matrix is referred to as one unit vector length operation taking say, t seconds. 48t seconds will be needed for all operations to go in serial. Matrix formation requires values from neighbouring i-1 and i+1 stations only. Thus a separate iteration cycle can start at a preceded station of i-2 (Figure 1). In such a case the total time consumed is 18t seconds. Process involving a total of 48 vector operations in serial mode taking 48t seconds, will involve 18 vector operations of varying length taking 18t seconds.

For the serial operations, numerical values

s, q, $\bar{p}$ are given as:

$$s = \sum_{1}^{48} q \times 1 = 48$$

$$q = \sum_{1}^{l_{max}} q_1 = 48, \quad \bar{p} = 1 \qquad (5)$$

For the parallel operations, numerical values in s, q, $\bar{p}$ are given as:

$$s = \sum_{l=1}^{l_{max}} q_l \, p_l = 48,$$

$$q = 18, \quad \bar{p} = 48/18 \qquad (6)$$

However, in the case of supersonic flow preceded calculations will fall at (i - 3)rd station. This will alter the $\bar{p}$ value.

The average parallelism in the scheme is 48/18 and is the index of requirement order of n_p/n_s i.e. $n_p/n_s \backsim O(48m/18)$. This is so, since the calculation of ϕ at any line itself can be paralleled. For large number of iterations, $n_p/n_s \backsim O(4m)$, since the start and finish time in this case will be negligible. Calculations in respect of matrix formation at any line do not involve much computations, therefore, $n_p/n_s \backsim O(4)$ is considered. With the increasing number of grid points, there is multifold increase in start and finish time. For the N grid points in the ith direction in a 2-d problem, the $n_p/n_s \backsim O(N/2)$ can be considered. Solution to 3-d problems involve few million grid points, computing each point several thousand times. In a 3-d problem, the preceded calculations will fall by (i-2)th station in the (i x k) plane. For the N x N grid in the (i x k) plane, there will be multifold increase in the start and finish time. This may even further increase with the developments in efficient numerics. Thus, the n_p/n_s in a 3-d domain appear to be, $n_p/n_s \backsim O(N^2/4)$. The 100 x 100 x 100, 3-d meshes having a million computational grid points are common in the present day calculations. In this case start and finish time will be very high, necessitating high n_s computer. For a high n_s value, the value of $n_p/n_s \backsim O(N)$ appear to go well synchronous with the parallelism in algorithm and provide low computational timings. Thus, high (n_s, n_p/n_s) pair value is needed for numerical algorithms.

Breaking down the sequential algorithm into a parallel one is aimed at carrying out simultaneous operations at a time. This however, results in slower convergence rate of the paralleled scheme. An example is cited with a 2-d finite difference formula. In the case of serial calculations at a line, ϕ^n build up is as below:

$$\phi^n (i, j) \backsim f [\phi^n (i-1,j), \ \phi^{n-1} (i+1, j), \ \phi^n (i, j-1), \ \phi^{n-1} (i, j+1)] \qquad (7)$$

Points on entire line can be updated simultaneously with a parallel computer, as below:

$$\phi^n (i, j) \backsim f [\phi^n (i-1, j), \ \phi^{n-1} (i+1, j), \ \phi^{n-1} (i, j-1), \ \phi^{n-1} (i, j+1)] \qquad (8)$$

In case of sequential algorithm previous iteration (n - 1) data is drawn from two stations. In case of parallel computations, previous iteration data is drawn from 3 stations. This makes convergence of parallel algorithm slow. This problem can be overcome by progressing diagonals simultaneously. Scanning a full row results in a tridiagonal matrix. Due to wastages involving start and finish timings, the value of k could be fairly less than unity. N^2

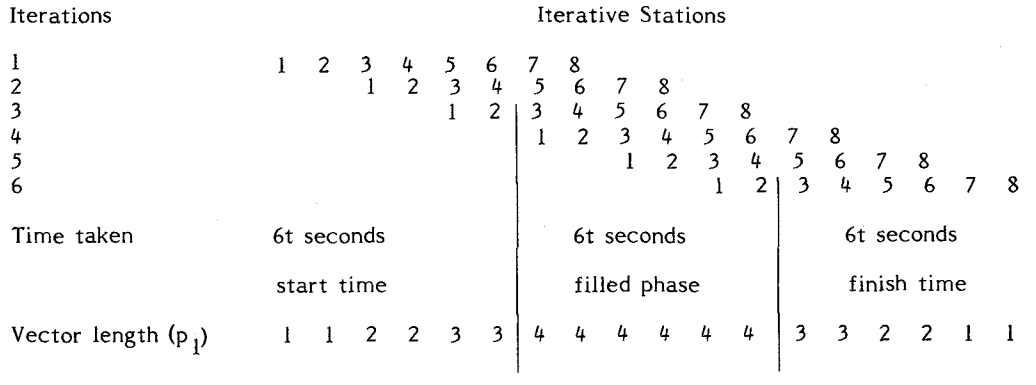

Figure 1. Pipelining the Computational Work.

matrix need be developed in each iteration for N^3 grid point 3-d mesh system. $_2$ N matrix requires solution in each iteration. N^2 iterations are not uncommon for the convergence of solutions. This means N^4 matrix solution are needed for any problem. Sequential Gauss can be paralleled for fast solution of matrix. Several variants of parallel Gauss are on the horizon. Tridiagonal systems have become of particular interest in parallelism because of their recursive and sequential nature. There are several algorithms for tridiagonal system that exploit parallelism. Use of cyclic or odd-even reduction techniques are common. An example of odd-even reduction on all equations is cited here. A tridiagonal set given below, eqn. (9), after one stage of odd-even elimination is given as eqn. (10). Next the non-zero off-diagonals are removed. It takes $\text{Log}_2 Q$ reduction steps to make the resulting system diagonal, where Q is the matrix size. Choice of best algorithm depends upon the computer hardware. Phase-diagrams to this effect are brought out in reference-1.

$$\begin{bmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & c_3 & a_3 & b_3 & \\ & & c_4 & a_4 & \end{bmatrix} \qquad (9)$$

$$\begin{bmatrix} a_1^1 & 0 & b_1^1 & & \\ 0 & a_2^1 & 0 & b_2^1 & \\ c_3^1 & 0 & a_3^1 & 0 & b_3^1 \\ & c_4^1 & 0 & a_4^1 & 0 \end{bmatrix} \qquad (10)$$

Computational effort required in the case of numerical iterative optimisation[3,4] is $\sim O(N^N)$. Use of numerical optimisation is getting important for the design of supercritical shapes. A high $(n_s, n_p/n_s)$ pair value shall be much desirable for such a task. Numerical optimization can be benefitted from parallelism in two ways: 1) the objective function for optimization involves solutions in ϕ say, by finite difference approximates, that can be heavily paralleled and 2) evaluation of difference approximates to gradients in search directions where, the function evaluations can be computed on separate processors. High SR ratio however, results in drop in machine efficiency. This aspect is broughtout later on.

III. Experimental Computer Aerodynamics

Computer is needed in the experimental aerodynamics for the purpose of control automation of experiments, data acquisition and analysis techniques[5]. Test facilities need idealization of working section close to that of flying conditions. Therefore, wall interference need be quickly removed during experimentation from the test section. This is done by wall removal through self adaptive walls. Computer based iterative procedure helps achieving this goal. Pressure and surface shape close to free stream conditions are developed without loosing much of time. Computer based aerodynamic software can be relied upon for quick iterative wall removal. Flight testing of unmanned scaled aircraft is becoming of interest for several important reasons. This work requires fast data acquisition and data analysis, which is possible through high speed computations. Air combat simulation and captive techniques require real time computations. Several equations of motion with each involving large number of variables are required to be quickly solved on line. This necessitates high speed computations.

IV. Artificial Intelligence/Computational Aerodynamics

Aspects such as geometric reasoning for the geometry definition, physics of the numerics convergence and reasoning the methodology adopted form the part of perceptual knowledge which can be automated to make AI/CA base[6]. Knowledge based expert systems can be developed to couple AI and numerical computations. Knowledge level is attained from the past experties which develop over a period of time. Job of geometry definition, discretization, intermediate solution assessment can be delegated to the computer. Knowledge base expert system with symbolic codification form the tools for writing AI software. Software towards parametric design evaluation, flow field zoning, solution adaptive grid refinement and failure recovery numerics are aid to CA researchers.

Fig. 2 shows, how a computational design aerodynamicst could look at his AI/CA system. The initial, Level-I knowledge shall provide the important characteristic type of information, thereby, preventing wrong geometry selections. Level-II knowledge provides selection of appropriate discretization schemes. Level-II AI-software could be aimed at providing solution adaptive grid refinement and flow field zoning based upon the analysis of synthesized solutions. Grid adaption consists of successive grid refinement depending upon flow field features[7]. Expert zonal grid generation is based upon : 1) description of geometry shape, 2) zoning knowledge, and 3) zonal boundary construction. Opinion on best zoning is based on reasoning, which consist in recognizing pressure patterns, geometric contours, viscous dissipations and shock locations. Level-III software is aimed at aiding code developers and code users in providing methodology description. Code developers come across laying of doublet design work, formation of Jacobian matrix and doing functional evaluation. Such a task could be automated. Physics of numerics, convergence criteria and numerical oscillations could be described, e.g. in case of numerical oscillations the execution of program could stop. Accelerating the convergence of numerics based on flow field zoning could be in-built AI feature. An example is cited here. During the shock capturing, the shock location shifts during the iterative process until settling down at some final iteration value. The geometry refinement is much needed around the shock location. An intuitive base system could place the refined geometry at an expected location of shock as the iterations proceed. Such an AI base system could be desired from the accelerations with which the shock location shifts during iterations. Thus, effort involved in excessive geometry refinement at every shock location could be reduced.

Level I Parametric Knowledge	Level II Geometry Discretization	Level III Knowledge of Relevant Codes
. Influence of parametric variations on aircraft start combat weight, turn rate performance and horizontal acceleration time etc.	. Surface discretization, panel networks, hyper-boloidal panels, enriched grids etc.	. Methodology, Integral or differential.
. Influence of wing shape on constraint criteria in a multi-constraint camber line definition.	. Space discretization, expert zonal grid generation, solution adaptive grids etc.	. Numerics, failure recovery handling. . Analysis of synthesized solutions. . User manual.

Figure 2. AI/CA Expert System Knowledge.

Symbolic manipulation (SM) in computational aerodynamic consist in writing symbolic operators[8]. Equation (11) gives the expression in ϕ (i, j) at any nth iteration in a 2-d finite difference scheme. This could be symbolically written in the form of eqn. (12).

$$\phi^n (i, j) \backsim \phi^{n-1} (i + 1, j) + \phi^n (i - 1, j) +$$

$$\phi^n (i, j - 1) + \phi^{n-1} (i, j + 1) \qquad (11)$$

$$\phi^n (i, j) \backsim X1 + X2 + X3 + X4 \qquad (12)$$

Where $X1 = \phi^{n-1} (i + 1, j)$

$X2 = \phi^n (i - 1, j)$

$X3 = \phi^n (i, j - 1)$ and

$X4 = \phi^n (i, j + 1)$

Symbolic operators X1, X2, X3, X4 could become the in-built features designating various difference operators. It could prevent writing long or erroneous programmes. Thus, programmes can be written to recognise the formulae in fortran subroutines. Coding of unsteady time dependent multiple equations and validation procedures could be written with SM. AI/CA/SM systems need be developed in the present day context.

V. Computer Parallelism

Calculations involved in the boundary integral methodology are replication in nature, therefore, require a computer with replication based architecture. The calculations in A_{ij} could be carried out at individual PEs. Calculations involved in computational method are of pipelining nature, therefore, require a computer with pipelined architecture.

A common principle machine with the capability of taking replication based or pipelined based algorithms is much desired for the computational aerodynamics. Mathematical calculations involved in the estimation of one value of A_{ij} in the boundary integral methodology are very large, in-contrast calculations involved in the estimation of ϕ at a point in the numerical approach are few. A group of pipelined PEs can be arranged to optionally operate for the larger calculation task (task that cannot be further pipelined). Alternatively, a ring type architecture connecting PEs with the arrangement of having a data storing memory bus can serve this purpose. Figure 3 shows such architecture where the calculations in A_{ij} can be placed at various intervals, also the computations of ϕ in numerical approach can be pipelined. Provision for scheduling parallelism is of importance. One prime difference in performing calculations in [A] and processing vectors in ϕ exists. Calculations in [A] require an array processor in which local memory parallel architecture (each processor having its own memory) is required. Processing vectors in ϕ involve large data from previous iterations. This requires a vector processor operating on shared memory architectures. Calculations in ϕ involve operations under high data environment. Access to common memory via global bus is much needed. Several random access memory units operating in parallel can be arranged on the bus. Data in terms of geometry and space discretization schemes can be carried on the bus. Access to such data (storing and fetching) is not of priority concern when dealing with boundary integral method. However, in the case of numerical approach rapid access to such data is of prime requirement. Since, the previous iteration values form the data for the next iteration, active data accessing is required. Accessing and processing data comes much from the machine organisation. While the boundary integral approach involves much of data processing (fortran functions for arithmetic calculations), the numerical methods involve much of data accessing. For data processing, jump processors need be available for the PEs in the ring. For quick data accessing, multi-operation computer requires several random access memory units operating in parallel. For x memory units and y operands to be accessed at any time, $x \geq y$ readily implies. Frequent rearrangements in memory are time consuming and create difficulty in allocating proper storage schemes for various computations in progress. Therefore, fixed storage scheme is desirable, which

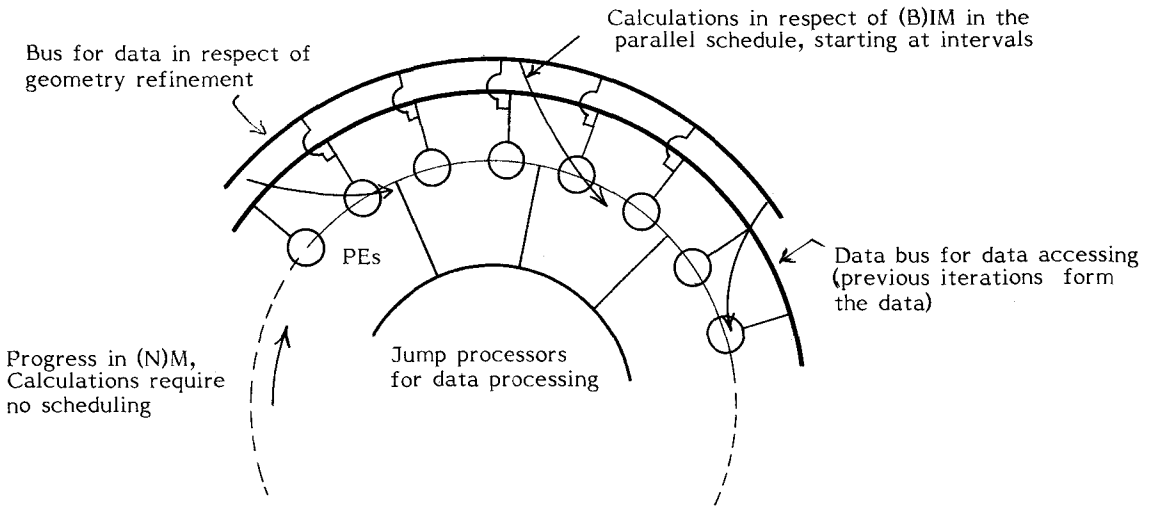

Fig. 3. Ring Architecture for CFD Work.

is much feasible with the suggested ring structure. For $x = y$ with a straight or skew type storage, the multi-operation machine slows down during data accessing in some particular directions. More redundant memory units[9] i.e. $x > y$ are needed for efficient data accessing. Geometry discretization involves rapid geometry refinements near the shock locations. The shock location shifts during the iterative process until settling at a final iteration value. Geometry data under such circumstances varies very fast as the iterations proceed. For carrying large and rapidly varying geometry data, a second bus is suggested, Fig. 3.

Do loops involved in the boundary integral approach are DOALL (loops without cross iteration dependences). In the case of numerical approach, loops are FORALL (loops with cross iteration dependences). FORALL loops can be executed as DOALL if data from iterations are synchronized and made available at an early time step. Data aligning and conflict-free data accessing is required. Indexing a memory system, provisioning more redundance memory units, appropriate scheduling of parallel programmes and memory management during the execution of parallel Do loops are much desired characteristics for optimal functioning of the computer. Nodes of the PEs are assigned with the task of large data handling, data transfer and data acquisition for processing vectors. Additional task of artificial intelligence and symbolic manipulation can be assigned to the algorithms. After a careful judicial task allotment, the availability of extra nodes in a PE need be assessed. Such nodes can be further pipelined.

While the high speedups are possible through large number of PEs, along with increase in speedup comes the decrease in system efficiency. If more processors are added to the system, processor idle time significantly goes up. This is because of reasons such as the contention for shared memory, time involving communication between processors and the software structure that cannot be tailored to keep busy any arbitrary number of processors. Tradeoff between the speedup and efficiency need be assessed, limiting the feasible parallelism[10]. To achieve a given speedup, efficiency penalty must be decided. Average parallelism ($\bar{p}$) is a good index of the software structure[10]. In an ideal situation $SR = n_p/n_s$ and $n_p/n_s \backsim \bar{p}$. Thus when $SR = \bar{p}$ i.e. the number of processors available is equal to the average parallelism, the speedup and efficiency are at the tradeoff point. The $\bar{p}$ however, varies with algorithms that are called upon to handle different tasks. In the case of matrix computations, $\bar{p} \backsim kM$ is reported in several references. In the case of solution to tridiagonal set of simultaneous equations, $\bar{p} \backsim kM/\text{Log } M$ is broughtout in some references[9,10]. Apparently, maximum benefit of parallel operations lies in matrix preparation part of the numerical algorithm. Hardware advances are the miracles of chips. Parallelism and modularity will be much desired architecture of futuristic computers for the AI/CA work.

VI. Conclusions

Computational task involved in aerodynamic calculations can be heavily pipelined. (Low n_s, high n_p/n_s) pair computer can be relied upon for fast computations in the boundary integral approach. High (n_s, n_p/n_s) pair value computer is required for fast numerical work. Paralleling an algorithm can result in loss of efficiency of the numerics convergence. Schemes to compute such algorithms need be altered to recover the convergence rate. High speedups do not guarantee faster processings, there could be decrease in machine efficiency. Decrease in machine efficiency could be severe for certain cases of algorithms. Appropriate scheduling of Do loops and memory management for conflict-free data accessing are much desired characteristics of machine for the CFD work.

VII. Acknowledgement

Author is thankful to Professor UC Durgapal, Chairman, Faculty of Guided Missiles for giving the guidelines to write this paper. Author expresses

his gratitude to Dr. E. Bhagiratharao, Director and Dean, Institute of Armament Technology, Pune, for the resources and encouragement given to pursue this work.

VIII. References

1. Hockney, R.W., "Characterization of Parallel Computers and Algorithms," Numerical Methods in Aeronautical Fluid Dynamics, Academic Press, 1982.

2. Ghosh, S.S., and Gupta, S.C., "Accelerating the Finite Difference Scheme", Proceedings of the 4th National Conference on Aerodynamics, IIT, Madras, India, Dec. 1988.

3. Piers, W.J. and Slooff, J.W., "Calculations of Transonic Flow by Means of a Shock Capturing Field Panel Method," AIAA paper 79, 1459.

4. Hicks, R.M., and Vanderplaats, G.N., "Application of Numerical Optimization to the Design of Low-Speed Airfoils", NASA TMX-3213, March 1975.

5. Hancock, G.J., "Aerodynamics - the Role of the Computer", Aeronautical Journal, August/ September 1985.

6. Alison E. Andrews, "Progress and Challenges in Application of Artificial Intelligence to Computational Fluid Dynamics", AIAA Journal Vol. 26, No. 1, January 1988.

7. Joe F. Thompson, "Grid Generation Techniques in Computational Fluid Dynamics", AIAA Journal Vol. 22, No. 11, November 1984.

8. Patrick J. Roache and Stanly Steinberg, "Symbolic Manipulation and Computational Fluid Dynamics", AIAA Journal Vol. 22, No. 10, October 1984.

9. David J. Kunck, "Multioperation Machine Computational Complexity", Complexity of Sequential and Parallel Numerical Algorithms, Academic Press, 1973.

10. Derek L. Eager, John Zahorjan and Edward D. Lazowska, "Speedup Versus Efficiency in Parallel System", IEEE Transaction Computers, Vol. 38, No. 3, March 1989.