Parallel iterative solvers for 2D Poisson equation on shared-memory architectures

Luigi Di Sotto

Università di Pisa
Dipartimento di Informatica

27 July 2016

The continuous problem

The 2-D Poisson equation

Let $f(x,y)$ be a given function defined on the rectangle

$$\Omega = \{(x,y) \in \mathbb{R}^2 : \alpha_x \leq x \leq \beta_x,\ \alpha_y \leq y \leq \beta_y\}$$

we wish to find a function u that satisfies

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f(x,y),\ \ (x,y) \in \Omega$$

From continuous to discrete space

Finite difference method

Our plan is to approximate u at the grid points

$$(\alpha_x + ih_x, \alpha_y + jh_y)$$

with

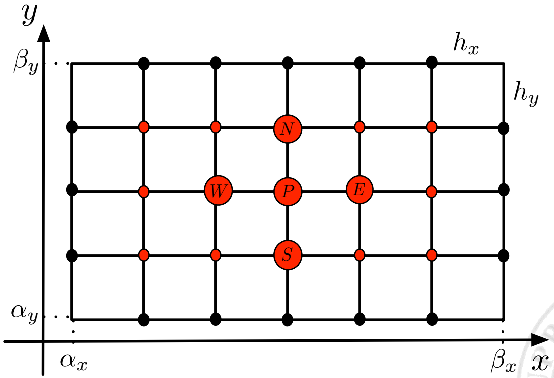
$$i = 1, \cdots, n-2,\ j = 1, \cdots, m-2$$

and

$$h_x = \frac{\beta_x - \alpha_x}{n},\ h_y = \frac{\beta_y - \alpha_y}{m}$$

From continuous to discrete space

The correspondent grid has the form



with top-down natural ordering of nodes, where red nodes and black nodes are the internal and external points respectively.

From continuous to discrete space

Finite difference method (contd.)

An interior grid point P has a north (N), east (E), south (S), and west (W) neighbour. Using this "compass point" notation we obtain the following approximation to Poisson equation at point P

$$\frac{\frac{u(E)-u(P)}{h_x} - \frac{u(P)-u(W)}{h_x}}{h_x} + \frac{\frac{u(N)-u(P)}{h_y} - \frac{u(P)-u(S)}{h_y}}{h_y} = F(P)$$

Discretizing Poisson equation

Finite difference methods (contd.)

The linear equation at point P has the form

$$u(P) = \frac{1}{\alpha}\left(\gamma\beta F(P) + (u(E) + u(W))\,\beta + (u(N) + u(S))\,\gamma\right)$$

where

$$\alpha = 2\left(h_x^2 + h_y^2\right)$$

$$\gamma = h_x^2,\ \beta = h_y^2$$

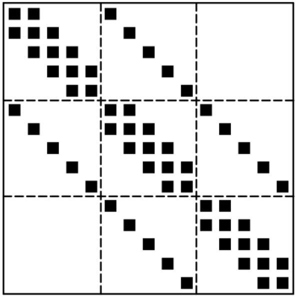
Poisson system

The linear system

Giving rise to a system of $n \times m$ linear equations in $n \times m$ unknowns

$$\mathbf{Au} = \frac{\gamma\beta}{\alpha}\mathbf{f}$$

The pattern of matrix A of linear equations appers to be



Poisson system

The linear system

As an example, the block structure of matrix A is the following

$$A = \begin{pmatrix} B & -\gamma I & \\ -\gamma I & B & -\gamma I \\ & -\gamma I & B \end{pmatrix}$$

with

$$B = \begin{pmatrix} \alpha & -\beta & & \\ -\beta & \alpha & -\beta & \\ & -\beta & \alpha & -\beta \\ & & -\beta & \alpha \end{pmatrix}$$

Numerical solving of Poisson systems: intro to iterative methods

The problem

As we have seen, we wish to solve numerically

$$\mathbf{Au} = \frac{\beta\gamma}{\alpha}\mathbf{f}$$

to this aim we could choose from a variety of approaches. We could use, for example, direct methods (as Gaussian elimination), but knowing the pattern exhibited by matrix A such methods they could introduce the **fill-in phenomenon**. Conversely, with iterative methods we could exploit the particular structure of A.

Numerical solving of Poisson systems: intro to iterative methods

Definition

A vector series $\{u^{(k)}\} \in \mathbb{R}^n$ is convergent to a vector $u^* \in \mathbb{R}^n$ if exists a vector norm such that

$$\lim_{k\to+\infty} ||u^{(k)} - u^*|| = 0$$

namely

$$\lim_{k\to+\infty} u^{(k)} = u^*$$

Numerical solving of Poisson systems: intro iterative methods

Theorem

Let $A \in \mathbb{R}^{nxn}$, then

$$\lim_{k \to +\infty} A^k = O \text{ iff } \rho(A) < 1$$

Numerical solving of Poisson systems: intro to iterative methods

Some generalities

Let A be a square non-singular matrix and consider its decomposition

$$A = M - N$$

where M is a non-singular matrix. Substituting it in a system of linear equations

$$Au = f$$

we get

$$Mu - Nu = f$$

namely

$$u = M^{-1}Nu + M^{-1}f$$

Numerical solving of Poisson systems: intro to iterative methods

Some generalities (contd.)

Now let $P = M^{-1}N$ and $q = M^{-1}f$, so we have

$$\mathbf{u} = \mathbf{P}\mathbf{u} + \mathbf{q}$$

equivalent to the given system of linear equations. Such a relation gives rise to an **iterative method** that, starting from an initial vector $u^{(0)}$, solution is approximated by means of a vector series $\{u^{(k)}\}$. We say that P is the **iteration matrix**.

Numerical solving of Poisson systems: intro to iterative methods

Theorem

The iterative method

$$x^{(k)} = Px^{(k-1)} + q$$

is convergent iff $\rho(P) < 1$

Numerical solving of Poisson systems: intro to iterative methods

Proof

Let x^* be the solution vector satisfying

$$x^* = Px^* + q$$

from wich we subtract $x^{(k)} = Px^{(k-1)} + q$, obtaining

$$x^* - x^{(k)} = P\left(x^* - x^{(k-1)}\right), k = 1, 2, \cdots$$

Let

$$e^{(k)} = x^* - x^{(k)}, k = 1, 2, \cdots$$

be the **error vector** at the k-th iteration

Numerical solving of Poisson systems: intro to iterative methods

Proof (contd.)

So we have

$$e^{(k)} = Pe^{(k-1)}, k = 1, 2, \cdots$$

and thus

$$e^{(k)} = Pe^{(k-1)} = P^2 e^{(k-2)} = \cdots = P^k e^{(0)}$$

If $\rho(P) < 1$, by the Theorem seen before, we have

$$\lim_{k \to +\infty} P^k = O$$

it follows that

$$\lim_{k \to +\infty} e^{(k)} = 0$$

Numerical solving of Poisson systems: intro to iterative methods

Proof (contd.)

Viceversa, if method converges, we have that the last limit holds for every $x^{(0)}$, in particular holds if $x^{(0)}$ is such that vector $e^{(0)} = x^* - x^{(0)}$ is eigenvector of P relative to an eigenvalue of maximum modulus, namely $|\lambda| = \rho(P)$. In such case results that

$$Pe^{(0)} = \lambda e^{(0)}$$

and thus

$$e^{(k)} = P^k e^{(0)} = \lambda^k e^{(0)}$$

Numerical solving of Poisson systems: intro to iterative methods

Proof (contd.)

Follows that

$$\lim_{k \to +\infty} [\rho(P)]^k = 0$$

and then

$$\rho(P) < 1$$

Numerical solving of Poisson systems: intro to iterative methods

The previous necessary and sufficient condition for convergence is not easy to test. So, whenever is possibile, we could use a sufficient condition that is more easy to prove. So we claim that

Theorem

If exists an induced matrix norm $||.||$ for which $||P|| < 1$, the iterative method

$$x^{(k)} = Px^{(k-1)} + q$$

is convergent. (Proof follows from theorem seen above and by property $\rho(P) \leq ||P||$)

Numerical solving of Poisson systems: intro to iterative methods

Checking convergence

Choosing a vector norm $||.||$ and the corresponding induced matrix norm, we have

$$||e^{(k)}|| \leq ||P^k|| \cdot ||e^{(0)}||$$

So $||P^k||$ expresses the reduction, with respect to the initial error, of the error at the k-th step. But such a measure is not very useful for evaluating speed convergence of the considered method.

Numerical solving of Poisson systems: intro to iterative methods

Checking convergence (contd.)

If $e^{(k-1)} \neq 0$, quantity given by

$$\frac{||e^{(k)}||}{||e^{(k-1)}||}$$

expresses error reduction at k-th iteration and the geometric mean of error reduction in the first k iterations

$$\sigma_k = \sqrt[k]{\frac{||e^{(1)}||}{||e^{(0)}||}\frac{||e^{(2)}||}{||e^{(1)}||}\cdots\frac{||e^{(k)}||}{||e^{(k-1)}||}} = \sqrt[k]{\frac{||e^{(k)}||}{||e^{(0)}||}}$$

expresses the average error reduction.

Numerical solving of Poisson systems: intro to iterative methods

Checking convergence (contd.)

It follows that

$$\sigma_k \leq \sqrt[k]{||P^k||}$$

Numerical solving of Poisson systems: intro to iterative methods

Theorem

Let $A \in R^{n \times n}$ and let $||.||$ be a generic induced norm. Thus,

$$\lim_{k \to +\infty} \sqrt[k]{||A^k||} = \rho(A)$$

Observation

Quantity $\rho(P)$, norm and iteration index independent, is supposed to be the measure of convergence speed of an iterative method.

Numerical solving of Poisson systems: intro to iterative methods

Definition

We define the constant

$$R = -\log_{10} \rho(P)$$

as the **asymptotic rate of convergence** of an iterative method: the smaller is $\rho(P)$, the higher is the rate of the convergence, i.e. the greater is the number of correct decimal places computed per iteration.

Example

$$[\rho(P)]^k \approx \frac{1}{10} \implies k \approx -\frac{1}{\log_{10} \rho(P)}$$

Numerical solving of Poisson systems: intro to iterative methods

Stopping criteria

Fixed a tolerance ϵ, the two most common stopping criteria are

$$||x^{(k)} - x^{(k-1)}|| \leq \epsilon$$

$$\frac{||x^{(k)} - x^{(k-1)}||}{||x^{(k)}||} \leq \epsilon$$

Numerical solving of Poisson systems: intro to iterative methods

Stopping criteria (contd.)

Note that such conditions do not guarantee that approximated solution has precision ϵ. In fact we have

$$x^{(k)} - x^{(k-1)} = \left[x^* - x^{(k-1)}\right] - \left[x^* - x^{(k)}\right] = e^{(k-1)} - e^{(k)}$$

$$= (I - P)\, e^{(k-1)}$$

Numerical solving of Poisson systems: intro to iterative methods

Theorem

Let $||.||$ be an induced matrix norm and let $A \in \mathbb{R}^{n\times n}$ be such that $||A|| < 1$. Then matrix given by $I + A$ is non singular and the following holds

$$||\,(I+A)^{-1}\,|| \leq \frac{1}{1-||A||}$$

Stopping criteria (contd.)

By such theorem, we can state that

$$||e^{(k-1)}|| \leq ||\,(I-P)^{-1}\,|| \cdot ||x^{(k)} - x^{(k-1)}|| \leq \frac{||x^{(k)} - x^{(k-1)}||}{1-||P||}$$

Numerical solving of Poisson systems: intro to iterative methods

Stopping criteria (contd.)

So it can happen that even if stopping criteria conditions are satisfied, quantity $||e^{(k-1)}||$ is high.

Numerical solving of Poisson systems: intro to iterative methods

Stopping criteria: observation

In fact, in implementing an iterative method, we must check when we reach too many iterations due to the fact seen above. Is also possibile that if $\rho(P) < 1$, as a consequence of round-off errors, iterative methods are not convergent, especially if matrix A is ill-conditioned and $\rho(P) \approx 1$.

Numerical solving of Poisson systems: intro to iterative methods

Observation

Iterative methods w.r.t direct methods are less sensitive to error propagation. In fact, $x^{(k)}$ can be seen as the vector generated by only one iteration starting with initial guess $x^{(k-1)}$, and so it is affected by round-off errors generated by the last iteration.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

Derivation

Among the methods we can derive by a particular decomposition of matrix A, we can have the **Jacobi** and **Gauss-Seidel** methods, for wich is possible to prove sufficient conditions for convergence that most matrices arising from differential problems satisfy.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

Derivation (contd.)

Let's consider decomposition of A as follows

$$A = D - B - C$$

where

$$d_{ij} = \begin{cases} a_{ij}, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

$$b_{ij} = \begin{cases} -a_{ij}, & \text{if } i > j \\ 0 & \text{if } i \leq j \end{cases}$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

Derivation (contd.)

$$c_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -a_{ij}, & \text{if } i < j \end{cases}$$

By choosing $M = D$, $N = B + C$, we obtain the **Jacobi method**, and by choosing $M = D - B$, $N = C$, we have the **Gauss-Seidel method**. Fo such decompositions we have that $\det(M) \neq 0$ iff all diagonal elements of A are non-zero.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method

Let J be the iteration matrix of Jacobi method. By what we have seen before, J is as follows

$$J = D^{-1}(B + C)$$

Then

$$x^{(k)} = Jx^{(k-1)} + D^{-1}b$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices

Recalling our 2-dimensional Poisson system,

$$Au = \frac{\beta\gamma}{\alpha} f$$

the iteration matrix J has the following structure

$$J = \frac{1}{\alpha} \begin{pmatrix} H & \gamma I & & \\ \gamma I & \ddots & \ddots & \\ & \ddots & \ddots & \gamma I \\ & & \gamma I & H \end{pmatrix}$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices (contd.)

where

$$H = \beta \begin{pmatrix} 0 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 0 \end{pmatrix} \in \mathbb{R}^{p \times p}$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices (contd.)

Since iteration matrix can be expressed as

$$J = \alpha^{-1}\left(\beta^{-1}H \otimes \gamma I + I \otimes H\right)$$

Eigenvalues of J are

$$\lambda_{ij} = \alpha^{-1}\left(\beta^{-1}\lambda_i\gamma + \lambda_j\right),\ i,j = 1,\cdots,p$$

noting that λ_i, λ_j are eigenvalues of H, and they have the following form

$$\lambda_k = 2\beta\cos\left(\frac{k\pi}{p+1}\right),\ k = 1,\cdots,p$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices (contd.)

and so

$$\rho(J) = \frac{2}{\alpha}(\gamma + \beta)\cos\left(\frac{\pi}{p+1}\right) = \cos\left(\frac{\pi}{p+1}\right)$$

by Taylor series we have

$$\rho(J) \approx \left(1 - \frac{\pi^2}{2(p+1)^2}\right)$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices (contd.)

We require that

$$\rho(J) < 1$$

but unfortunately as *n* approaches infinity we have

$$\rho(J) \approx 1$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Jacobi method for structured matrices (contd.)

More in depth, the Jacobi method updates components with such rule

$$u_{ij}^{(k)} = \frac{1}{\alpha}\left[\gamma\left(u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)}\right) + \beta\left(u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)}\right) + \beta\gamma F_{ij}\right]$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Gauss-Seidel method

Let G be the iteration matrix of Gauss-Seidel method. By the decompoisition seen before , G is defined as follows

$$G = (D - B)^{-1} C$$

So

$$x^{(k)} = Gx^{(k-1)} + (D - B)^{-1} b$$

Observation

Since matrix $(D - B)$ is lower-triangular, components of vector $x^{(k)}$ can be updated by using the already computed compononents of $x^{(k)}$ itself, speeding-up the computation.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Gauss-Seidel method

So we have

$$x^{(k)} = D^{-1}Bx^{(k)} + D^{-1}Cx^{(k-1)} + D^{-1}b$$

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Gauss-Seidel method for structured matrices

We know that holds the following relation

$$\rho\left(G\right) = \rho^2\left(J\right) = \cos^2\left(\frac{\pi}{p+1}\right)$$

with J the Jacobi iteration matrix. As we did before, by approximation we have

$$\rho\left(G\right) \approx \left(1 - \frac{\pi^2}{2\left(p+1\right)^2}\right)^2$$

And follows that

$$\rho\left(G\right) \approx 1$$

as n approaches infinity.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

Observation

Asimptotically the Jacobi method needs approximately twice the number of iterations of Gauss-Seidel method.

Numerical solving of Poisson systems: the Jacobi and Gauss-Seidel iterative methods

The Gauss-Seidel method for structured matrices (contd.)

The Gauss-Seidel method updates components as follows

$$u_{ij}^{(k)} = \frac{1}{\alpha}\left[\gamma\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)}\right) + \beta\left(u_{i,j-1}^{(k)} + u_{i,j+1}^{(k-1)}\right) + \beta\gamma F_{ij}\right]$$

Numerical solving of Poisson systems: Stencil computations

Definition

In a stencil based computation, as in our case, a working domain values are updated according to some iterative kernel.

Definition

Formally we define a stencil as a 5-tuple $(\mathcal{I}, \mathcal{S}, \mathcal{K}, \mathcal{T}, \mathcal{W}_D)$.

Numerical solving of Poisson systems: Stencil computations

Definition

$\mathcal{I}$ is defined as follows

$$\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_k$$

where

$$\mathcal{I}_i = [\alpha_i, \cdots, \beta_i],\ \alpha_i, \beta_i \in \mathbb{Z}$$

is an interval of integer values. So $\mathcal{I}$ is the k-dimensional index set of the discrete domain, namely its topology. The generic point p in the domain is identified by its coordinates $p \in \mathcal{I}$. Its size is given by

$$|\mathcal{I}| = |\mathcal{D}_1| \times \cdots \times |\mathcal{D}_k|$$

Numerical solving of Poisson systems: Stencil computations

Definition

The discrete domain or simulation space, is defined as follows

$$\mathcal{W}_D^{(i)} : \mathcal{I} \to \mathcal{S}$$

namely it expresses the state of the working domain at timestep $i \in \mathbb{N}$, and $\mathcal{S}$ is a domain of values.

Numerical solving of Poisson systems: Stencil computations

Definition

The geometry of the stencil, $\mathcal{K}$, is given by

$$\mathcal{K} = \{e_1, \cdots, e_\ell\},\ \ e_i \in \mathbb{Z}^k$$

Definition

Let $\mathcal{T}$ be a function such that

$$\mathcal{T} : \mathcal{S}^\ell \to \mathcal{S}$$

namely, is the transition function which is used to determine an element new state by the state of its neighborhood.

Numerical solving of Poisson systems: Stencil computations

Stencil: moving around

We can now obtain for each point $p \in \mathcal{I}$ the tuple of its neighbors points generated via the canonical basis $\mathcal{K}$

$$\mathcal{I}_p = \{ j \mid \exists x \in \mathcal{K} : j = p + x \}$$

their states are given by mapping the tuple $\mathcal{I}_p$ to the corresponding tuple of states $\mathcal{N}_i\left(p\right)$, where

$$\mathcal{N}_i : \mathcal{I} \to \mathcal{S}^{\ell}$$

at time step i, and is defined as follows

$$\mathcal{N}_i\left(p\right) = \left(s_1, \cdots, s_\ell\right),\ s_j = \mathcal{W}_D^{(i)}\left(\mathcal{I}_p\left(j\right)\right)$$

Numerical solving of Poisson systems: Stencil computations

Stencil: moving around (contd.)

State of $\mathcal{W}_D^{(i+1)}$ is computed as follows

$$\mathcal{W}_D^{(i+1)}(p) = \begin{cases} \mathcal{T}(\mathcal{N}_i(p)) & p \in \mathcal{I} \\ \mathcal{W}^{(i)}(p) & p \in \mathbb{Z}^k \setminus \mathcal{I} \end{cases}$$

Sometimes the elements of $\mathcal{I}_p$ may be defined by a vector addition modulo the simulation space dimension to realize toroidal topologies

$$\mathcal{I}_p = \{j \mid \exists x \in \mathcal{K} : j = ((p + x) \bmod (n_1, \cdots, n_k))\}$$

Numerical solving of Poisson systems: implementing 2D Jacobi method

Preliminar considerations

So the 2D Jacobian stencil can be defined as follows

$$\mathcal{S} = \mathbb{R}$$

$$\mathcal{K} = \{(0,-1)^T, (-1,0)^T, (1,0)^T, (0,1)^T\}$$

$$\mathcal{W}^{(i+1)}(p) = \begin{cases} \mathcal{T}(\mathcal{N}_i(p)) = \frac{1}{\alpha}(\gamma(s_e + s_w)\beta(s_n + s_s) + \beta\gamma F_p),\ p \in I \\ \mathcal{S}_i(p),\ p \in \mathbb{Z}^2 \setminus I \end{cases}$$

with $\mathcal{N}_i(p) = (s_n, s_s, s_e, s_w)$

Numerical solving of Poisson systems: implementing 2D Jacobi method

Sequential algorithm

```
while $||u_{i,j}^{(k)} - u_{i,j}^{(k-1)}|| < \epsilon$ do
    for all $(i,j) \in \mathcal{I}$ do
        $u_{i,j}^{(k)} = \frac{1}{\alpha}\left[\gamma\left(u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)}\right) + \beta\left(u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)}\right) + \beta\gamma F_i\right]$
```

the inner-most loop has linear complexity $\mathcal{T}_{\mathcal{J}}(\mathcal{I}) = \mathcal{O}(|\mathcal{I}|)$.

Numerical solving of Poisson systems: implementing 2D Jacobi method

Preliminar considerations to the parallel implementation

As the sequential computation naturally expresses, function $\mathcal{T}$ only depends by the state of $\mathcal{W}_D^{(i)}$; making computation data-depencies free. As a consequence, we can use any topology sorting strategy for traversing $\mathcal{I}$. More precisely, we are dealing with what we call as a **data parallel** computation, characterized by **data partitioning** and **function replication**, as we will see further.

Numerical solving of Poisson systems: implementing 2D Jacobi method

Hypothesis

By definition, we have that 2-dimensional space $\mathcal{I}$ is as follows

$$\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$$

So we could find a partitioning of it, say

$$\mathcal{I}_k = \mathcal{D}_{k_1} \times \mathcal{D}_{k_2}$$

such that $\mathcal{I}_k \subseteq \mathcal{I}$, so that any partition could be updated in parallel.

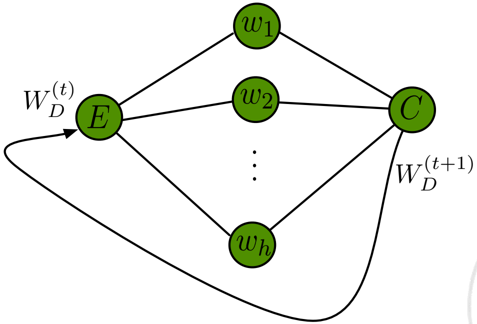
Numerical solving of Poisson systems: implementing 2D Jacobi method

Hypothesis (contd.)

We model function $\mathcal{T}$ as a **computational graph** Σ (as in figure) in which a node C finds a decomposition of $\mathcal{I}$, and the found set of $\{\mathcal{I}_k\}$ is distributed among the $\{w_q\}$ by following a certain policy; the generic w_j is in charge to update the assigned partition, sending it to a node C. Then C gathers all partitions and re-builds the entire (now updated) domain.

Numerical solving of Poisson systems: implementing 2D Jacobi method

The **computational graph $\boldsymbol{\Sigma}$**



Numerical solving of Poisson systems: implementing 2D Jacobi method

A general cost model

Let $\mathcal{T}_E$ and $\mathcal{T}_C$ be the time spent by nodes E and C respectively, and let $\mathcal{T}_{w_j}$ be the time spent by the generic worker node

$$\mathcal{T}_{\Sigma}^{\mathcal{J}} = \max\{\mathcal{T}_E,\ \frac{\mathcal{T}_{w_j}}{n_w},\ \mathcal{T}_C\}$$

where n_w is the number of worker nodes. More in depth, we have that

$$\mathcal{T}_{w_j} = \mathcal{T}_{\text{receive}} + \mathcal{T}_{\mathcal{J}}\left(\mathcal{I}_k\right) + \mathcal{T}_{\text{send}}$$

Numerical solving of Poisson systems: implementing 2D Jacobi method

A general cost model (contd.)

Let's slightly complicate our model, but first we made some considerations:
- working domain $\mathcal{W}_D$ could be implemented as a matrix whose displacement in memory could be of central importance, since accessing it via the pattern exhibited by our computation we could incur in cache misses as the size of matrix grows, so we want to better exploit spatial and temporal locality of references.

Numerical solving of Poisson systems: implementing 2D Jacobi method

A general cost model (contd.)

So, to $\mathcal{T}_{w_j}$ we add a factor $\mathcal{T}_\mu$ defined as

$$\mathcal{T}_\mu = \mathcal{N}_{\text{miss}} \cdot \mathcal{T}_{\text{mem}} = \sigma^{-1}|\mathcal{I}_k| \cdot \mathcal{T}_{\text{mem}}$$

where σ is the size of a cache line and $\mathcal{T}_{\text{mem}}$ is the memory latency. We could alleviate the impact of the previous phenomenon by using **cache blocking** in which $\mathcal{I}$ is decomposed into blocks such that $|\mathcal{I}_k| << \mathcal{N}_{\text{cache}}$, increasing spatial and temporal locality of references.

Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Sequential algorithm

```
while $||u_{i,j}^{(k)} - u_{i,j}^{(k-1)}|| < \epsilon$ do
    for all $(i,j) \in \mathcal{I}$ do
        $u_{ij}^{(k)} = \frac{1}{\alpha}\left[\gamma\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)}\right) + \beta\left(u_{i,j-1}^{(k)} + u_{i,j+1}^{(k-1)}\right) + \beta\gamma F_{ij}\right]$
```

where the inner-most loop has linear complexity $\mathcal{T}_{\mathcal{GS}}(\mathcal{I}) = \mathcal{O}(\mathcal{I}|)$

Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Drawback

Function $\mathcal{T}$ in this case depends by the states of $\mathcal{W}_D^{(i)}$ and $\mathcal{W}_D^{(i+1)}$. In this way we raise data-depencies and the previously seen approach to parallelize stencil is of no use.

Solution

We have to find a topology sorting of $\mathcal{I}$ such that computation semantics is preserved and data-depencies removed.

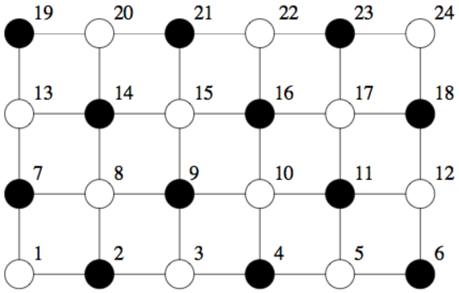
Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Red-Black ordering

Such technique is found to be useful in improving parallelism in iterative solvers. The problem addressed by multicoloring is to determine a coloring of the nodes of the adjacency graph of a matrix such that any two adjacent nodes have different colors.

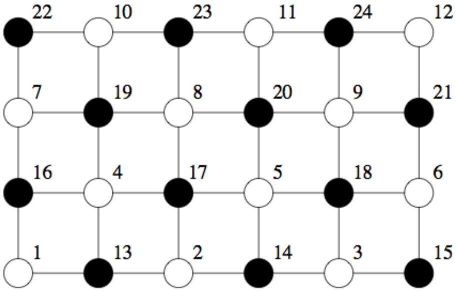
Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

For example, the red-black coloring is illustrated in the following figure for a 6×4 mesh where the black nodes are represented by filled circles.



Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Assume that the unknowns are labeled by listing the red unknowns first together, followed by the black ones. The new labeling of the unknowns is shown in figure



Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Solving Red-Black systems

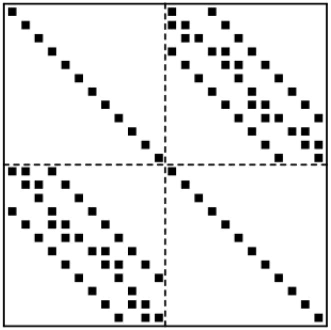
Since the red nodes are not coupled with other red nodes and, similarly, the black nodes are not coupled with other black nodes, the system that results from this reordering will have the following structure

$$\begin{pmatrix} D_r & F \\ E & D_b \end{pmatrix} \begin{pmatrix} x_r \\ x_b \end{pmatrix} = \begin{pmatrix} b_r \\ b_b \end{pmatrix}$$

in which D_1 and D_2 are diagonal matrices.

Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

The reordered matrix associated with this new labeling has the following pattern



Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Solving Red-Black systems (contd.)

Namely is equivalent to solve

$$\begin{cases} x_r^{(k)} = D_r^{-1}\left(-Fx_b^{(k-1)} + b_r\right) \\ x_b^{(k)} = D_b^{-1}\left(-Ex_r^{(k)} + b_b\right) \end{cases}$$

Numerical solving of Poisson systems: implementing 2D Gauss-Seidel method

Considerations

That is equavilent to say that we apply the Jacobi method two times on half of $\mathcal{W}_D$ each time. So the considerations on parallelization and the cost model are still sound. So let $\mathcal{T}_{\mathcal{RB}}$ be the sequential time of the Gauss-Seidel method with red-black ordering, we have that

$$\mathcal{T}_{\mathcal{RB}} \approx \mathcal{T}_{\mathcal{J}} \implies \mathcal{T}_{\Sigma}^{\mathcal{RB}} \approx \mathcal{T}_{\Sigma}^{\mathcal{J}}$$

up to a multiplicative constant.

Numerical solving of Poisson systems: experimental results

Hardware specs

Experiments were done on a Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz with a 8-core chip with 2 threads per core. As a computational addendum, it has two co-processors on which we have actually made tests, and they are equipped with a 60-core chip with 4 threads per core.

Software specs

Code is written in C++11 and **FastFlow** library for parallel code.

Numerical solving of Poisson systems: experimental results

The problem

We look for an approximated solution of

$$\begin{cases} -\nabla^2 u = 2\pi^2 \sin(\pi x) \sin(\pi y) & (x, y) \in \Omega \\ u = 0 & (x, y) \in \partial\Omega \end{cases}$$

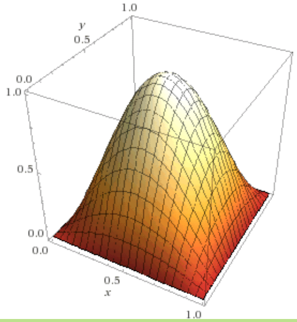
in the unit square $[0, 1] \times [0, 1]$ with Dirichlet boundary condition.

Numerical solving of Poisson systems: experimental results

The problem (contd.)

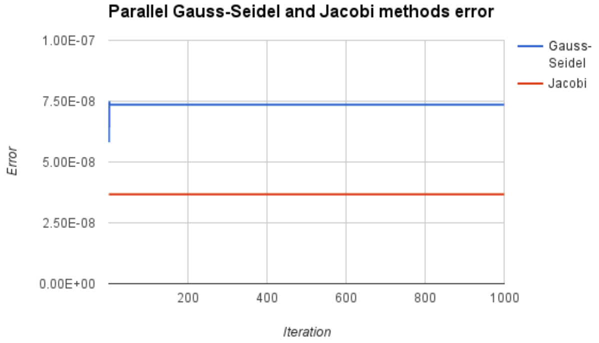
Where the **exact solution** is

$$u = \sin(\pi x) \sin(\pi y)$$



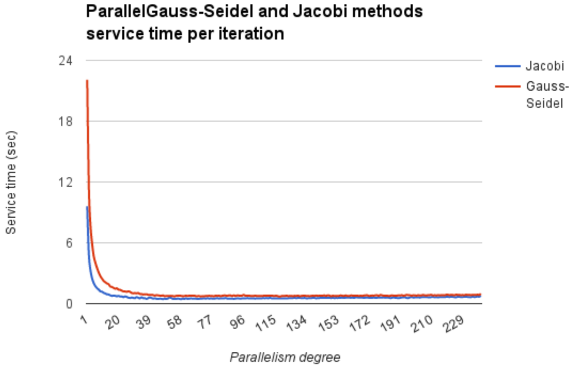
Numerical solving of Poisson systems: experimental results

Error: experiments done on a discrete grid of $8192 \times 8192 \approx 67M$ points and precision $\epsilon = 10^{-9}$



Numerical solving of Poisson systems: experimental results

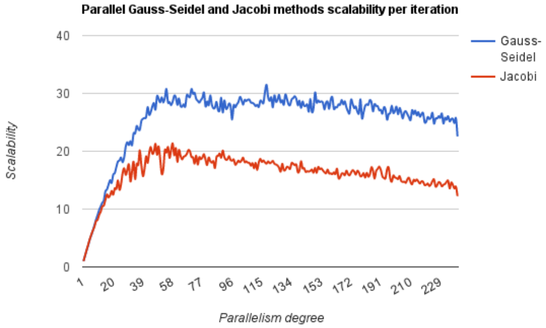
Service time per iteration: experiments done on a discrete grid of $8192x8192 \approx 67M$ points.



Numerical solving of Poisson systems: experimental results

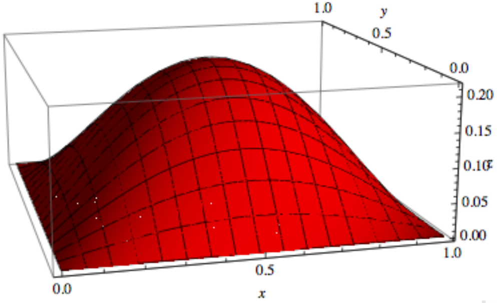
Scalability:

$$\frac{\mathcal{T}_{\text{seq}}}{\mathcal{T}_{\text{par}}} \approx \mathcal{O}\left(n_w\right)$$



Numerical solving of Poisson systems: experimental results

The approximated solution on a $\approx 70M$ grid points with $e \approx 10^{-1}$



Numerical solving of Poisson systems: experimental results

Bibliography

[1] Metodi numerici per l'algebra lineare - Bini, Capovani, Menchi

[2] Matrix computations - Golub, Van Loan, 4th Edition

[3] Grid-Computing: Eine Basistechnologie für Computational Science - Fey, Dietmar et al. (2010)

[4] High performance computing - Marco Vanneschi (2014)