CSIS7303 Tutorial

Introduction to Assignment 1

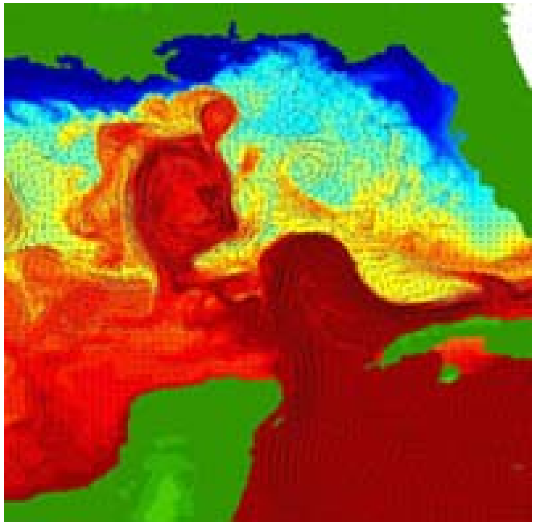
Part 1. OpenMP Programming

Part 2. MPI Programming

Session 2008 2nd semester

Date: 13 Feb 2008

Assignment 1:
Parallel Solver for the Steady-State Heat Distribution Problem



A well-known problem in physics.

Application relevance to weather forecasting …

The Heat Distribution Problem

- Find the temperature distribution within an area having known temperatures along each of its edges.
- Modeled by the *Laplace's* equation

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t} = 0$$

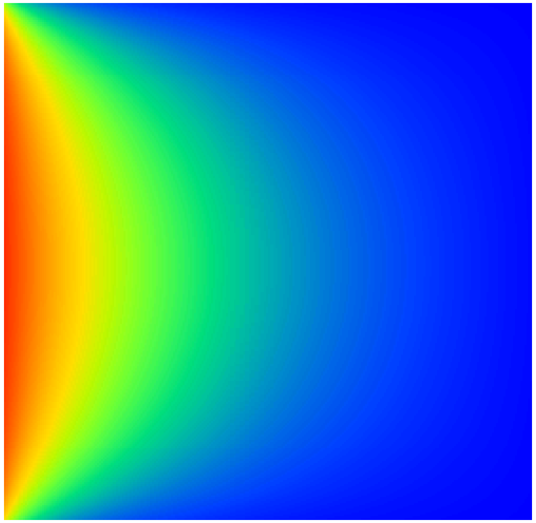
- Initial condition:

$$u(x, y, 0) = 25 \qquad 0 < x < 1, \quad 0 < y < 1$$

- Boundary conditions:

$$u(x, 0, t) = 1000 \qquad u(0, y, t) = 0$$

$$u(x, 1, t) = 0 \qquad u(1, y, t) = 0$$



Finite Difference Methods

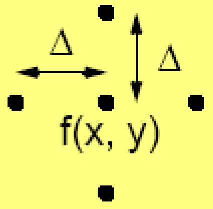
- Approximate an solution to the Laplace’s equation $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$

If distance between points, Δ, made small enough:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{1}{\Delta^2}[f(x+\Delta, y) - 2f(x, y) + f(x-\Delta, y)]$$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{1}{\Delta^2}[f(x, y+\Delta) - 2f(x, y) + f(x, y-\Delta)]$$

Discretize the solution space



Substituting into Laplace’s equation, we get

$$\frac{1}{\Delta^2}[f(x+\Delta, y) + f(x-\Delta, y) + f(x, y+\Delta) + f(x, y-\Delta) - 4f(x, y)] = 0$$

Rearranging, we get

$$f(x, y) = \frac{[f(x-\Delta, y) + f(x, y-\Delta) + f(x+\Delta, y) + f(x, y+\Delta)]}{4}$$

Rewritten as an iterative formula:

$$f^k(x, y) = \frac{[f^{k-1}(x-\Delta, y) + f^{k-1}(x, y-\Delta) + f^{k-1}(x+\Delta, y) + f^{k-1}(x, y+\Delta)]}{4}$$

$f^k(x, y)$ - kth iteration, $f^{k-1}(x, y)$ - $(k-1)$th iteration.

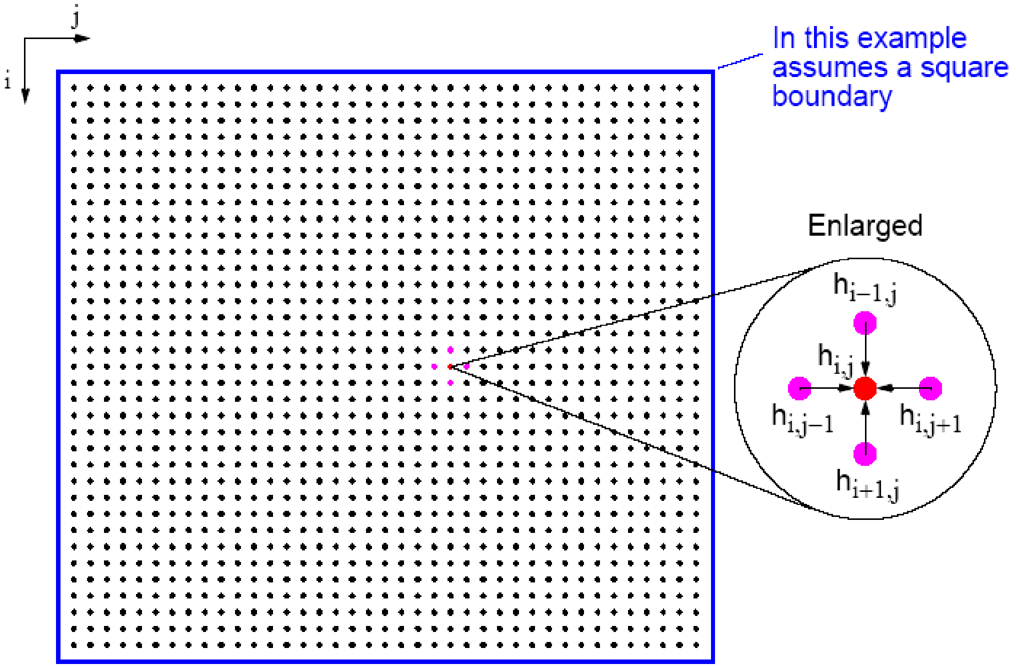
Finite Difference Methods

- Temperature of each point by iterating the equation

$$h_{i,j} = \frac{h_{i-1,j} + h_{i+1,j} + h_{i,j-1} + h_{i,j+1}}{4}$$

- Loop until predefined error tolerance (temperature diff % successive iterations) or maximum iterations is reached
- Several algorithm variants:

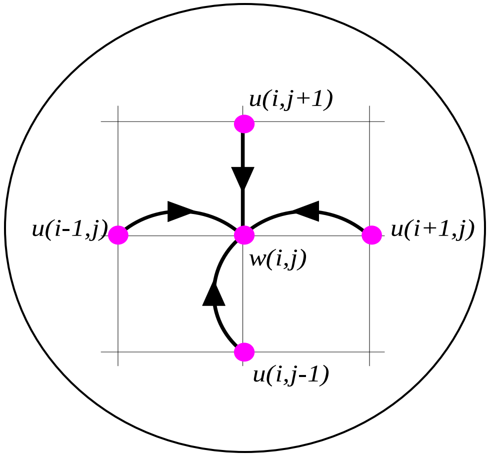
  Jacobi, Gauss-Seidel, SOR



In this example assumes a square boundary

Jacobi Iterative Method

$$w[i,j] = \frac{u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1]}{4}$$



Sequential Code:

```
for (its = 0; its < max_its; its++) {
   diff = 0.0;
   for (i = 1; i < M-1; i++) {
      for (j = 1; j < N-1; j++) {
         w[i][j] = 0.25 * (u[i-1][j] + u[i+1][j] +
                           u[i][j-1] + u[i][j+1]);
         if (fabs(w[i][j] - u[i][j]) > diff)
            diff = fabs(w[i][j] - u[i][j]);
      }
   }
   for (i = 1; i < M-1; i++)
      for (j = 1; j < N-1; j++)
         u[i][j] = w[i][j];

   /* Terminate if temperatures have converged */
   if (diff <= EPSILON)
      break;
}
```

Let M = N always.
What's the time complexity?

Cost per iteration: O(N^2)

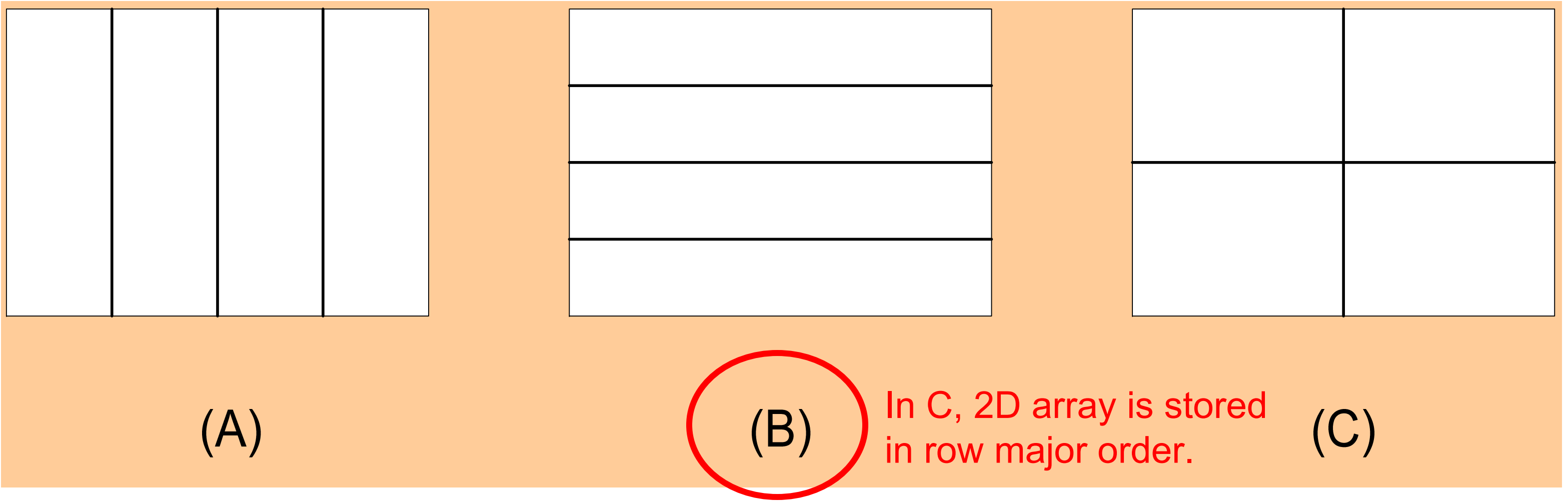
Length of execution depends on the convergence rate (algorithm-specific), initial *u* and *EPSILON*.
Below is a rough approximation:

No. of iterations: ~ $[(N+1)/\pi]^2$ = O(N^2) for large *N* (See proof.)

Answer: ~ $\Theta(N^4)$

Parallel Implementations – Data/Domain Decomposition

- Dividing computation and data into pieces
- Domain could be decomposed in three ways:
  - Column-wise: adjacent groups of columns (A)
  - Row-wise: adjacent groups of rows (B)
  - Block-wise: adjacent groups of two dimensional blocks (C)



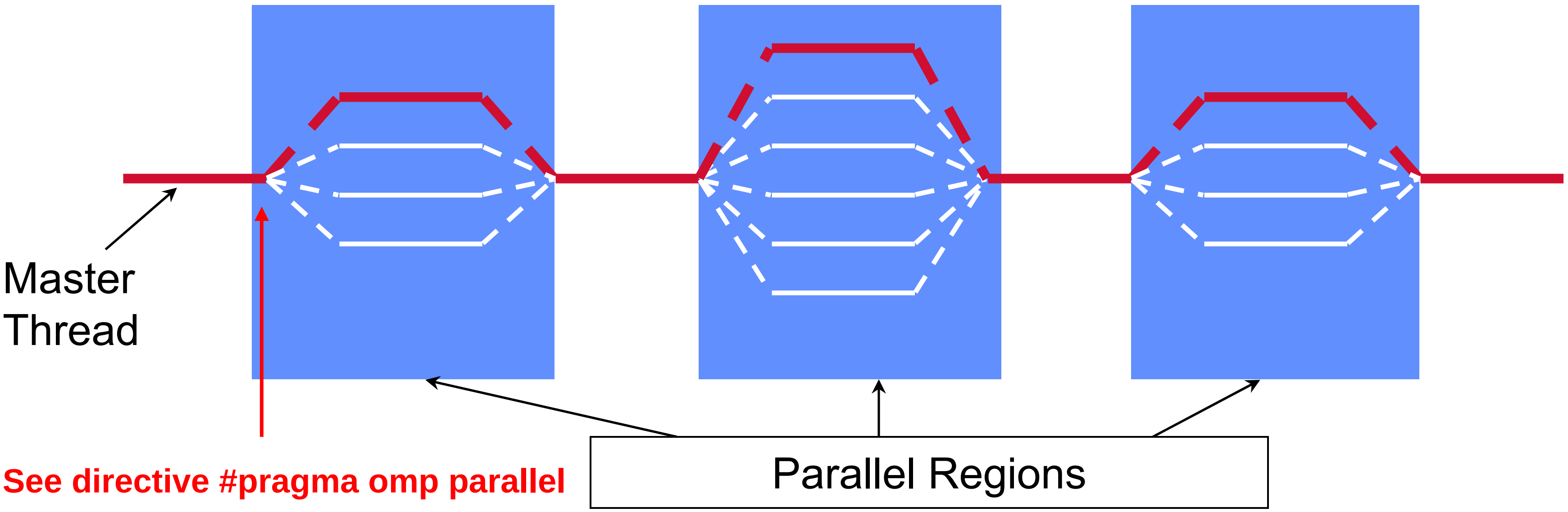
In C, 2D array is stored in row major order.

Part I

OpenMP Programming

OpenMP Programming Model

- Shared-Memory Programming
- Fork-Join Parallelism
  - Master thread spawns a team of threads as needed.
  - Parallelism is added incrementally: i.e. the sequential program evolves into a parallel program.



Master Thread

See directive #pragma omp parallel

Parallel Regions

Make the Code Multithreaded! How?

- Add omp directives to create loop parallelization

#pragma omp parallel private (...) – *which one applicable?*

```
for (its = 0; its < max_its; its++) {
   diff = 0.0;
   for (i = 1; i < M-1; i++) {
      for (j = 1; j < N-1; j++) {
         w[i][j] = 0.25 * (u[i-1][j] + u[i+1][j] + u[i][j-1] + u[i][j+1]);
         if (fabs(w[i][j] - u[i][j]) > diff)
            diff = fabs(w[i][j] - u[i][j]);
      }
   }
   for (i = 1; i < M-1; i++)
      for (j = 1; j < N-1; j++)
         u[i][j] = w[i][j];

   /* Terminate if temperatures have converged */
   if (diff <= EPSILON)
      break;
}
```

Any data dependence?

Which variable(s) be shared or not shared (private)?

Any data race?

Static, dynamic, guided scheduling; Optimal chunk size?

Program Development

- Log in any CS Unix server using your CS account and password
- Start programming with text editor (vi, pico, emacs, etc) or FTP your source file to your home directory.
- Suggest not to work on the *Genius* server since it is of a different architecture - SPARC rather than x86.
- Check your program correctness carefully
    - Parallel and sequential programs should give the same output. Precision of the output file is correct to 2 decimal places only.

```
diff jacobi_seq.dat jacobi_omp.dat
```

→ Show nothing if they equal

Compilation and Execution

- Use the Sun Studio compiler

```
export PATH=/opt/SUNWspro/bin/:$PATH
```

- Sequential Program

```
cc -xO3 -o <pgm> <pgm.c>
./<pgm> <parameters>
```

- OpenMP Program

```
cc -xO3 -xopenmp=parallel -o <pgm> <pgm.c>
```

Adjust the number of processor cores before execution:

```
export PARALLEL=<num_of_proc>
./<pgm> <parameters>
```

or

Use **`omp_set_num_threads`** in **`pgm.c`**

Performance Benchmarking

- Experimental Platform:
  - **Belief** - 2 × Xeon X5355 (8 cores) 2.6GHz, 8GB RAM, Solaris 10 x86 64-bit
- Reserved period: **March 8-14 (one week)**
- Job Scheduling:
  - All benchmarking jobs must be submitted through the Linux ‘at’ command.
    - Try and practice the ‘at’ command or prepare your own shell scripts before the reserved period.
  - **REMEMBER**: During reserved period, need your cooperation to keep Belief dedicated for benchmarking:
    - Compile and test your programs on other servers: Honest or Virtue
    - Don’t run any interactive jobs like VNC on Belief
  - Process running for more than 10 min will automatically be aborted by the system.

Submit Batch Jobs on Belief

- Submit job:

```
echo "export PARALLEL=4; ./jacobi_omp 300 300" > jacobi.job
at –q d –f jacobi.job now  [or at –q d now < jacobi.job]
```

  - where d is the shared batch job queue for this assignment.
    - Only one job from the queue can be run
    - If one is already running, your job will be deferred and wait for 10 seconds before re-probing the system for availability.

- Other scheduling ways:

```
at -q d –f jacobi.job now + 5 minutes
at -q d –f jacobi.job now + 1 hour
at -q d –f jacobi.job midnight
at -q d –f jacobi.job 0930pm
```

Other Batch Job Management Commands

- List your submitted jobs: `at -l`

```
1202294722.d    Wed Feb  6 18:45:22 2008
1202313600.d    Thu Feb  7 00:00:00 2008
1202340600.d    Thu Feb  7 07:30:00 2008
1202297400.d    Wed Feb  6 19:30:00 2008
1202340601.d    Thu Feb  7 07:30:01 2008
```

- Check job queue: `atq -q d`
  - Check how many users are also waiting on the queue.

```
 Rank      Execution Date      Owner      Job              Queue    Job Name
  1st    Feb  6, 2008 18:00    ktlam      1202292000.d       d      jacobi.job
  2nd    Feb  6, 2008 18:45    ktlam      1202294722.d       d      stdin
  3rd    Feb  6, 2008 19:30    ktlam      1202297400.d       d      stdin
  4th    Feb  7, 2008 00:00    ktlam      1202313600.d       d      stdin
  5th    Feb  7, 2008 07:30    ktlam      1202340600.d       d      stdin
```

- Delete job: `at -r job_id`
  - Only the job owner can do so.

Other Batch Job Management Commands (Cont')

- Program Output:
  - By default, program output is sent to your CS mail account.

```
Your "at" job on belief
"/var/spool/cron/atjobs/1202289688.d"

produced the following output:

Problem size: M=300, N=300
Converged after 54413 iterations with error:
0.00099998.
Elapsed time = 29.236654 sec.
```

  - If email is not preferred, you can redirect output to a file in your job. For example,

```
echo "./jacobi_omp 300 300 > ~/jacobi.out" > jacobi.job
```

Start an X Terminal on CS Unix Server to Run Gnuplot

- Please refer to CS Intranet’s Technical Support
  - VNC How-to
  - SSH Tunneling (Port Forwarding)

display no.

Start VNC server:

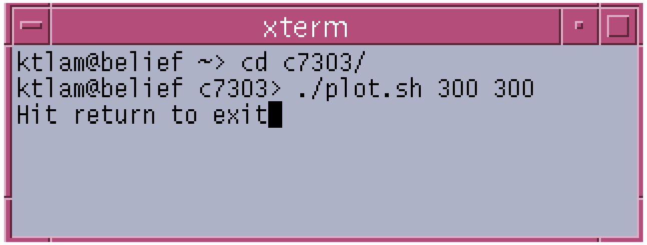
```
vncserver :1
```

Stop VNC server:

```
vncserver -kill :1
```

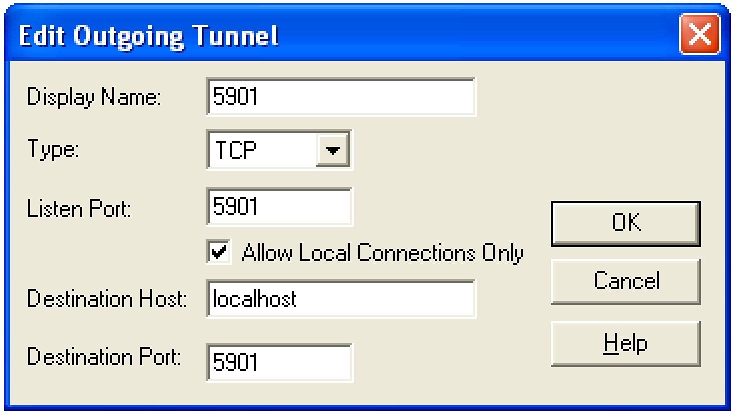
Under the VNC screen:

Press Ctrl+T to open a terminal
Type `xterm` to open an X terminal
`./plot.sh <parameters>`



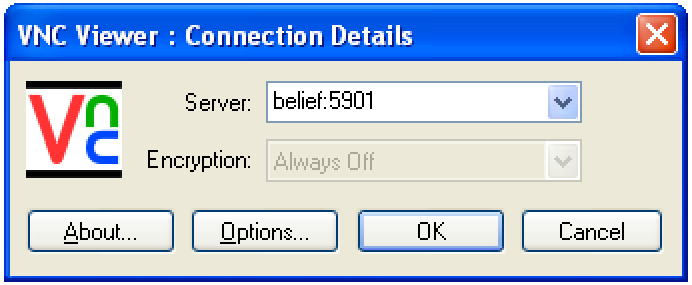
xterm

```
ktlam@belief ~> cd c7303/
ktlam@belief c7303> ./plot.sh 300 300
Hit return to exit
```



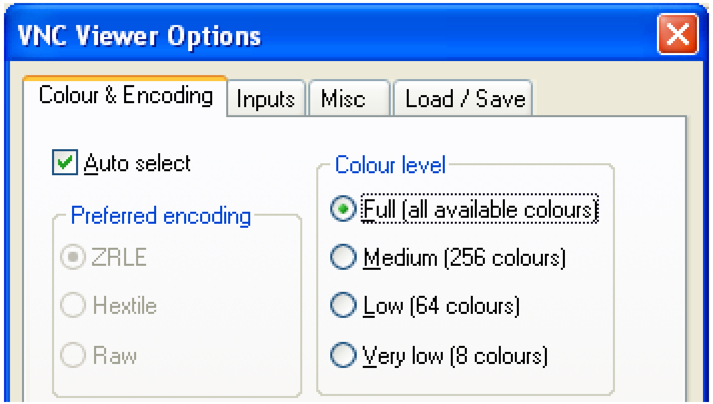
Edit Outgoing Tunnel

Display Name: 5901
Type: TCP
Listen Port: 5901
☑ Allow Local Connections Only
Destination Host: localhost
Destination Port: 5901



VNC Viewer : Connection Details

Server: belief:5901
Encryption: Always Off



VNC Viewer Options

☑ Auto select

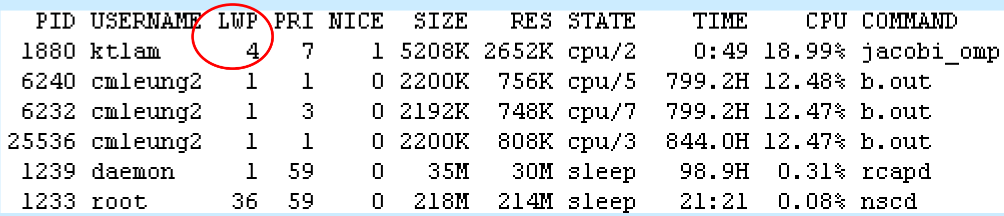
Preferred encoding
● ZRLE
○ Hextile
○ Raw

Colour level
● Full (all available colours)
○ Medium (256 colours)
○ Low (64 colours)
○ Very low (8 colours)

Demonstration

- Login Belief
- Set up VNC server
- Start VNC viewer to connect to the VNC server
- Start an X terminal
- export PATH=/opt/SUNWspro/bin/:$PATH
- cc -xO3 -o jacobi_seq jacobi_seq.c -lm -DINTERACTIVE
- cc -xO3 -xopenmp=parallel -o jacobi_omp jacobi_omp.c -lm -DINTERACTIVE jacobi_omp.c -lm -DINTERACTIVE
- ./jacobi_seq 200 200
- export PARALLEL=4
- ./jacobi_omp 200 200

top shows your process has 4 LWP’s



```
  PID USERNAME LWP PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
 1880 ktlam      4   7    1 5208K 2652K cpu/2    0:49 18.99% jacobi_omp
 6240 cmleung2   1   1    0 2200K  756K cpu/5  799.2H 12.48% b.out
 6232 cmleung2   1   3    0 2192K  748K cpu/7  799.2H 12.47% b.out
25536 cmleung2   1   1    0 2200K  808K cpu/3  844.0H 12.47% b.out
 1239 daemon     1  59    0   35M   30M sleep   98.9H  0.31% rcapd
 1233 root      36  59    0  218M  214M sleep   21:21  0.08% nscd
```

Sample benchmark result

Part II

MPI Programming

Sample MPI Program Skeleton

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
```

Initialization

```
    MPI_init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_world, &p);
    MPI_Comm_rank(MPI_COMM_world, &id);
```

Distribution of data

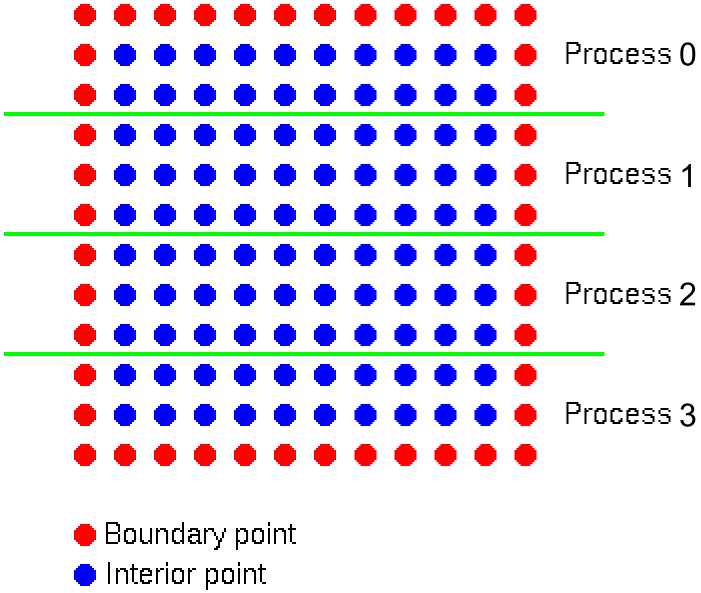
```
    if (id == 0) {
        /* Input data from inputfile */
        /* Partition the data and distribute them to all processes */
    } else {
        /* Get data from process 0 */
    }
    MPI_barrier();
    stime = MPI_Wtime();
```

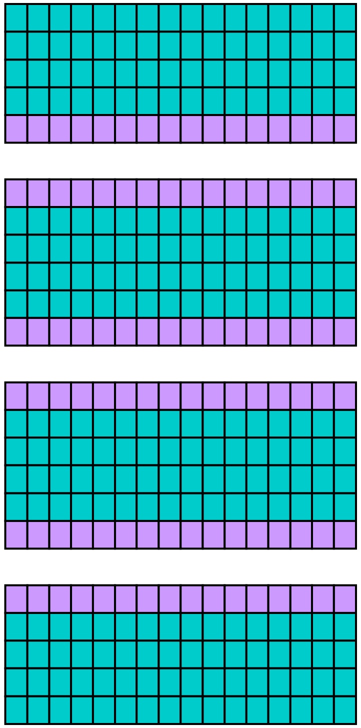
Parallel computation

```
    /* All processes begin their computation, */
    /* There may be some communication between some processes */
    /* Finally process 0 collects the result from other processes */
    MPI_barrier();  /* Or a collective operation like
                         MPI_Reduce(&count, &global_count, 1, MPI_INT,
                                       MPI_SUM, 0, MPI_COMM_WORLD);   */
    etime = MPI_Wtime();
    if (!id) printf(“Result:” ...);
    MPI_finalize();
}
```

Domain Decomposition Revisit

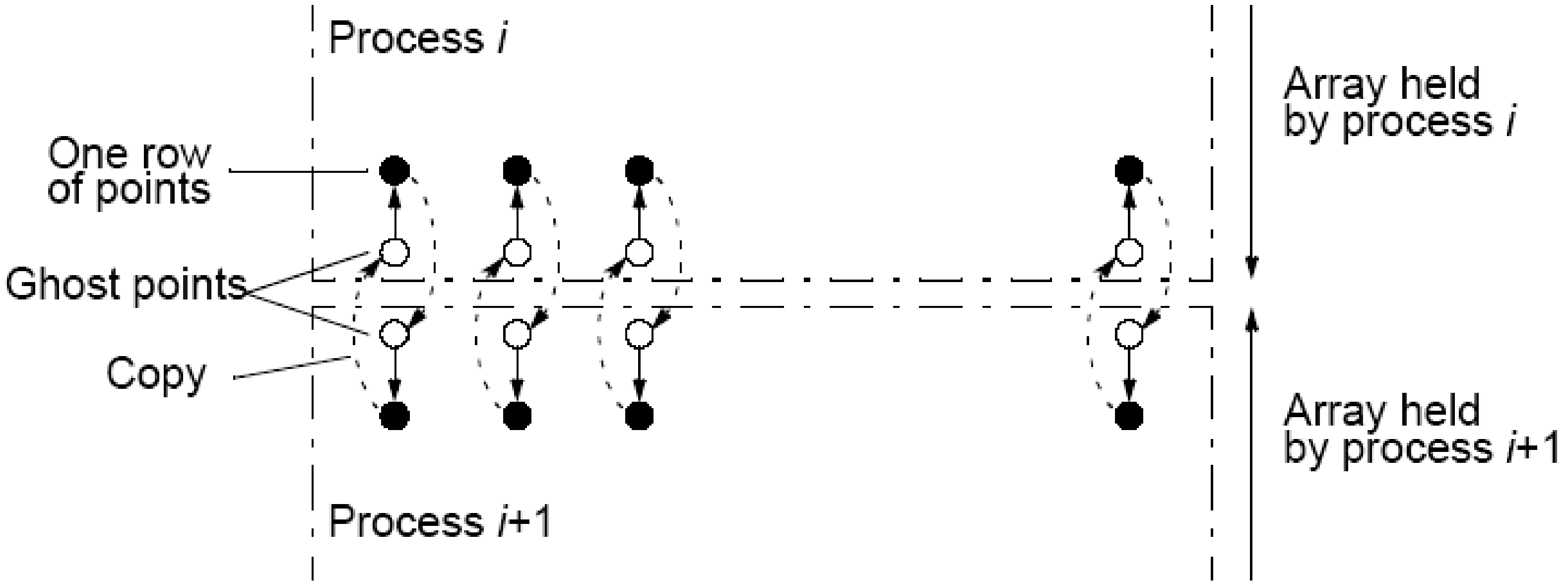
- Below is a 16 × 16 grid divided among 4 processors





Ghost Points

- Additional row of points at each edge that hold values from adjacent edge. Each array of points increased to accommodate ghost rows.



Login the Gideon 300 Cluster

- Login gatekeeper.cs.hku.hk or any CS Unix server (faith, virtue, honest) using SSH

- Login cserver1 (entry point of Gideon)

```
ssh –p 2004 [cs_user_id]@cserver1
```

(You will be automatically redirected to **GD001**. *Don't run program there*)

- Login a Gideon node

```
ssh GDxxx
```

  - For **development**: Login your assigned node in the range GD201-208 (8 nodes) according to your last digit of university number.
  - For **benchmarking**: Login GD208 and submit your job there through OpenPBS which will dispatch your job to GD134-149 (16 nodes).
  - Remember to completely '`logout`' when leave.

File Management on Gideon

- Your home directory: **`/home/home_7303/[cs_user_id]`**
- Protect your work from unauthorized access. Please do "**`chmod 700 ~`**".
- Files under your home directory are accessible from any Gideon node as all nodes share a common NFS file system (with cserver1).
- File transfer between Gideon and CS Unix Server:
  - **By FTP** - on a development node, type commands like below.

```
ftp virtue.cs.hku.hk
… (Type your CS Unix account and password)
ftp> get jacobi_seq.c
ftp> mget *.c
ftp> put jacobi.out
ftp> bye
```

  - **By Secure Copy** - on a development node, type commands like below.

```
scp virtue.cs.hku.hk:~/jacobi_seq.c .
scp -r virtue.cs.hku.hk:~/mydir .
scp jacobi.out virtue.cs.hku.hk:~
```

Machine File Configurations

- .rhosts File:
  - Used to bypass password prompts in `rsh`, `rcp`.
  - For MPICH to spawn a remote process (avoid “permission denied”)
  - For OpenPBS to remote-copy output files
  - List all development nodes (GD201-208) and benchmarking nodes (GD134-149) **with and without suffix B** + GD001 (optional)
  - Make sure .rhosts is in your **home directory** and:
    - **owned by your login_id**
    - **with permission of 600** (Do "`chmod 600 .rhosts`“ if not)

```
GD001
GD201
GD201B
GD202
GD202B
…
GD149
GD149B
```

- MPI Machine File (of arbitrary name, say machines.txt):

```
#Comment: Machines for Development
GD201B
GD202B
…
GD208B
```

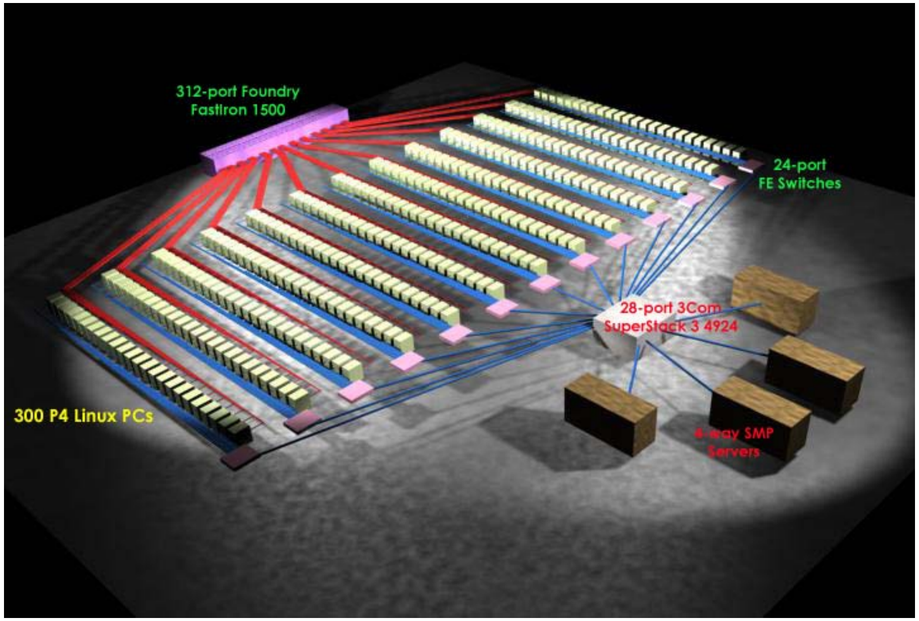
  - List all development nodes **with suffix B**. **Don’t** include any benchmarking nodes (GD134-149). Put it in the current directory containing your programs.

Gideon’s Network Interfaces

- **Network A (Hostname without suffix B)** - Hierarchical management network (for login, I/O access and cluster management)
  - 13 x 24-port Fast-Ethernet switches with uplink to one 24-port Gigabit-Ethernet switch
- **Network B (Hostname with suffix B)** - High-performance network (for inter-process communication)
  - Foundry Networks' Fast-Ethernet switch with 312 ports

So, please

- define GDxxx (for login) and GDxxx**B** (for MPI’s IPC) in .rhosts.
- define GDxxx**B** only in the MPI machine file used in the development stage.



Compilation and Execution

- Use the MPI library

```
export PATH=/usr/local/mpich-1.2.7p1/bin:$PATH
```

- Sequential Program

```
cc -O3 -o <pgm> <pgm.c>
./<pgm> <parameters>
```

- MPI Program

```
mpicc -O3 -o <pgm> <pgm.c>
```

```
mpirun –np <p> –machinefile <machine_file> –nolocal <pgm> <parameters>
```

Required if your machine file includes the local node calling the mpirun command; otherwise a slave process runs also on the local node.

where <p> is the number of processes for the execution; 1 master and (p-1) slave processes

Note:

1. Compile with optimization level -O3, or else your program will be too slow.
2. If you will use <math.h> in your program, remember to link with the math library by compiling with option -lm.
3. If you include <MyMPI.h>, place MyMPI.h in the same directory with your C source file.

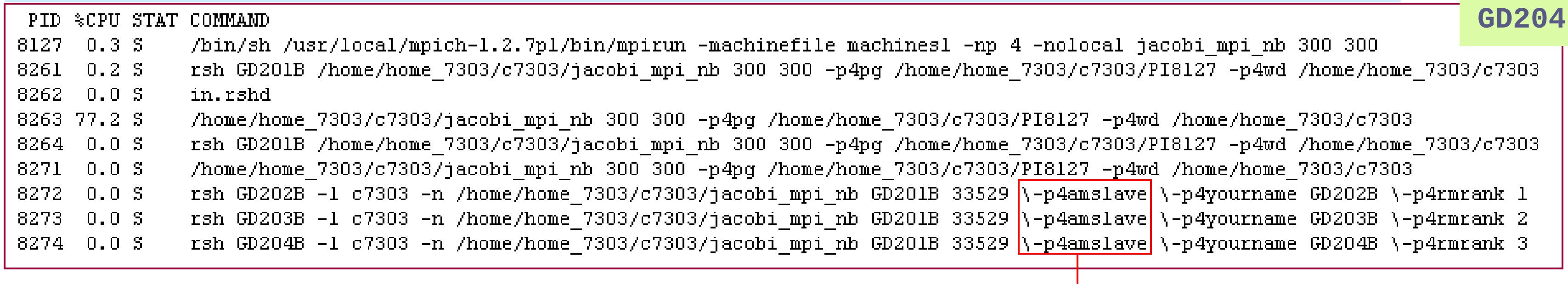
MPI Processes (Local/Remote)

machines1

GD201B
GD202B
GD203B
GD204B

Case 1: ✓

```
[GD201]$ mpirun -np 4 -machinefile machines1 -nolocal jacobi_mpi_nb 300 300
```



```
 PID %CPU STAT COMMAND
8127  0.3 S    /bin/sh /usr/local/mpich-1.2.7p1/bin/mpirun -machinefile machines1 -np 4 -nolocal jacobi_mpi_nb 300 300
8261  0.2 S    rsh GD201B /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI8127 -p4wd /home/home_7303/c7303
8262  0.0 S    in.rshd
8263 77.2 S    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI8127 -p4wd /home/home_7303/c7303
8264  0.0 S    rsh GD201B /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI8127 -p4wd /home/home_7303/c7303
8271  0.0 S    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI8127 -p4wd /home/home_7303/c7303
8272  0.0 S    rsh GD202B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201B 33529 \-p4amslave \-p4yourname GD202B \-p4rmrank 1
8273  0.0 S    rsh GD203B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201B 33529 \-p4amslave \-p4yourname GD203B \-p4rmrank 2
8274  0.0 S    rsh GD204B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201B 33529 \-p4amslave \-p4yourname GD204B \-p4rmrank 3
```

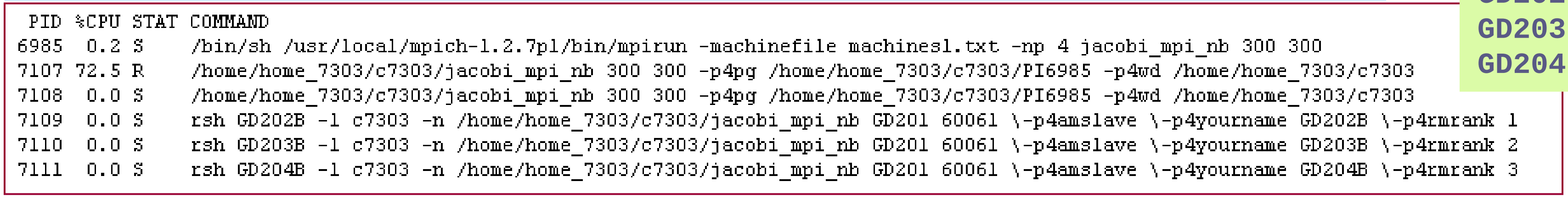
3 slave processes on 3 nodes

machines2

#GD201B
GD202B
GD203B
GD204B

Case 2: ✓

```
[GD201]$ mpirun -np 4 -machinefile machines2 jacobi_mpi_nb 300 300
```



```
 PID %CPU STAT COMMAND
6985  0.2 S    /bin/sh /usr/local/mpich-1.2.7p1/bin/mpirun -machinefile machines1.txt -np 4 jacobi_mpi_nb 300 300
7107 72.5 R    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI6985 -p4wd /home/home_7303/c7303
7108  0.0 S    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI6985 -p4wd /home/home_7303/c7303
7109  0.0 S    rsh GD202B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 60061 \-p4amslave \-p4yourname GD202B \-p4rmrank 1
7110  0.0 S    rsh GD203B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 60061 \-p4amslave \-p4yourname GD203B \-p4rmrank 2
7111  0.0 S    rsh GD204B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 60061 \-p4amslave \-p4yourname GD204B \-p4rmrank 3
```

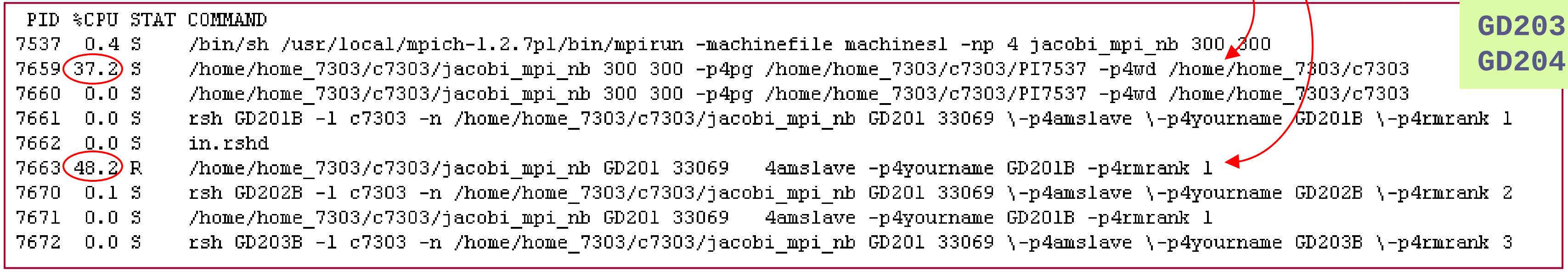
machines1

GD201B
GD202B
GD203B
GD204B

Case 3: ✘ Slower!

GD204 is idle while GD201 is time-shared by both master and slave processes

```
[GD201]$ mpirun -np 4 -machinefile machines1 jacobi_mpi_nb 300 300
```



```
 PID %CPU STAT COMMAND
7537  0.4 S    /bin/sh /usr/local/mpich-1.2.7p1/bin/mpirun -machinefile machines1 -np 4 jacobi_mpi_nb 300 300
7659 37.2 S    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI7537 -p4wd /home/home_7303/c7303
7660  0.0 S    /home/home_7303/c7303/jacobi_mpi_nb 300 300 -p4pg /home/home_7303/c7303/PI7537 -p4wd /home/home_7303/c7303
7661  0.0 S    rsh GD201B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 33069 \-p4amslave \-p4yourname GD201B \-p4rmrank 1
7662  0.0 S    in.rshd
7663 48.2 R    /home/home_7303/c7303/jacobi_mpi_nb GD201 33069   4amslave -p4yourname GD201B -p4rmrank 1
7670  0.1 S    rsh GD202B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 33069 \-p4amslave \-p4yourname GD202B \-p4rmrank 2
7671  0.0 S    /home/home_7303/c7303/jacobi_mpi_nb GD201 33069   4amslave -p4yourname GD201B -p4rmrank 1
7672  0.0 S    rsh GD203B -l c7303 -n /home/home_7303/c7303/jacobi_mpi_nb GD201 33069 \-p4amslave \-p4yourname GD203B \-p4rmrank 3
```

Debugging MPI Program

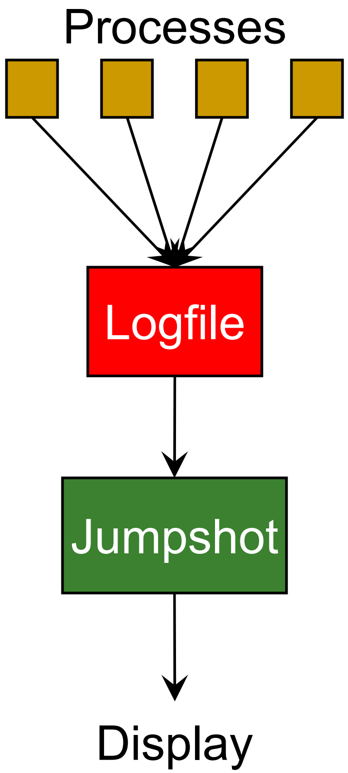
- Use “printf” (Recommended)
  - Output intermediate calculated values
  - Prefix each message printed by printf with process rank
  - Add fflush(stdout) after printf (because under UNIX, output is placed in a buffer to be printed, but not actually printed yet)
- Compile with option "**`-mpitrace`**"
  - This prints out all MPI communications executed by each process for your checking or profiling.

MPI Program Performance Visualization

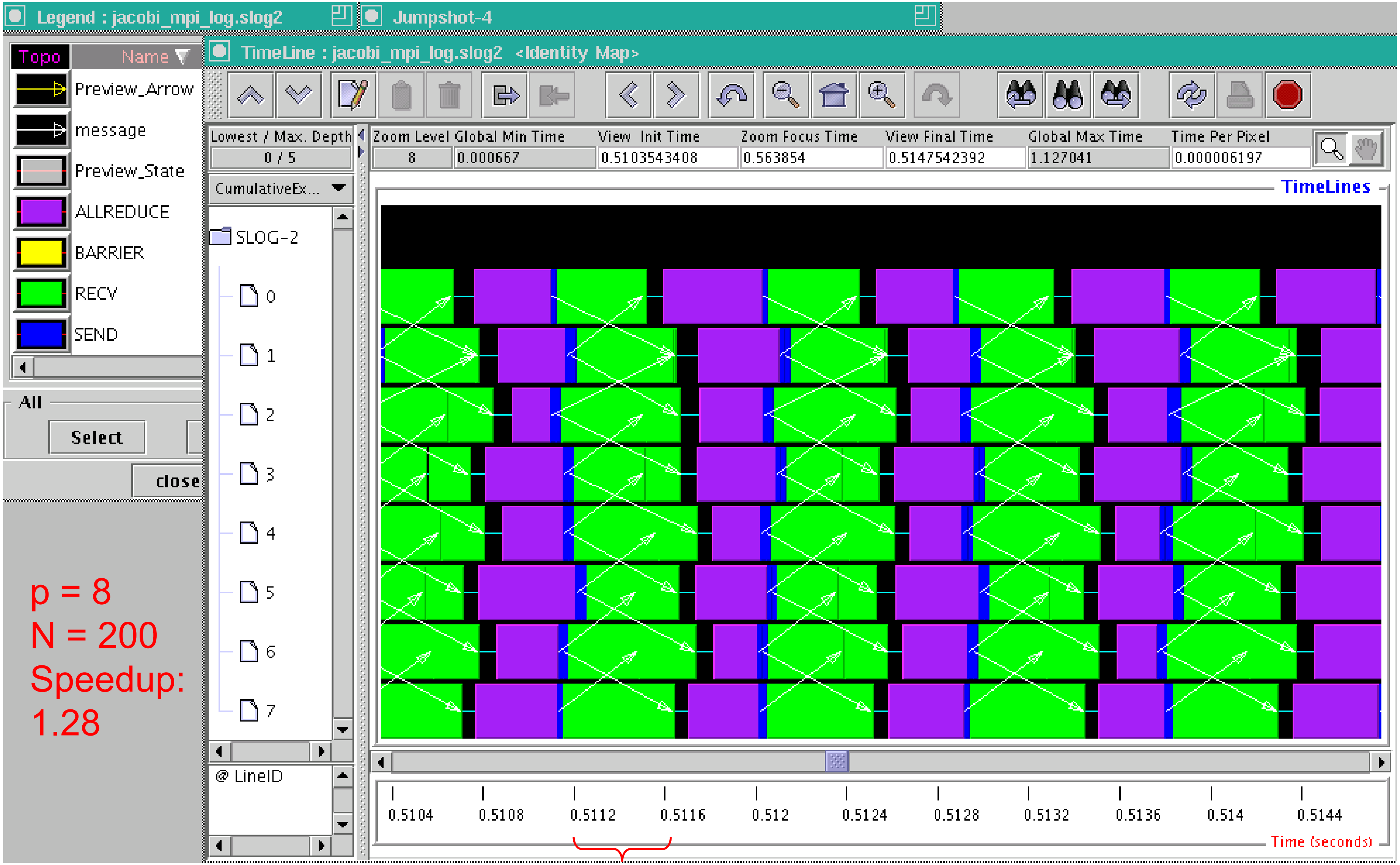
- A log file of the program's behavior will be generated when compiled with -mpilog option.

```
mpicc -o pgm pgm.c -mpilog
```

- File generated: pgm.clog
- Log file can be visualized graphically by **Jumpshot**.
- Download Jumpshot from here. Latest version is Jumpshot-4. It will convert the file from clog to slog-2 format before viewing.
- We don't allow VNC on Gideon. So please ftp your clog file to your desktop machine for viewing.
- For large *n* and *p*, running the program to the end will generate a clog file as big as several hundreds MB. So please limit your iterations to within a few hundreds. And delete the clog files asap.



Jumpshot: Standard Sends + AllReduce



Legend : jacobi_mpi_log.slog2

Topo	Name
→	Preview_Arrow
→	message
⬜	Preview_State
🟪	ALLREDUCE
🟨	BARRIER
🟩	RECV
🟦	SEND

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 5	8	0.000667	0.5103543408	0.563854	0.5147542392	1.127041	0.000006197

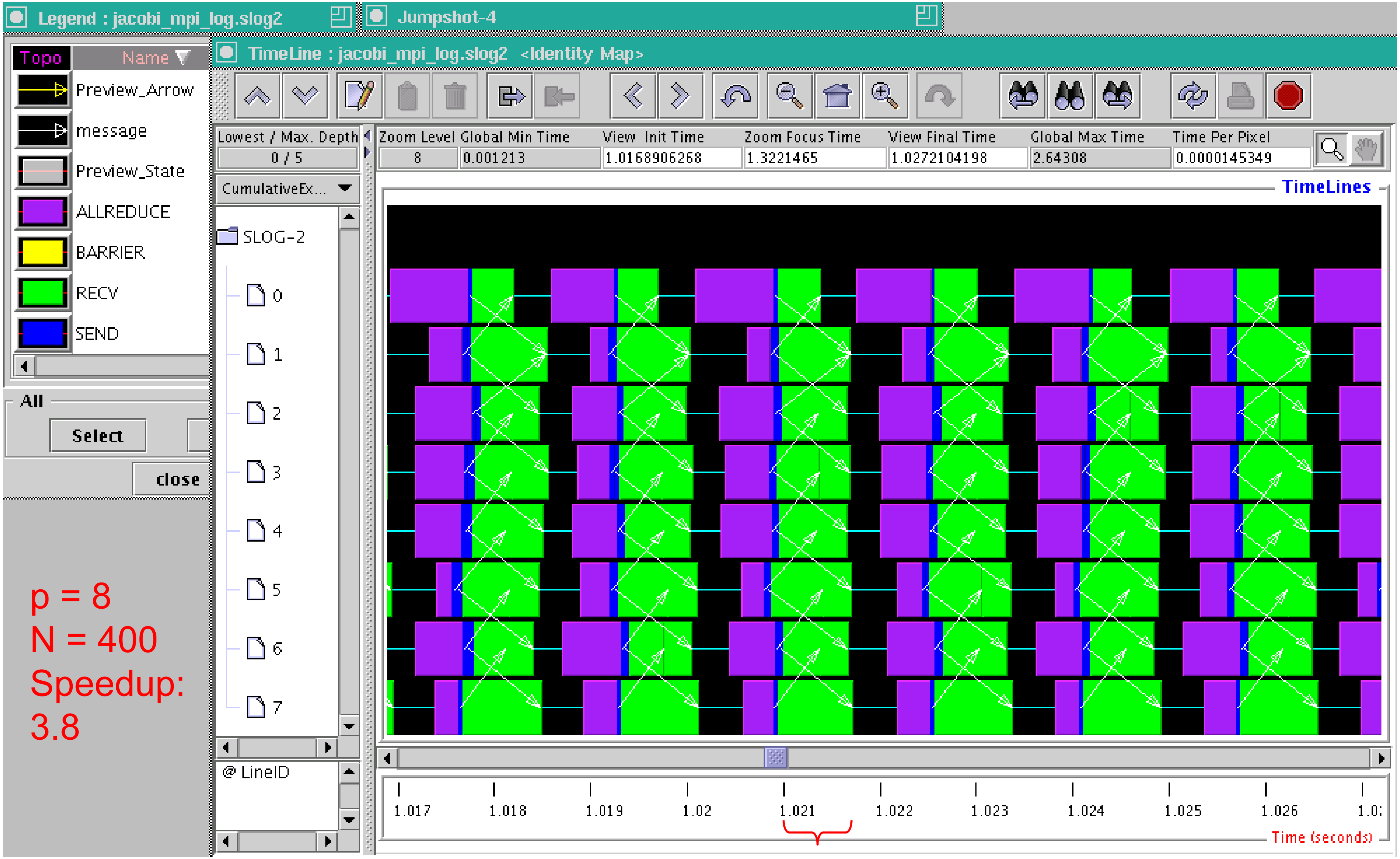
p = 8

N = 200

Speedup: 1.28

~400μs (latency for waiting 2 message arrivals)

Jumpshot: Standard Sends + AllReduce



Legend : jacobi_mpi_log.slog2

Topo	Name
→	Preview_Arrow
→	message
▭	Preview_State
🟪	ALLREDUCE
🟨	BARRIER
🟩	RECV
🟦	SEND

Jumpshot-4

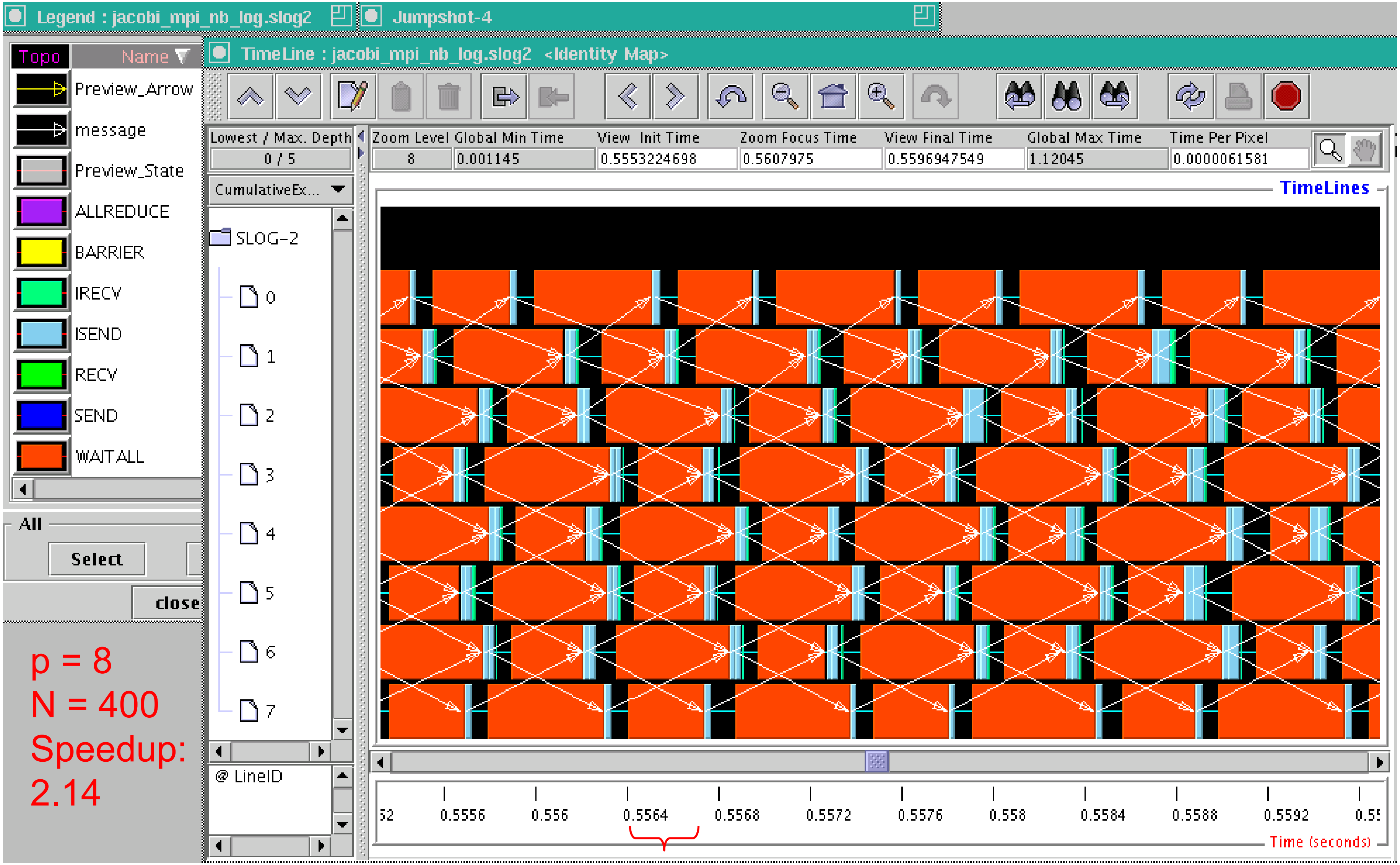
TimeLine : jacobi_mpi_log.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 5	8	0.001213	1.0168906268	1.3221465	1.0272104198	2.64308	0.0000145349

p = 8
N = 400
Speedup: 3.8

~700μs

Jumpshot: Non-blocking Communication



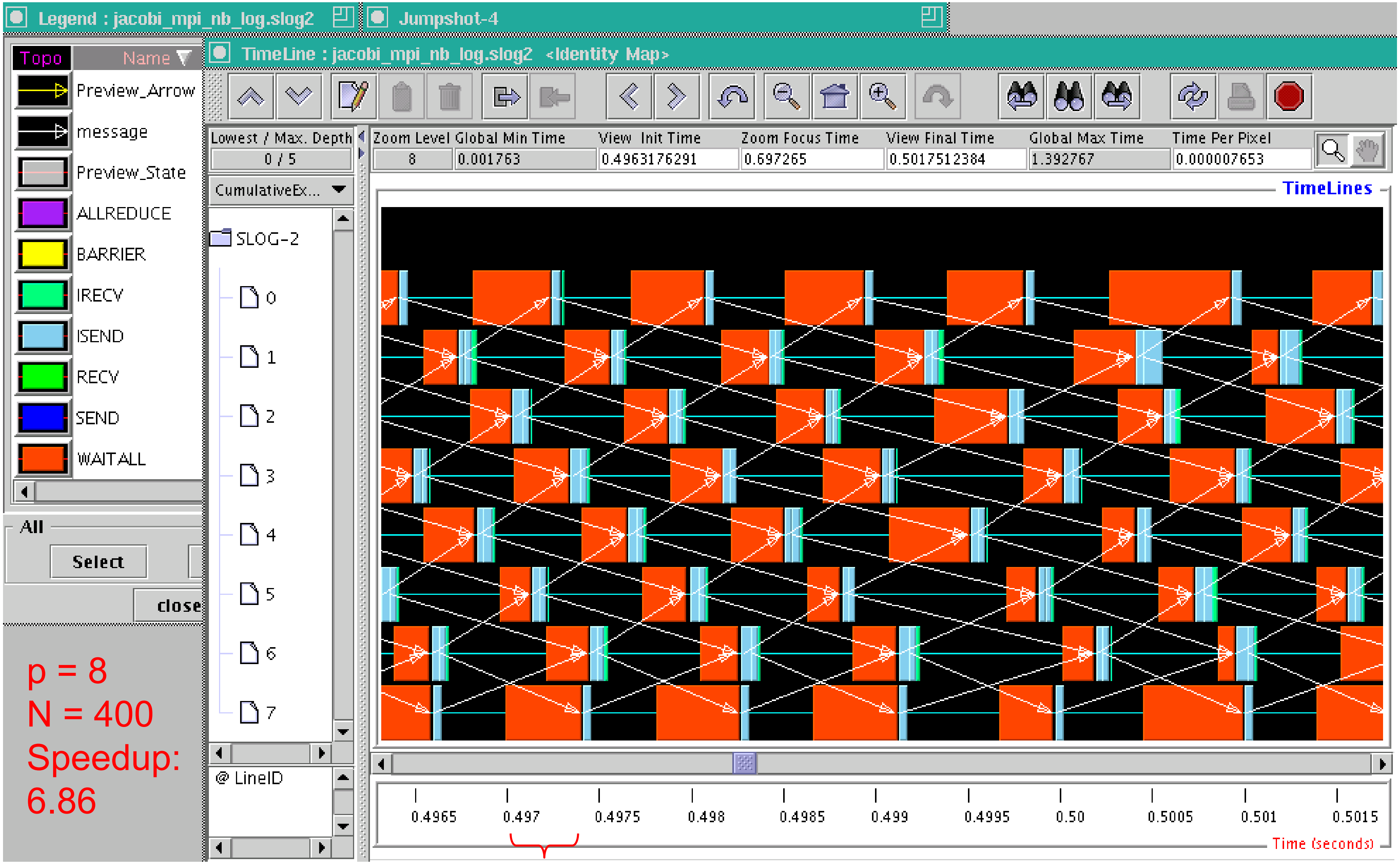
Topo	Name
→	Preview_Arrow
→	message
▭	Preview_State
🟪	ALLREDUCE
🟨	BARRIER
🟩	IRECV
🟦	ISEND
🟩	RECV
🟦	SEND
🟧	WAITALL

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 5	8	0.001145	0.5553224698	0.5607975	0.5596947549	1.12045	0.0000061581

p = 8
N = 400
Speedup: 2.14

~400-700μs (bigger variance)

Jumpshot: Non-blocking Communication



Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 5	8	0.001763	0.4963176291	0.697265	0.5017512384	1.392767	0.000007653

p = 8

N = 400

Speedup: 6.86

~100-500μs

Performance Benchmarking

- Gideon's Configuration:
  - Pentium 4 2.0 GHz w/ 512 Kbytes L2 cache
  - 512 MB (PC2100) DDR SDRAM
  - Fast-Ethernet adaptors x 2
  - Linux OS (Fedora 3, kernel 2.4.22, 32-bit)

- Reserved period: from now up to Mar 14 (> one month)
- Again, for dedicated benchmarking environment, you need to submit your job to a batch job system.
- An **OpenPBS** job server is up on **GD208**, which assigns submitted processes to the 16 benchmarking machines (GD134-149).
- You need to write an OpenPBS job script to submit your job on GD208.

Reminder:

1. Test and debug your program on development machines and submit your job running a correct program ONLY. Runaway process (for > 15 min) will be killed by our job monitor daemon.
2. There is no easy way to provide input to submitted batch job during runtime. Don't prompt for input with scanf in your program.

Sample OpenPBS Job Script

- An example C-MPI job script

```
#!/bin/bash

#PBS -l nodes=8

echo "The path to PBS machine file is $PBS_NODEFILE"
echo "The machine lists are:"
cat $PBS_NODEFILE

echo "Begin execution:"
export PATH=/usr/local/mpich-1.2.7p1/bin:$PATH
mpirun -np 8 -machinefile $PBS_NODEFILE -nolocal $HOME/jacobi_mpi
```

Ask OpenPBS to allocate 8 dedicated compute nodes for this job.

- **$PBS_NODEFILE** is a dynamically created list of the allocated machines. Please don’t use your own machine file.
- Jobs requesting fewer nodes will be scheduled first.
- Jobs requesting more resources (i.e. nodes) may wait longer (i.e. starved). Maximum starvation time is set to 30 min.

Job Script Tailored for Gideon

- The previous script has not used network “B”. The following job script is tailor-made for Gideon to use GDxxxB:

```
#!/bin/bash
#PBS -l nodes=8

echo "The path to PBS machine file is $PBS_NODEFILE"

# Wrap the allocated nodes to use network B
MY_NODEFILE="/tmp/"`basename $PBS_NODEFILE`
echo "The path to PBS machine file (network B) is $MY_NODEFILE"
cat $PBS_NODEFILE | awk '{print $1"B"}' > ${MY_NODEFILE}

echo "The machines allocated are:"
cat $MY_NODEFILE

echo "Begin execution:"
export PATH=/usr/local/mpich-1.2.7p1/bin:$PATH
mpirun -np 8 -machinefile $MY_NODEFILE –nolocal $HOME/jacobi_mpi

# Remove the wrapped file from /tmp
rm $MY_NODEFILE
```

Output from a PBS execution

- Suppose the job script file name is jacobi.job. You will see two files generated in your current directory. If you don’t see them, please check and fix your .rhosts setting.
  - <job_name>.e<job_id> stores the standard error, e.g. jacobi.job.e26
  - <job_name>.o<job_id> stores the standard output, e.g. jacobi.job.o26
- For a normal execution, <job_name>.e<job_id> will contain nothing while <job_name>.o<job_id> will contain results such as below:

```
The path to PBS machine file is /etc/PBS/aux/16.GD208
The path to PBS machine file (network B) is /tmp/16.GD208
The machines allocated are:
GD141B
GD140B
GD139B
GD138B
GD137B
GD136B
GD135B
GD134B
Begin execution:
Problem size: M=200, N=200
Converged after 30627 iterations with error: 0.00099999.
Elapsed time = 16.357088 sec.
```

Submit Jobs on Gideon Using OpenPBS Commands

These commands are to be called on GD208 only.

- Submit job:

```
qsub <job_script_file>
```

  - Job file name (including extension) should be within 15 characters for display.

- View job status:

```
qstat [-a]
```

  - Show your job’s id, script name and current status –
    R: running, Q: queued, E: exiting
  - Check which other users are also on the queue.

- Delete job:

```
qdel <job_id>
```

  - Only the job owner can do so.

A Shell Script Wrapper of qsub

- To ease your work, you can use this script to submit job by one step:

```
[GD208]$ ./jacobi_pbs.sh <num_of_proc> <pgm> <rows> <cols>
```

jacobi_pbs.sh (Shell script that fills in a job template with input arguments and call 'qsub')

```
#!/bin/sh
...
p=$1
pgm=$2
rows=$3
cols=$4
sed "s/<p>/${p}/g; s/<pgm>/${pgm}/g; s/<rows>/${rows}/g; s/<cols>/${cols}/g"
./jacobi.tmpl > ./jacobi.job
qsub jacobi.job
```

jacobi.tmpl (Job template)

```
#!/bin/bash
#PBS -l nodes=<p>
...
mpirun -np <p> -machinefile $MY_NODEFILE -nolocal $HOME/<pgm> <rows> <cols>
```

Demonstration

```
ssh –p 2004 c7303@cserver1
cat .rhosts
cat machines.txt
```

Development Stage:

```
rsh GD201
mpicc -O3 -o jacobi_mpi jacobi_mpi.c –lm
mpicc -O3 -o jacobi_mpi_nb jacobi_mpi_nb.c –lm
mpicc -O3 -o jacobi_dead jacobi_dead.c -lm
mpirun -machinefile machines.txt -np 4 –nolocal jacobi_dead 200 200
```

Note: rsh will not prompt for password only if .rhosts is set up correctly. If rsh prompts for password, then your setup is wrong. Your CS password is only for ssh but not rsh.

```
./dsh.sh machines.txt "kill -9 `ps -fu c7303 | grep amslave | grep -v grep | awk '{print $2}'`"
./dsh.sh machines.txt "kill -9 `ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]' | awk '{print $2}'`“
```

For killing all dangled processes in case of deadlock or other errors

```
mpirun -machinefile machines.txt -np 4 –nolocal jacobi_mpi 200 200
```

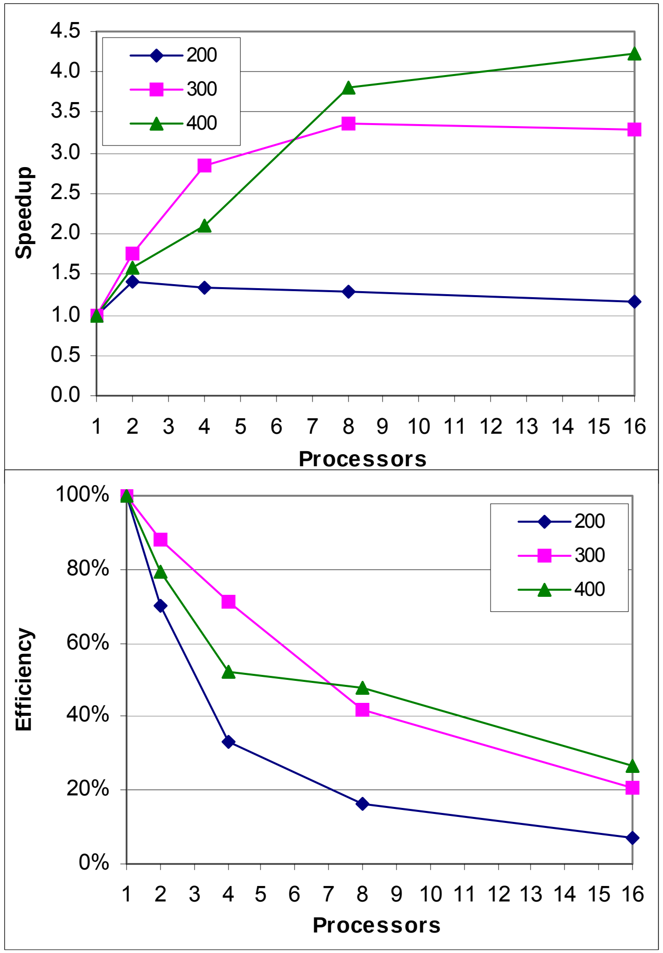
Benchmarking Stage:

```
rsh GD208
./jacobi_pbs.sh 16 jacobi_mpi_nb 200 200
qstat
pbsnodes -a
```

An OpenPBS command to check all nodes’ status

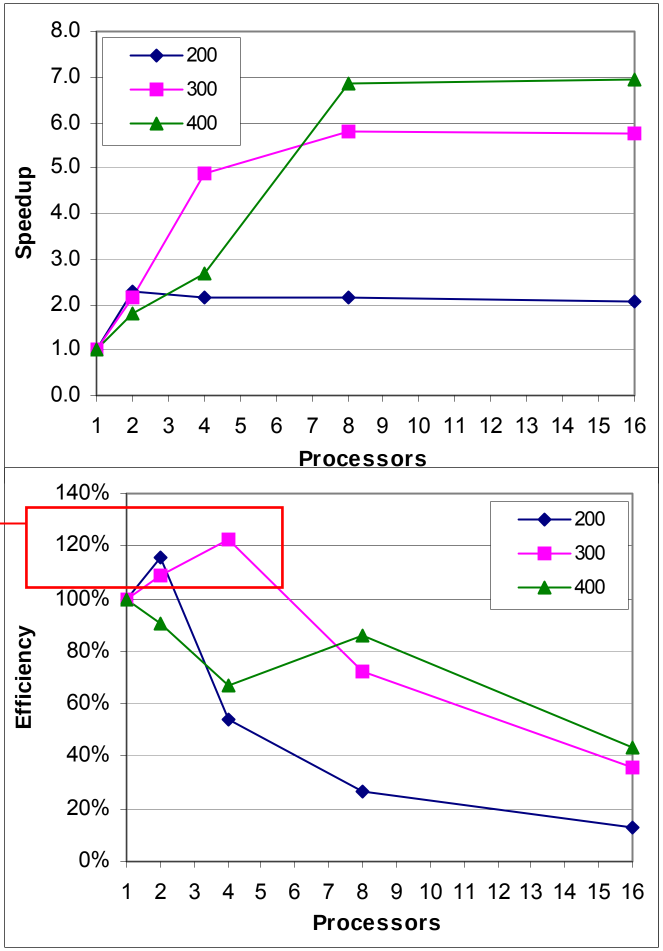
Sample Performance Results

- Average work vs. good work



MPI (Initial version)

Superlinear speedup! Why?



MPI (Enhanced version)

Final Reminder
- Things to Submit

- Deadline: **Mar 14 at 12:00 noon**
- Please submit your assignment via MSc Intranet
- Required Submission:
  - Source code of OpenMP and MPI program in C
  - Report of performance charts
- Note:
  - Place sufficient and useful comments to make your programs readable.
  - Be sure that your programs can be compiled correctly and get correct results on our platforms.

The End. Thank You!