FAS and Solver Performance

Matthew Knepley

Mathematics and Computer Science Division
Argonne National Laboratory

Fall AMS Central Section Meeting
Chicago, IL
Oct 05–06, 2007

Why should I care?

1. Optimal multilevel solvers are necessary
2. Processor flops are increasing much faster than bandwidth
3. Nonlinear algorithms can be efficient than linear algorithms
4. Presents an opportunity for numerical algebraic geometry

Payoff

Why should I care?

1. Optimal multilevel solvers are necessary
2. Processor flops are increasing much faster than bandwidth
3. Nonlinear algorithms can be efficient than linear algorithms
4. Presents an opportunity for numerical algebraic geometry

Payoff

Why should I care?

1. Optimal multilevel solvers are necessary
2. Processor flops are increasing much faster than bandwidth
3. Nonlinear algorithms can be efficient than linear algorithms
4. Presents an opportunity for numerical algebraic geometry

Payoff

Why should I care?

1. Optimal multilevel solvers are necessary
2. Processor flops are increasing much faster than bandwidth
3. Nonlinear algorithms can be efficient than linear algorithms
4. Presents an opportunity for numerical algebraic geometry

Outline

Necessity Of Simulation

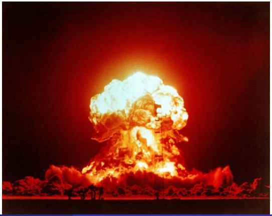
Experiment are ...

Expensive

Difficult

Impossible

Dangerous



Why Optimal Algorithms?

- The more powerful the computer, the greater the importance of optimality
- Example:
  - Suppose Alg_1 solves a problem in time CN^2, N is the input size
  - Suppose Alg_2 solves the same problem in time CN
  - Suppose Alg_1 and Alg_2 are able to use 10,000 processors
- In constant time compared to serial,
  - Alg1 can run a problem 100X larger
  - Alg2 can run a problem 10,000X larger
- Alternatively, filling the machine's memory,
  - Alg1 requires 100X time
  - Alg2 runs in constant time

Outline

What Is Optimal?

I will define *optimal* as an $\mathcal{O}(N)$ solution algorithm

These are generally hierarchical, so we need

- hierarchy generation
- assembly on subdomains
- restriction and prolongation

The Bratu Problem

$$\Delta u + \lambda e^u = f \quad \text{in} \quad \Omega \tag{1}$$

$$u = g \quad \text{on} \quad \partial\Omega \tag{2}$$

- Also called the Solid-Fuel Ignition equation
- Can be treated as a nonlinear eigenvalue problem
- Has two solution branches until $\lambda \cong 6.28$

Newton's Method

$$0 = F(u + \delta u) \cong F(u) + J(u)\delta u \tag{3}$$

so that

$$u + \delta u = u - J(u)^{-1} F(u) \tag{4}$$

- Quadratic convergence
- J can be solved approximately (Dembo-Eisensat-Steihaug)

Linear Multigrid

Smoothing (typically Gauss-Seidel)

$$x^{new} = S(x^{old}, b) \tag{5}$$

Coarse-grid Correction

$$\begin{aligned} J_c \delta x_c &= R(b - Jx^{old}) && (6) \\ x^{new} &= x^{old} + R^T \delta x_c && (7) \end{aligned}$$

Linear Convergence

Convergence to $||r|| < 10^{-9}||b||$ using GMRES(30)/ILU

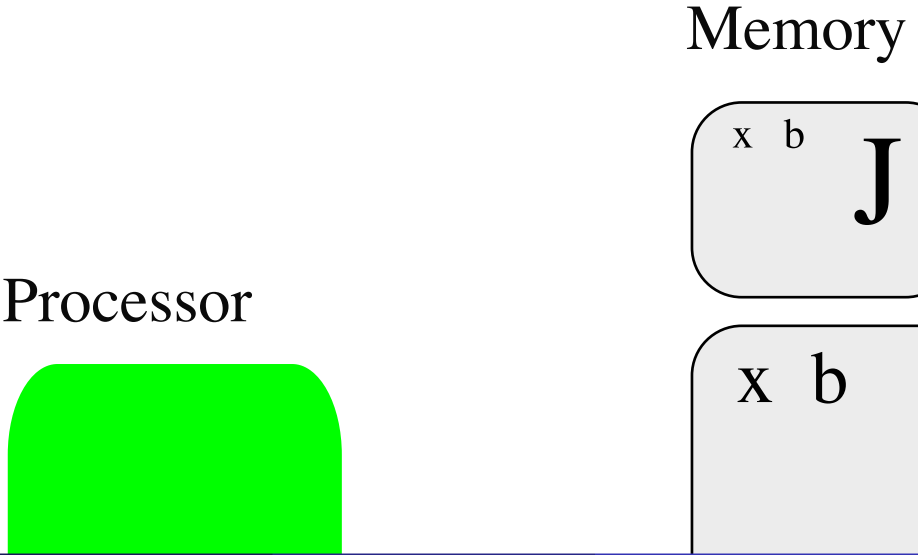
Elements	Iterations
128	10
256	17
512	24
1024	34
2048	67
4096	116
8192	167
16384	329
32768	558
65536	920
131072	1730

Linear Convergence

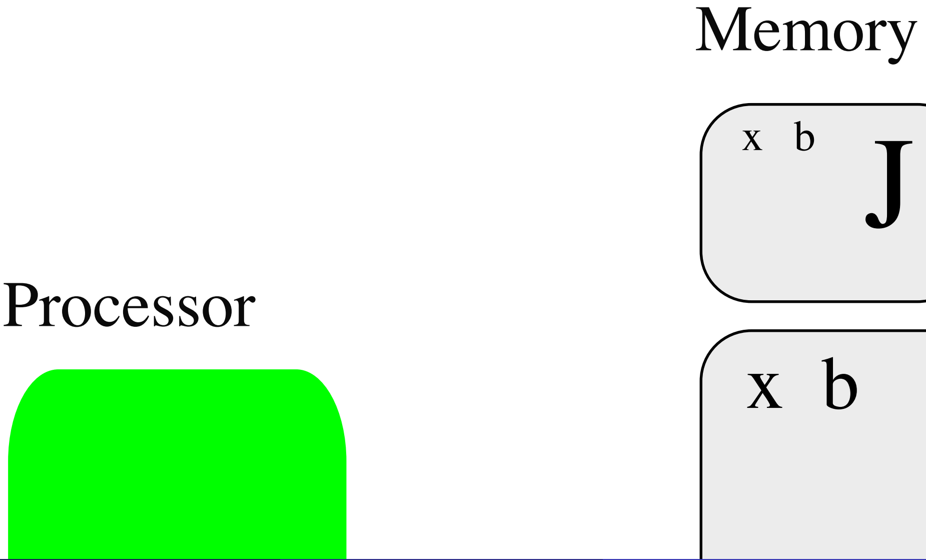
Convergence to $||r|| < 10^{-9}||b||$ using GMRES(30)/MG

Elements	Iterations
128	5
256	7
512	6
1024	7
2048	6
4096	7
8192	6
16384	7
32768	6
65536	7
131072	6

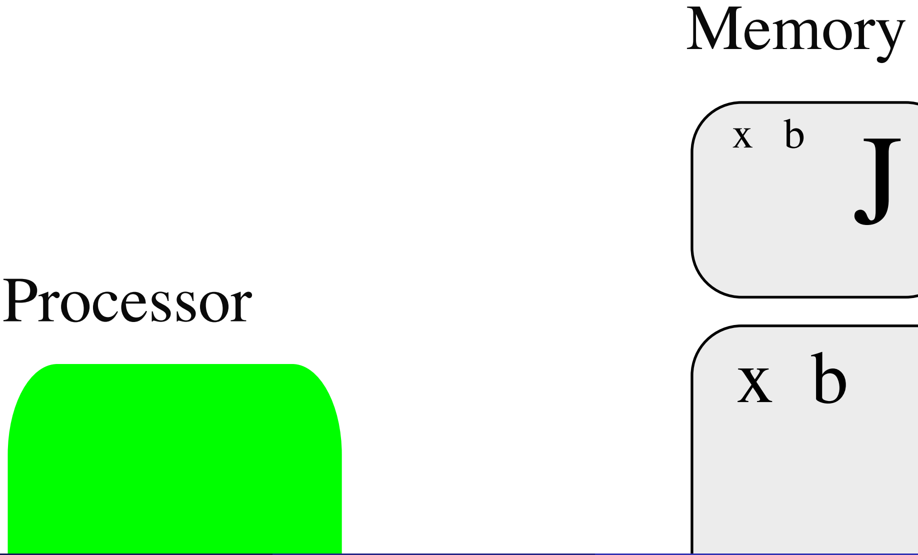
Linear Multigrid Memory Access



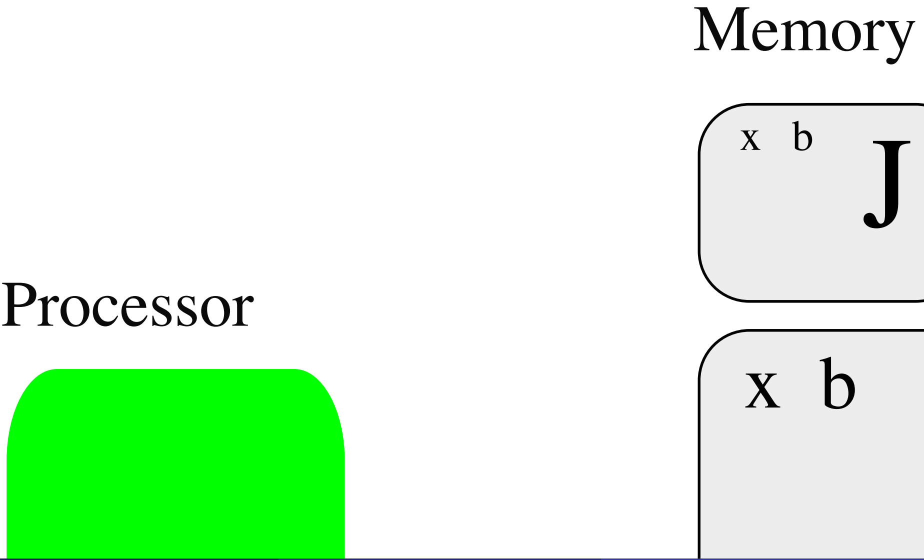
Linear Multigrid Memory Access



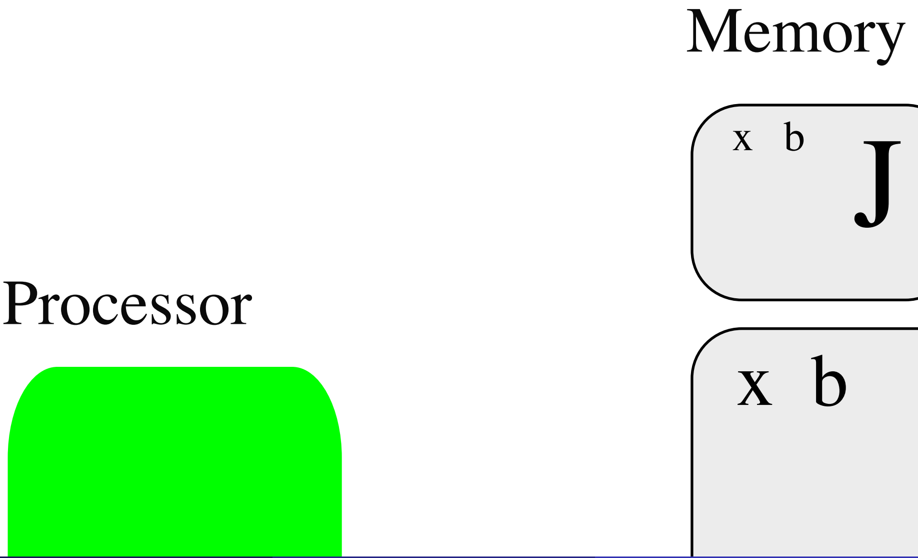
Linear Multigrid Memory Access



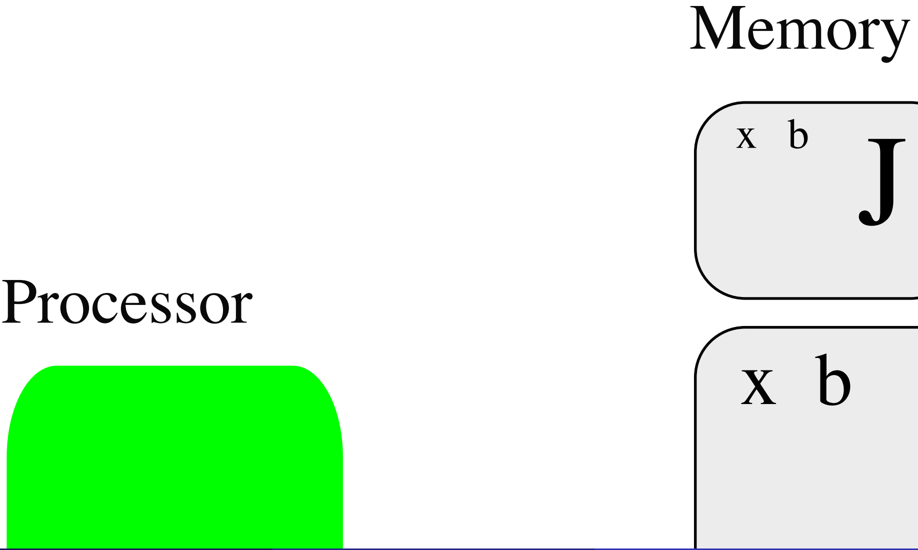
Linear Multigrid Memory Access



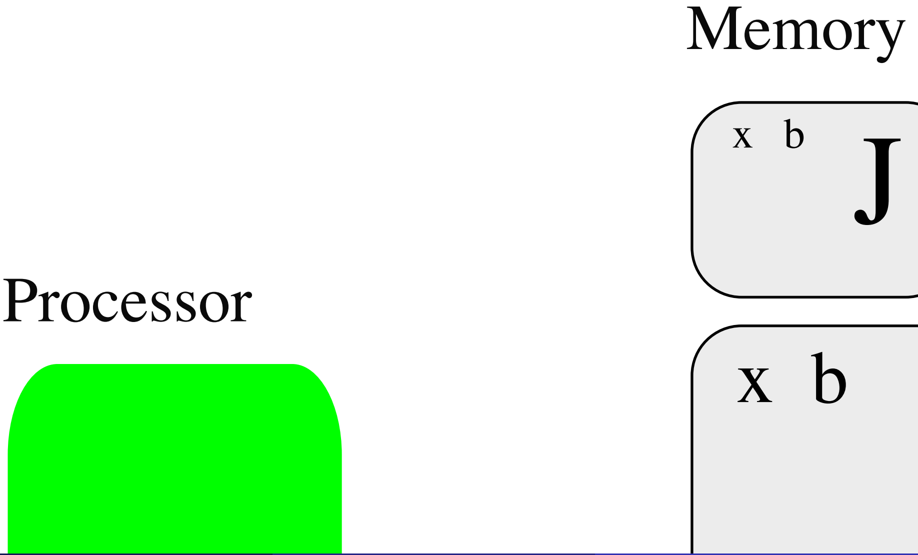
Linear Multigrid Memory Access



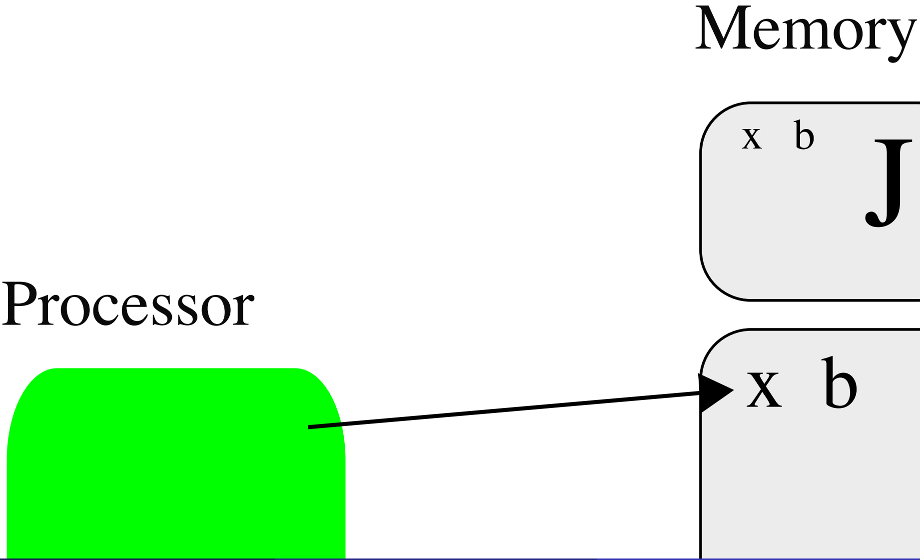
Linear Multigrid Memory Access



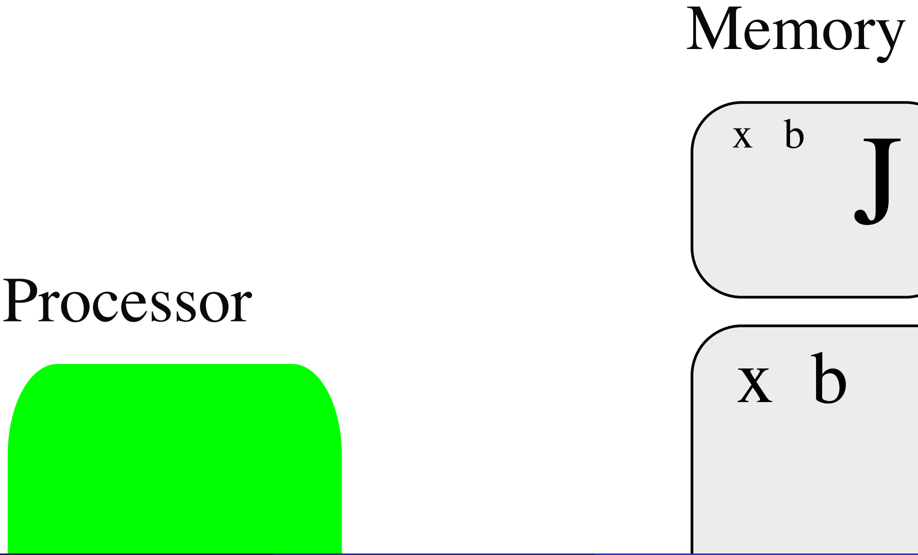
Linear Multigrid Memory Access



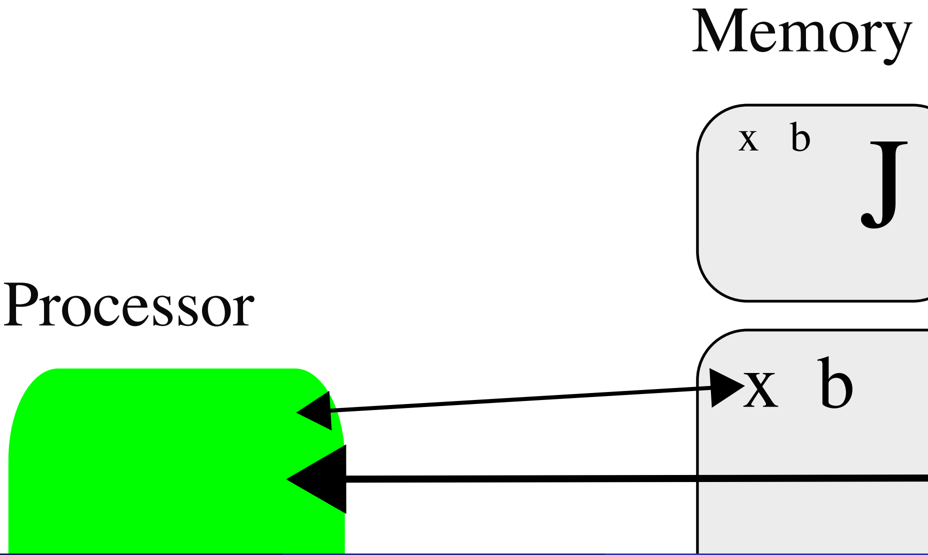
Linear Multigrid Memory Access



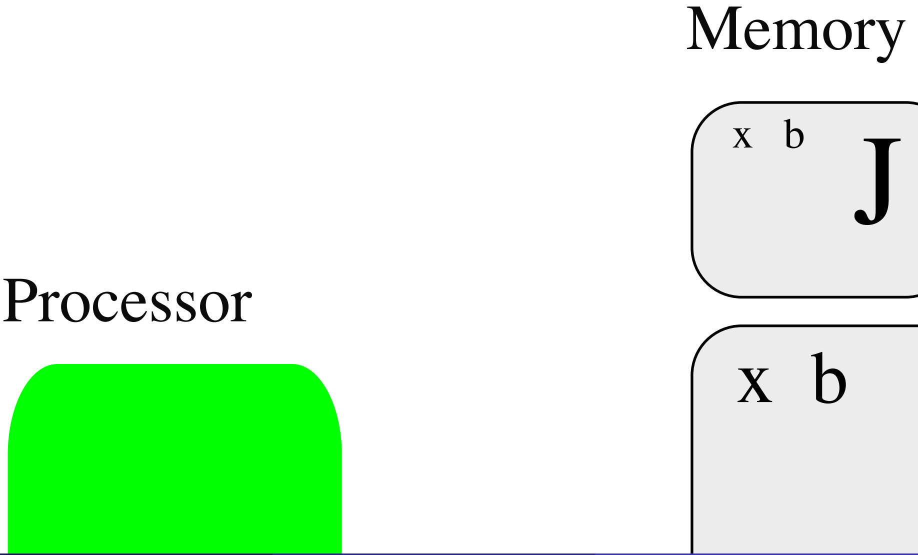
Linear Multigrid Memory Access



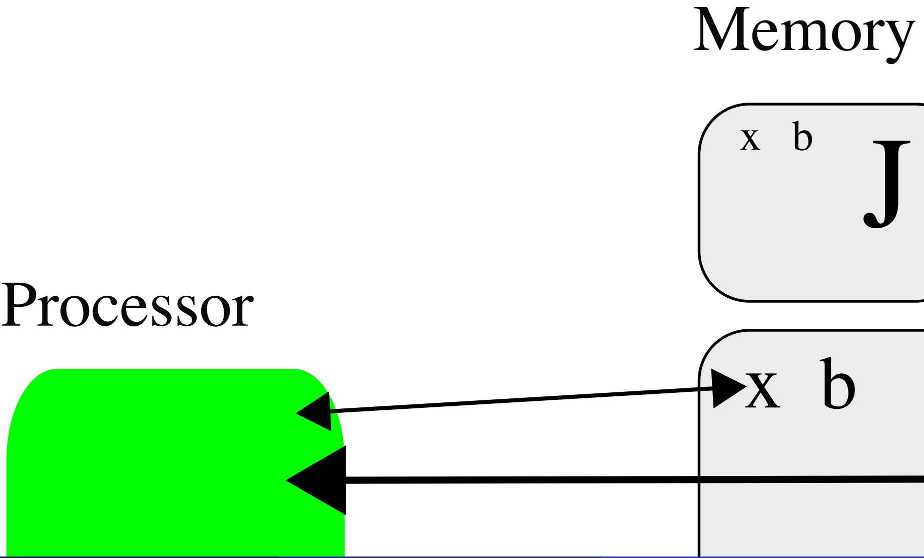
Linear Multigrid Memory Access



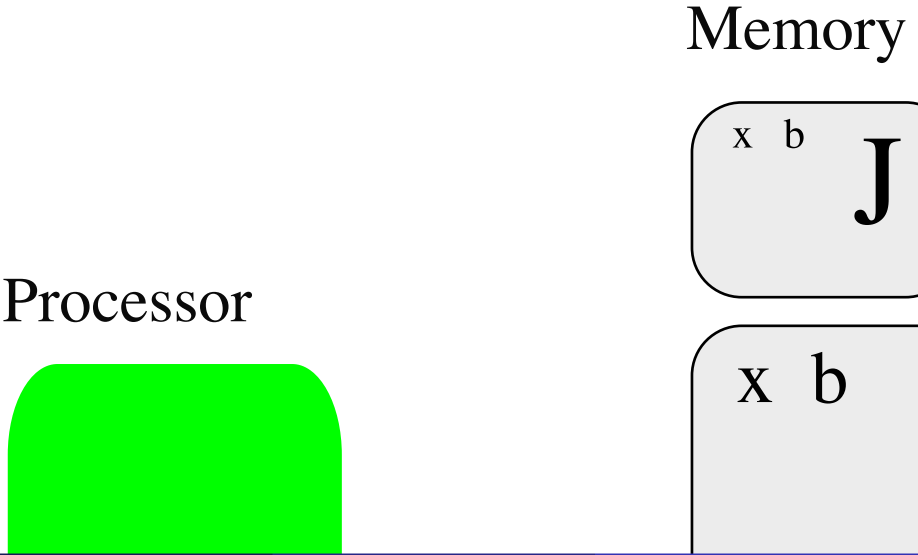
Linear Multigrid Memory Access



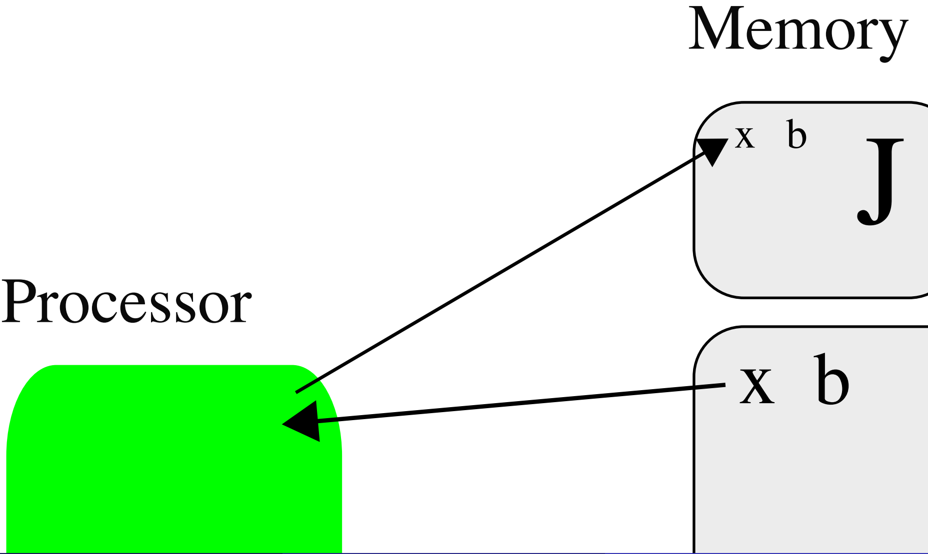
Linear Multigrid Memory Access



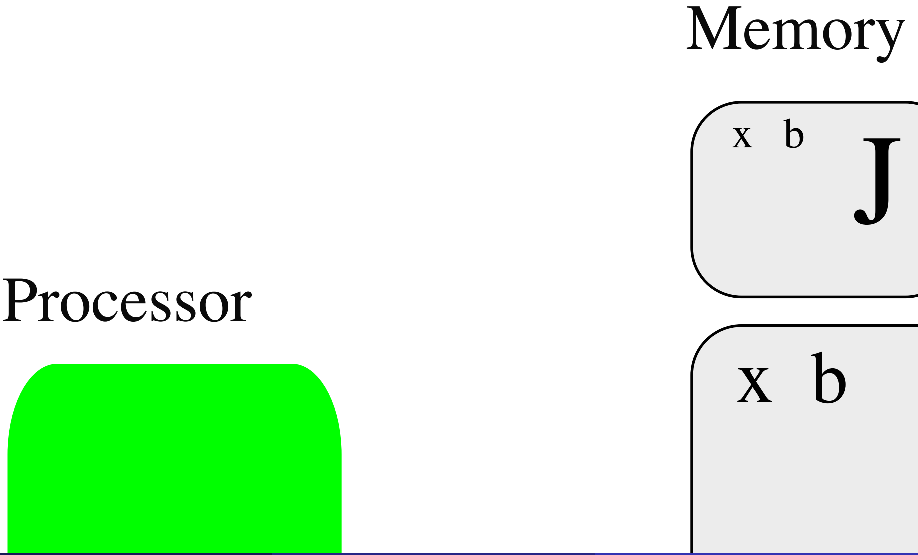
Linear Multigrid Memory Access



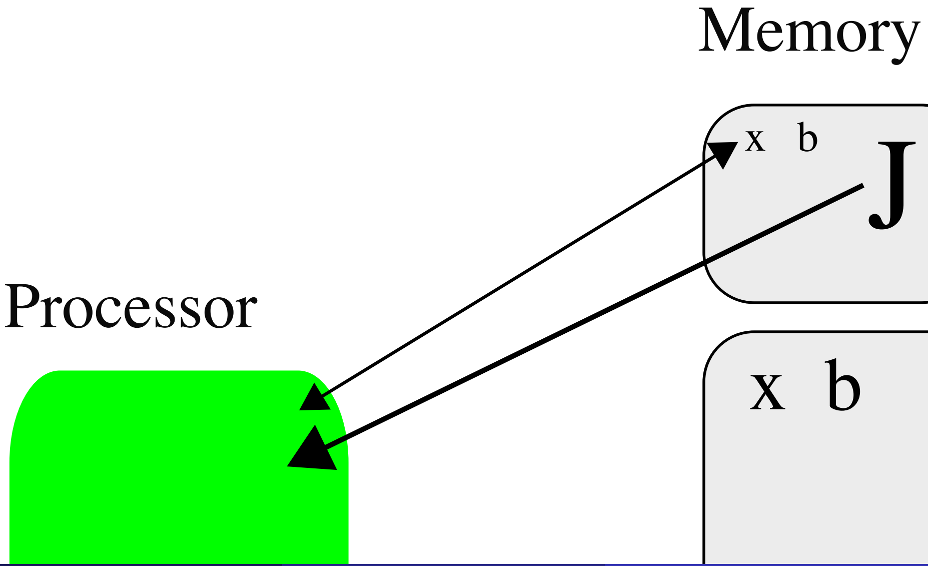
Linear Multigrid Memory Access



Linear Multigrid Memory Access



Linear Multigrid Memory Access



Outline

STREAM Benchmark

Simple benchmark program measuring sustainable memory bandwidth

- Protoypical operation is Triad (WAXPY): $\mathbf{w} = \mathbf{y} + \alpha\mathbf{x}$
- Measures the memory bandwidth bottleneck (much below peak)
- Datasets outstrip cache

Machine	Peak (MF/s)	Triad (MB/s)	MF/MW	Eq. MF/s
Matt's Laptop	1700	1122.4	12.1	93.5 (5.5%)
Intel Core2 Quad	38400	5312.0	57.8	442.7 (1.2%)
Tesla 1060C	984000	102000.0*	77.2	8500.0 (0.8%)

Table: Bandwidth limited machine performance

http://www.cs.virginia.edu/stream/

Analysis of Sparse Matvec (SpMV)

Assumptions

- No cache misses
- No waits on memory references

Notation

m Number of matrix rows

nz Number of nonzero matrix elements

V Number of vectors to multiply

We can look at bandwidth needed for peak performance

$$\left(8 + \frac{2}{V}\right)\frac{m}{nz} + \frac{6}{V}\ \mathrm{byte/flop} \tag{8}$$

or achieveable performance given a bandwith BW

$$\frac{Vnz}{(8V+2)m + 6nz} BW\ \mathrm{Mflop/s} \tag{9}$$

Towards Realistic Performance Bounds for Implicit CFD Codes, Gropp, Kaushik, Keyes, and Smith.

Improving Serial Performance

For a single matvec with 3D FD Poisson, Matt's laptop can achieve at most

$$\frac{1}{(8+2)\,\frac{1}{7}+6}\ \text{bytes/flop}(1122.4\ \text{MB/s}) = 151\ \text{MFlops/s}, \qquad (10)$$

which is a dismal 8.8% of peak.

Can improve performance by

- Blocking
- Multiple vectors

but operation issue limitations take over.

Improving Serial Performance

For a single matvec with 3D FD Poisson, Matt's laptop can achieve at most

$$\frac{1}{(8+2)\frac{1}{7}+6}\ \text{bytes/flop}(1122.4\ \text{MB/s}) = 151\ \text{MFlops/s}, \tag{10}$$

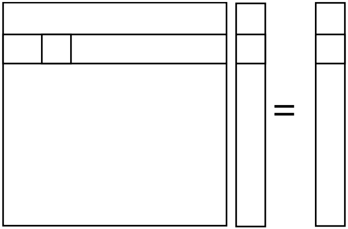
which is a dismal 8.8% of peak.

Better approaches:

- Unassembled operator application (Spectral elements, FMM)
  - N data, N^2 computation
- Nonlinear evaluation (Picard, FAS, Exact Polynomial Solvers)
  - N data, N^k computation

Outline

Matrix-Free Smoothing



We can use point Jacobi

$$x_i^{new} = x_i^{old} + J_{ii}^{-1}(b_i - J_i^T x^{old}) \tag{11}$$

In the nonlinear case,

$$J_i^T x^{old} = e_i^T \nabla F(x) x^{old} \tag{12}$$

which might be calculated automatically using AD.

Nonlinear Gauss-Seidel

If we have an initial guess u, $b = 0 - F(u)$,

$$\begin{aligned}
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(b_i - J_i^T x^{old}) && (13)\\
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(-F_i(u) - \nabla F_i(u)^T x^{old}) && (14)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1}(F_i(u) + \nabla F_i(u)^T x^{old}) && (15)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1} F_i(u + x^{old}) && (16)\\
u_i^{new} &= u_i^{old} - J_{ii}^{-1} F_i(u^{old}) && (17)
\end{aligned}$$

This is just Newton's method on a single equation at a time …

which is Nonlinear Gauss-Seidel.

Nonlinear Gauss-Seidel

If we have an initial guess u, $b = 0 - F(u)$,

$$
\begin{aligned}
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(b_i - J_i^T x^{old}) && (13)\\
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(-F_i(u) - \nabla F_i(u)^T x^{old}) && (14)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1}(F_i(u) + \nabla F_i(u)^T x^{old}) && (15)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1} F_i(u + x^{old}) && (16)\\
u_i^{new} &= u_i^{old} - J_{ii}^{-1} F_i(u^{old}) && (17)
\end{aligned}
$$

This is just Newton's method on a single equation at a time . . .

which is Nonlinear Gauss-Seidel.

Nonlinear Gauss-Seidel

If we have an initial guess u, $b = 0 - F(u)$,

$$
\begin{aligned}
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(b_i - J_i^T x^{old}) && (13)\\
x_i^{new} &= x_i^{old} + J_{ii}^{-1}(-F_i(u) - \nabla F_i(u)^T x^{old}) && (14)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1}(F_i(u) + \nabla F_i(u)^T x^{old}) && (15)\\
x_i^{new} &= x_i^{old} - J_{ii}^{-1} F_i(u + x^{old}) && (16)\\
u_i^{new} &= u_i^{old} - J_{ii}^{-1} F_i(u^{old}) && (17)
\end{aligned}
$$

This is just Newton's method on a single equation at a time . . .

which is Nonlinear Gauss-Seidel.

Nonlinear Multigrid

Most authors just offer an ansatz with nonlinear smoothing

$$x^{new} = S(x^{old}, b) \tag{18}$$

and coarse-grid correction

$$\begin{aligned} F_c(x_c) &= F_c(\tilde{x}_c) + \gamma R(b - F(x^{old})) && (19) \\ x^{new} &= x^{old} + \frac{1}{\gamma} R^T (x_c - \tilde{x}_c) && (20) \end{aligned}$$

where $\tilde{x}$ is an approximate solution.

If F is a linear operator L, the correction reduces to

$$\begin{aligned} L_c(x_c) &= L_c(\tilde{x}_c) + \gamma R(b - L(x^{old})) && (21) \\ L_c(x_c - \tilde{x}_c) &= \gamma R(b - L(x^{old})) && (22) \\ L_c \delta x_c &= \gamma R r && (23) \end{aligned}$$

Nonlinear Multigrid

Most authors just offer an ansatz with nonlinear smoothing

$$x^{new} = S(x^{old}, b) \tag{18}$$

and coarse-grid correction

$$\begin{aligned} F_c(x_c) &= F_c(\tilde{x}_c) + \gamma R(b - F(x^{old})) && (19) \\ x^{new} &= x^{old} + \frac{1}{\gamma} R^T (x_c - \tilde{x}_c) && (20) \end{aligned}$$

where $\tilde{x}$ is an approximate solution.

and the update becomes

$$\begin{aligned} x^{new} &= x^{old} + \frac{1}{\gamma} R^T \delta x_c && (21) \\ x^{new} &= x^{old} + R^T \hat{L}_c^{-1} R r && (22) \end{aligned}$$

Nonlinear Multigrid

It is instructive to look at the alternate derivation of Barry Smith

Begin with the nonlinear generalization $F(u) = 0$, for a correction

$$\begin{aligned} J_c x_c &= R(b - Jx^{old}) && (23) \\ J_c x_c &= -R(F(u) + Jx^{old}) && (24) \end{aligned}$$

and then using Taylor series

$$\begin{aligned} F(u^{old}) &= F(u) + J(u^{old} - u) + \dots && (25) \\ F_c(u_c^{old} + x_c) &= F_c(u_c^{old}) + J_c x_c + \dots && (26) \end{aligned}$$

we have the correction

$$\begin{aligned} F_c(u_c^{old} + x_c) - F_c(u_c^{old}) &= -RF(u^{old}) && (27) \\ F_c(u_c^{old} + x_c) &= F_c(u_c^{old}) - RF(u^{old}) && (28) \end{aligned}$$

Nonlinear Multigrid

It is instructive to look at the alternate derivation of Barry Smith

Begin with the nonlinear generalization $F(u) = 0$, for a correction

$$\begin{aligned} J_c x_c &= R(b - J x^{old}) && (23) \\ J_c x_c &= -R(F(u) + J x^{old}) && (24) \end{aligned}$$

and then using Taylor series

$$\begin{aligned} F(u^{old}) &= F(u) + J(u^{old} - u) + \dots && (25) \\ F_c(u_c^{old} + x_c) &= F_c(u_c^{old}) + J_c x_c + \dots && (26) \end{aligned}$$

and the same update

$$x^{new} = x^{old} + R^T x_c \tag{27}$$

Spectrum of Methods

Newton-Multigrid FAS

- When does linearization happen?
- Which Jacobian entries are updated?

Nonlinear Convergence

Convergence to $||r|| < 10^{-9}||r_0||$ using Newton/GMRES(30)/ILU

Elements	Iterations
32	4
64	4
128	4
256	4
512	4
1024	4
2048	4
4096	4
8192	4
16384	4
32768	4
65536	4
131072	4

Nonlinear Convergence

Outline

Polynomial Solvers

A great opportunity exists for polynomial solvers

- Better performance
  - Bandwidth considerations only intensify on multicore chips
  - Petascale systems will need these improvements
- More robust
  - Most practical engineering calculations are quadratic
- New algorithms
  - Can multiple solutions speed up convergence?

Conclusions

Newton-Multigrid provides

- Good nonlinear solves
- Simple interface for software libraries
- Low computational efficiency

Multigrid-FAS provides

- Good nonlinear solves
- Lower memory bandwidth and storage
- Potentially high computational efficiency
- Requires formation on small systems “on the fly”

PETSc Resources

- http://www.mcs.anl.gov/petsc
- Can download tarballs or clone a Mercurial repository
- Hyperlinked documentation
  - Manual
  - Manual pages for evey method
  - HTML of all example code (linked to manual pages)
- FAQ
- Full support at petsc-maint@mcs.anl.gov