CS 258 Parallel Processing
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Spring 1995 D. E. Culler
Handout 2

Assignment 1

Due Friday, Feb 3

This assignment includes some fairly straight-forward problems to check that you a up on the material in chapter 1. You are welcome to do them alone or with a friend. It should take a couple of hours. Please bring your solutions to class on friday.

Technology Growth Trends

uP-Name	Year	Transistors	Clk Rate	Die Size (mm^2)
i4004	1971	2300	0.5	9
i8008	1972	3500	0.8	12.25
i8080	1974	5000	3	20.25
i8085	1976	6500	5	25
i8086	1978	20000	8	6
i80286	1982	130000	10	64
i80386	1985	275000	16	90.25
i80486	1989	1200000	33	169
Pentium	1993	3300000	66	
i960CA	1990	575000		
i860xp	1991	2500000	50	
M6800	1974	5000	1	25
M68000	1979	68000	12.5	43.56
M68010	1982	69000	12.5	
M68020	1984	180000	25	84.64
M68030	1987	300000	33	
M68040	1990	1200000		
M88100	1989	1200000	33	
MIPS R3010	1988	75000	25	
MIPS R4000	1992	1100000	100	
MIPS R4400	1993	2300000	150	
R10000	1995.5	5900000	200	298
DEC 21064	1992	1700000	150	
DEC 21164	1994	9300000	300	298
Sparc	1990		33	

Table 0-1 Design parameters for several recent microprocessors

uP-Name	Year	Transistors	Clk Rate	Die Size (mm^2)
Supersparc	1993	3100000	50	
UltraSparc	1995	3800000	167	315
PA7100	1992	850000	100	
IBM PPC601	1993	2800000		
IBM PPC620	1995	6900000	167	311

Table 0-1 Design parameters for several recent microprocessors

Exercise 0.1 Compute the annual growth rate in number of transistors, die size, and clock rate by fitting an exponential to the available data in Table 0-1.

Exercise 0.2 Compute the annual performance growth rates for Table 0-2P

Table 0-2

Year	Machine	SpecInt	SpecFP	Linpack	n=1000	Peak FP
Sun 4/260	1987	9	6	1.1	1.1	3.3
MIPS M/120	1988	13	10.2	2.1	4.8	6.7
MIPS M/2000	1989	18	21	3.9	7.9	10
IBM RS6000/540	1990	24	44	19	50	60
HP 9000/750	1991	51	101	24	47	66
DEC alpha AXP	1992	80	180	30	107	150

Programming Models

Exercise 0.3 In message passing models each process is provided with a special variable or function that gives its unique number or rank among the set of processes executing a program. Show that this is not necessary for a shared memory program by using fetch&add to assign each process a unique number. Canyou determine the number of processes comprising a shared memory parallel program in a similar way?

Exercise 0.4 To move an n-byte message along H links in an unloaded store-and-forward network takes time $H\left(\frac{n}{W}+R-1\right)$, where W is the raw link bandwidth, and R is the routing delay per hop. In a network with cut-through routing this takes time $\frac{n}{W}+(H-1)R$. Consider a 64 node grid consisting of 40 MB/s links, routers with 250ns of delay. What is the minimum, maximum, and average time to move a 64 byte message through the network? A 246 byte message?

Exercise 0.5 Consider a simple 2D finite difference scheme where at each step every point in the matrix updated by a weighted average of its four neighbors, $A[i,j] = A[i,j] - w(A[i-1,j] + A[i+1,j] + A[i,j-1] + A[i,j+1])$ All the values are 64-bit floating point nuimbers. Assuming one element per processor and 1024x1024 elements, how much data must be communicated per step? Explain how this computation could be mapped onto 64 processors so as to minimize the data traffice. Compute how much data must be communicated per step.

Latency and bandwidth

At the core of our communication cost equation is a familiar linear model of the time for a data transfer as a function of the transfer size. This is often used in empirical evaluations of the performance of data transfers between components of the system: for example moving data between cache and memory or across a network. Transfers of various sizes are used to compute the asymptotic peak bandwidth and the start-up cost, according to the following formula.The overhead, latency, and overlap contribute to the start-up costs.

$$\text{ne}(n) = \text{Startup} + \text{Transfer}(n), \quad \text{wh}$$
$$\text{nsfer}(n) = \frac{n}{\text{Peak Bandwidth}} \qquad \text{(EQ 0.1)}$$

If we consider loading a cache line from memory the transfer time is the time to actually transmit the data across the bus. The start-up includes the time to obtain access to the bus, convey the address, access the memory, and possibly to place the data in the cache before responding to the processor. However, in a modern processor with dynamic instruction scheduling, the overhead may include only the portion spent accessing the cache to detect the miss and placing the request on the bus. The memory access portion contributes to latency, which can potentially be hidden by the overlap with execution of instructions that do not depend on the result of the load.

Exercise 0.6 Suppose we have a machine with a 64-bit wide bus running at 40 MHz. It takes 2 bus cycles to arbitrate for the bus and present the address. The cache line size is 32 bytes and the memory access time is 100ns. What is the latency for a read miss? What bandwidth is obtained on this transfer?

Exercise 0.7 Suppose this 32-byte line is transferred to another processor and the communication architecture imposes a start-up cost of 2µs and data transfer bandwidth of 20MB/s. What is the total latency of the remote operation?

If we consider sending an n-byte message to another processor, we may use the same model. The start-up can be thought of as the time for a zero length message; it includes the software overhead on the two processors, the cost of accessing the network interface, and the time to actually cross the network. The transfer time is usually determined by the point along the path with the least bandwidth, i.e., the bottleneck.

Exercise 0.8 Suppose we have a machine with a message startup of 100µs and a asymptotic peak bandwidth of 80MB/s. At what size message is half of the peak bandwidth obtained?

Exercise 0.9 Derive a general formula for the“half-power point” in terms of the start-up cost and peak bandwidth.

The model makes certain basic trade-offs clear. For example, longer transfers take more time, but obtain higher bandwidth because the start-up cost is amortized over more data movement. This observation can help guide design trade-offs, at least up to a point where the collection of data transfers interact to increase the start-up cost or reduce the effective bandwidth.

In some cases we will use the model in Equation 1.2 for estimating data transfer performance based on design parameters, as in the examples above. In other cases, we will use it as an empirical tool and fit measurements to a line to determine the effective start-up and peak bandwidth of a portion of a system. Observe, for example, that if data undergoes a series of copies as part of a transfer

Exercise 0.10 Assuming that before transmitting a message the data must be copied into a buffer. The basic message time is as in Exercise 1.7, but the copy is performed at a cost of 5 cycles per 32-bit words on a 100 MHz machine. Given an equationfor the expected user-level message time. How does the cost of a copy compare with a fixed cost of, say, entering the operating system.

Exercise 0.11 Consider a machine running at 100 MIPS on some workload with the following mix: 50% ALU, 20% loads, 10% stores, 10% branches. Suppose the instruction miss rate is 1%, the data miss rate is 5%, the cache line size is 32 bytes. For the purpose of this calculation, treat a store miss as requiring two cache line transfers, one to load the newly update line and one to replace the dirty line. If the machine provides a 250 MB/s bus, how many processors can it accomodate at 50% of peak bus bandwidth? What is the bandwidth demand of each processor?

The scenario is Exercise 1.10 is a little rosy because it looks only at the sum of the average badwidths, which is why we left 50% headroom on the bus. In fact, what happens as the bus approaches saturation is that it takes longer to obtain access for the bus, so it looks to the processor as if the memory system is slower. The effect is to slow down all of the processors in the system, thereby reducing their bandwidth demand. Let's try a analogous calculation from the other direction.

Exercise 0.12 Assume the instruction mix and miss rate as in Exercise 1.10, but ignore the MIPS, since that depends on the performance of the memory system. Assume instead that the processor run as 100 MHz and has an ideal CPI (with an perfect memory system). The unloaded cache miss penalty is 20 cycles. You can ignore the write-back for stores. (As a starter, you might want to compute the MIPS rate for this new machine.) Assume that the memory system, i.e., the bus and the memory control is utilized throughout the miss. What is the utilization of the memory system, U_1, with a single processor?

Assume that the miss penalty increases with the number of processors, n, placing a load on the memory system according to the following formula. Compute the MIPS rate and total utilization of the memory system as processors are added.

$$Penalty_n = Penalty_1\left(1 + \frac{nU_1}{1 - nU_1}\right)$$