Lecture 24: Board Notes: Introduction to Parallel Processing

Part A:

Users can see a system that alternates between 2 states of delivered service:

1. Service accomplishment: where the service is delivered as specified
2. Service interruption: where the delivered service is different from specified

Transitions from state 1 to state 2 are caused by *failures*, transitions from state 2 to state 1 are called *restorations*.

- Failures can be permanent or intermittent

Reliability is a measure of the continuous service accomplishment … or equivalently, the time to failure.

- *Mean Time To Failure (MTTF)* is a reliability measure
- *Annual Failure Rate (AFR)* is the percentage of devices that would be expected to fail in a year for a given MTTF
- Service interruptions are measured by the *Mean Time to Repair (MTTR)*
- *Mean Time Between Failures (MTBF)* is the sum of MTTF and MTTR
- Availability is a measure of service accomplishment with respect to the alternation between the two states of accomplishment and interruption
    - Availability = MTTF / (MTTF + MTTR)

To increase MTTF, can improve the quality of the components or design systems to continue operation in presence of components that have failed

- A failure in a component may not lead to a failure of the system

To make distinction clear, term “fault” used to mean failure of a component:

Here are 3 ways to improve MTTF:

- Fault avoidance:
    - Prevent fault occurrence by construction
- Fault tolerance:
    - Using redundancy to allow the service to comply with the service specification despite faults occurring, which applies primarily to hardware faults.
- Fault forecasting:
    - Predicting the presence and creation of faults, which applies to HW and SW faults, allowing the component to be replaced before it fails

Question:

- Assume that you have a system that uses 10000 disks
- The MTTF is 1,200,000 hours
- The disks are used 24 hours a day
- If a disk fails, you replace it with one that has the same reliability characteristics
- How many disks fail per year?

Failed Disks: (10000 drives) x (8760 hours / drive) / (1,200,000 hours/ failure) = 73

Thus, the AFR is 0.73%

But if in a supercomputing system, what if an entire computation must halt to replace???

Part B:

We have considered CPI largely from an architectural perspective (e.g. how does some datapath enhancement or new instruction affect the average numbers of clock cycles per instruction?). However, we can modify the CPI formula to consider the impact of communication in MIMD machines.

$$CPI = CPI_{base} + (\text{Remote Request Rate}) \times (\text{Cost})$$

Assume that:

- $CPI_{base} = 0.4$
- A remote request takes 400 ns
- The clock cycle time is 0.33 ns (3 GHz clock rate)
- The remote request rate is 0.2%

What is the impact on CPI?

First, figure out the penalty in terms of CCs:

- 400 ns / 0.33 ns = 1200 CCs

Second, calculate a new CPI:

- 0.4 x (0.002 x 1200) = 0.4 + 2.4 = 2.8!

Third, consider impact on performance:

- 2.8 / 0.4 = 7
- (Our system needs 7 processors to “break even”)

Part C:

Consider a multi-core processor with heterogeneous cores: A, B, C and D where core B runs twice as fast as A, core C runs three times as fast as A and cores C and A run at the same speed (i.e. have the same processor frequency, micro architecture etc). Suppose an application needs to compute the square of each element in an array of 256 elements. Consider the following two divisions of labor:

(a)
Core A	32 elements
Core B	128 elements
Core C	64 elements
Core D	32 elements

(b)
Core A	48 elements
Core B	128 elements
Core C	80 elements
Core D	Unused

Compute (1) the total execution time taken in the two cases and (2) cumulative processor utilization (Amount of total time the processors are not idle divided by the total execution time). For case (b), if you do not consider Core D in cumulative processor utilization (assuming we have another application to run on Core D), how would it change? Ignore cache effects by assuming that a perfect prefetcher is in operation.

Option 1:

We can normalize all times to Time A.

- Time B = Time A / 2
- Time C = Time A / 3
- Time D = Time A

Total execution time = max(32 / 1, 128 / 2, 64 / 3, 32 / 1)
= max(32, 64, 21.3, 32)

Max is 64 time units.

Utilization: ((32/1) + (128/2) + (64/3) + (32/1)) / 4 cores
149 time units / 4 cores

129 time units / 4 cores / 64 units
58% utilization for job

Option 2:

Total execution time = max(48 / 1, 128 / 2, 80 / 3, 32 / 1)
= max(48, 64, 26.7, 32)

Max is still 64 time units:

Utilization: ((48/1) + (128/2) + (80/3) + (0/1)) / 4 cores / 64 units

54% utilization for job

If core 4 is not included, utilization goes to 72%

- This number makes more sense (and this option makes more sense) as we could be running another job on Core 4