JACOBI ITERATIVE SOLVER

Udhayasankar, Naveen - CSE 708, Fall 2022

Final Presentation – 12/06/2022

University at Buffalo The State University of New York

Solving a system of linear equations

$$Ax + B$$

$$a_{11}x_1 + \cdots + a_{1n}x_n = b_1$$
$$\vdots$$
$$a_{n1}x_1 + \cdots + a_{nn}x_n = b_n$$

1. Direct or Exact methods – Result is obtained in a finite number of steps. Error is zero.
2. Iterative methods – Result is an array of approximate values, which converge to the exact result.

Jacobi Method

$$x_1 = b_1/a_{11} - \frac{a_{12}x_2}{a_{11}} - \cdots \frac{a_{1n}x_n}{a_{11}}$$

$$x_1 = b_1/a_{11} - \frac{\sum_{\substack{j=2 \\ j\neq 1}}^{n} a_{1j}x_j}{a_{11}}$$

$$x_2 = b_2/a_{22} - \sum_{\substack{j=1 \\ j\neq 2}}^{n} \frac{a_{2j}x_j}{a_{22}}$$

$$x_i = b_i/a_{ii} - \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{a_{ij}x_j}{a_{ii}}$$

1. Solve for an unknown using other unknowns.
2. Assume an initial value for all the unknowns, a commonly used value is 0.
3. For each iteration, calculate the unknowns – approximate solutions (vector of size n)
4. Run the iterations until an acceptable solution (a solution close to the exact value) is reached.
5. Each of the iterations produce an approximate solution for the real values of the system.

Jacobi Method – Very Large Number of Unknowns

- Matrix representation

$$A * x = b$$

A: $n \times n$ matrix

x, b: vector of size n

- For n = 4,

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \quad B = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Jacobi Method – Very Large Number of Unknowns

- Matrix A is the sum of Lower Triangle, Diagonal and Upper Triangle

$$A = \begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$$

$$A = L + D + U$$

$$Ax = b$$

$$(L + D + U)x = b$$

$$Lx + Dx + Ux = b$$

$$Dx = b - Lx - Ux$$

$$Dx = b + Dx - Lx - Ux - Dx$$

$$Dx = b + Dx - Ax$$

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

$$x^{k+1} = x^k + \Delta x, \text{ where } \Delta x = D^{-1}(b - Ax)$$

Jacobi Method – Algorithm

Given A, b, n, x_{old} and tolerance,
for max_iterations:
    $\Delta x = D^{-1}\left(b - (A * x_{old})\right)$
    $x_{old} = x_{old} + \Delta x$
    if abs(Δx) < tolerance:
        break
end for

- Run the algorithm for maximum number of iterations or until the algorithm converges.
- Calculate Δx
- Check for convergence i.e., if Δx is less than the given tolerance, the algorithm has converged or produced an acceptable result.
- Update the approximate solution x_{old} to be used in the next iteration.

Jacobi Method – Why?

Why should the Jacobi method be parallelized?

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

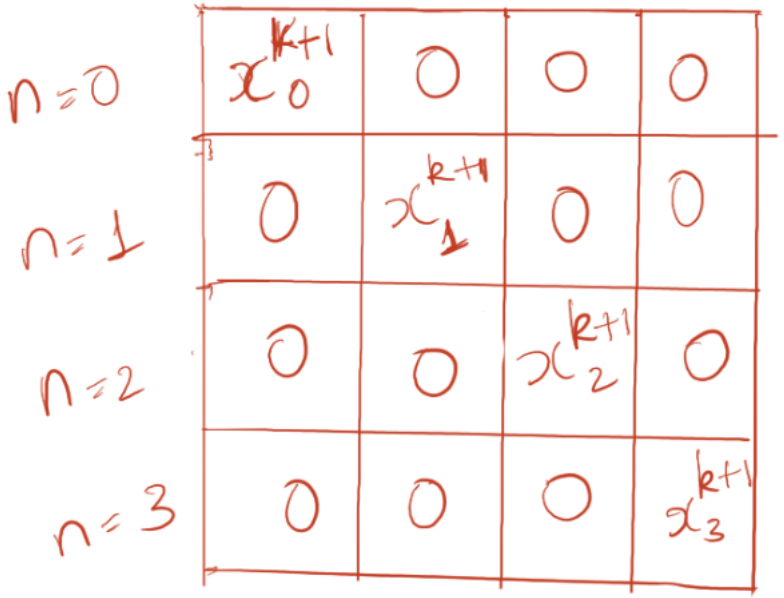
$$i = 1, 2, \ldots n$$

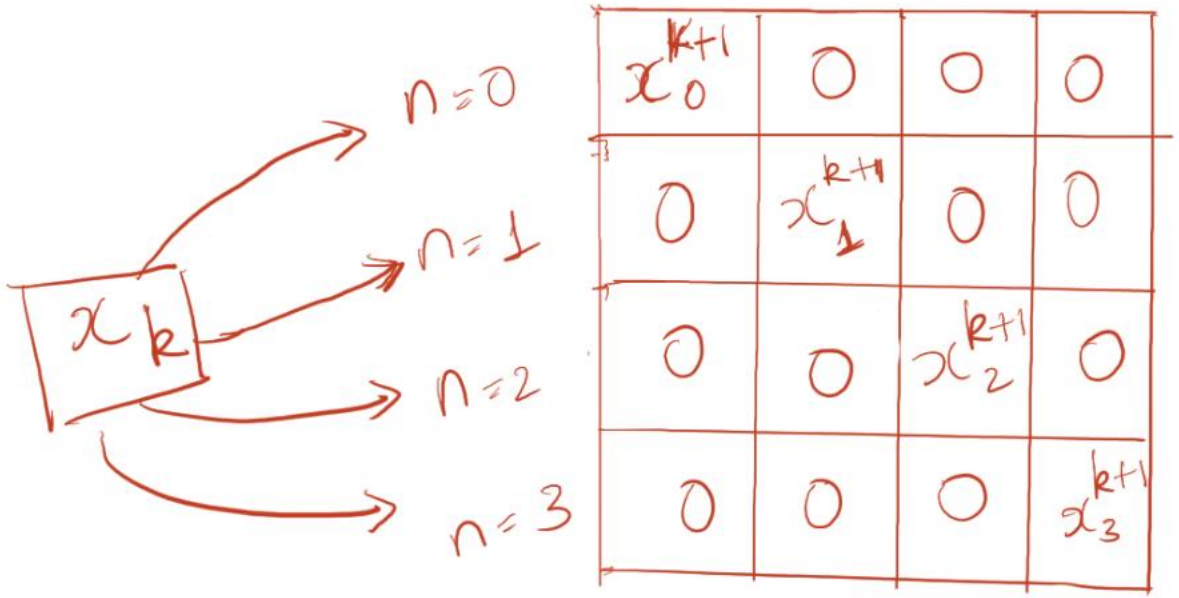
$k+1 \rightarrow$ current iteration

$k \rightarrow$ previous iteration

Jacobi Method – Why?

The ith *processor can calculate the* i^{th} *value to be used in the next iteration, parallely*





Jacobi Method – How?

The ith *processor calculates the* i^{th} *value to be used in the next iteration, parallely, HOW?*

The value of x from previous iteration is known to all processors.

num_proc = 4

$$n=0 \quad x_0^{k+1} = [b_0 - a_{00}\ a_{01}\ a_{02}\ a_{03} * x^k_0]/a_{00}$$

$$n=1 \quad x_1^{k+1} = [b_1 - a_{10}\ a_{11}\ a_{12}\ a_{13} * x^k_1]/a_{11}$$

$$n=2 \quad x_2^{k+1} = [b_2 - a_{20}\ a_{21}\ a_{22}\ a_{23} * x^k_2]/a_{22}$$

$$n=3 \quad x_3^{k+1} = [b_3 - a_{30}\ a_{31}\ a_{32}\ a_{33} * x^k_3]/a_{33}$$

Jacobi Method – How?

The value of x from previous iteration is known to all processors, HOW?

MPI_Allgather [3]:

MPI Allgather allows us to “gather” to all processes information from all processes with the communicator. The action of “all gather” is as if we were to gather to one process using MPI Gather, and then send from that process to all other processes the assembled information.

MPI_Allgather:

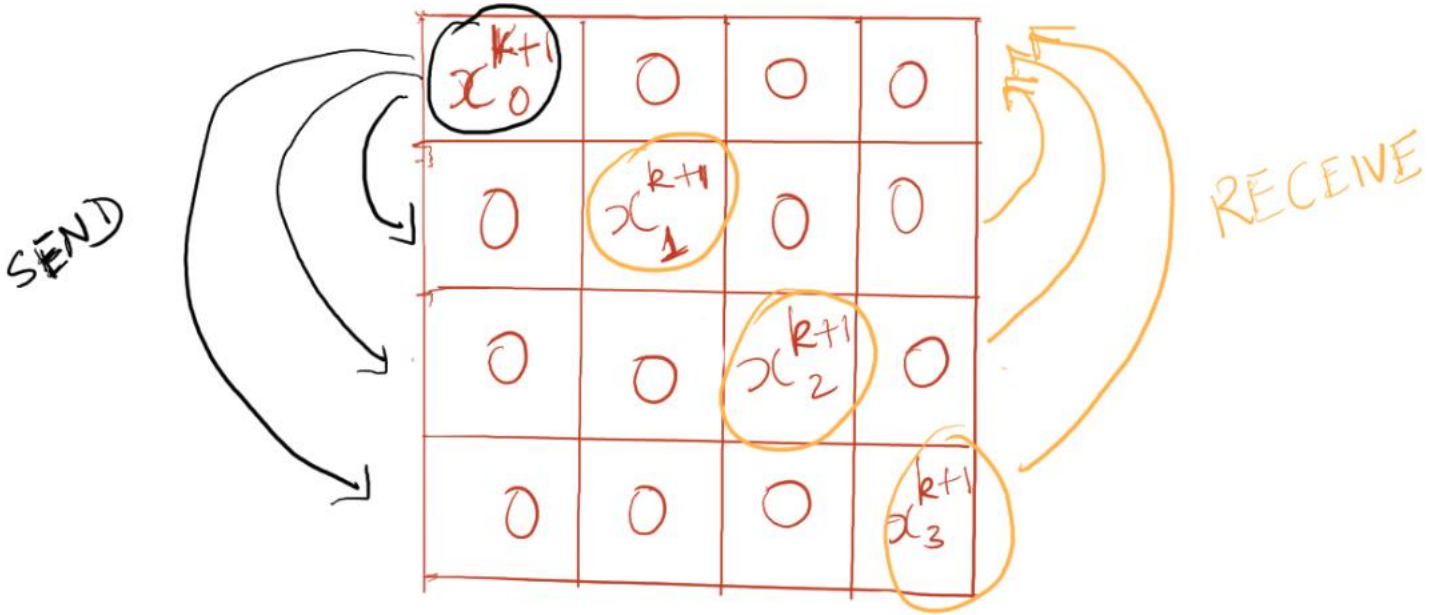
Function Call Syntax

```
int MPI_Allgather(
        void*          sendbuf     /*  in   */,
        int            sendcount   /*  in   */,
        MPI_Datatype   sendtype    /*  in   */,
        void*          recvbuf     /*  out  */,
        int            recvcount   /*  in   */,
        MPI_Datatype   recvtype    /*  in   */,
        MPI_Comm       comm        /*  in   */)
```

Understanding the Argument List

- *sendbuf* - starting address of the send buffer.
- *sendcount* - number of elements in the send buffer.
- *sendtype* - data type of the elements in the send buffer.
- *recvbuf* - starting address of the receive buffer.
- *recvcount* - number of elements for any *single* receive.
- *recvtype* - data type of the elements in the receive buffer.
- *comm* - communicator.

Jacobi Method – How?



Jacobi Method – When?

When does the Jacobi method converge?

- The Jacobi method converges for strictly row-wise or column-wise diagonally dominant matrices.
- The diagonal elements of the matrix must not be zero.
- For strictly row-wise or column-wise matrices, the Jacobi relaxation will converge “to a good solution” in log n steps. In these cases, the parallel time complexity is potentially O(log n) with n processors.

$$a_{i,i} > \sum_{j \neq i} |a_{i,j}|$$

(or)

$$a_{i,i} > \sum_{j \neq i} |a_{j,i}|$$

$$i = 1, 2, \ldots, n$$

Jacobi Method – Inputs and Outputs

- Diagonally dominant matrices.

- $A_{i,i} = 1 + n, all\ other\ elements\ in\ the\ matrix\ are\ 1.$
- x_old for the first iteration is zero vector of size n.
- b is a vector of size n containing the value 2 * n.

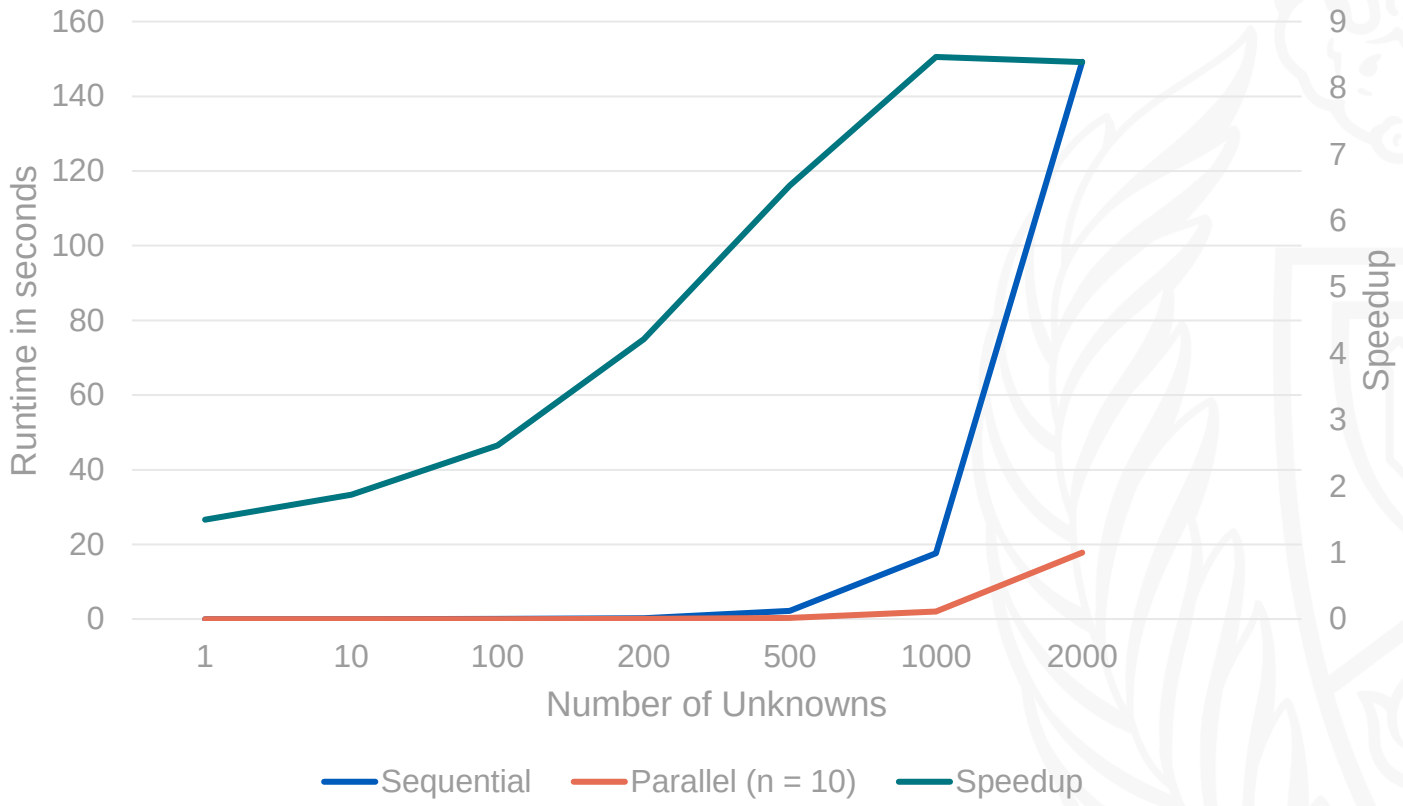
- The solution is always going to be a unit vector of size n.
- Since, for a single row of size n, the diagonal value is 1 + n and there are (n - 1) 1's and the right hand side is 2 * n.

$n=4$

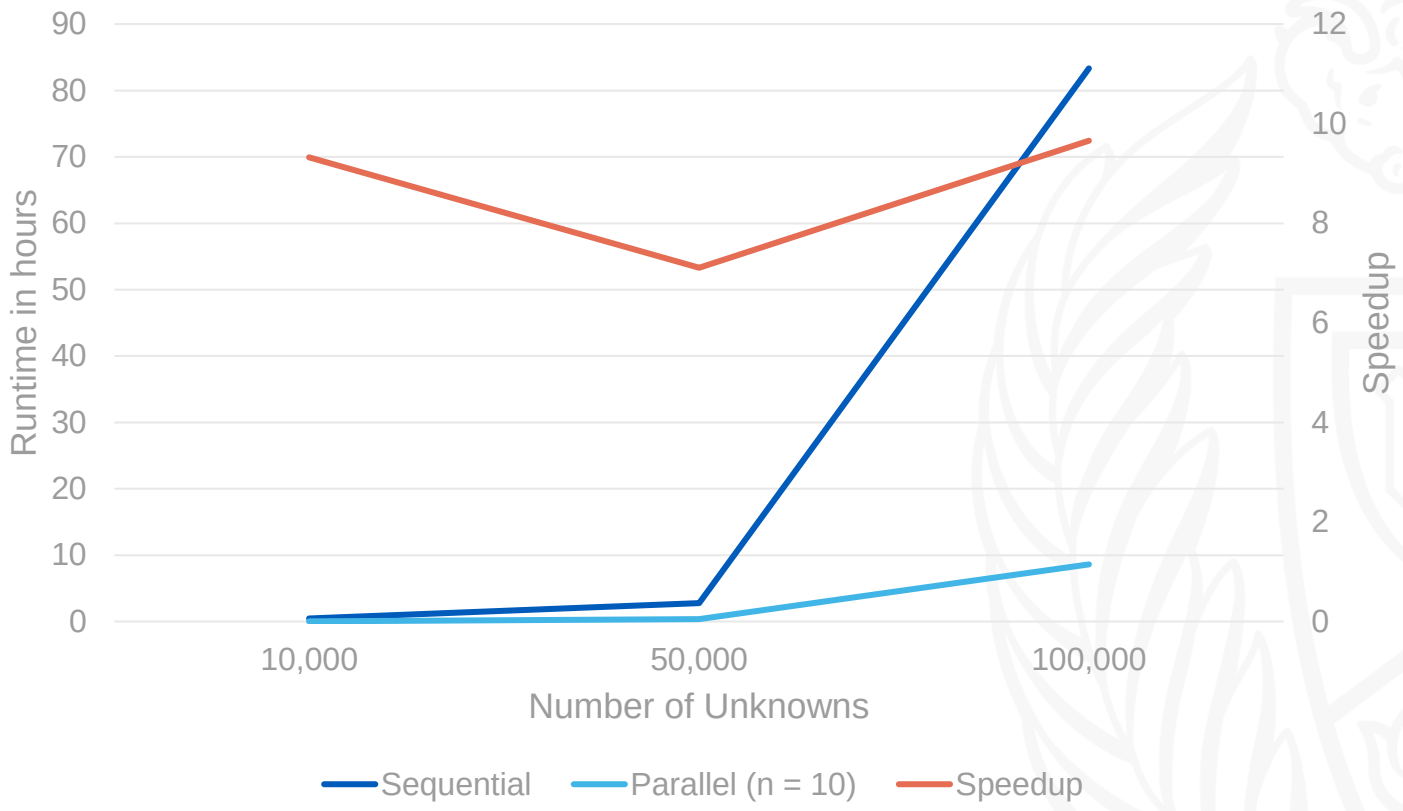
$8x = 8 \Rightarrow x = 1$

$$\underbrace{\begin{bmatrix} (1+n) & 1 & 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x \end{bmatrix} = \underbrace{\begin{bmatrix} 2n \end{bmatrix}}_{B}$$

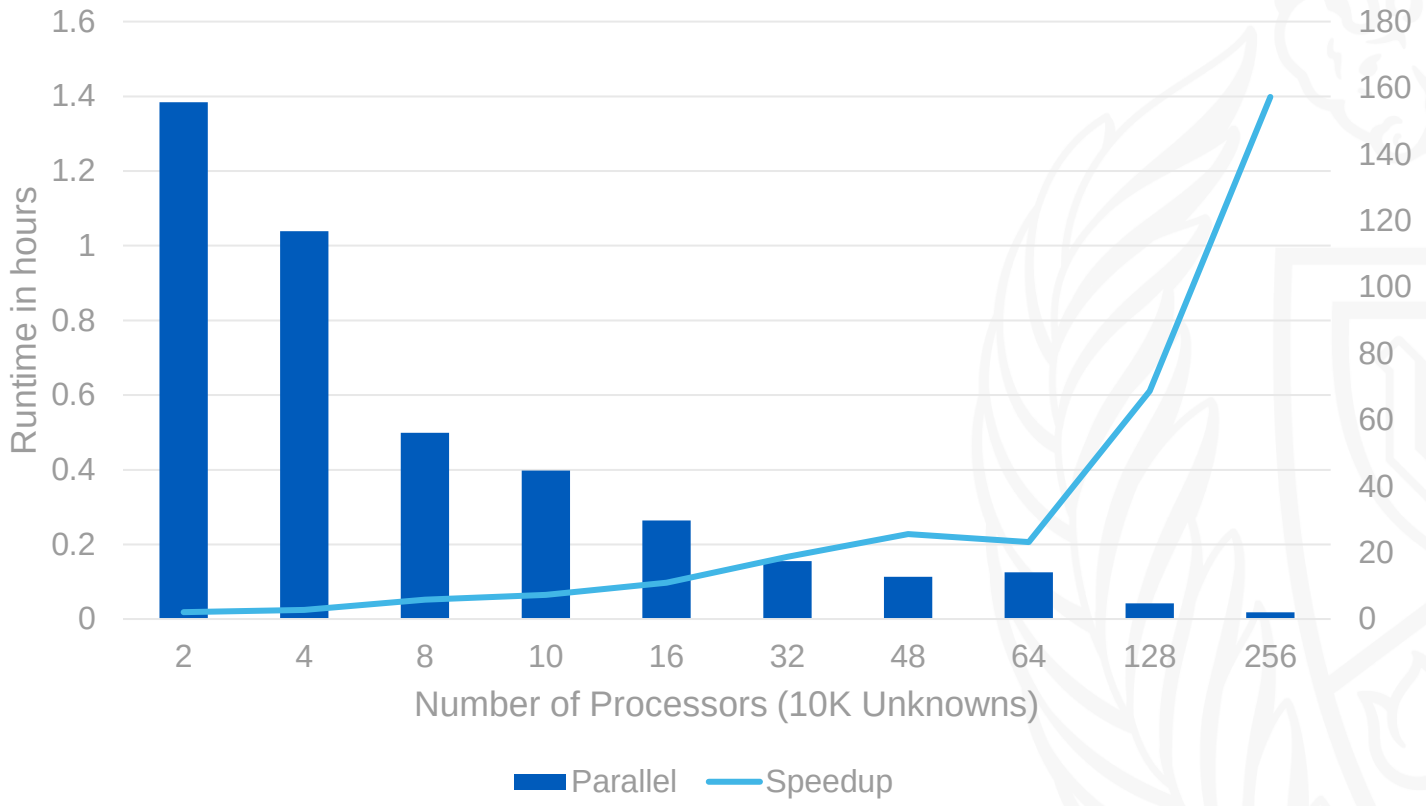
Jacobi Method – Runtime Graphs



Jacobi Method – Runtime Graphs (Larger Data)



Jacobi Method – Runtime Graphs (Scaling out)



Jacobi Method – Profiling Results - % of communication – 1 Processor

unknown

- Load Balance
- Communication Balance
- Message Buffer Sizes
- Communication Topology
- Switch Traffic
- Memmory Usage
- Executable Info
- Host List
- Environment
- Developer Info

powered by IPM

command: ./mpi_jacobi 1000

codename:	unknown	state:	unknown
username:		group:	
host:		mpi_tasks:	1 on 1 hosts
start:		wallclock:	3.01195e+01 sec
stop:		%comm:	0.0402785570809608
total memory:	0.0546227 gbytes	total gflop/sec:	0
switch(send):	0 gbytes	switch(recv):	0 gbytes

Computation

Event	Count	Pop

Communication

% of MPI Time

Jacobi Method – Profiling Results - % of communication – 2 Processors

unknown

- Load Balance
- Communication Balance
- Message Buffer Sizes
- Communication Topology
- Switch Traffic
- Memmory Usage
- Executable Info
- Host List
- Environment
- Developer Info

command: ./mpi_jacobi 1000

codename:	unknown	state:	unknown
username:		group:	
host:		mpi_tasks:	2 on 1 hosts
start:		wallclock:	1.21119e+03 sec
stop:		%comm:	96.7734211808222
total memory:	0.1115722 gbytes	total gflop/sec:	0
switch(send):	0 gbytes	switch(recv):	0 gbytes

Computation

Event	Count	Pop

Communication

% of MPI Time

Jacobi Method – Profiling Results - % of communication – 5 Processors

unknown

- Load Balance
- Communication Balance
- Message Buffer Sizes
- Communication Topology
- Switch Traffic
- Memmory Usage
- Executable Info
- Host List
- Environment
- Developer Info

powered by IPM

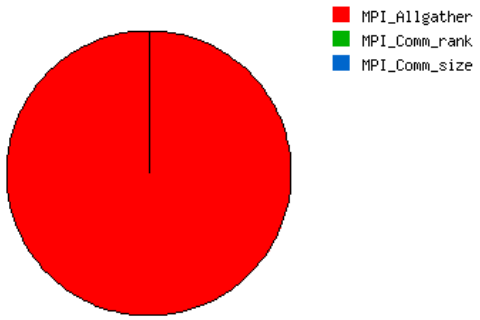
command: ./mpi_jacobi 1000			
codename:	unknown	state:	unknown
username:		group:	
host:		mpi_tasks:	5 on 1 hosts
start:		wallclock:	1.69974e+03 sec
stop:		%comm:	99.5641686375563
total memory:	0.448189 gbytes	total gflop/sec:	0
switch(send):	0 gbytes	switch(recv):	0 gbytes

Computation

Event	Count	Pop

Communication

% of MPI Time



Jacobi Method – Profiling Results - % of communication – 10 Processors

unknown

- Load Balance
- Communication Balance
- Message Buffer Sizes
- Communication Topology
- Switch Traffic
- Memmory Usage
- Executable Info
- Host List
- Environment
- Developer Info

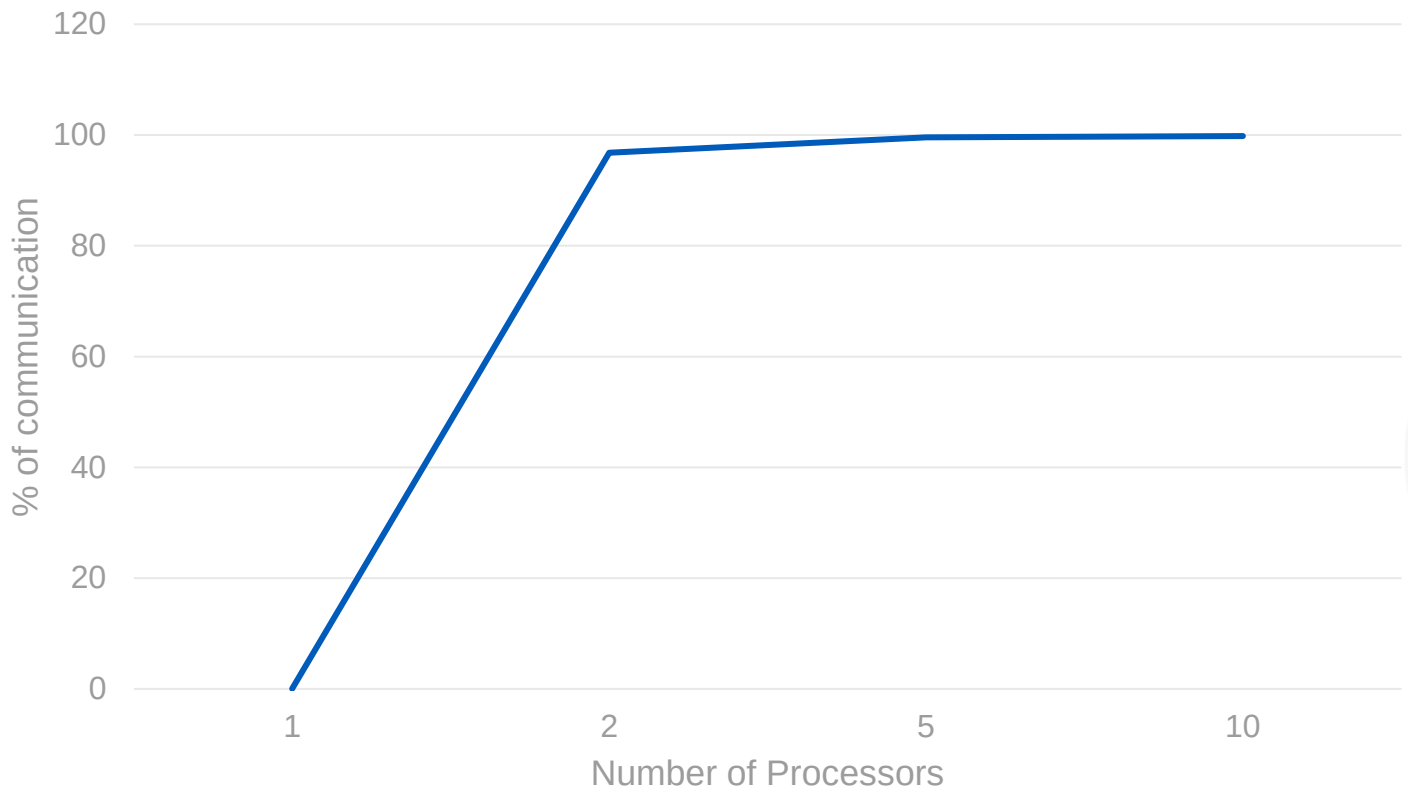
powered by IPM

command: ./mpi_jacobi 1000

codename:	unknown	state:	unknown
username:		group:	
host:		mpi_tasks:	10 on 1 hosts
start:		wallclock:	5.19544e+03 sec
stop:		%comm:	99.7983038972638
total memory:	0.908165 gbytes	total gflop/sec:	0
switch(send):	0 gbytes	switch(recv):	0 gbytes

Computation			Communication
Event	Count	Pop	% of MPI Time

Jacobi Method – Profiling Results - % of communication



References

1. Introduction to Running Computations on the High Performance Clusters at the Center for Computational Research – L. Shawn Matott, Center for Computational Research University at Buffalo, SUNY
2. Wilkinson, Allen - Parallel Programming Techniques and Applications using Networked Workstations and Parallel Computers, 2nd Edition, Section 11.3, 11.4.
3. Snir, Gropp – MPI, The Complete Reference, Section 2.5
4. Karniadakis, Kirby – Parallel and Scientific Computing in C++ and MPI, Section 7.2
5. Edmond Jajaga and Jolanda Kllobocishta - MPI Parallel Implementation of Jacobi, ICT Innovations 2012 Web Proceedings
6. https://www.mpi-forum.org/docs/mpi-2.2/mpi22-report/node99.htm#Node99
7. https://www.mcs.anl.gov/research/projects/mpi/mpi-standard/mpi-report-2.0/node145.htm

THANK YOU!

University at Buffalo The State University of New York