A Parallel Multigrid Tutorial

Jim E. Jones

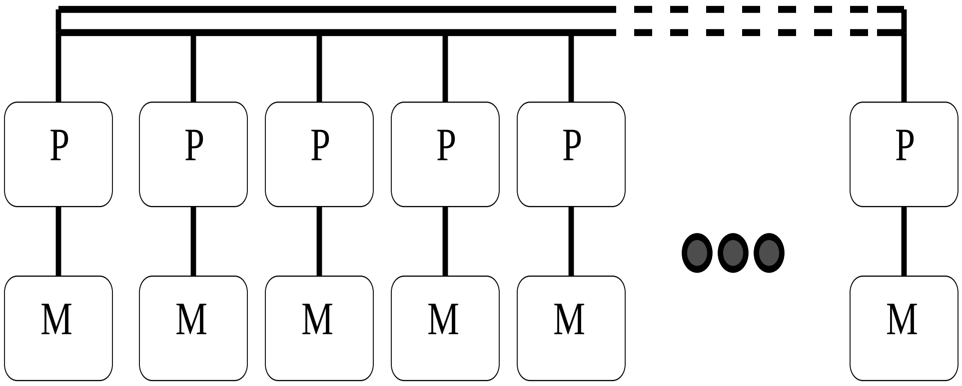
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Outline

- Parallel Computing Model
- Domain Partitioning
- Parallel Performance Metrics
- Model Analysis of Parallel Multigrid
- Numerical Examples
- Further Topics

Parallel Computing Model

Parallel computing achieved by multiple processes each with its own address space. A process can work on data in its own address space independently of the other processes. Processes exchange data through communication.

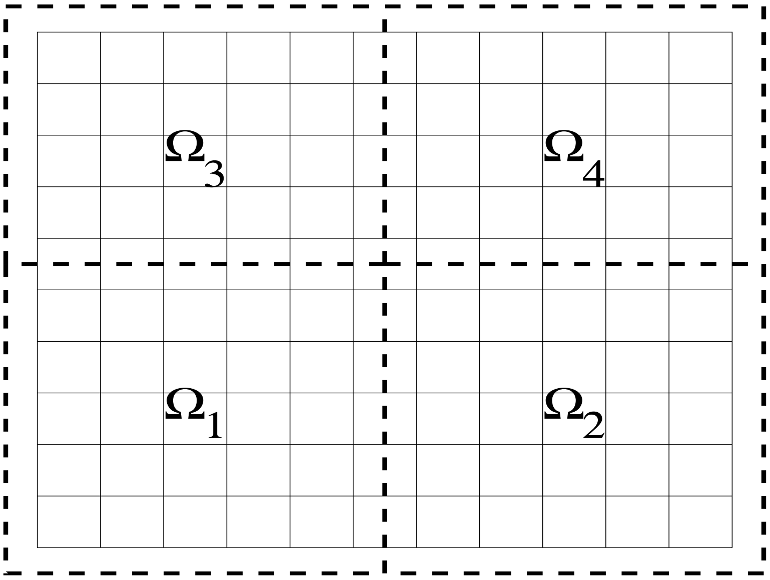


This model most closely matches the architecture of distributed memory machines where each process can be thought to correspond to one of the nodes of the machine. A node consists of one (or more) processor(s) and some local memory. Communication is by message passing using the network connecting the nodes.

Efficient programs for shared memory architectures may also correspond to this model where a the local address space may correspond to private data for a thread and and communication to copying data or synchronization.

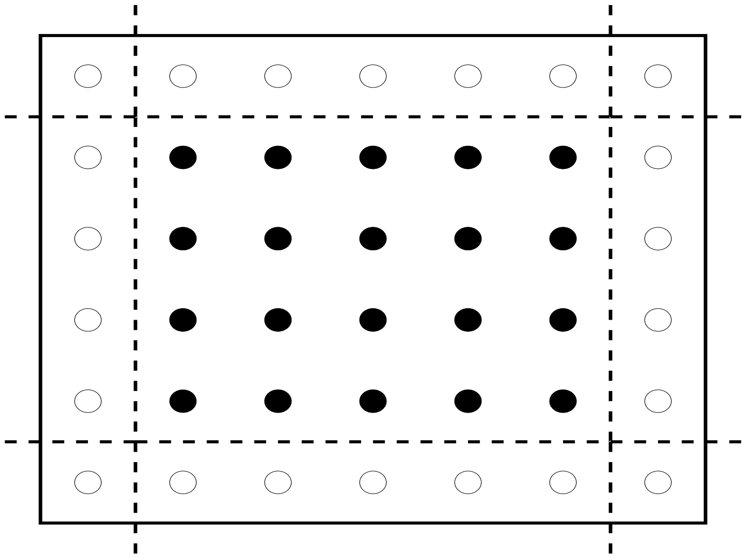
Domain Partitioning

For P processes, the grid Ω is partitioned into P non-overlapping subgrids Ω_q. Each process is responsible for calculations that result in changes of the approximate solution within its subgrid. To perform these calculations, a processor may need to receive values of the current approximate solution from other processes.



Load Balancing: To equally divide the work, would like each process to have nearly the same number of grid points in its subgrid.

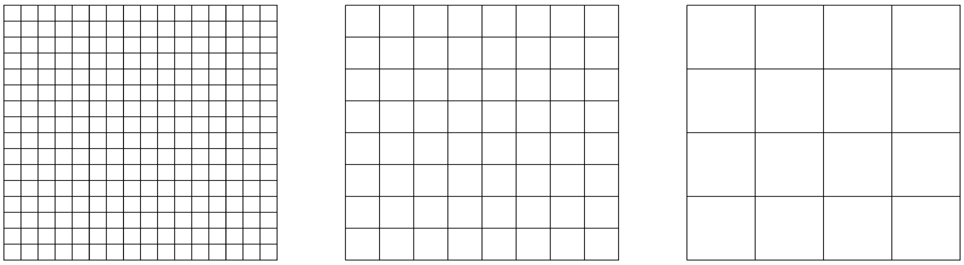
To avoid sending many small messages between processes, its better to have each process store not only data for its subgrid but also some small portion of data (ghost layers) from its neighboring processes.



Jacobi relaxation:

For i from 1 to $num\_iterations$

1) Each process q performs Jacobi relaxation within Ω_q.
2) Send messages to update ghost layers.



Grid h Grid 2h Grid 4h

Multigrid $V(\nu_1, \nu_2)$ cycle

1. Perform ν_1 smoothing steps on $A^hU^h = F^h$.

2. Set $F^{2h} = R_h^{2h}(F^h - A^hU^h)$.

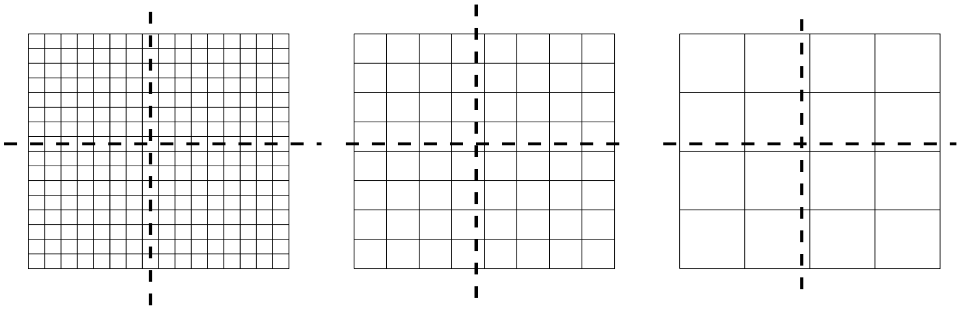
3. "Solve" $A^{2h}U^{2h} = F^{2h}$ by recursion.

4. Correct $U^h \leftarrow U^h + P_{2h}^hU^{2h}$.

5. Perform ν_2 smoothing steps on $A^hU^h = F^h$.

MG parallelized by domain partitioning

Partition the fine grid, coarse grid partitions are aligned with fine grid, i.e. the point $i \in \Omega_{2h}$ belongs to processor q if the corresponding point in Ω_h does.

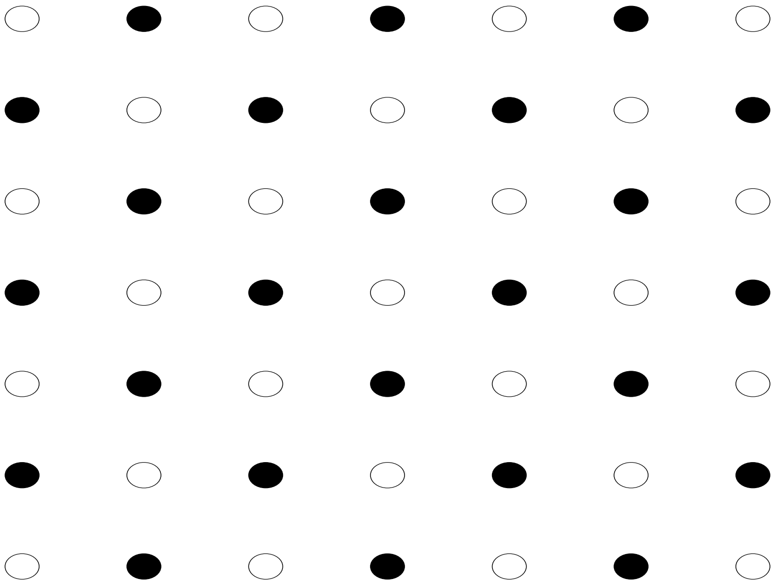


Grid h Grid 2h Grid 4h

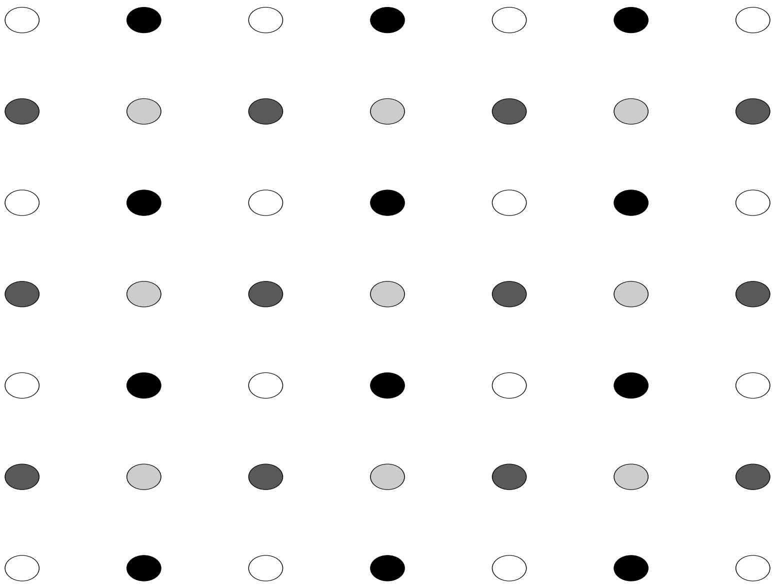
Standard multigrid components, say, linear interpolation and full weighting for restriction.

Smoothing by Jacobi or multicolor Gauss-Seidel.

No changes to the algorithm. The parallel code will produce the same numerical results as the serial code and will produce the same numerical results no matter how many processors it runs on.



Red-Black Gauss-Seidel



4-Color Gauss-Seidel

Parallel computing performance metrics

Let $T(N, P)$ be the time to solve a problem with N unknowns on a computer using P processors.

Speedup:

$S(N, P) = T(N, 1)/T(N, P)$, perfect utilization of resources when $S(N, P) = P$.

Scaled Efficiency:

$E(N, P) = T(N, 1)/T(PN, P)$, perfect utilization of resources when $E(N, P) = 1$.

A code is *scalable* if

$$E(P, N) \geq E(N) > 0, \text{ as } P \to \infty.$$

Scalable linear solvers

Algorithm Scalability:

- Computational work (per iteration) is $\mathcal{O}(N)$. (Jacobi, Gauss-Seidel, Multigrid, not Gaussian elimination).

- Convergence factor per iteration is $\mathcal{O}(1)$. (Multigrid, Multilevel Domain Decomposition, not Gauss-Seidel or Jacobi.)

Scalable linear solvers (cont.)

Implementation Scalability:

- A single iteration is scalable on the parallel computer. (Multigrid?).

Both algorithmic and implementation scalability are required for a code to be scalable.

Scalability Model

Time to communicate n doubles between processors is

$$T_{\text{comm.}} = \alpha + \beta n,$$

here α is the *latency* or startup time and β is the time to transfer a single double. The *bandwidth* of a communication channel is $1/\beta$.

Time to do n floating point operations is

$$T_{\text{comp.}} = fn,$$

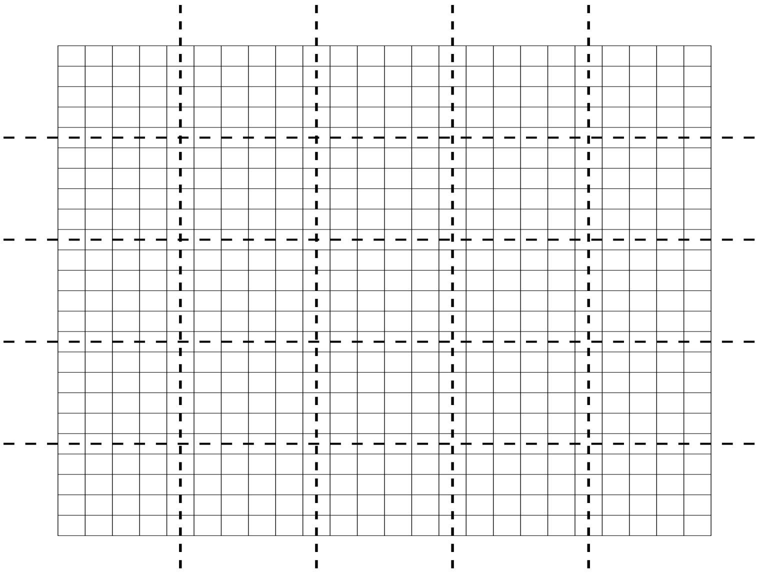
here $1/f$ is the Mflops achieved.

Approximate values for SP2 (Gropp 97)

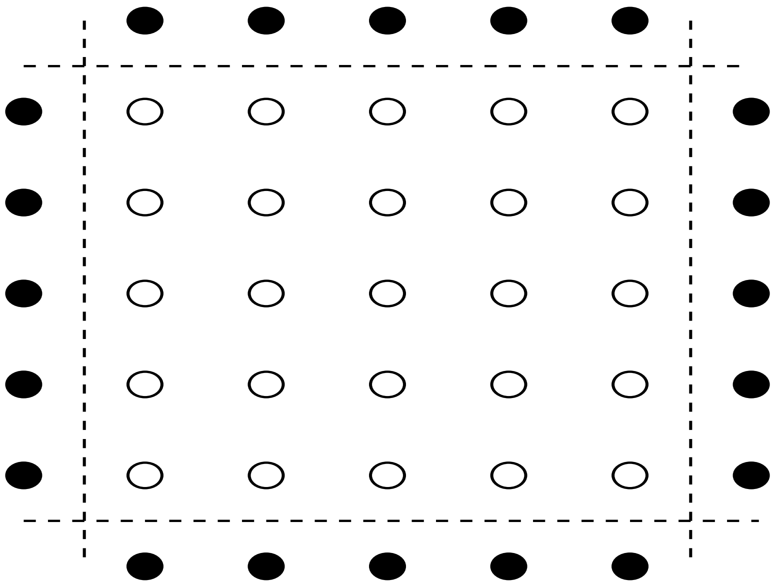
$$\alpha = 5 \times 10^{-5}\text{sec}$$

$$\beta = 1 \times 10^{-6}\text{sec}$$

$$f = 8 \times 10^{-9}\text{sec}$$



Assume 2d problem of size $(pN)^2$ is distributed to p^2 processors so that each processors subgrid is N^2. Operator is 5-point. Model accounts for communication and computation in relaxation only.



Time for relaxation on finest grid (level 0):

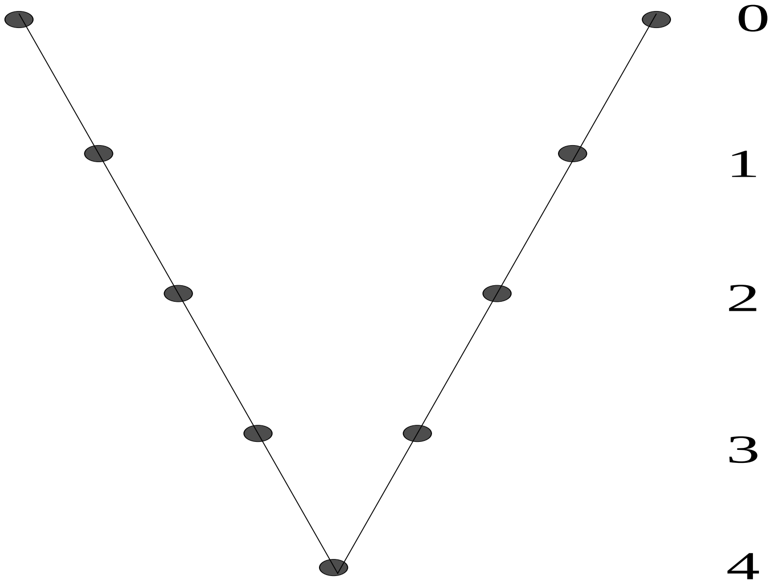
$$T_0 \approx 4\alpha + 4N\beta + 5N^2 f.$$

Time for relaxation on first coarse grid (level 1):

$$T_1 \approx 4\alpha + 4(N/2)\beta + 5(N/2)^2 f.$$

Time for relaxation on level l:

$$T_l \approx 4\alpha + 4(N/2^l)\beta + 5(N/2^l)^2 f.$$



For V-cycle

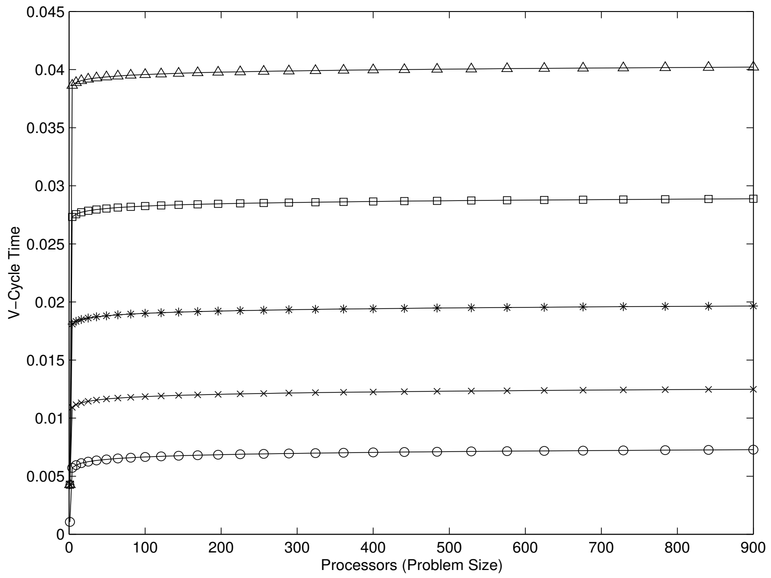
$$
\begin{aligned}
T_v &\approx \sum_l 2T_l \\
&\approx 2\sum_l \left[4\alpha + 4(N/2^l)\beta + 5(N/2^l)^2 f\right] \\
&\approx 8\alpha(1+1+1+\ldots) \\
&+ 8N\beta(1+1/2+1/4+\ldots) \\
&+ 10N^2 f(1+1/4+1/16+\ldots) \\
&\approx 8L\alpha + 16N\beta + 40/3N^2 f.
\end{aligned}
$$

Here L is the total number of multigrid levels: $L \approx \log_2(pN)$.

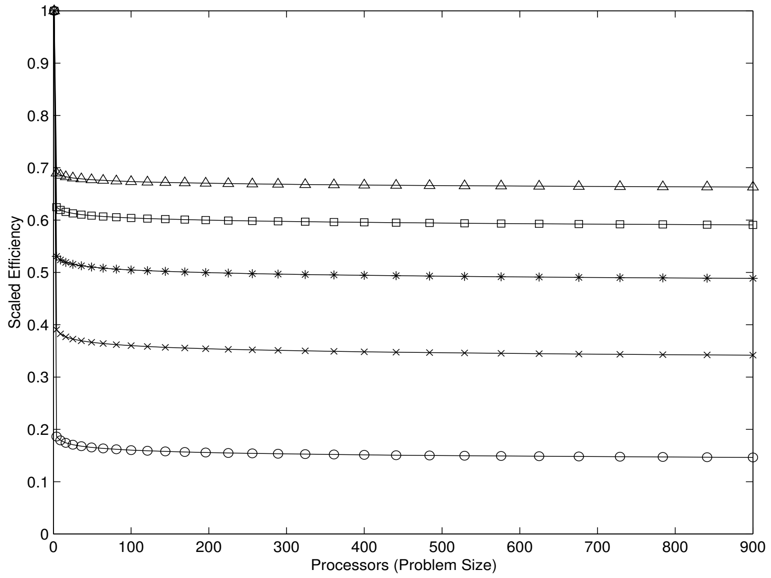
Scalability: $\lim_{P\to\infty} T_v(N,1)/T_v(PN,P) = \mathcal{O}(1/\log(P))$.

Model Predictions for solution times and scaled efficiencies. Problem size per processor 100×100, 200×200, 300×300, 400×400, and 500×500.

$\alpha = 5 \times 10^{-5}$sec, $\beta = 1 \times 10^{-6}$sec, $f = 8 \times 10^{-9}$sec



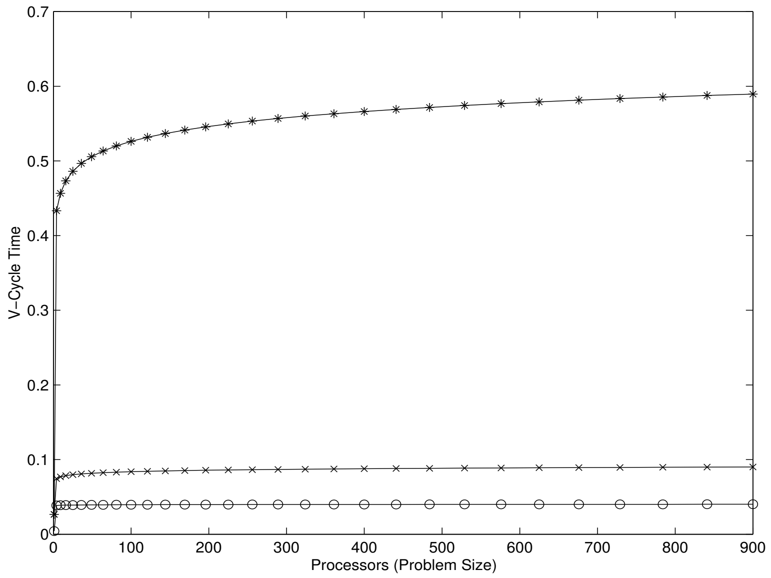
Solution Times



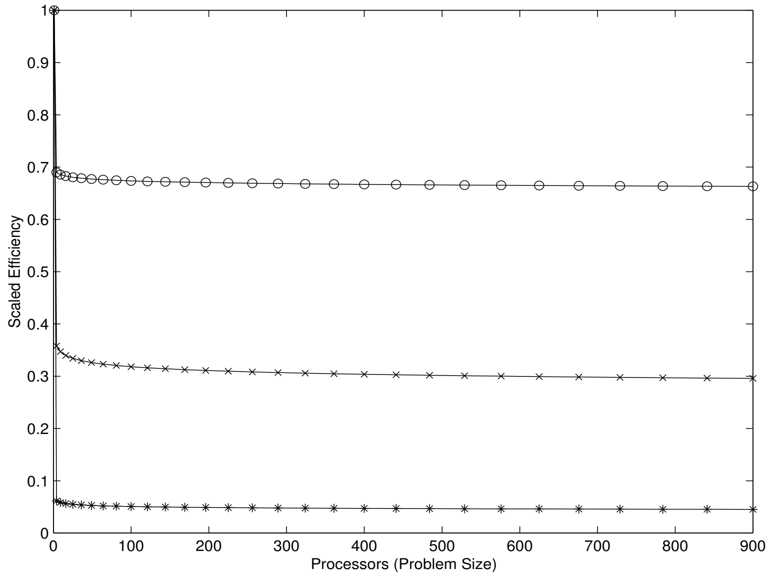
Scaled Efficiencies

Model Predictions for solution times and scaled efficiencies. Latencies $\alpha = 5\times10^{-5}$sec, 5×10^{-4}sec, 5×10^{-3}sec.

$N = 500 \times 500, \beta = 1 \times 10^{-6}\text{sec}, f = 8 \times 10^{-9}\text{sec}$



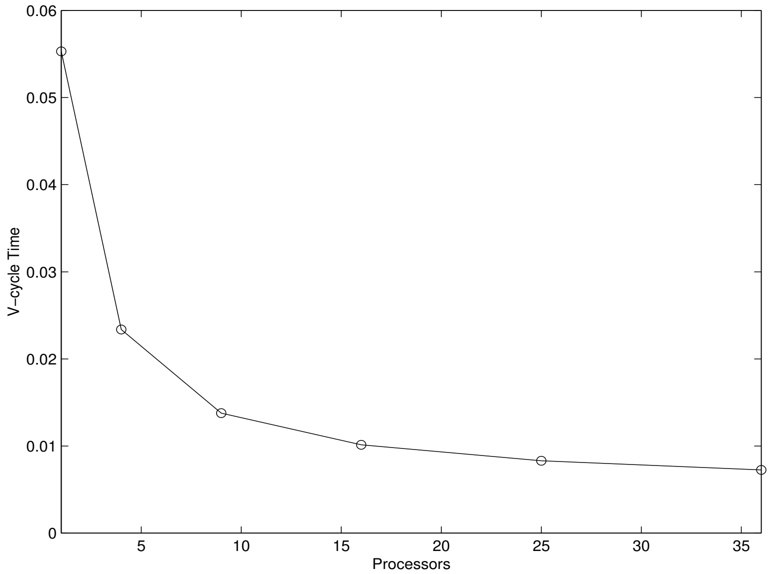
Solution Times



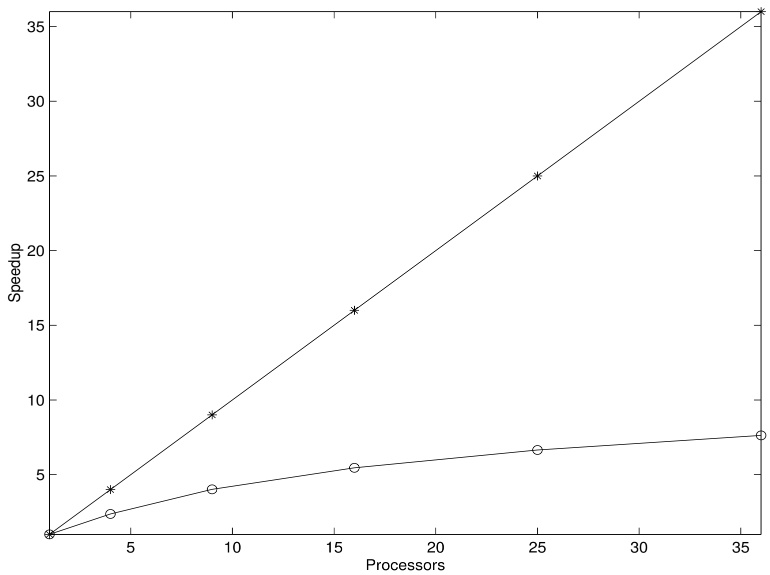
Scaled Efficiencies

Model Predictions for solution times and speedup. Fixed problem size of 720×720.

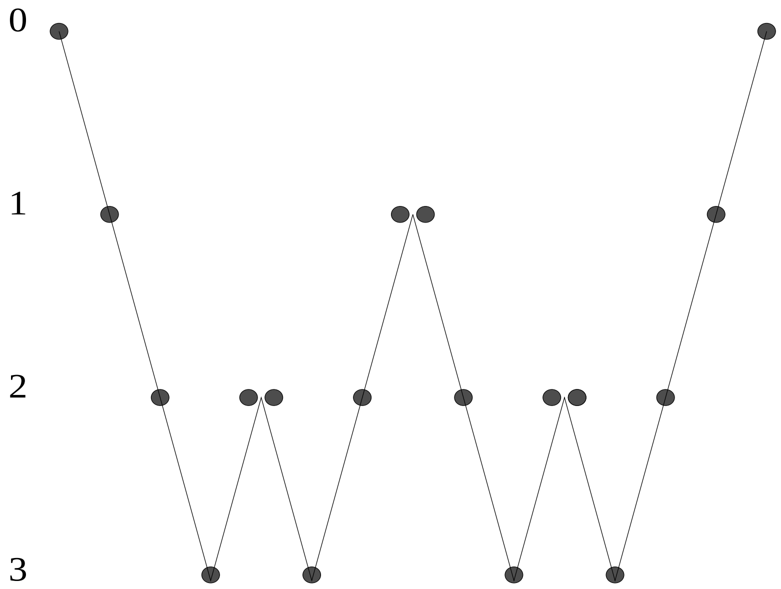
$\alpha = 5 \times 10^{-5}\text{sec}, \beta = 1 \times 10^{-6}\text{sec}, f = 8 \times 10^{-9}\text{sec}$



Solution Times



Speedups



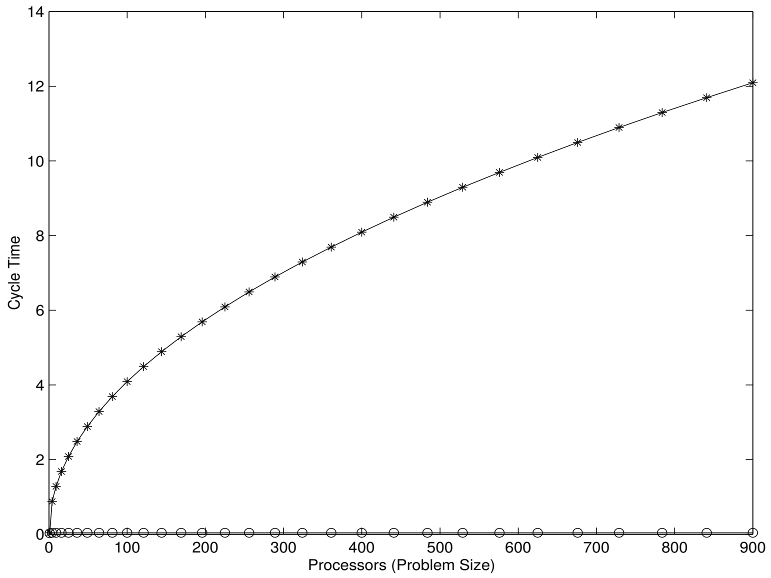
For W-cycle

$$\begin{aligned}
T_w &\approx \sum_l 2 \times 2^l T_l \\
&\approx 2 \sum_l 2^l \left[4\alpha + 4(N/2^l)\beta + 5(N/2^l)^2 f \right] \\
&\approx 8\alpha(1 + 2 + 4 + \ldots) \\
&+ 8N\beta(1 + 1 + 1 + \ldots) \\
&+ 10N^2 f(1 + 1/2 + 1/4 + \ldots) \\
&\approx 8(2^{(L+1)} - 1)\alpha + 8LN\beta + 20N^2 f.
\end{aligned}$$

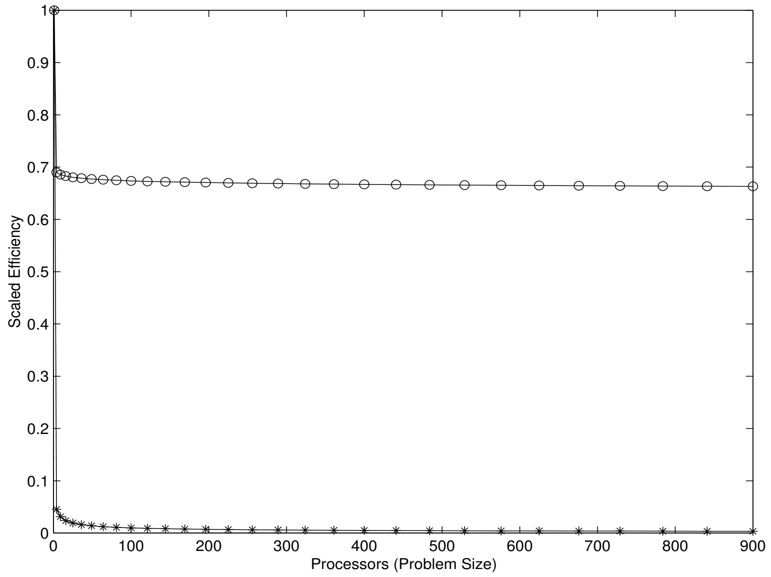
Here again L is the total number of multigrid levels: $L \approx \log_2(pN)$.

Model Predictions for solution times and scaled efficiencies for V and W cycles. Problem size per processor 500×500.

$\alpha = 5 \times 10^{-5}$sec, $\beta = 1 \times 10^{-6}$sec, $f = 8 \times 10^{-9}$sec



Solution Times

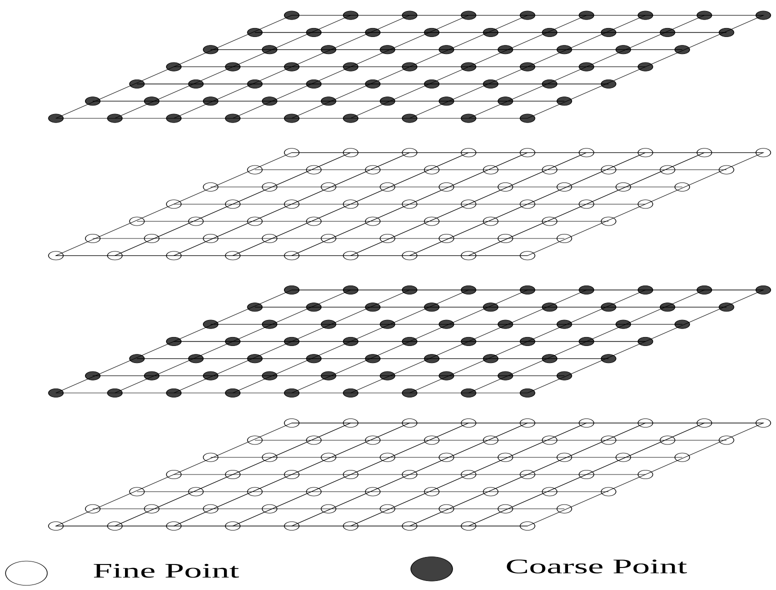


Scaled Efficiencies

Numerical Results for SMG

Multigrid code using Schaffer's algorithm. Designed for 3D diffusion problems with anisotropic and/or discontinuous coefficients.

- Semicoarsening
- Plane relaxation
- Operator induced interpolation



Parallelization by domain partitioning.

Written in C with MPI for message passing.

Scalability model for SMG

Recall previous result:

$$T_v \approx 8L\alpha + 16N\beta + 40/3N^2 f.$$

For SMG:

$$T_{SMG} \approx 4L(1 + 2L(L+1))\alpha + 20N^2 L\beta + 48N^3 f.$$

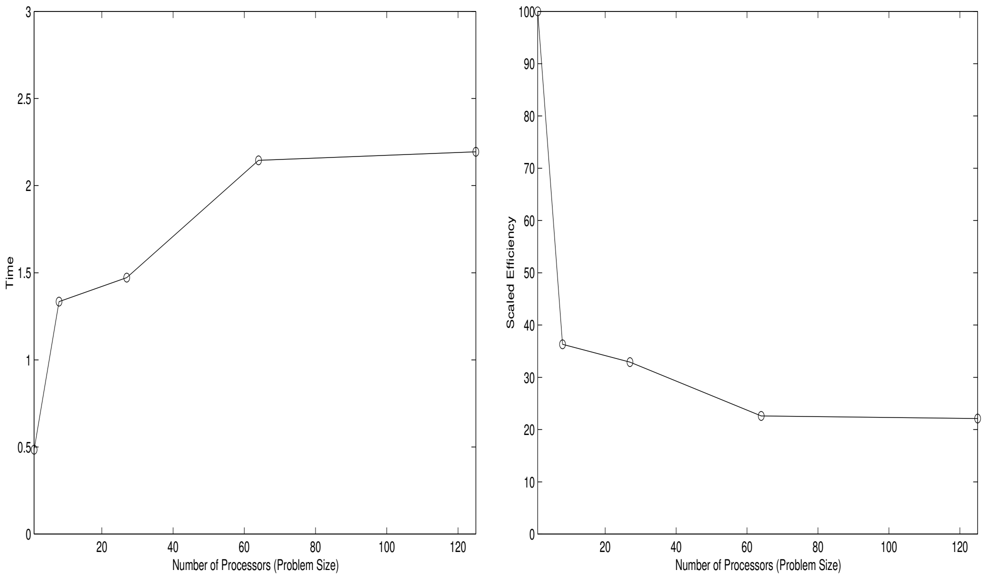
SMG has more communication due to the plane solves (accomplished by 2D multigrid using line solves accomplished by cyclic reduction).

Implementation scalability for SMG

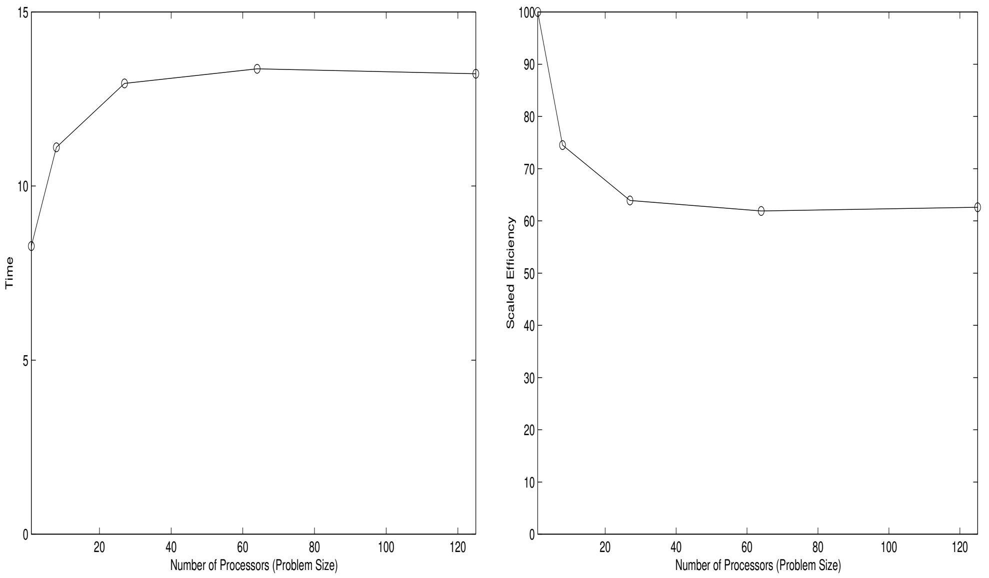
Scalability study of SMG $V(1,0)$ cycle, performed on the IBM SP at LLNL.

Each processor has an $n \times n \times n$ subgrid. The global problem is $pn \times pn \times pn$ and is solved on $p \times p \times p$ processors.

Poisson problem discretized by finite differences yielding a 7-point operator.



$20 \times 20 \times 20$ subgrids.



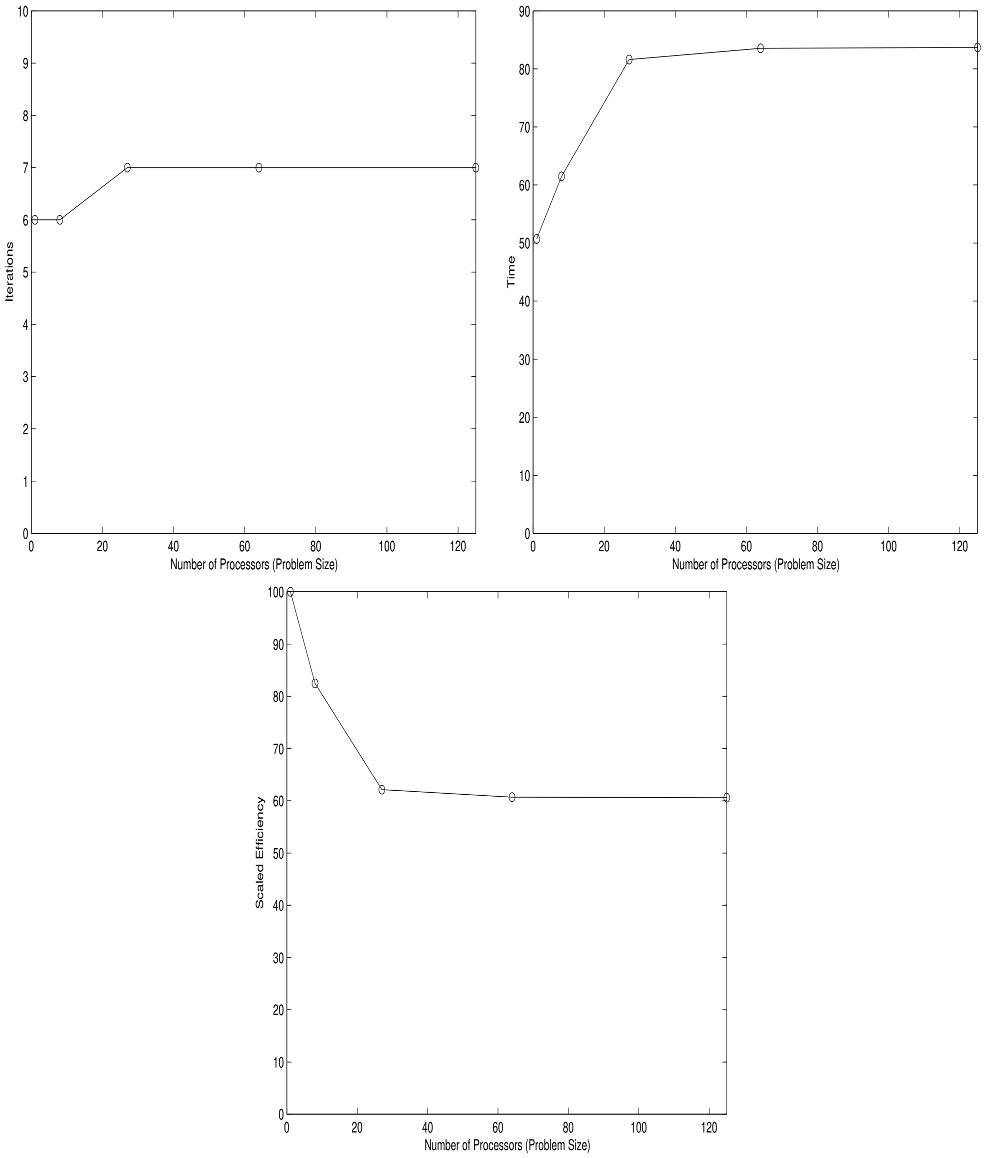
$50 \times 50 \times 50$ subgrids.

Scalability for SMG

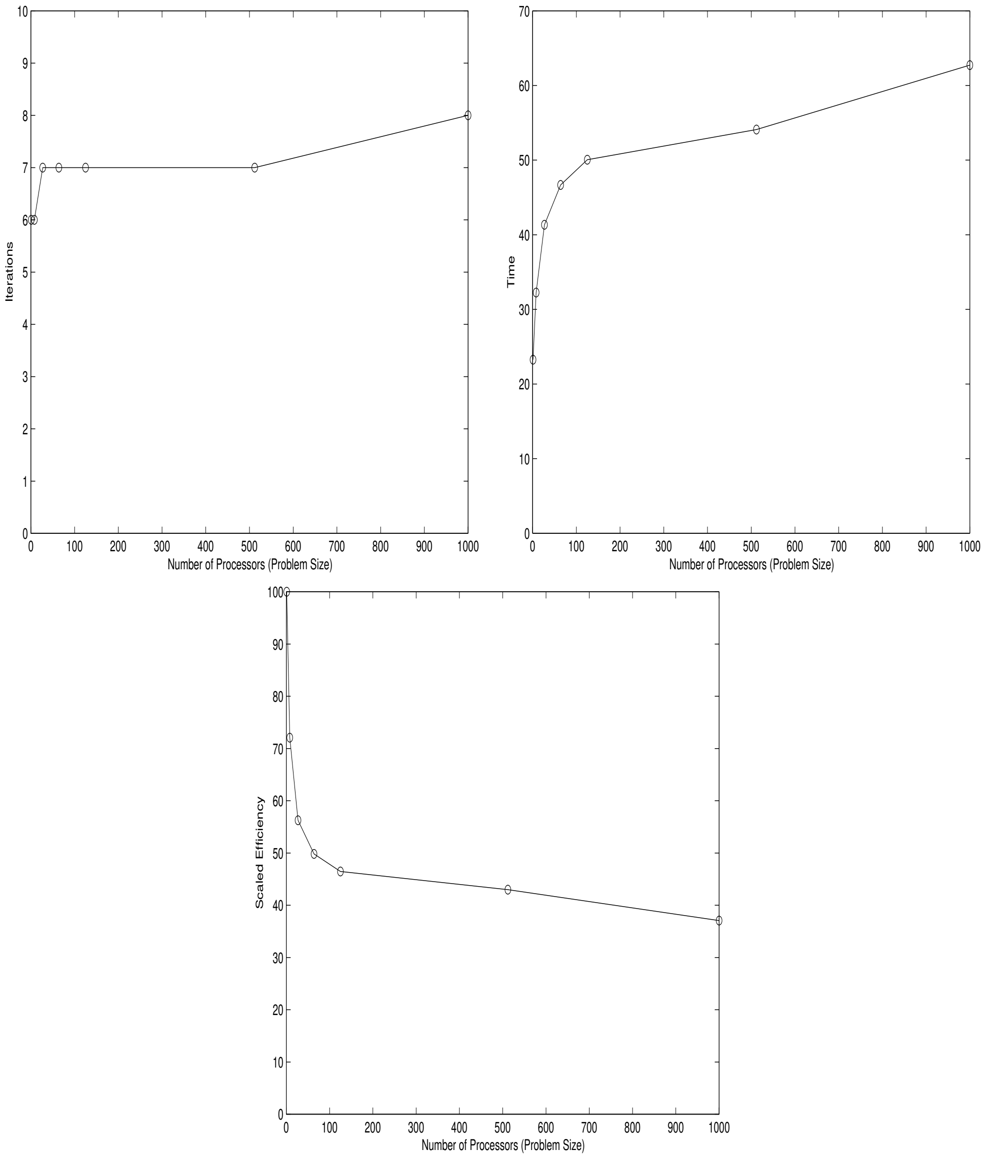
Scalability study of SMG $V(1,0)$ cycle, performed on the Intel TeraFlop machine at Sandia.

Each processor has an $n \times n \times n$ subgrid. The global problem is $pn \times pn \times pn$ and is solved on $p \times p \times p$ processors.

Anisotropic diffusion problem discretized by finite differences yielding a 7-point operator.



$50 \times 50 \times 50$ subgrids.



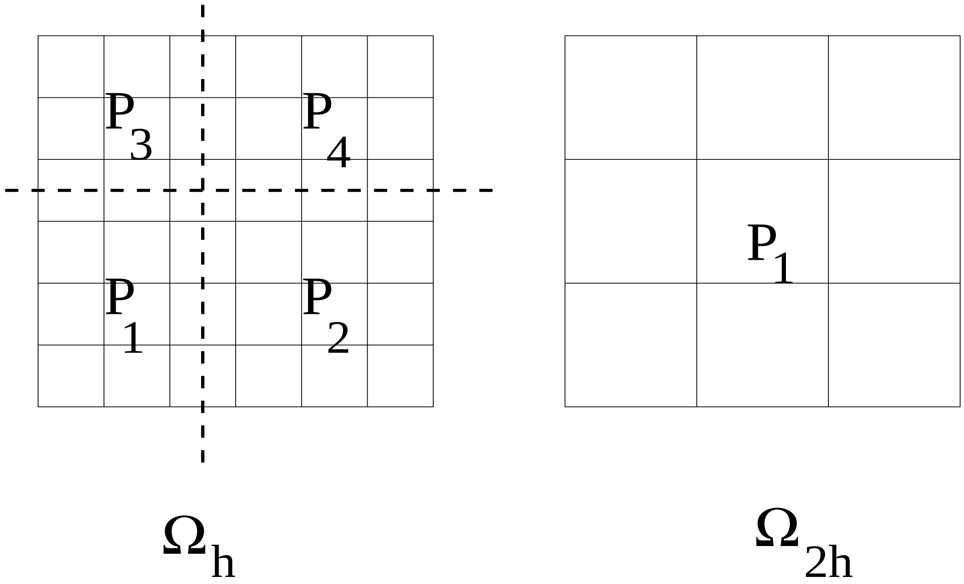
$40 \times 40 \times 40$ subgrids.

Further topics

Agglomeration of coarsest grids

For most applications $N \gg P$ on the fine grid. However, on the very coarsest grids when $N \approx P$ multigrid may not be efficient. Recall the scalability results for small N. It can be argued from the scalability model (and is often seen in practice) that this inefficiency on the coarsest grids has negligible effect on overall efficiency as most time is spent of finer grids. But this result depends on the fine grid problem size, number of processors, and the relative speeds of communication and computation.

On the coarsest grids, multigrid may be faster using fewer processors than the number used on fine grids. In agglomeration the coarse grid partitions are no longer aligned with the finer grid partitions. More communication in grid transfer, less in coarse grid solve.

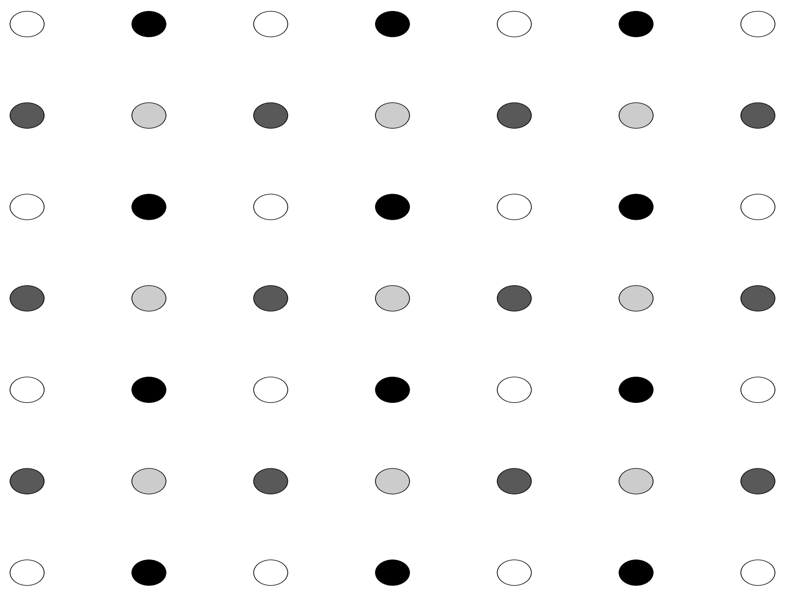


Example: Moulton, Thur. 8:00.

Further topics

Multiple coarse grids

Using multiple coarse grids produces more work on coarser levels. This means more processors can be kept busy. Perhaps more importantly, it can improve the convergence rate of the multigrid cycle.



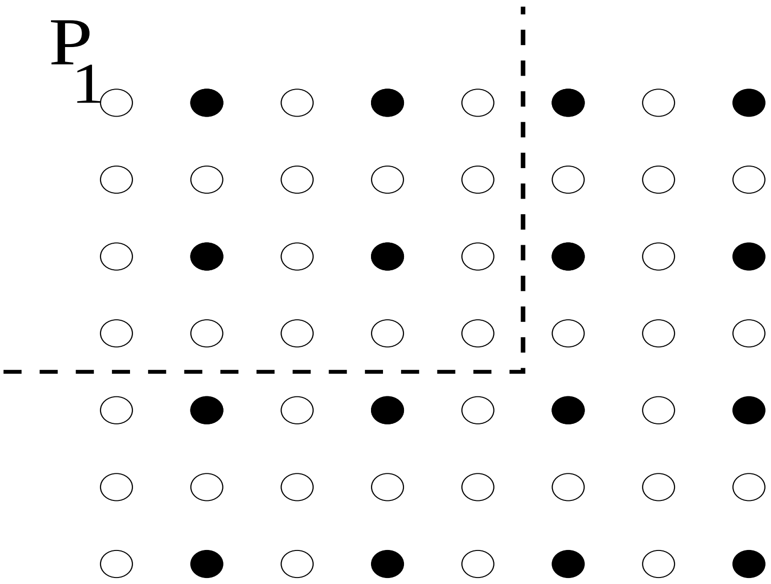
Multiple coarse grids used in Parallel Superconvergent Multigrid to create useful coarse grid work for processors that would otherwise be idle. Used in frequency decomposition multigrid to improve convergence for anisotropic problems.

Example: Frederickson, Wed 11:00.

Further topics

Extended overlap

Increasing the size of the ghost layers can reduce the number of communication steps in the v-cycle.



In Full Domain Partition method, the overlap region is increased but it is at a coarser resolution. Resulting algorithm is not equivalent to serial multigrid, but one communication per v-cycle can be enough for acceptable performance.

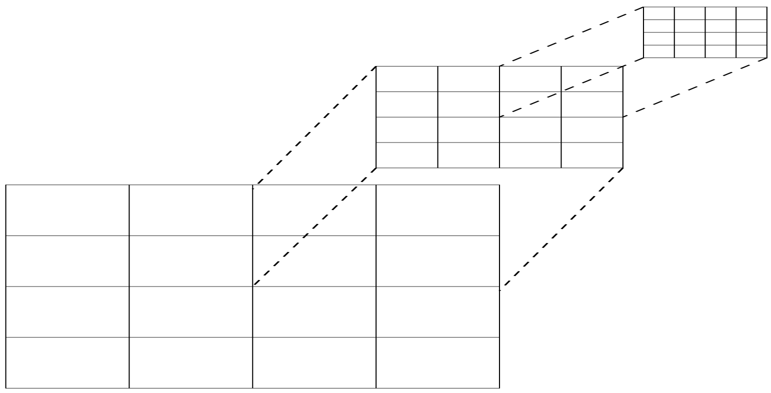
Example: Mitchell, Wed 11:50.

Further topics

Adaptive refinement and additive multigrid

The scalability model and numerical results show that domain partitioning parallelization of standard multigrid can be efficient. However, this is partly because most work is spent at fine grid levels where $N \gg P$.

With adaptive refinement, this may not be the case.



Additive variants of multigrid allow simultaneous processing on different levels. Standard (multiplicative) multigrid is twice as fast to converge (per iteration), but in some applications the additive method may result in faster run time.

Example: Zumbusch, Thur. 11:00 - Keyes, Fri. 11:25.

Further topics

- Reduce communications on coarsest grids by using fewer processors. Agglomeration of coarsest grids - Moulton, Thur. 8:00.

- Increase computational work on coarser grids by introducing additional grids. Parallel Superconvergent Multigrid - Frederickson, Wed 11:00.

- Reduce communications per v-cycle by adding larger overlap regions but at coarser resolution. Full Domain Partition - Mitchell, Wed. 11:50.

- Simultaneous processing on different levels. Additive multigrid - Zumbusch, Thur. 11:00 - Keyes, Fri. 11:25.

- Parallel multigrid with adaptive refinement - Zumbusch, Thur. 11:00 - Mitchell, Wed. 11:50.

Further topics

- Parallel multigrid for unstructured problems. Parallel Algebraic Multigrid - many!

- Parallel space–time multigrid to eliminate sequential time stepping for parabolic problems.

This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under contract number: W-7405-Eng-48.