Lab 1: Multi-tasked Food Distribution

Sept 19th, 2022

William Pearce

COSC 3319

Section 01

MWF

“A” Option

Added filter for recalled food

**“D-“ Requirements:**

Final Result:

[bill@Manjaro-HP DSA-Lab-1]$ vim Data.txt // Enter data 1 2

[bill@Manjaro-HP DSA-Lab-1]$ ./bin/productdistributionmain < Data.txt > res1.txt

// I/O redirection to use Data.txt as input and print results to result-3-1-2.txt

In file result-3-1-2.txt:

How many Product Generators?

How many points of sale?

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 5.66092E-02 Time units!

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 9.14843E-02 Time units!

B delivered.

Rejected by GateKeeper:

SQUASH B

Rejected = 1. Sent to another distribution facility!

Next grain shipment arrives 4.34619E-01 Time units!

...

Retail Sales successfuly sold STEAK M

FOWEL M Removed by GateKeeper for shipment.

B delivered.

Retail Sales successfuly sold WHEAT B

Retail Sales successfuly sold FOWEL M

GateKeeper insert accepted WHEAT B

Next grain shipment arrives 6.39829E-01 Time units!

WHEAT B Removed by GateKeeper for shipment.

M delivered.

GateKeeper insert accepted FOWEL M

Next grain shipment arrives 1.97321E-01 Time units!

Retail Sales successfuly sold WHEAT B

B delivered.

Hours of operation prior to closing: 40.367263000

Meat Packs Processed: 4

Non-meat Packs Processed: 9

Total Packets Processed: 13

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Queue Size | # of Gen | # of POS | Total food packs | Meat packs | Non-meat packs | sales |
| 10 | 1 | 1 | 10 | 2 | 8 | 10 |
| 10 | 1 | 2 | 13 | 2 | 11 | 13 |
| 10 | 1 | 3 | 16 | 4 | 12 | 19 |
| 10 | 1 | 4 | 15 | 4 | 11 | 21 |
| 10 | 2 | 1 | 3 | 0 | 3 | 3 |
| 10 | 2 | 2 | 13 | 2 | 11 | 13 |
| 10 | 2 | 3 | 13 | 2 | 11 | 13 |
| 10 | 2 | 4 | 13 | 2 | 11 | 13 |
| 10 | 3 | 1 | 3 | 0 | 3 | 3 |
| 10 | 3 | 2 | 3 | 0 | 3 | 3 |
| 10 | 3 | 3 | 3 | 0 | 3 | 3 |
| 10 | 3 | 4 | 3 | 0 | 3 | 3 |
| 15 | 3 | 4 | 3 | 0 | 3 | 3 |
| 20 | 3 | 4 | 3 | 0 | 3 | 3 |
| 5 | 1 | 2 | 13 | 3 | 10 | 13 |
| 4 | 1 | 2 | 13 | 4 | 9 | 13 |
| 3 | 1 | 2 | 13 | 4 | 9 | 13 |
| 2 | 1 | 2 | 5 | 2 | 3 | 7 |

In order to determine the minimum queue size to get the maximum number of sales, I first determined which combination of product generators and points of sale would result in the most sales. Highlighted in read are sales numbers that are likely the result of a race condition between the sales threads, since they are greater than the total number of packets processed. After excluding the race condition cases, the trial with the most sales and least threads used is the second trial, with 1 generator and 2 points of sale.

After determining the optimal combination of generators and points of sale, I gradually lowered the queue size until it began to adversely affect the performance. This method managed to get the queue size all the way down to 3.

**“C” Requirements:**

Added to circularque.ads:

procedure insertFront (msg : in message);

Added to circularque.adb

procedure insertFront(msg : in message) is

begin

if mesnum < maxMessages then

box(front) := msg; -- insert first since front is on an empty space (unless overflow)

if front = 0 then -- if front is 0, loop around

front := maxMessages;

else -- otherwise subtract 1

front := front - 1;

end if;

mesnum := mesnum + 1;

else

Put ("ERROR - Message rejected - queue is full!");

New\_Line (2);

end if;

end insertFront;

Modified gatekeeperservice.adb:

accept acceptMessage(newFood : in Food\_Pack) do

if not (circularQueFull) then

if getFood\_PackFoodType(newFood) in GrainVegetable then

CircularQueue.acceptMessage(newFood);

Put("GateKeeper insert accepted ");

PrintFood\_Pack(newFood);

New\_Line;

else

CircularQueue.insertFront(newFood);

Put("Gatekeeper insertFront accepted ");

PrintFood\_Pack(newFood);

New\_Line;

end if;

else

....

Final Result:

Retail Sales successfuly sold STEAK M

Retail Sales successfuly sold POTATOES B

GateKeeper insert accepted WHEAT B

Next grain shipment arrives 6.39829E-01 Time units!

TOMATO B Removed by GateKeeper for shipment.

M delivered.

Gatekeeper insertFront accepted FOWEL M

Next grain shipment arrives 1.97321E-01 Time units!

Retail Sales successfuly sold TOMATO B

B delivered.

Hours of operation prior to closing: 40.365416000

Meat Packs Processed: 4

Non-meat Packs Processed: 9

Total Packets Processed: 13

...

How many Product Generators?

How many points of sale?

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 5.66092E-02 Time units!

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 9.14843E-02 Time units!

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Queue Size | # of Gen | # of POS | Total food packs | Meat packs | Non-meat packs | sales |
| 5 | 1 | 2 | 13 | 4 | 9 | 13 |
| 5 | 1 | 3 | 16 | 4 | 12 | 19 |

After implementing the InsertFront procedure for meat products, the queue size had to be increased to accommodate the increased prep time for meat products to be shipped out, since they were now cutting in front of non meat products that may already be prepared.

I didn’t test the program with more generator threads, because in previous testing I found that additional generators put too much extra load on the main gatekeeper thread to allow the sales threads to work efficiently. I did, however test with one more sales thread, and once again ran into the issue of race conditions.

**“B” Requirements:**

Modified in gatekeeperservice.adb:

End\_Time := Start\_Time + 1.0 \* 8.0 \* 2.0;

-- 1.0 sec./hour \* 8 hours/days \* 2 days

**Results:**

How many Product Generators?

How many points of sale?

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 5.66092E-02 Time units!

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 9.14843E-02 Time units!

B delivered.

...

Gatekeeper insertFront accepted FISH M

Next grain shipment arrives 6.55132E+00 Time units!

FISH M Removed by GateKeeper for shipment.

Hours of operation prior to closing: 16.668208000

Meat Packs Processed: 2

Non-meat Packs Processed: 4

Total Packets Processed: 6

Retail Sales successfuly sold FOWEL M

B delivered.

Retail Sales successfuly sold FISH M

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Queue Size | # of Gen | # of POS | Total food packs | Meat packs | Non-meat packs | sales |
| 10 | 1 | 1 | 5 | 2 | 3 | 5 |
| 10 | 1 | 2 | 6 | 2 | 4 | 6 |
| 10 | 1 | 3 | 6 | 2 | 4 | 6 |
| 10 | 2 | 1 | 3 | 2 | 1 | 3 |
| 10 | 2 | 2 | 4 | 2 | 2 | 4 |
| 10 | 2 | 3 | 4 | 2 | 2 | 4 |
| 10 | 3 | 1 | 2 | 2 | 0 | 2 |
| 10 | 3 | 2 | 2 | 2 | 0 | 2 |
| 10 | 3 | 3 | 2 | 2 | 0 | 2 |
| 5 | 1 | 2 | 6 | 2 | 4 | 6 |
| 3 | 1 | 2 | 6 | 2 | 4 | 6 |
| 2 | 1 | 2 | 5 | 2 | 3 | 6 |

Using the same process shown in option “D”, I once again found queue size 3 with 1 generator and 2 points of sale to be the optimal combination.

**“A” Requirements:**

For my “A” option, I added the ability to ‘recall’ certain food types in case of contamination. The recalled food type is determined at the start of runtime, and each food pack is checked as it comes in. If the incoming food pack matches the type that was recalled, the pack is dropped and a rejection message is displayed.

Modified gatekeeperservice.adb:

with Ada.Text\_IO; use Ada.Text\_IO;

with Ada.Calendar; use Ada.Calendar;

with Food\_DataStructures; use Food\_DataStructures;

package body GateKeeperService is

package IntegerIO is new Ada.Text\_IO.Integer\_IO(*Integer*);

use IntegerIO;

task body GateKeeper is

package CircularQueue is new CircularQue(Food\_Pack, 10); -- default size 10.

use CircularQueue;

rejected : *Integer* := 0;

total\_packs : *Integer* := 0;

meat\_packs : *Integer* := 0;

other\_packs : *Integer* := 0;

-- Declare food packet counters here.

Start\_Time : Ada.Calendar.*Time*;

End\_Time : Ada.Calendar.*Time*;

recalled\_food : Food\_Pack;

begin

delay 0.5; -- allow 1/2 hour to initialize facility.

Start\_Time := Ada.Calendar.Clock;

End\_Time := Start\_Time + 1.0 \* 8.0 \* 2.0;

-- 1.0 sec./hour \* 8 hours/days \* 2 days

setFood\_PackFoodType (recalled\_food, Food\_DataStructures.RandomFoodType);

-- Terminate after losing 5 customers or time to close has arrived.

while rejected < 5 and Ada.Calendar.Clock < End\_Time

loop -- Terminate after losing 5 customers

-- In Ada, a "select" statement with multiple "or" options must uniformly

-- process (randomly) the "accept" statements. This prevents any single

-- "accept" from starving the others from service.

--

-- Rules for "Select":

-- 1) If no task are waiting for service, the task sleeps. Wakes when any “accept” is requested.

-- 2) If only one of the "accept" entries has a task waiting that task is served.

-- 3) If sleeping and a task or tasks arrive simultaneously, awake a service the

-- the first arrival.

-- 4) If multiple "accepts" have task waiting, service them in random order

-- to prevent starvation.

--

select

-- new arrivals of food

accept acceptMessage(newFood : in Food\_Pack) do

if getFood\_PackFoodType(newFood) = getFood\_PackFoodType(recalled\_food) then

Put(" Rejected by Gatekeeper: ");

New\_Line;

PrintFood\_Pack(newFood);

Put(" recalled");

New\_Line(3);

elsif not (circularQueFull) then

...

**Result:**

How many Product Generators?

How many points of sale?

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 9.93067E-01 Time units!

B delivered.

GateKeeper insert accepted RICE B

Next grain shipment arrives 1.27761E+00 Time units!

B delivered.

...

Rejected by Gatekeeper:

SQUASH B recalled

...

Hours of operation prior to closing: 16.507522000

Meat Packs Processed: 2

Non-meat Packs Processed: 2

Total Packets Processed: 4

B delivered.

M delivered.

Retail Sales successfuly sold RICE B