Welcome to ACS TA session 5

Alexander, Lasse, Svend, Yiwen, and Yuan

Department of computer science, University of Copenhagen

Academic year 2018-2019, Block 2

Agenda for today

Feedback on Assignment 3

Distributed transactions

- The two-phase commit (2PC) protocol. Exercises

Reliability

- Eventual consistency. Exercises
- Availability. Exercises

Assignment 3 – Feedback 1 (Code)

- Latency should be calculated for each worker individually.
  - Average them afterwards over the number of workers.
- Throughput should be calculated as described in the assignment text.
- The interaction methods should not call the bookstore api in a loop.
  - AddCopies, AddBooks and BuyBooks all take sets, use it like such.
- Sorting the entire database on each interaction is expensive.

Assignment 3 – Feedback 2 (Q1: workload, setup)

- Justification of choices are more important than implementation.
- Always justify the numbers for the experiment:
  - Size of dataset;
  - How you decided those values
    (e.g. a bookstore with very few books, is it a realistic one?)
- Comment on the book generation procedure:
  - What type of workload do you model?
  - Why do you think that workload is good?
- Consider the test system:
  - E.g. if the test set is small enough to fit into the CPU cache, this would be a meaningless experiment.
  - Also, is your system “enough”, why?

Assignment 3 – Feedback 3 (Q2: plots, discussion)

- Read the questions properly.
  - “Explain the trends” part was missed by most.
  - Showing the graphs != Understanding the graphs
- If the RPC one didn’t work out, write your expectations of how you think it would look. Better to show some understanding than give up.
- Start from a reasonable number of threads.
  - This bookstore can only handle 1 request at a time.
  - Should we start at 10 worker threads?

Assignment 3 – Feedback 4 (Q3: reliability, metrics)

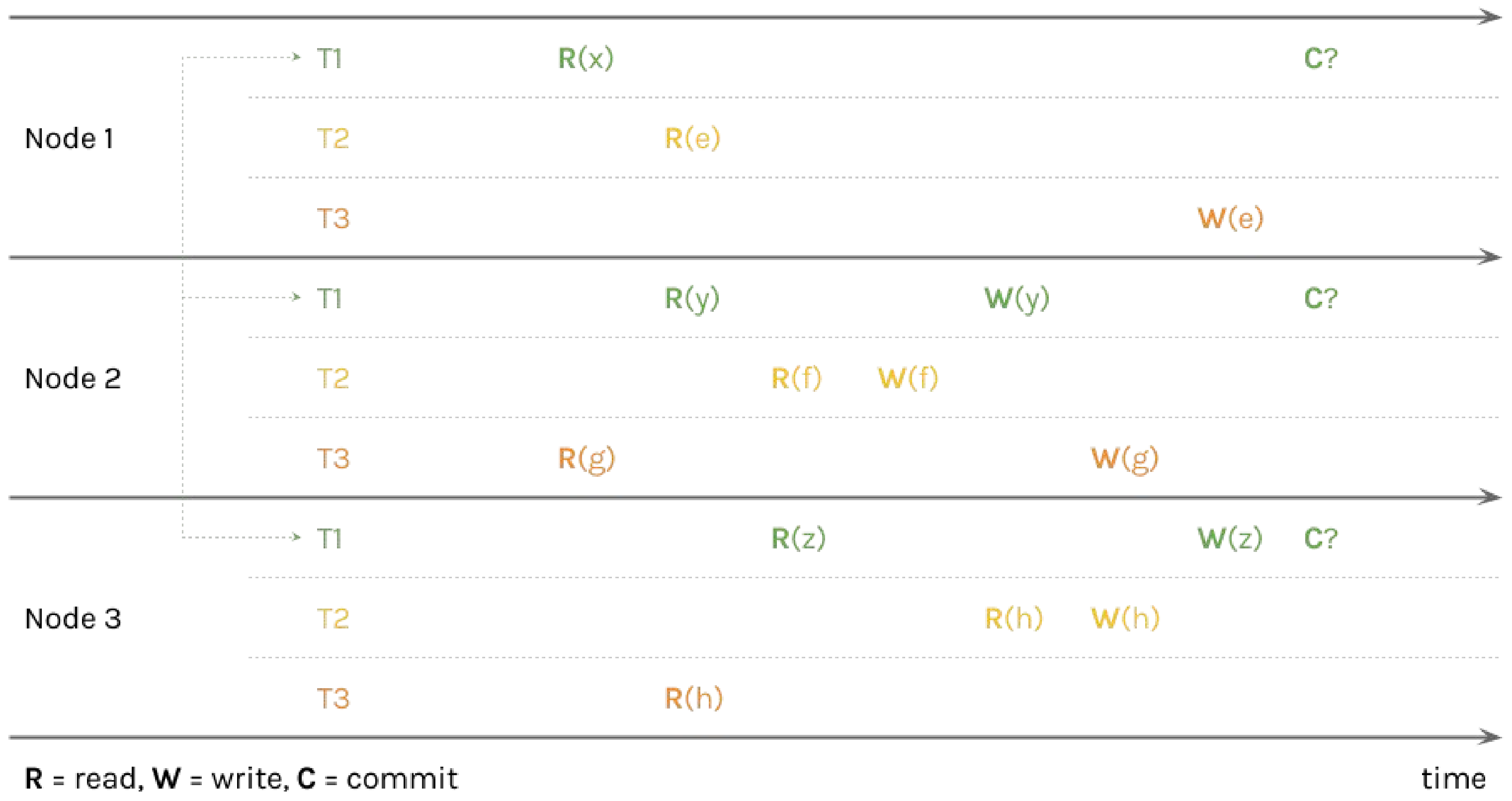
- Reliability is not whether this metric we have chosen can reflect the real-world.
  - A reliable metric reflects some invariant characteristics of the system. (Regardless of workload and setup)
- Please explain WHY the metrics you used are reliable.
  - What characteristics of the system does the metric show?
  - Will its behavior changes in different runs?
- Metrics results can be different in a new setup, but performance have the same trend when doing the same tests for the same system in comparable setups.
- Successful Requests pr. time --> throughput. Not correctness.
- Fault-tolerance and scalability are not performance metrics.
  - Properties of the system we are testing.

Exercise 1: Distributed transactions

For the distributed system on the next slide:

1. What are the local wait-for graphs for the three nodes?
2. Is there any deadlock in the system?
3. At the end of the schedules, would T1 be able to issue a commit?

Exercise 1: Distributed transactions



Exercise 2 - problem statement

$$[P_{coord}] \rightarrow [P_1] \rightarrow \ldots \rightarrow [P_n]$$

A process P waits for a message from its left neighbor. If P receives YES and its own vote Is YES, then P forwards YES to its right neighbor. Otherwise, P forwards NO.

The rightmost process P_n will have all the information it needs to make a decision. If it receives a YES and its own vote is YES , then the decision is COMMIT, otherwise the decision is ABORT. Then it forwards the decision to the left neighbor.

Each process that receives the decision message decides accordingly and then forwards that message to its left neighbor until the coordinator receives the message.

Exercise 2 - questions

$$[P_{coord}] \rightarrow [P_1] \rightarrow \ldots \rightarrow [P_n]$$

1. If there are $n > 2$ participants, how many messages are sent when performing **centralized 2PC** and **linear 2PC**?

2. Assume a communication system with unlimited bandwidth. Assume further, that the processor time necessary to process a 2PC algorithm is 0 both for the coordinator and the participants. If message transmission time is t, how much time at least will it take when performing a **centralized 2PC** and a **linear 2PC**?

3. What should happen if one process fails in the **linear 2PC**?

Exercise 3 - questions

1. What is the idea behind **eventual consistency**? Why is it eventual?
2. Can you describe any mechanism of applying **eventual consistency**? What are the trade-offs?
3. (Discussion) Provide use cases where the gains in performance from **eventual consistency** make it useful in practice.

Exercise 4 - availability

How many machines do you need to build a system that has an availability probability of 99.9999%, knowing that you can use (bad quality, cheap) machines that have only 50% availability probability?[1] [2]

1. the system is available if one machine is available
2. all failure events are assumed independent

Thank you

Alexander, Lasse, Svend, Yiwen and Yuan

Department of computer science, University of Copenhagen

Academic year 2018-2019, Block 2