12

A Critical Analysis of Multigrid Methods on Massively Parallel Computers

Lesley R. Matheson[1] and Robert E. Tarjan[2]

ABSTRACT The hierarchical nature of multigrid algorithms leaves domain parallel strategies with a deficiency of parallelism as the computation moves to coarser and coarser grids. To introduce more parallelism several strategies have been designed to project the original problem space into non-interfering subspaces, allowing all grids to relax concurrently. Our objective is to understand the potential efficiency of standard and concurrent multigrid algorithms on existing and proposed massively parallel machines. We study model problems on simple domains discretized with finite difference techniques on block structured meshes in two and three dimensions with up to 10^6 and 10^9 points, respectively. Performance of the standard domain parallel V and F cycle schemes is compared to several proposed concurrent algorithms. The multigrid strategies are studied in several models of parallel computation, designed to reflect both the characteristics of existing machines and those of the proposed next generation of multicomputers. These models include a SIMD fine-grain model which contains a large number ($10^4 - 10^6$) of small (bit-serial) processors as well as a SPMD medium-grain model with a more modest number (256-16,384) of powerful (single chip) processors interconnected through a single stage or multistage communication network. Our analysis suggests that obtaining acceptable levels of performance requires optimization techniques which address the salient characteristics of each architectural class of massively parallel computers. With the appropriate optimization techniques, a comparison indicates that the F-cycle is potentially more efficient than the V-cycle despite the relative increase in coarse grid activity. In addition, the analysis suggests that subspace parallelism is too expensive to be practical, except under a very limited set of conditions, on either class of massively parallel computers.

---

[1]Department of Computer Science, Princeton University and NEC Research Institute
[2]Research at Princeton University partially supported by the National Science Foundation, Grant No. CCR-8920505, the Office of Naval Research, Contract No. N0014-91-J-1463, and by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), a National Science and Technology Center, Grant No. NSF-STC88-09648.

Introduction

Massively parallel computers can potentially deliver the large increase in processing speeds required for the solution of many multigrid applications. Yet the architectural characteristics of these machines present a set of well defined algorithm design incentives which need to be addressed if this potential is to be realized. We've studied a representative set of parallel multigrid strategies in several different models of computation. The purpose of this analysis was to try to gain some insight into how general strategies might compare and perform on different architectural classes of massively parallel machines. In addition, we hoped to gain insight into which implementation issues and strategies might have a substantial impact on the overall performance of multigrid algorithms on these machines.

1 Massively Parallel Computers

The evolution of massively parallel computers has produced at least two significant architectural classes. The machines of both classes are networks of physically distributed processing and memory elements but differ substantially in other architectural characteristics. The early generation of massively parallel computers were fine grained with from 1K to 64K processing elements. The processing elements were small, bit-serial processors commonly packed eight to a chip. The floating point rates of these processors were slow. The processors communicated through various permutation networks consisting of a wide range of topologies. This class of machines executes in either synchronous SIMD or asychronous MIMD mode. Though this generation consists of older machines such as the Intel Hypercube, several more recent machines include the Thinking Machines CM200 and the Maspar MP1 and MP2 series. The current generation of massively parallel computers has been motivated by the the proliferation and decreased cost of powerful, "workstation-size" microprocessors. These machines are multicomputers, interconnection networks of physically distributed processors and memory, linked in a variety of different topological configurations. The processors are generally "off the shelf" single-chip RISC microprocessors. They can perform integer and floating point computation significantly faster than the bit-serial processors which characterized many machines of the previous generation. The increased size, cost and speed of the individual processing elements has delineated a medium grain size for the current generation. Most of the machines are targeted for the range of 1K processors, with larger machines possibly ranging up to 16K processors. The current machines generally exhibit slow interprocessor communication speeds relative to on-chip events. This is frequently a result of handling the network communications processing in the software layer. Unlike the more rigid SIMD and asynchronous MIMD patterns of the previous generation most of the newer machines execute the same program on each processing element with different data, enforcing synchronization only as required by interprocessor communication. The current generation includes: the CM5 by Thinking Machines, a network of Sun SPARC processor nodes potentially with vector accelerators, connected in a fat tree topology; the Touchstone Delta, developed by Intel and Caltech, a three dimensional mesh of two Intel i860s per node; the Paragon by Intel, a 3D mesh topology with one to four i860 processors per node; the Kendall Square Research machines, a hierarchy of concentric rings with shared virtual memory, with two custom designed chips per node; a Cray Research machine with DEC Alpha processors connected by a yet unrevealed topology.

2 Analysis Methodology

The emergence of these two architectural classes motivated us to study the potential performance of multigrid algorithms using practical models of computation which reflect the salient characteristics of each generation of massively parallel computers. The guiding philosophy behind the development of these models was to strike a reasonable balance between machine independence and practicality, simplicity and accuracy. The models used in this analysis are abstract, based not on a particular machine but on the substantive characteristics of a particular class. The models have parameters to facilitate analysis under different ratios of problem to machine size, and allow the incorporation of changes in technology, such as increases in on-chip computation speed or a decrease in network communication latency. On these models we implemented a representative set of parallel multigrid algorithms in order to try to compare the efficiency of several general strategies on each class of machine. These algorithms included several standard multigrid algorithms as well as more exotic concurrent methods. These implementations produced performance predictions in the form of analytic complexity measures. The complexity measures were interpreted with sensitivity analysis to provide insight into the impact of changes in parameters such as problem size, machine size, the latency of a network communication and on-chip computing rates. We examined these analytic and numerical results to try to understand their implications for algorithm design and implementation on each class of massively parallel computers.

3 Overview

The analysis presented in this paper implements four multigrid algorithms on three models of computation. The results are ordered by model. The first model is the abstract PRAM model. The last two are the practical models based on the early and current generation of massively parallel computers. The four multigrid algorithms include two standard cycling schemes, the V-cycle and the F-cycle, and two concurrent methods, the Gannon-Van Rosendale algorithm and the Chan-

Tuminaro algorithm. Following the results within each model overall conclusions are drawn.

4 Parallel Multigrid Algorithms

Four representative parallel multigrid algorithms, two standard and two concurrent strategies, were analyzed. We looked at simple problems on simple domains in order to provide an initial understanding of the interplay between algorithmic strategies and architectural characteristics. The algorithms solved model problems discretized onto block structured grids using second order finite difference techniques. The problem domains were square and cubic with up to 10^6 and 10^9 points respectively. The iteration schemes were simple explicit weighted Jacobi schemes. The grid hierarchy was generated using a coarsening ratio of two. Inter-grid transfer operators are simple, 5-9 full-weight linear operators. The implementations were parameterized to allow the analysis of more complicated, practical problems such as the Euler and Navier Stokes equations. In addition to considering the standard V-cycle and F-cycle algorithms [5], the analysis considered the performance of two concurrent algorithms. The first algorithm, the Chan-Tuminaro algorithm [4], decomposes the residual from several pre-relaxation sweeps on the finest grid into a set of approximately orthogonal subspaces, projects the subspaces onto the hierarchy of grids, relaxes concurrently on all grids and synthesizes the solution on the finest grid. This process is repeated until truncation error accuracy is obtained on the finest grid. In our analysis the Chan-Tuminaro algorithm decomposes the space over log N steps into log N subspaces by repeatedly bifurcating the residual with a product of the restriction and interpolation matrices. For each grid the residual is ideally split into high and low frequency error components with respect to the relaxation matrix. The high frequency component remains on the fine grid while the low frequency component is projected to the coarser grid. Concurrent relaxation proceeds for O(log N) steps, to amortize the O(log N) lower bound on decomposition, and then the solution is interpolated to the finest grid using standard linear operators. The second concurrent algorithm, the Gannon-Van Rosendale algorithm [1], generates subspace parallelism by decomposing the original problem space using the same frequency bifurcation techniques as those used in the Chan-Tuminaro algorithm but radically redirects the algorithm data flow. In this algorithm as the residual components are decomposed and projected to coarse grids, coarse grid correction information is computed and interpolated to fine grids. This bi-directional flow of information occurs after $O(\log \log N)$ concurrent relaxation sweeps. Thus the Gannon-Van Rosendale concurrent strategy differs fundamentally from the Chan-Tuminaro strategy in that it immediately propagates new approximation information. In addition, the Gannon-Van Rosendale strategy maintains the subspace decomposition by frequent local adjustments without propagating all information through the finest grid.

5 The Parallel Random Access Model

The Parallel Random Access Model of Computation (PRAM) is an ideal model which is somewhat impractical but can provide a fundamental measure of how much parallelism is available in an algorithm. The PRAM posits an arbitrairily large number of processors executing independently and a global shared memory. In the model there is a unit cost for an arithmetic operation. On a PRAM (CRCW) there are no inter-processor communication costs, no memory coherency costs and no synchronization costs. A clear definition of the PRAM can be found in [2].

6 Results on the PRAM

Standard Algorithms

Though the F-cycle is computationally more efficient that the V-cycle by a logarithmic factor, as the number of processors grows large relative to the size of the domain the performance of the two schemes becomes asymptotically equivalent on a PRAM. The asymptotic complexity of the V-cycle and the F-cycle on a PRAM are $O((N^d/P)\log N + \log^2 N)$ and $O(N^d/P + \log^2 N)$ respectively, on a domain with N^d points in d dimensions with P processors.

At or near full concurrency a comparison of the two standard schemes depends on constant factors such as the number of relaxation sweeps and the number of iterations of the F-cycle. If the F-cycle converges in one iteration, it is more efficient than the (2,1) V-cycle if the number of relaxation sweeps is less than 10 regardless of the number of available processors. If more than one F-cycle iteration is required, and more than a few relaxation sweeps are performed per iteration, the outcome depends on the number of processors [3]. With four iterations of the F-cycle and 64K processors (approximately 10 points/processor) processors, the V-cycle is always more efficient.

Concurrent Algorithms

On a PRAM the concurrent algorithms are asymptotically inferior to both of the standard algorithms except at full concurrency. At full concurrency the Gannon-Van Rosendale algorithm is asymptotically more efficient by a factor of $O(\log \log N/\log N)$, while the Chan-Tuminaro algorithm is asymptotically equivalent. Analysis on the PRAM predicts that any advantage of the Gannon-Van Rosendale algorithm degrades rapidly as the number of processors decreases. The number of concurrent iterations in the Gannon-Van Rosendale algorithm is bounded by O(log N) when $O(\log \log N)$ relaxation sweeps are performed each iteration. A Gannon-Van Rosendale iteration is somewhat less effective than a V-cycle iteration. When the number of concurrent iterations is more than a small constant factor of the number of V-cycle iterations, the V-cycle is more efficient regardless of the number of processors. If the ratio of Gannon-Van Rosendale to V-cycle iterations is less than four, the Gannon-Van Rosendale algorithm can produce a small improvement with a very large number of processors. However, if the number of processors is small enough so that there are more than five points per processor, the Gannon-Van Rosendale algorithm is less efficient than the V-cycle regardless of the number of processors [3]. The Chan-Tuminaro algorithm must converge in a small constant number of concurrent iterations to be at least as efficient as the standard algorithms. The decomposition steps of the Chan-Tuminaro algorithm cannot be executed concurrently as in the Gannon-Van Rosendale algorithm and requires that information propagate through the entire hierarchy to the finest grid. A concurrent iteration is at least as effective as a V-cycle iteration. The analysis on a PRAM suggested that unless the algorithm can converge in a small fraction of the number of V-cycle iterations, the standard methods are more efficient. If the algorithm can converge in a small constant number of iterations, the strategy could be effective if there are a very large number of processors available [3]. The analysis of both algorithms suggests that in order for a concurrent method to be an efficient strategy the decomposition techniques must be fast and accurate and there must be $O(N^d)$ processors available, where N^d is the size of the domain of the finest grid. Without an accurate decomposition, interference inhibits converge and without a large number of processors costly subspace parallelism goes unexploited.

7 The Early Generation Model

The Early Generation Model attempts to capture the salient architectural characteristics of the previous generation of massively parallel computers. This class is characterized by fine grain networks of small processing elements operating in either SIMD or MIMD execution mode. The model is a two dimensional mesh of processors which communicate locally through single-stage nearest neighbor hops along the physical connections of the mesh (as well as along diagonal connections). A single stage communication is assigned a constant fixed cost and the cost of a multiple hop communications is approximately linear in the number of hops. The length of a message is a small constant number of bytes. The cost of an arithmetic operation is treated separately. An arithmetic operation is assigned a constant fixed cost which is based on an average of multiplication and addition floating point rates (without vector acceleration units) across this class of machines. The model executes all instructions in SIMD mode.

The model is a tool to try to better understand the potential of fine grain parallelism. Small (bit serial) processing elements allow these machines to potentially range up to tens or hundreds of thousands of processors. Yet small processing elements tend to have slow floating point rates. The ratio of the cost of an arithmetic computation to a communicaiton is approximately 3:1. The fine granularity also suggests that local communication will be significantly more efficient than more distant communication. Because of the prohibitive cost of global or irregular communications on this class of machines, the model contains no over-arching multi-stage permutation network. Thus, distant communication incurs a high variable cost relative to the fixed cost of a single node communication. These defining characteristics are captured through constructs which are simple and abstract, but parameterized with data from working computers. Thus, the model provides abstract measures of performance but facilitates sensitivity analysis which can potentially clarify the practical implications of performance measures. A more detailed description of the model can be found in [3].

8 Results on the Early Generation Model

Standard Algorithms

Analysis on the Early Generation model suggests that the performance of multigrid algorithms on fine grain machines with high variable communication costs is highly dependent on the data mapping. With a simple static approach where the square domains are repeatedly quartered to create a 2D mesh of P subblocks and each subblock is assigned to its analog on the 2D mesh of processors, coarse grid communications become substantial. With a small number of processors fine grid computation dominates the execution time of both algorithms. As the number of processors grows, coarse grid communications dominate. With the static approach the asymptotic complexity of the coarse grid communications of the V-cycle and the F-cycle are equivalent. Thus, performance of the two standard algorithms depends on the number of processors available as well as the constant factors. The F-cycle is more efficient than the V-cycle only when there are a small number of processors, the algorithm converges quickly and executes only a small number of relaxation sweeps per iteration. With a dynamic mapping where the coarse grid point sets are contracted and expanded across the processor topology as grids coarsen and refine to maintain a compact data mapping, coarse grid communications can be significantly reduced. With this dynamic implementation the F-cycle is more efficient than the V-cycle even with a large number of processors and slow convergence. Figure 1 shows the total number of machine cycles required for both the V-cycle and the F-cycle for a two dimensional problem of 1,000,000 points when the number of processors varies from 10,000 to 250,000. The figure illustrates the greater efficiency of the F-cycle even with two iterations regardless of the number of processors available.

While the dynamic mapping significantly reduces coarse grid communication costs, the optimal number of processors for both standard multigrid algorithms is less than the number of points in the domain of the problem. As the number of processors grows, fine grid computation costs decrease as coarse grid communication costs increase. At some point the marginal reduction in computation costs is less than the incremental increase in communications costs and performance degrades with the addition of more processors. This optimum can be approximated analytically as a function of a small number of model parameters [3]. Figure 2 illustrates the optimum for the V-cycle which is approximately four points per processor. Thus, the hierarchical nature of the algorithms coupled with the penalty for distant communications results in excess parallelism degrading the performance of standard multigrid algorithms.

Concurrent Algorithms

When the concurrent multigrid algorithms are implemented using simple, practical data mappings, inter-grid communication costs become significant. With simple mappings at the completion of each concurrent iteration, data must travel across the entire processor mesh. In the simplest strategy, the Chan-Tuminaro algorithm, this inter-grid communication takes place every log N relaxation sweeps. This improves performance relative to the standard V-cycle because the concurrent algorithm exhibits a lower ratio of communication to computation. Figure 3 shows the performance of the Chan-Tuminaro algorithm and the standard V-cycle on a two dimensional problem when the number of concurrent iterations is one-half the number of standard V-cycle iterations. The figure shows that the concurrent algorithm only becomes competitive, under favorable convergence assumptions, at full concurrency. The figure also shows the difference in optimal performance of the two algorithms. Because the inter-grid communications costs of the concurrent method are lower than the coarse grid communications costs of the V-cycle, the optimal number of processors in this range of processors $(0 - N^d)$ is equal to the number of points in the finest grid. The Gannon-Van Rosendale algorithm does not perform as well as the standard V-cycle regardless of the number of processors. The more frequent decomposition and inter-grid transfers of the Gannon-Van Rosendale algorithm make it more expensive than the V-cycle regardless of the number of processors available. Like the V-cycle the Gannon-Van Rosendale algorithm achieves optimal performance below full concurrency. Figure 4 shows the performance of the Gannon-Van Rosendale algorithm and the standard V-cycle in two dimensions with 1,000,000 points when the number of concurrent iterations is equal to the number of V-cycle iterations. The figure illustrates the greater efficiency of the V-cycle and the optimum number of processors for both algorithms below full concurrency. The asymptotic performance of the Gannon-Van Rosendale algorithm can be improved by a logarithmic factor if a more complicated data mapping is used. This mapping reduces the inter-grid communication distances by interspersing the coarse grids, while keeping intra-grid communication distances small. The details of this mapping can be found in [3]. Though the mapping provides an asymptotic improvement, it is substantially more complicated and therefore unlikely to be practical.

9 The Current Generation Models

The Current Generation Models are designed to reflect the architectural characteristics of the current generation of massively parallel computers. These models reflect the medium grain size of this generation with the number of processors ranging from 256 to 16K. The models also reflect the fast floating point rates of the powerful workstation-size processing elements. Motivated by the large disparity between the speeds of on-chip and network events, the models reflect the costs of a two level memory hierarchy. The cost of a local memory access is included in the cost of an arithmetic operation while the cost of a remote memory access is treated separately. The models assume the processors operate in a Single Program Multiple Data mode of execution. The characteristics of the models are similar, differing only in their treatment of communications costs. These different treatments include a simple topologically blind model which simply assigns a fixed cost to any network communication pattern, a model which adds to the fixed cost a variable cost based on a topological approximation of the communications distance and a topologically specific single stage model. Each of these treatments reflects the large fixed cost of a network communications which is characteristic of this generation of massively parallel machines. Conservative model parameters result in a ratio of the cost of a floating point operation to the cost of a communication of approximately 1:250-500. A more detailed description of these models can be found in [3].

10 Results on the Current Generation Models

Standard Algorithms

Analysis on the Current Generation models of the standard V-cycle and F-cycle algorithms indicates that for large problem sizes medium granularity increases fine grid communications costs to unacceptable levels. On moderate sized machines, those with 1K to 4K processors, with a 2500 fixed communications cost (approx. 75 microseconds with a 33 MHz clock speed), the models predicted speed-ups of only 130 times over the serial implementation with approximately 87% of execution time spent on communication. The table below shows the speed-ups and efficiency of the V-cycle, in three dimensions, for different machine sizes under different assumptions of fixed and variable communications costs. The problem size is 10^9 points or 1000 points along each dimension. The mapping used is a simple variant of the two dimensional partitioning scheme.

Interpreting the data is not straightforward. From a theoretical perspective these speed-ups are far from linear. On the other hand computing the wall-clock times associated with these predictions, then scaling these model problem times to reflect the increased complexity of actual applications, produces running times which are unacceptably slow. These discouraging predictions are a result of very

TABLE 1. Speed-Up and Efficiency

Three Dimensional V-cycle, $N^3 = 1,000,000,000$
Current Generation Models

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	49.1 19.19%	130.6 12.75%	338.1 8.25%	359.3 5.26%
500, 100	129.5 50.58%	389.2 38.01%	1102.2 26.92%	2969.2 18.12%
5000, 0	30.1 11.7%	79.2 7.7%	205.3 5.01%	526.7 3.22%
3600, 0	39.9 15.59%	106.8 10.42%	279.7 6.83%	722.5 4.4%

high communications latencies. In this range of processors, the fine grid communications costs dominate both the cost of the computation and the cost of coarse grid communications. The predictions of these models are in contrast to the asymptotic predictions of more abstract models of computation. Asymptotic analysis suggests the fine grid communications costs become negligible as the problem size gets large for a fixed range of machine sizes. These results suggest the huge imbalance between the cost of communication per word and the cost of a floating point computation causes communication time to dominate the time spent on computation, even with one billion points. The high fixed cost of a network communication coupled with a low spooling rate per word motivates trying to lower the average communication cost per word by transmitting large blocks of words per message. With large messages, the fixed cost of initiating a network communication can be amortized over a larger number of words, lowering the effective fixed cost per word. With the three dimensional V-cycle, the models suggest the ability to send variable length messages, up to 1000 words, produces a marked increase in solution speed in this range of processors, on problems up to one billion points. Table 2 below shows the speed-ups predicted for the three dimensional algorithm by Current Generation models.

With variable length messages, computation costs dominate the total execution time producing almost linear speed-ups in this range of problem to processor size. The corresponding efficiency levels are above 90%. These results suggest the average communications cost per word can be driven down far enough through the efficient transmission of large messages to effectively leverage the increased computational speeds of the current generation of microprocessors. Thus, the ability to package messages into large blocks, up to a 1000 word maximum, can potentially bring these machines closer to the goal of design tool performance on these prob-

TABLE 2. Speed-Up

Three Dimensional V-cycle
with Variable Length Messages
$N^3 = 1{,}000{,}000{,}000$

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	253.3	1006.1	3965.4	15,192.1
1000, 200	253.9	1010.3	3999.2	15,555.7
500, 100	254.2	1012.3	4016.9	15,773.9
100, 50	254.4	1013.7	4029.3	15,924.2
3600, 0	253.0	1004.2	3953.3	15,105.8

lems. The ability to bundle messages requires an optimized data partition which minimizes the maximum length of any subdomain border as well as minimizing the number of neighboring regions and load balancing the computation. With structured grids this type of optimized partition is straightforward. With unstructured meshes, however, more sophisticated partitioning techniques are required to produce well-shaped partitions. Several directions for unstructured multigrid partitioning strategies can be found in [3]. With both fixed, constant length and variable length message transmission, analysis on the Current Generation Models suggests that the F-cycle can be more efficient than the V-cycle. With fixed message lengths, this is due mainly to the reduced amount of fine grid communication of the F-cycle. With the ability to send large messages, the F-cycle outperformed the V-cycle in three dimensions because of the reduction in the amount of required computation.

Concurrent Algorithms

The medium granularity of the current generation leaves unexploited much of the subspace parallelism in the concurrent algorithms, making them far too expensive to be practical. With a machines in the range of 256-4096 processors producing more fine grained parallelism is inefficient for even moderate problems sizes. Figure 5 shows the performance of the standard V-cycle and the Chan-Tuminaro algorithm in three dimensions on a problem with 1,000,000 points. Within the medium grain range of machine sizes, the V-cycle is more efficient by at least a factor of two than the concurrent algorithm, even with favorable convergence assumptions.

11 Conclusions

The architectural characteristics of each class of massively parallel computers motivate different optimization strategies to facilitate the realization of the considerable processing power of these machines. Fine grain machines with a high variable cost component in communication costs motivate the optimization of the domain to processor topology mapping. The mapping needs to minimize the topological distance of the required communications. The medium grain machines of the current generation with their high fixed cost of a communication motivate an optimized domain partition. The partition needs to consist of "well-shaped" subdomains in which the maximum size of a subdomain border is minimized and the number of neighboring regions is small. This facilitates the transmission of large messages which allow the fixed cost to be amortized over a large number of words, significantly reducing the average fixed cost per word. With simple optimized strategies the F-cycle is a potentially more efficient than the V-cycle on massively parallel machines. The reduced fine grid activity potentially outweighs the additional coarse grid communications costs on many problems. Concurrent methods even with optimized strategies, require an accurate subspace decomposition and an unreasonably large number of processors to provide a practical alternative to the standard algorithms. This analysis suggests that this remains true regardless of the characterization of inter-processor communications costs.

References

[1] Gannon, D., and Van Rosendale, J., "On the Structure of Parallelism in a Highly Concurrrent PDE Solver", Journal of Parallel and Distributed Computing, vol. 3, pp. 106-135.

[2] JAJA J., "An Introduction to Parallel Algorithms", Addison-Wesley, Reading, MA., 1992.

[3] Matheson, L., "Multigrid Algorithms on Massively Parallel Computers", Phd. Dissertation, Princeton University, 1993.

[4] Tuminaro, R., "Multigrid Algorithms on Parallel Processing Systems", Phd. Dissertation, Stanford University, 1989.

[5] Briggs, W., Multigrid Tutorial, SIAM, Philadelphia, 1987

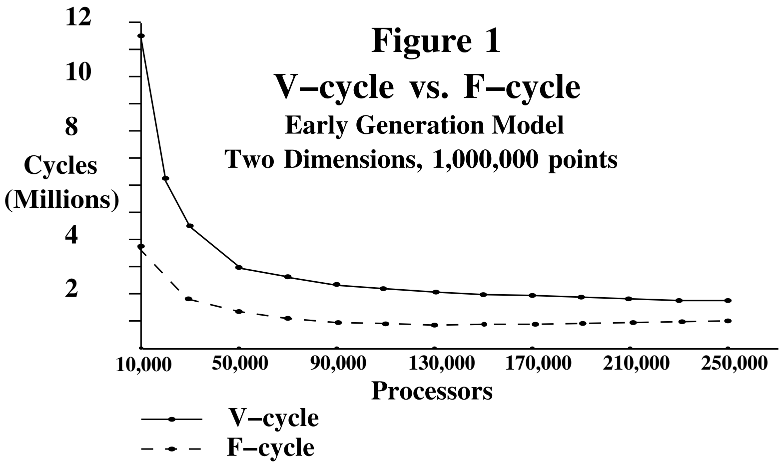
Figure 1
V−cycle vs. F−cycle
Early Generation Model
Two Dimensions, 1,000,000 points
12
10
8
Cycles
(Millions)
4
2
10,000
50,000
90,000
130,000
170,000
210,000
250,000
Processors
V−cycle
F−cycle

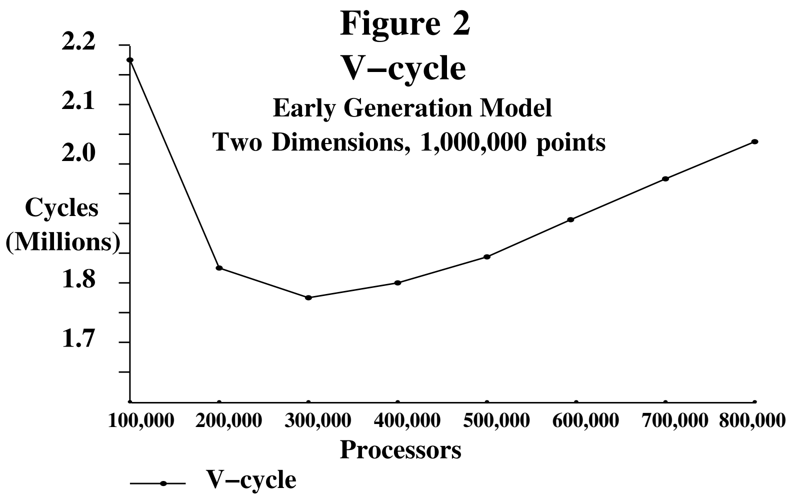
Figure 2
V−cycle
Early Generation Model
Two Dimensions, 1,000,000 points
2.2
2.1
2.0
Cycles
(Millions)
1.8
1.7
100,000
200,000
300,000
400,000
500,000
600,000
700,000
800,000
Processors
V−cycle

Figure 3

Chan–Tuminaro versus V–cycle

Early Generation Model, 1,000,000 Points, Ratio Iterations 1:2

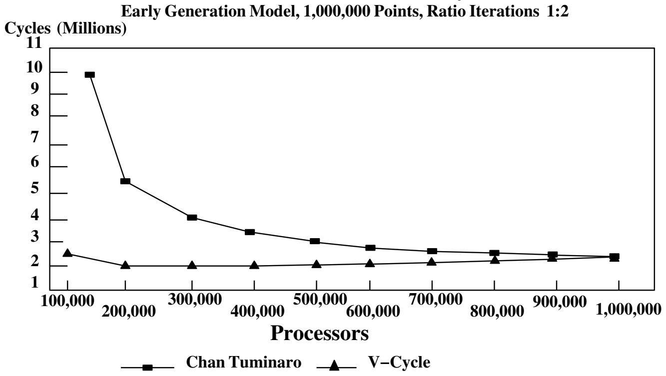

Figure 4

Gannon–Van Rosendale versus V–cycle

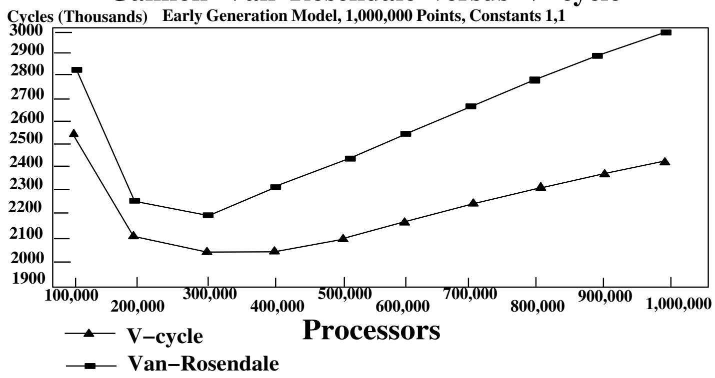

Figure 5

Standard versus Concurrent

Current Generation Model, 1,000,000,000 Points, Three Dimensions