Ghent University

Parallel and Distributed Software Systems

Homework Assignment 5

Tibo Vanheule

Academic Year 2023-2024

1 Introduction and Chang-Roberts implementation

(Q1.1): Concisely describe why the seed for the pseudo-random integer generator depends on the value of processRank. What could potentially happen if the seed was not dependent on processRank

The seeds based on only timestamps are too close together due to the processes are created too quickly. In fact here we use std::time()[1], this returns the seconds since epoch. This creates a problem since multiple processes can end up with the same UID. The algorithm described in the syllabus loops over all nodes but eventually it stops without sending information to the next node. The election never finds an elected node. We conclude that this algorithm is not very fault tolerant.

This can easly verified by altering the code in "process.hpp".

(Q1.2.1): Run your solution 10 times, with 16 processes. Give the minimum, maximum and average (averaged over 10 runs) amount of messages that your solution needed to elect a leader.

Minimum	Maximum	Avarage
17	31	25,7

Table 1: We look at the messages to elect a node.

(Q1.2.2): Is this in line with the theoretical message complexity given 16 processes are used? Calculate (show your calculations!) the best-case, worst-case and average-case message complexity.

- **Best-case:** 1+ n-1= 16
- **Worst-case:** n+ n-1= 31
- **Avarage-case:** n*log(n)= 20

(Q1.3.1): Run your solution 10 times using the ‘time‘ program to measure the execution time of your solution. Give the minimum, maximum and average (averaged over 10 runs) ‘real‘ execution time.

Minimum	Maximum	Avarage
1.633	1.848	1.692

Table 2: We look at the real time to elect a node. using hpc dophan cluster and 6 cores.

(Q1.3.2): Why do we want to know the ‘real‘ time and not the time spent in the ‘user‘- or ‘sys‘-mode?

Following the man page[2] of the time command we fin the following definitions.

[1] https://en.cppreference.com/w/cpp/chrono/c/time
[2] https://www.man7.org/linux/man-pages/man1/time.1.html

1. **real:** Elapsed real time
2. **sys:** Total number of CPU-seconds that the process spent in kernel mode.
3. **usr:** Total number of CPU-seconds that the process spent in user mode.

Cpu seconds do not include the IO-time and when a process is blocked.

2 Franklin

(Q2.1.1): Concisely explain the difference between the Franklin algorithm and the Chang-Roberts algorithm

There both ring algorithms, but Franklin uses passive nodes as router that only forward messages. Active can become passive if they receive an id bigger than their own ID. They can receive from both there left or right neighbour. Thus Franklin is bidirectional and Chang-Roberts is not.

(Q2.1.2): Why would anyone choose to use the Franklin algorithm over the Chang-Roberts algorithm or vice versa?

Franklin has higher complexity due the bidirectional message passing, but o, each round at least n/2 active nodes become passive. This could be intresting in bigger rings.

2.1 (Q2.2.1): Run your solution a couple of times, with 16 processes. What is the observed minimum and maximum amount of rounds that your solution needs to elect a leader (including the announcement round)?

Minimum	Maximum	Avarage
7	9	7.7

Table 3: We look at the real time to elect a node. using hpc dophan cluster and 6 cores.

(Q2.2.2): Calculate (show and explain your calculations!) the theoretical amount of rounds at most needed to elect a leader by 16 processes (including the announcement round). Is this in line with the result from the previous question?

/

2.2 (Q2.3.1): Measure the 'real' execution 'time' of your solution 10 times, with 16 processes. Give the minimum, maximum and average (averaged over 10 runs) 'real' execution time.

Minimum	Maximum	Avarage
1.921	2.539	2.243

Table 4: We look at the real time to elect a node. using hpc dophan cluster and 6 cores.

(Q2.3.2): Is there a noticeable execution time difference between Chang-Roberts and Franklin? If so, which one of the two algorithms is more efficient and why? If not, can you explain why these algorithms perform in the same way?

There is an significant difference in performance. The Franklin is notably slower, but the difference is only seen in real time mode and not in sys and usr time modes. This gives the impression that extra messages in the Franklin algorithm causes some extra process blocks.

(Q2.4): As a passive process, is there a certain order in which messages must be sent or received? If so, explain why that order is important. If not, explain why the order doesn't matter.

Yes, the opposite order as the active node has to be followed. Otherwise a deadlock will occur.

(Q3.1): How much time did you spend on this assignment, excluding the time spent studying the lecture notes?

About 6 hours.