Pachinko

Brody Larsen

January 27, 2022

Pachinko

When designing my pachinko code, I decided to make a two element list called "ball" that stored what pin/processor the ball was at and it stored what level the ball was currently at. My pachinko board has eight levels of pins. This means the ball gets passed back and forth randomly 8 times. Every time the ball is passed it records what processor it was sent to and decrements what level it is at. I made a separate list that is eight elements. Each element is bucket that catches the balls at the bottom of the board.

I put all my ball dropping logic inside a while loop that ran until a Boolean is set to false. The first step in my while loop was to receive a ball. After a ball was received, the level that the ball is at is decremented by one. The first check is to see if the ball is at level negative two. The only way to get a ball at negative two is if it was set they by an outside force as a trigger to break out of the while loop by setting the boolean to false. The next step in the while loop is to check to see if the ball is at level zero. If the ball is at level zero, the boolean is set to false and the while loop ends. If the past two checks fail then the ball is randomly get sent off to processor + 1 or processor - 1.

Outside the while loop, they is another check to catch the processor that has the ball. This catch will only happen if the ball is at level zero and is ready to be put into a bucket. The ball gets put into the element in the list that corresponds to either the pin's number or the pin's number + 1.

After the bucket list has been updated, the processor sets the ball's level to -2 and sends that information to all the other processors. This causes all the other processors to exit the while loop and terminate.

I put all the while loop logic inside a for loop that iterates 100 times. This simulates 100 balls being dropped through the pachinko board. Outside of the for loop, I iterated through the bucket list and printed out how many balls are in each bin/element. Since this program is running on all processors, the for loop will print out 8 times how ever many processors there are. To combat this extra printing of data, I ran a check in the for loop to only print if the bin was

greater than zero.

I ran my pachinko board with eight levels and 7 processors/pins. I excluded processors 0 from receiving a ball. This may or may not have prevented a deadlock situation. Initially when coding this assignment I was getting deadlock issues and when I prevented processor zero from receiving a ball the deadlock issue stopped. I'm still not sure why this is the case. I thought I had the necessary checks in place that prevented a process from sending a message to itself. However things started working once I prevented processor zero from receiving balls.

My pachinko game gives a normal distribution of balls. When I was testing the code while writing it, I only used 10 balls. 10 balls is not enough balls to get a normal distribution regularly. I can also imagine a pachinko board with only 2 or 3 levels won't get a normal distribution since it is impossible for a ball to reach either edge of the board. A normal distribution also wont happen if there are only 2 processors.