Welcome to ACS TA session 3

Julian, Leonardo, Li, Nikolaus, Rodrigo, Tilman, Yijian

Department of computer science, University of Copenhagen

Academic year 2021-2022, Block 2

Agenda for today

Feedback on Programming Assignment 1
Exercise: techniques for performance
JAVA materials for Programming Assignment 2
notes on graphs (Assignment 3)

Questions/Exercises on ARIES are moved to next week

Programming Assignment 1: Feedback

1) Provide short description in report including:
   - List of tests, where your tests are located

2)Implement all-or-nothing properly:
   - Validate the input (in one loop),
   - Perform the updates (in another loop).

3)Test all-or-nothing properly:
   - Add multiple items (with one or more invalid),
   - Check that datastore is not affected after exception.

4)Think about algorithms and data structures you use

Common mistakes in code

1) Implementation on rateBooks():
   - Forgot to validate all books/ISBN are in bookstore collection

2)Implementation on getTopRatedBooks():
   - Did not handle when K > total number of books in BookStore
   - Should compare the average rating, not total rating
   - Forget to return immutable class

3)Implementation on getBooksInDemand()
   - Forget to return immutable class

4)Test
   - Complete test cases are rare
   - Should test exception msg content, not just exception type (or better: modify/extend exception classes)

Common misconceptions

1) what type of semantics is implemented?
   - most got this right
   - though some provided weak arguments
2) in what sense is the system modular
   - java interfaces do NOT indicate modular design
   - isolation and encapsulation does
3) Where is/are the bottlenecks?
   - We expect the specific component
   - details of the implementation
4)Q7:caching at web proxy
- Consider read and write requests (mask failure from server)

Common misconceptions

1） Scalability: scalability is the capability of a system to be enlarged
(both in scaling out and scaling up)
to accommodate the growing amount of work.

2） Safety ≠ Security:
Many submissions explain from the aspects of security like authentication, encryption or filtering invalid requests.
Safety: the system internally
Security: dangers from outside

Exercise 1: Techniques for Performance

1. What is latency?

2. What is throughput?

3. How latency and throughput are related in the case of a serial execution? Concurrent execution?

4. How concurrency affects latency and throughput?

Exercise 1: Techniques for Performance

1. What is latency?

   The delay between a change at the input to a system and the corresponding change at it output

2. What is throughput?

3. How latency and throughput are related in the case of a serial execution? Concurrent execution?

4. How concurrency affects latency and throughput?

Exercise 1: Techniques for Performance

1. What is latency?

   The delay between a change at the input to a system and the corresponding change at it output

2. What is throughput?

   The rate of useful work done by a service for some given workload of requests

3. How latency and throughput are related in the case of a serial execution? Concurrent execution?

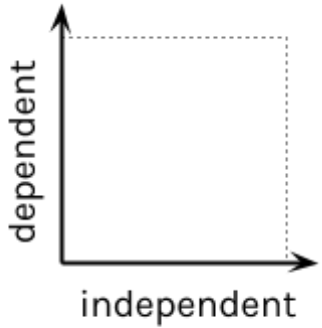
4. How concurrency affects latency and throughput?

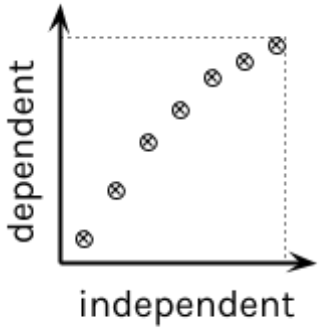
Exercise 1: Techniques for Performance

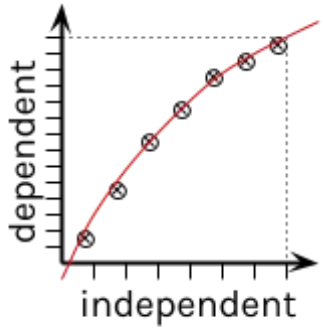
1. What is latency?
   The delay between a change at the input to a system and the corresponding change at it output
2. What is throughput?
   The rate of useful work done by a service for some given workload of requests
3. How latency and throughput are related in the case of a serial execution? Concurrent execution?
   Serial: latency = 1/throughput
   Concurrent: no direct relationship between them
4. How concurrency affects latency and throughput?

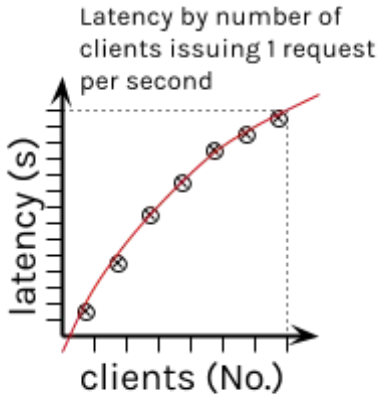
Notes on graphs: draw your graph

- Use the x axis for the independent variable
- Use the y axis for the dependent variable
- Choose scales such that the data points spread across the surface



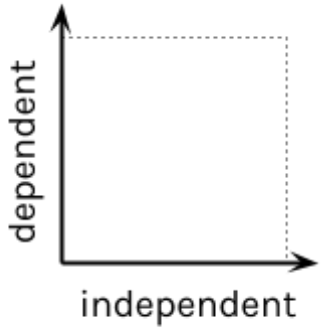


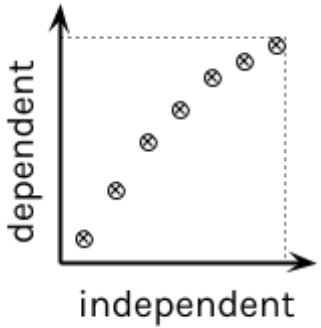


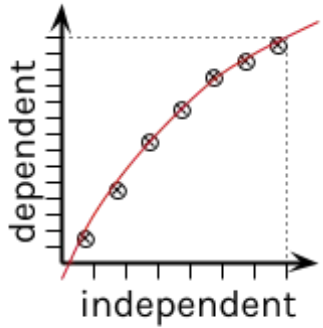


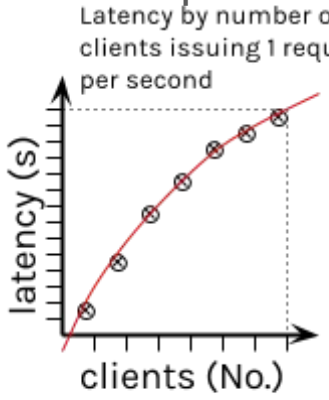
Notes on graphs: fill your graph with points

- Locate experimental points by small, sharp dots
- Draw around each point a small circle (cross, plus)
- (optional) Draw a smooth curve, passing near as many points as possible



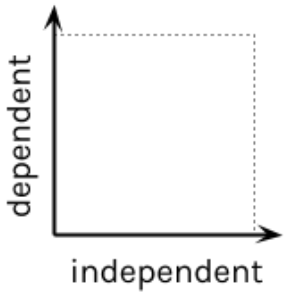


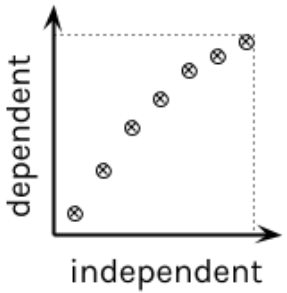


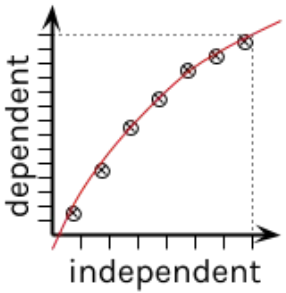


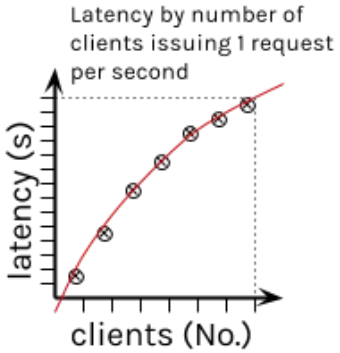
Notes on graphs: present your graph

- Write the title of the curve
- On each axis, type what it is (e.g., latency)
- On each axis, type what unit it is measured in (e.g., seconds)
- Distinguish different curves in the same plot by colors and/or markers





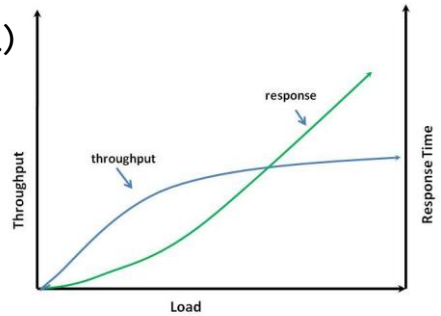




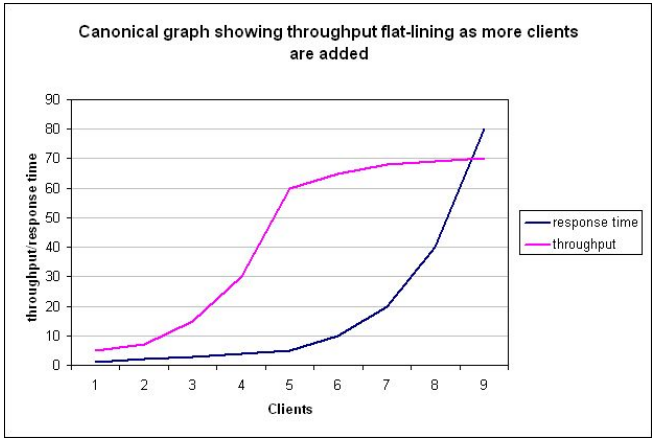
Exercise on graphs

What is missing from each of these graphs?

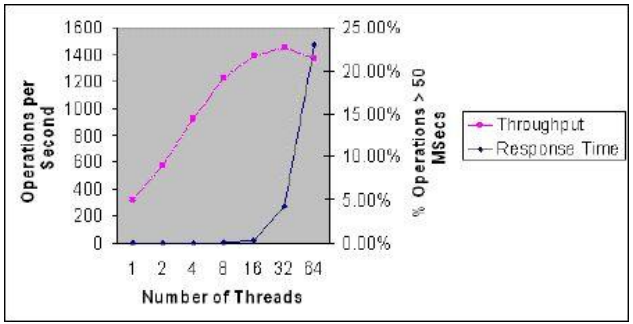
a.)



b.)



c.)



Stack Exchange computer science
http://cs.stackexchange.com/questions/40868/why-does-response-time-increase-with-throughput

Creating threads in JAVA

Method 1: Implement the java.lang.Runnable interface.

```
public class HelloWorldRunnable implements Runnable {
    public void run() {
        System.out.println("Hello World");
    }

    public static void main(String[] args) {
        new Thread(new HelloWorldRunnable()).start();
    }
}
```

Creating threads in JAVA

Method 2: Extend the java.lang.Thread class.

```
public class HelloWorldThread extends Thread {
    public void run() {
        System.out.println("Hello World");
    }

    public static void main(String[] args) {
        new HelloWorldThread().start();
    }
}
```

JAVA Concurrency - ReadWriteLock

A java.util.concurrent.locks.ReadWriteLock interface allows multiple threads to read at a time but only one thread can write at a time.

- **Read Lock −** If no thread has locked the ReadWriteLock for writing then multiple thread can access the read lock.
- **Write Lock −** If no thread is reading or writing, then one thread can access the write lock.

JAVA Concurrency - Example

```
1  import java.util.concurrent.locks.ReentrantReadWriteLock;
2
3  public class TestThread {
4      private static final ReentrantReadWriteLock lock = new
           ReentrantReadWriteLock(true);
5      private static String message = "a";
6
7      public static void main(String[] args) throws InterruptedException {
8          Thread t1 = new Thread(new WriterA());
9          t1.setName("Writer A");
10
11         Thread t2 = new Thread(new WriterB());
12         t2.setName("Writer B");
13
14         Thread t3 = new Thread(new Reader());
15         t3.setName("Reader");
16         t1.start();
17         t2.start();
18         t3.start();
19         t1.join();
20         t2.join();
21         t3.join();
22     }
```

JAVA Concurrency - Example

```
26    static class Reader implements Runnable
27    {
28
29        public void run()
30        {
31            lock.readLock().lock();
32            System.out.println(Thread.currentThread().getName() +": "+ message );
33            lock.readLock().unlock();
34        }
35    }
36
37    static class WriterA implements Runnable
38    {
39        public void run()
40        {
41            lock.writeLock().lock();
42            System.out.println(Thread.currentThread().getName() + "write aa");
43            message = message.concat("aa");
44            lock.writeLock().unlock();
45        }
46    }
47
48    static class WriterB implements Runnable
49    {
50        public void run()
51        {
52            lock.writeLock().lock();
53            System.out.println(Thread.currentThread().getName() + "write bb");
54            message = message.concat("bb");
55            lock.writeLock().unlock();
56        }
57    }
```

JAVA Concurrency – Possible results

```
Writer A: write aa
Writer B: write bb
Reader: aaabb
```

```
Reader: a
Writer A: write aa
Writer B: write bb
```

```
Writer B: write bb
Writer A: write aa
Reader: abbaa
```

.....

Thank you

Julian, Leonardo, Li, Nikolaus, Rodrigo, Tilman, Yijian

Department of computer science, University of Copenhagen

Academic year 2021-2022, Block 2