Questions Regarding Homework

• Any questions?
• Have you looked at the Example Homework Problem?
• Do you know what to put into your Ø Design Document?
  Ø Results and Analysis Document?
• Use the turn in script in
  /home/course/cs425/public/bin/turnin
  Ø One additional requirement of turnin is a single directory
    with all associated files that must be named “turnin”.
• Don’t turn in object files or executable binaries. I will recompile/link the code anyway!!
Logistics

• Recitation
  ➢ Reassigned to Gilman 0312
  ➢ Wed 12:10 – 1:00pm
  ➢ When needed.

• Laboratory Time
  ➢ Moved due to conflicts with the TA schedule
  ➢ Now Thursday Afternoon 12:10 – 4:00pm

• Office Hour
  ➢ Ricky: Thursday’s 2:00pm
    ▪ 336A Wilhelm Hall
Logistics [2]

- Homework #2
  - See the hint on the web page about comparing two matrices.
    - The computed solution vs. the analytical one!
      - Note that there are in all three (3) computed solutions
        - Simple loop or ddot
        - daxpy (j and k loop rearranged)
        - blocked (reblocking your code from 3 loops to 6).
  - Don’t forget to compile with optimization!!
    - gcc –O or g77 –O
    - This will reduce the time you wait significantly 😊
  - The three computed solutions DO NOT REQUIRE an interface to get part of the solution!!!!!!!
Memory management for Homework #2

• The largest case requires 33Mbytes for the minimum storage implementation!!
  ➢ 1001 x 489 x 1500

• The minimum is
  ➢ A,
  ➢ B,
  ➢ C\text{comp}
  ➢ Canal

• Compute difference norm on the fly.
Blocking a loop and the side effects.

• **Blocking a loop does what?**
  - Presents the same number and order of the values of the loop index!
  - Presents them only in groups of ordered, loop indexes.
  - A loop:
    - 0, 1, 2, 3, 4, 5, 6, 7, 8, …
  - A blocked (or grouped) loop
    - 0, 1, 2
    - 3, 4, 5
    - 6, 7, 8
  - What are the side effects??
    - Really 2 loops instead of one!!
What if we have two blocked loops

- Reality is 4 loops.
- $i=0, 1, 2, 3, 4, 5, \ldots \ j=0, 1, 2, 3, 4, 5, \ldots$
- Without blocking:
  - $i=0, j=0, 1, 2, 3, 4, 5, \ldots$
  - $i=1, j=0, 1, 2, 3, 4, 5, \ldots$
  - $i=2, j=0, 1, 2, 3, 4, 5, \ldots$
- With blocking size $i=2$ size $j=3$
  - $i=0, j=0, 1, 2$
  - $i=0, j=3, 4, 5$
  - $i=1, j=0, 1, 2 \ldots$
- How does this help us??
  - It doesn’t unless we carefully arrange the loops.
Unblocked Access
Code for 2 loops and blocking (1)

```c
for (i=0;i<6;i++)
    for (j=0;j<6;j++)
        printf("normal : %d,%d \n",i,j);

bsi = 2; bsj = 3; nbi = 6/bsi + 1; nbj = 6/bsj + 1;
for(ib=0;ib<nbi;ib++) {
    ilo = ib*bsi;         ihi = MYMIN((ilo+bsi),6);
    for(i=ilo;i<ihi;i++) {
        for(jb=0;jb<nbj;jb++) {
            jlo = jb*bsj;     jhi = MYMIN((jlo+bsj),6);
            for(j=jlo;j<jhi;j++) {
                printf("blocked 1: %d, %d\n",i,j);
            }
        }
    }
}
```
Block loops as outer loops (2)

\[
\begin{align*}
bsi &= 2; \quad bsj = 3; \quad nbi = 6/bsi + 1; \quad nbj = 6/bsj + 1; \\
&\text{for}(ib=0;ib<nbi;ib++) \{ \\
&\quad ilo = ib*bsi; \\
&\quad ihi = MYMIN((ilo+bsi),6); \\
&\quad \text{for}(jb=0;jb<nbj;jb++) \{ \\
&\quad \quad jlo = jb*bsj; \\
&\quad \quad jhi = MYMIN((jlo+bsj),6); \\
&\quad \quad \text{for}(i=ilo;i<ihi;i++) \{ \\
&\quad \quad \quad \text{for}(j=jlo;j<jhi;j++) \{ \\
&\quad \quad \quad \quad \text{printf("blocked 2: \%d, \%d\n",i,j); \\
&\quad \quad \quad \} \\
&\quad \quad \} \\
&\quad \} \\
&\} \\
\end{align*}
\]
## output

<table>
<thead>
<tr>
<th>normal : 0,0</th>
<th>blocked 1 : 0,0</th>
<th>blocked 2: 0, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal : 0,1</td>
<td>blocked 1 : 0,1</td>
<td>blocked 2: 0, 1</td>
</tr>
<tr>
<td>normal : 0,2</td>
<td>blocked 1 : 0,2</td>
<td>blocked 2: 0, 2</td>
</tr>
<tr>
<td>normal : 0,3</td>
<td>blocked 1 : 0,3</td>
<td>blocked 2: 1, 0</td>
</tr>
<tr>
<td>normal : 0,4</td>
<td>blocked 1 : 0,4</td>
<td>blocked 2: 1, 1</td>
</tr>
<tr>
<td>normal : 0,5</td>
<td>blocked 1 : 0,5</td>
<td>blocked 2: 1, 2</td>
</tr>
<tr>
<td>normal : 1,0</td>
<td>blocked 1 : 1,0</td>
<td>blocked 2: 0, 3</td>
</tr>
<tr>
<td>normal : 1,1</td>
<td>blocked 1 : 1,1</td>
<td>blocked 2: 0, 4</td>
</tr>
<tr>
<td>normal : 1,2</td>
<td>blocked 1 : 1,2</td>
<td>blocked 2: 0, 5</td>
</tr>
<tr>
<td>normal : 1,3</td>
<td>blocked 1 : 1,3</td>
<td>blocked 2: 1, 3</td>
</tr>
<tr>
<td>normal : 1,4</td>
<td>blocked 1 : 1,4</td>
<td>blocked 2: 1, 4</td>
</tr>
<tr>
<td>normal : 1,5</td>
<td>blocked 1 : 1,5</td>
<td>blocked 2: 1, 5</td>
</tr>
<tr>
<td>normal : 2,0</td>
<td>blocked 1 : 2,0</td>
<td>blocked 2: 2, 0</td>
</tr>
<tr>
<td>normal : 2,1</td>
<td>blocked 1 : 2,1</td>
<td>blocked 2: 2, 1</td>
</tr>
<tr>
<td>normal : 2,2</td>
<td>blocked 1 : 2,2</td>
<td>blocked 2: 2, 2</td>
</tr>
<tr>
<td>normal : 2,3</td>
<td>blocked 1 : 2,3</td>
<td>blocked 2: 3, 0</td>
</tr>
<tr>
<td>normal : 2,4</td>
<td>blocked 1 : 2,4</td>
<td>blocked 2: 3, 1</td>
</tr>
<tr>
<td>normal : 2,5</td>
<td>blocked 1 : 2,5</td>
<td>blocked 2: 3, 2</td>
</tr>
<tr>
<td>normal : 3,0</td>
<td>blocked 1 : 3,0</td>
<td>blocked 2: 2, 3</td>
</tr>
<tr>
<td>normal : 3,1</td>
<td>blocked 1 : 3,1</td>
<td>blocked 2: 2, 4</td>
</tr>
<tr>
<td>normal : 3,2</td>
<td>blocked 1 : 3,2</td>
<td>blocked 2: 2, 5</td>
</tr>
<tr>
<td>normal : 3,3</td>
<td>blocked 1 : 3,3</td>
<td>blocked 2: 3, 3</td>
</tr>
<tr>
<td>normal : 3,4</td>
<td>blocked 1 : 3,4</td>
<td>blocked 2: 3, 4</td>
</tr>
<tr>
<td>normal : 3,5</td>
<td>blocked 1 : 3,5</td>
<td>blocked 2: 3, 5</td>
</tr>
<tr>
<td>normal : 4,0</td>
<td>blocked 1 : 4,0</td>
<td>blocked 2: 4, 0</td>
</tr>
</tbody>
</table>
Blocked 1 vs. Blocked 2
Secure Shell (ssh)

• Provides an encrypted channel between computers
  ➢ Prevents password snooping
  ➢ Allows SECURE X-window sessions

• Replaces telnet or rsh completely

• The ssh command behaves much like rsh
  ➢ ssh –l rickyk popeye.cs.iastate.edu
  ➢ ssh rickyk@popeye.cs.iastate.edu

• Is the required access mechanism for both NERSC and the SCL
  ➢ NERSC requires the –P flag for ssh
    ▪ E.g., don’t use a “privileged port”
Secure CoPy (scp)

• Provides an encrypted channel between computers
  ➢ Prevents password snooping
• Replaces ftp or rcp completely
• The scp command behaves much like rcp
  ➢ scp local_file rickyk@seaborg.nersc.gov:.
  ➢ scp –r rickyk@seaborg.nersc.gov:cs425 ./
• Is the required file copy mechanism for both NERSC and the SCL
$HOME/.ssh/config file

• Sets options for all ssh/scp commands
• So you don’t have to remember the command line arguments 😊
  ➢ One specification per line.
• My file looks like:
  UsePrivilegedPort no
  ForwardX11 yes
  ForwardAgent yes
SSH do’s

• DOs
  Ø You may use ssh-agent.
    ▪ This is a mechanism to use your secure keys with a passphrase.
    ▪ Allows you to enter a passphrase once and connect to any place you have installed your secure keys.
  Ø You should use a real passphrase except when it is necessary to use a null passphrase.
    ▪ Required on most clusters
    ▪ This is why clusters are usually self-firewalled.
SSH don’ts

• Don’ts
  ➢ Never telnet from the workstation in front of you to another computer and then ssh to another machine.
    ▪ The link from your workstation is not secure.
    ▪ Compromises the security of the systems that require ssh access.
    ▪ ISU systems now require kerberized telnet 😊
  ➢ Don’t set your password and passphrase as the same thing.
    ▪ They are intended for different purposes.
SSH clients

• Links to free windows and Unix clients are on the course web site, so you may configure ssh on your personal systems.
SSH and SSH-AGENT

- SSH uses various encryption and compression algorithms depending upon which version you actually have.
- SSH forwards X11 connection information
  - Other ports too😊
  - Often not by default! So you have to turn it on.
- SSH has a passphrase mechanism associated with computer keys.
- To generate your keys:
  - `ssh-keygen -t rsa`
    - `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`
  - `ssh-keygen -t dsa`
    - `~/.ssh/id_dsa` and `~/.ssh/id_dsa.pub`
SSH and SSH-AGENT [2]

At the beginning empty or non-existent directories
SSH and SSH-AGENT [3]

% ssh-keygen –t dsa

You will enter a passphrase
It should be different from your password

CYASIFO::$HOME/.ssh

id_dsa.pub

id_dsa

CYW2A ::$HOME/.ssh
SSH and SSH-AGENT [4]

% cd $HOME/.ssh
% scp identity.pub CYW2A::ssh/authorized_keys
password:

CYASIFO::$HOME/.ssh
id_dsa.pub
id_dsa

CYW2A ::$HOME/.ssh
authorized_keys
SSH and SSH-AGENT [5]

% ssh-agent tcsh
% ssh-add
    passphrase:
% ssh-add -l
adfa987425145adf0982435 mylogin@CYASIFO
% ssh CYW2A
% prompt@CYW2A:

CYASIFO::$HOME/.ssh
    id_dsa.pub
    id_dsa

CYW2A ::$HOME/.ssh
    authorized_keys
CS System Access

• To come in from anywhere other than the *.iastate.edu domain you must:
  
  ➢ Generate a set of SSH keys on your system.
    ▪ Or any system you plan to access the CS systems with.
  ➢ Install your SSH public key on the CS systems
  ➢ This requires access to the CS systems to install your keys
    ▪ You have to go to Atanasoff 116A and move them.
  ➢ They don’t trust ameslab.gov 😊
    ▪ How Rude (Quoting Jar-Jar Binks)
  ➢ They also don’t trust “msn.com” or “mchsi.com”
How to install Ricky’s Keys

% > cd $HOME/.ssh
% > wget http://www.cs.iastate.edu/~cs425/rickys-keys
IFF you don’t already have an authorized_keys file.
  % > mv rickys-keys authorized_keys
OR
IF you do have an authorized_keys file
  % > cat rickys-keys >> authorized_keys
This will allow me to help with homework if you have issues!
Programming Parallel Computers

• **Essentially 4 ways to attack the issue**
  1. Extend an existing compiler to translate sequential programs into parallel programs
  2. Extend an existing language with new operations to express parallelism
  3. Add a new parallel language layer on top of an existing sequential language.
  4. Define a new language and compiler system.
Extending a Compiler

• **Automagic Parallelizing Compilers.**
  - Still in it’s infancy after 20 years of research.
  - Still a research topic and lots of good work is being done in this area.
    - M. Gupta @ U. IL
    - W. Cohen Ph.D. work at Purdue
    - S. Kothari @ ISU
    - Parallel Software Products
    - SGI and other vendors
  - Often focused at Shared Memory systems
    - OpenMP
  - Distributed Memory is mostly Data Parallel
  - Requires a thorough understanding of how the compiler works so code can be constructed in such a way the compiler knows what to do with it.
Extending a Language

- The most popular approach
  - OpenMP, MPI, SHMEM, Global Arrays, and Pthreads.
- Need to create, terminate, synchronize, and communicate among processes involved.
- Compiler is not involved so there is no “correctness” checking at that stage.
  - Library calls
- Debugging is difficult
- The process is error prone
  - Programmers make mistakes
  - Usually only detectable at runtime.
- The focus of this class
Add a Parallel Programming Layer

• **Dual language mode**
  - Upper layer is a parallel aware language (IDE)
  - Lower layer is a normal sequential language.

• **Compiler puts the pieces together into a parallel application**

• **Still in the research phase.**

• **Based on “component” ideas.**
  - The big bang after Object Oriented programming.
Create a New Parallel Language

• A language from scratch or an addition to a current language by adding parallel constructs.
  - Sisal, Occam, Unified Parallel C, Co-Array Fortran, High Performance Fortran
• Transitioning from research quality to production quality.
  - A few applications are written in these languages.
• Totally new languages must be able to deal with legacy code somehow
  - Ability to call FORTRAN/C/C++ modules
  - Translation facilities
• New languages are hard to get into the community because of the 1000s of man-years that exists in current applications.
  - Just rewriting it is not an option because of manpower limitations.
How do you write applications for these MIMD-DM systems?

- **SPMD**
  - Single Program Multiple Data
    - Flynn-ism
  - Do something different based on Process or Thread ID.

- The same program executes on all processors.
  - Via processes, threads, or both.

- The method and speed of addressing memory on each node is the key to performance.
Memory Hierarchy on a Node

- **Speed and Cost** vs. **Size**
- Registers
- Cache(s)
- Main Memory
- Virtual Memory
Unifying Theme

• The only difference in parallel computer systems is the access to the memory hierarchy of the system.

• Non-Uniform Memory Access
  ➢ Applies to both Shared Memory and Distributed Memory MIMD parallel supercomputers.
  ➢ Not only speed (Bandwidth & Latency) but the method of access as well!
MPP Memory Hierarchy

- Registers
- Cache(s)
- Main Memory
- Virtual Memory
- Remote Main Memory
- Remote Virtual Memory

Speed vs. Size graph.
Programming Models

- **Shared Memory**
  - process based
  - thread based (OpenMP, Pthreads)
- **Message Passing**
  - MPI, MPI-2, PVM etc.
- **Combined Message Passing and SMP**
  - MPI and OpenMP
- **Globally Addressable (NUMA)**
  - Global Arrays
  - Unified Parallel C
  - Co-Array Fortran
Accessing Memory

• **Shared Memory:**

• **Message Passing**
  - Send ($B$); Receive($B$); Send($C$); Receive($C$);

• **Combined Message Passing and Shared Memory**
  - **BUT** you must know where the data is!

• **Globally Addressable (NUMA)**
  - get($B$); get($C$);
  - put($A$);
Supercomputers

• Integrate many resources into a single system
• Employ at least some bleeding edge technology
  ➢ First of a kind
  ➢ One of a few
• Have the highest performance or
• Offers the promise of the highest performance
  ➢ What determines the reality?

Applications and utilization
Scalable Architectures

• A computational resource is scalable if it can expand to accommodate additional demands.
  ➢ Better performance and/or more functionality.
    ▪ A linear increase in power with an increase in resources.
  ➢ Both hardware and software
  ➢ Reasonable cost scaling
    ▪ A factor of N in scaling should require a linear or NlogN cost scaling
    ▪ Can also scale down with a reduced cost
Scalable Architectures (2)

• A resource is scalable …

  ➢ Compatibility

  ▪ The same software, hardware, applications should work as the system grows.
  ▪ Best done scaling all resources but should work just adding a single resource
    ♦ A processor upgrade
    ♦ A memory upgrade
    ♦ Which is the more costly of these two?
Resource Pyramid

IBM Blue Gene

Handful of Supercomputers

Mainframes/Site mini-supercomputers

Group Servers (SMP) and clusters

Stand-alone workstations

Millions of Personal Computers

Billions of embedded systems (Cars, Refrigerators, ATMs etc.)
A Scalable Resource

• Advocates a common architecture along the pyramid.
  ➢ Satisfies multiple performance cost requirements
    ▪ Can always scale up when demand increases
  ➢ High End machines use low-end components
    ▪ Lots of them
    ▪ Compatibility from desktop to teraflop
  ➢ Bleeding Edge technology migrates down the pyramid over time.
    ▪ Cost of disks over the last 3 years
    ▪ Cost of memory
    ▪ Raw processor speed
Architecture Nomenclature

• Macro-architecture
  ➢ The overall structure of the computational components
    ▪ The “multiple-node” or repeated unit architecture.
    ▪ The interconnect for the nodes
    ▪ Multiple system images (single image in special cases).

• Micro-architecture
  ➢ The structure of a single instance of a computational component.
    ▪ The node itself.
    ▪ SMP or processor
    ▪ Always a single system image
Dimensions of Scalability

- Resources
- Applications
- Technologies
Resource Scalability

• Gaining higher performance or functionality by increasing machine size.
  ➢ Processors - size - performance
  ➢ Memory - size - performance
  ➢ Disks - size - performance
  ➢ Software - functionality
Resource Scalability

• All are interdependent
  - More processors/nodes may require higher communications bandwidth
  - More disk may require external physical storage space?
  - More memory may not be possible due to physical space in the nodes
  - A better interconnect may require an OS upgrade.
    - Extra $$$
Software Scalability

• A new Operating System
• New Batch Scheduling Algorithms
  ➢ Increased utilization of the system
• Better Optimizing Compilers
• More efficient Math/Eng. Libraries
• Efficient easy-to-use Applications
  ➢ Build it they will come???
• More developer-friendly programming Environments.
  ➢ Users don’t care about the programming environment!!!
What limits Scalability

• **Programming**
  - The ease of expression of algorithms
  - Most computer languages have been hacked to work in parallel
    - Via extensions to current languages
      - OpenMP, Vector Operations (F90/F95)
    - Via library systems that interact with languages.
      - MPI, GA, Pthreads,

• **Communication**
  - Getting the data where the available processing power lives. (The trick)
Traditional Fault Rate

“Bathtub”
Computer Usability (ideal)
Computer Usability (reality)
Application Scalability

• A scalable application should run proportionally better on a scaled up system.
  - Problem Size
    - Larger data sets
  - Machine Size
    - More processors, memory, disk
  - The algorithm or multiple algorithms mitigate both measures of application scalability.

• Scalable Applications utilize both computer technology AND algorithmic enhancements.
Technology Scalability

• **Generation or time**
  - How components change in time

• **Space**
  - Physical size
    - MPPs can only fit in the room (limited)
    - The internet fits the planet. (scalable or not?)

• **Heterogeneity**
  - Integration of different kinds of components.
  - Requires standards, well defined interfaces
    - Open Architecture
  - Most are homogenous systems
Generation or Time Scalability

- **Five sub-systems of any computer:**
  - **CPU**
    - Doubles in speed every 18 months (Moore’s Law)
  - **Memory**
    - Doubles in size/density every 3 years
  - **Communication (bus, switch, network)**
    - Doubles in bandwidth and reduced latency every 3 to 10 years.
  - **Secondary Storage (disks)**
    - Doubles in size every 2 to 3 years
    - Doubles in bandwidth every 5 to 10 years.
  - **Software (OS, Compilers, Languages)**
    - Improves over 1 or two generations of other components.
    - Sometimes rewritten for each generation.
Application Usability (ideal)

The Perfect Application
Application Usability (reality)

Time

- Shift in technology
- The Solid Application

Usability
Combined Computer/Application Usability (reality)

Shift in technology

The Semi-Solid Application e.g., Glass Butter
Combined Usability (reality)

- **A scalable application**
  - Will outlive any computer!
  - Must be portable to the next generation system
  - Multiple algorithms for the same computable property may be needed.
  - Algorithms should be “tunable” to mitigate different computer resource strengths and weaknesses.
Application Tradeoffs

- Algorithms can be tuned by making tradeoffs between computer sub-systems
  - The block size in the Example Homework Problem is a “tunable” parameter.
  - Trade reduced computation for storage
  - Trade reduce storage for computation
  - Trade reduced communication for storage.
  - Trade increased computation for better communication performance
Application Tradeoffs Example

• An iterative process that multiplies a large data set with a varying matrix until the variation is minimized.

➢ The large data set has several choices.
  ▪ Compute once and store in memory / disk
  ▪ Fill memory recompute the rest
  ▪ Fill memory with the most expensive to compute elements and recompute the rest.
  ▪ Recompute at every iteration.