

Homework #4: Midterm practice questions

Due in class on November 1, 2005

Worth 20 points

1. **Message passing [2 pts]**. You are given 2 message passing programs. The programs contain conditional statements so that each process may execute a different sequence of message passing calls.

- a) You run the first program with 3 processes. Each process executes the code sequence as shown.

Process P1	Process P2	Process P3
Send(ONE,P3)	Send(TWO,P3)	Recv(s) Recv(t)

Assume that all free variables are scalars. `Send(x,P3)` is a blocking primitive which sends a single data element `x` to processor `P3`. `Recv(s)` is a blocking primitive that receives a message from *any* processor and stores the data element in `s`. All processes use the same tag and the same communicator. When the program has completed, what are the contents of `s` and `t`?

- b) You next run a program on two processors that uses immediate sends `ISend()`. Assume that `x`, `y`, `u`, `v` are long arrays all with the same length.

Processor P1	Processor P2
ISend(x,P2) x[1] = -x[1] ISend(y,P2)	Recv(u) Recv(v)

What are the contents of `u` and `v` on processor P2 when the program terminates?

2. **Communication performance [4 pts].** We've used the simple α - β model of communication performance, in which the time to pass a message of length n is given by the formula $\alpha + \beta n$. For "very short" messages, the transmission time reduces to the *message start time* α . Upon closer examination, we may divide α into the *sender side overhead* o_s , the *receiver side overhead* o_r , and *latency* L , which is the time taken for the message to pass through the interconnect. On many systems, $\alpha = o_s + L + o_r$. This quantity is also known as the end-to-end latency or *EEL*.

Answer the following questions.

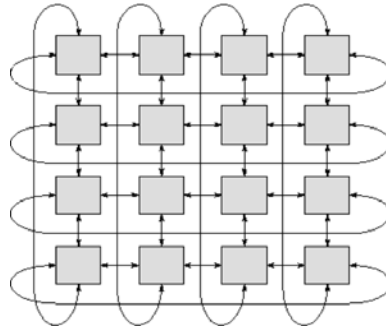
- If we pass messages back and forth between two processors for a total of t cycles, then the total communication time should be $2t\alpha$. However, on some systems we observe a noticeably shorter message passing time. Give a plausible explanation for the phenomena. Assume that there is no other message passing activity or any other form of contention in the processors or interconnect.
- You run another program on 2 processors in which the processes exchange data using immediate sends and receives. The program carries out some computation before waiting on completion of the communication:

```
IRecv(Buffer, Proc);
ISend(Data, Proc);
for (i=0; i<n; i++)
    SomeComputation;
Wait for the send and receive to complete;
```

Assume that **Buffer** and **Data** are arrays of the same length. When **n** is sufficiently small, and keeping the arrays fixed in size, the running time doesn't change significantly when we remove the **for** loop. What is causing this behavior and what is the constraint on **n**?

- When you run the computation loop alone (that is, without any communication), you notice that the computational loop speeds up. Why is this the case?
- If you reduce the size of **Buffer** and **Data** significantly in part (b), then removing the **for** loop always reduces the running time, no matter what the value of **n**. What is happening?

3. **Interconnect [4 pts].** A k -ary d -cube is an interconnection network with k^d nodes, and is a generalization of the mesh and hypercube interconnects. There are k nodes along all d axes and there are end around connections. A 4-ary 2-cube is pictured below.



Now, derive the following quantities for the general case of a k -ary d -cube, expressing all quantities in terms of k and d . Be sure to show your work.

- a. How many links are there?
- b. What is the diameter?
- c. What is the bisection bandwidth (assuming a link bandwidth of B)?
- d. What is the broadcast time for short messages, assuming a message start time α ?

4. Speedup [3 pts]

- a) What is the difference between ordinary speedup and isoefficient scaled speedup?
- b) Write the expression for *isoefficient* scaled speedup in terms of P , W , and $T_p(w,P)$, where $T_p(w,P)$ is the parallel running time for a computation with $W=w$. Assume that the isoefficiency function is $\Theta(P \lg P)$.
- c) Using the expression in (b), show how isoefficient scaled speedup overcomes the alleged limitations articulated by Amdahl's law.

5. **Algorithms [3 pts]**

- a. Come up with an efficient algorithm for total exchange (all to all communication) on long messages, giving the expected running time.
- b. Repeat for allreduce, but for *short* messages. Recall that an all reduce provides all processes with a copy of the result.
- c. Why is Canon's algorithm more efficient than the naive algorithm for performing matrix multiply? Are there any shortcomings of Canon's algorithm?

6. **Performance modeling [4 pts].** You are running a 3d version of the iterative solver `jac3d` on a local parallel computer, and you have a friend who is running the same computation on an identical system connected via the internet. You both decide to use a grid enabled version of MPICH to permit the two machines to solve a problem that is twice as large as either of you could run on a single machine. Make the following assumptions: the machines and network are dedicated to your job, the bandwidth connecting the two machines is much less than the bisection bandwidth inside either machine, and messages are long. Now, answer the following questions.

- a. You have a choice of two possible ways of increasing the problem size $N \times N \times N$.
 - i. increase all the dimensions x, y, z equally to retain a cubical problem domain
 - ii. increase one dimension only (slowest varying in memory), resulting in an elongated domain of size $N \times N \times 2N$.

Which do you prefer? Be sure to justify your answer, showing your work. Express any square or cube roots without simplifying them, e.g. $3^{1/3}$ rather than 1.259...

- b. Under ideal conditions we could run the larger problem on the two machines in the same time taken to run the original problem on one. However, communication overhead costs will result in a running time that is longer than the ideal time. You settle on the following performance constraint: *the completion time for the larger problem running on both machines must not be greater than 1.2 times the completion time for the smaller problem running on a single machine.*

Assume that you run with the elongated domain (which may or may not be optimal as determined in your previous answer). Let $\gamma n_0 n_1 n_2$ be the computational work for a problem of size $n_0 \times n_1 \times n_2$, where γ is the grind time to update a single point. What is the minimum bandwidth $1/\beta$ required of the network connecting the two machines so that they can meet the above performance constraint? Assuming full duplex communication between machines (each machine may simultaneously transmit and receive), express β as a function of N and γ .