Program Design

Lakshmi Narasimha Guptha Munuhur Rajagopal

Core Concepts

There are two coordinators: one responsible for pigs with odd port numbers, and one responsible for pigs with even port numbers. Naturally, the performance will be better if there are similar numbers of odd and even pigs. At the beginning of each bird launch, there is a chance that one of the coordinators will go down. The coordinators talk to each other about their status. It is impossible for both the coordinators to be down at any instance in the game. If both coordinators go down during the same bird launch, the one with the higher port number remains alive. A coordinator which has gone down during a game is not involved in the subsequent launches. The other coordinator will then not go down for the rest of the game.

At the end of each launch, the coordinator gathers the status of the pigs that it is responsible for at this point of the game. If it is the sole alive coordinator, it gathers the status of all the pigs. It then updates the database with the positions and status of all the pigs it is responsible for. This ensures that the database remains in sync, in case the coordinator fails. One possible optimization is for the sole alive coordinator to not update the database, but I felt it was better this way, in case the design changes such that a coordinator is able to come up again. It is trivial to change the code to allow a dead coordinator to come up again seamlessly. 4 messages, one to the other coordinator, one to the bird, one to the database, and one response from the database is all that is required to take care of that case.

The database has been implemented as an unordered_map, with the port number as the key. This allows for constant time lookup most times, with a worst case complexity linear in the number of pigs. The database is multithreaded, and accepts multiple connections at the same time. The integrity of the core database is protected using locks. The database is able to fetch the positions and status of multiple pigs in a single message. The database model is a stateless pull based one.

Program Flow

The bird sends the Start Game message at the beginning of every game to both the coordinators. Then, it sends the Bird Launch message. This bird launch message contains the position where the bird will land. The timing of the bird landing is a fixed amount of time after the bird launch message is received. Once the coordinators receive the bird launch message, they either become dead or stay alive. They then send the bird approach message to the other pigs. The other pigs see if they are affected by the bird. Two relaxations have been done for this assignment: the pigs are not affected by their neighbors.

The only way for a pig to get hit is if the bird lands directly on their position and they don't have time to move. The second relaxation is that whether they have time to move or not is random.

The coordinators then send the status request message to the pigs they are responsible for. The pigs respond to this with their status. One thing to note is that the coordinator can be active as a pig even after it has been hit by a bird. The responsibilities of a coordinator are completely separated from its status as a pig. The design is agnostic to whether a pig is a coordinator or not – the bird landing side of the process is completely independent.

Once the coordinators have collated results, they send their part of the score back to the bird. If the other coordinator is dead, the live coordinator indicates that in the message. The bird adds up the scores and displays them. The bird then launches itself again and informs the coordinators using the bird position message. This chain continues for 10 launches.

Once 10 launches are done, the bird sends the start game message to both the coordinators. This resets everything. Both the coordinators become live again, and all the pigs are live again. There is a repeat of the 10 launches again.

Messages

START_GAME_MSG – Sent from the bird to the coordinators to indicate start of a game

DB_RESP_MSG – Sent from the DB to the coordinator with the pig information requested

DB_REQ_MSG – Sent from the live coordinator to the DB soon after the other coordinator dies

DB_UPD_MSG – Sent at the end of each launch by the coordinators to the DB

BIRD_POSN_MSG – Sent by the bird to the coordinators at the start of each launch

PIG_COORD_MSG – Sent between coordinators to sync up their status

STATUS_REQ_MSG – Sent from the coordinator to each of the pigs under their purview

STATUS_RESP_MSG – Sent by the pigs to their coordinator

BIRD_APPROACH_MSG – Sent by the coordinator to each of the pigs under their purview

END_LAUNCH_MSG – Sent by the live coordinators to the bird at the end of a launch

Important Methods

birdStartNewGame – Starts off a new game

birdStartNewLaunch – Starts off a new launch

handlePigs – At the coordinator, this function takes care of notifying the pigs under their purview

dbGetElement – Gets a stored element at the DB

dbSetElement – Stores a given element in the DB

Note that I have not touched upon the send and handle methods for each of the messages. They are fairly straightforward.

Some Characteristics

1. All the processes are multithreaded, and are capable of handling multiple requests at the same time
2. The DB stores information in memory, and is assumed to be alive all the time
3. The cache consistency algorithm is pull based, and the DB server is stateless

Improvements

1. The DB should ideally be fault tolerant, either through use of replication, or by using the disk
2. Assigning pigs to coordinators should be done in a better way. Currently, the odd numbered pigs are assigned to one coordinator and the even, to the other. This often does not lead to a balanced network
3. Right now, the coordinators inform the other coordinator when they go down. In the real world, this happens only if there is a watcher process that can monitor the status of the coordinator. Ideally, this should be done using heartbeat messages, or timing out the isAlive message

Use Instructions

Go to the birdypigs3 directory and give make. This creates three executables: bin/bird, bin/db and bin/pig. Run bin/bird. This automatically spawns the pigs and the db.

If you want to adjust any of the macros in inc/comconst.h, change them and run “make”. Make timestamps suck, so it is always better to “make clean” and then “make.” The process is persistent, so to kill, just “chmod 777” the file called kill and run “./kill” All commands have to be given in the birdypigs3 directory