Welcome to ACS TA session 3

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 1

The ARIES algorithm

- Principles and properties
- Log record data structure: types and fields of log records
- Additional data structures: dirty page and transaction tables

Questions/Exercises on ARIES

Recap: techniques for performance

notes on graphs (Assignment 3)

Programming Assignment 1: Feedback

1) Provide short description in report including:
    - Notes on algorithms used
    - List of tests, where your tests are located
2) Write more tests!
3)Implement all-or-nothing properly:
    - Validate the input (in one loop),
    - Perform the updates (in another loop).
4)Test all-or-nothing properly:
    - Add multiple items (with one or more invalid),
    - Check that datastore is not affected after exception.
5)Think about algoritms and data structures you use
6) No need to sort everything in getTopRatedBooks()

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) the purpose of ImmutableBook classes
   - what they're not used for
     - prevent client from modifying our state
   - why and how do we use them?
     - see 2

Common misconceptions cont

1) Scalability: scalability is the capability of a system to be enlarged (both in scaling out[“cloud”] and scaling up[“free lunch”]) 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

3) Single server assumption:
   Some mention more servers should be added into the architecture, otherwise, it will not benefit from deploying web proxy servers. In fact, although with single server, web proxy servers do reduce the number of connections for the server.

Assignment 1: Feedback 3- Q5 – Q7

Q5) Web-Proxy
- it is safe to use web-proxies
- between BookStoreHTTPServer and BookStoreHTTPProxy
- reduce overhead/load to maintain connections on
- increase amount of clients
- transparent for clients

Q6) Bottleneck
- queuing can occur (requests and responses must be in 'equilibrium')
- BookStore doesn't allow real concurrent access (yet)

Q7) Failures
- degrading failures, client will still get a response
- caching is ok, but only reads (caching is not replication)

ARIES principles and properties

Properties

- **Atomicity**: undo the transactions that do not commit
- **Durability**: ensure all actions of committed transactions survive system crashes and media failures

Approach

- **Steal**: pages are written to disk in yet uncommitted transactions
- **No-force**: when a transaction commits, its pages are not forced to disk

ARIES principles and properties

Principles

- **Write-ahead logging**: log the operation before executing it
- **Repeat history**: re-bring system to its state when it crashed and then fix
- **Log the undo**: to fully repeat the history, including the undo operations. BUT we never undo the undo operations.

ARIES log record data structure

Log record

- **Log**: chronologic sequence of log entries
- **Log tail**: the portion of the log in main memory (not forced yet)
- **Log sequence numbers** (LSN): strictly increasing IDs for log records

Log record types and fields

- ***All****: prevLSN, transID, type*
- **Update**: pageID, length, offset, before-image, after-image
- **Commit**, **Abort**, **End**
- **Compensation** (CLR): undoNextLSN

ARIES additional data structures

Dirty pages table

- One entry per page not written to disk yet
- **Fields**: pageID, recLSN

Transactions table

- One entry per transaction
- **Fields**: transID, lastLSN, status
- Entries with status **committed** or **aborted** are removed from the Table when the corresponding transaction reaches the **end** state

Exercise 1

1) If we can guarantee that uncommitted data is never written to disk, is **undo** still necessary?
2) What about **redo**?

3) If updates are always forced to disk when a transaction commits, is **undo** still necessary?
4) What about **redo**?

Exercise 2

LSN	PrevLSN	TransID	Type	PageID
1	0	T1	update	C
2	0	T2	update	B
3	1	T1	commit	
4 5			begin checkpoint end checkpoint	
6	3	T1	end	
7	0	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	

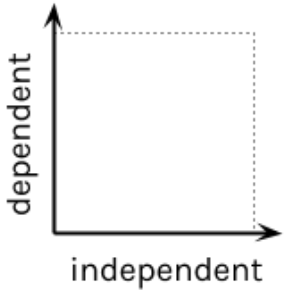
Show the transaction table and the dirty pages table at LSN 6,8,10

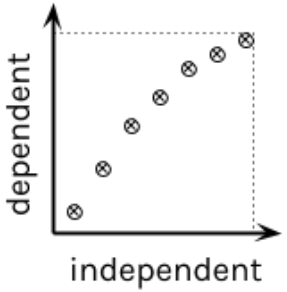
Exercise 3: Techniques for Performance

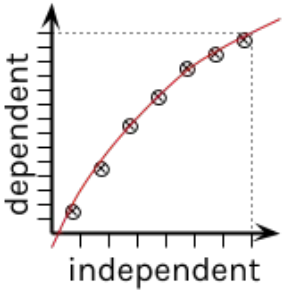
1. What is latency?
2. What is throughput?
3. How latency and throughput are related in the case of a serial execution? Concurrent execution?
4. How concurency affects latency and throughput?

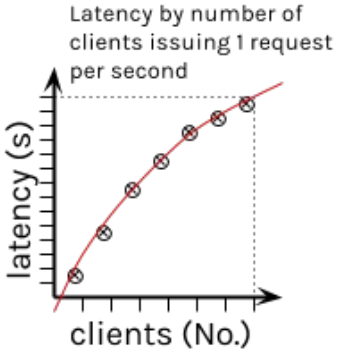
Notes on graphs: draw your graph

- 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



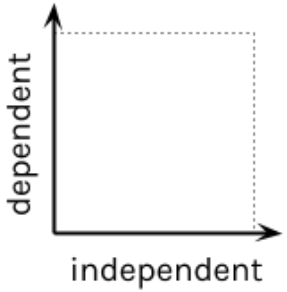


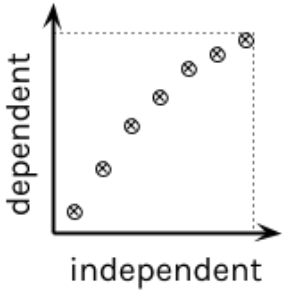


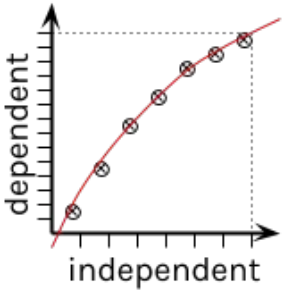


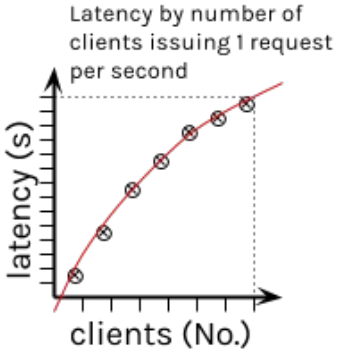
Notes on graphs: fill your graph with points

- 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
- Remove any strange bends in your curve



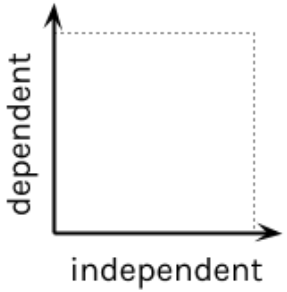


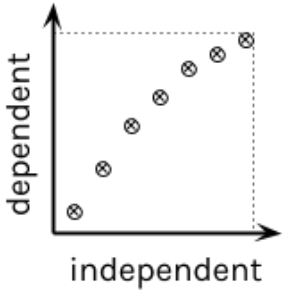


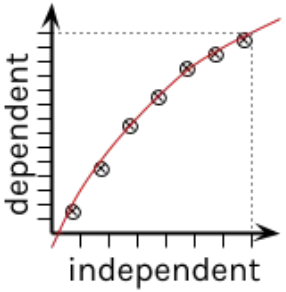


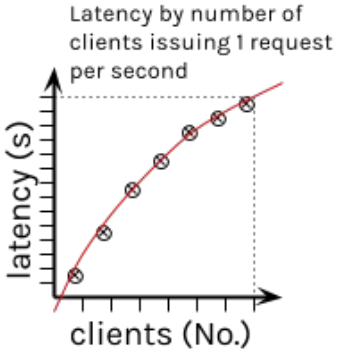
Notes on graphs: present your graph

- 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





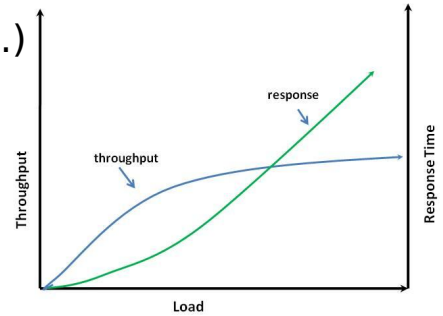




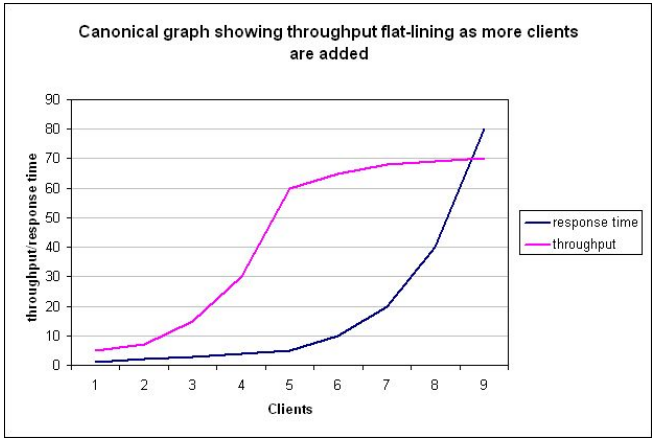
Exercise on graphs

What is missing from each of these graphs?

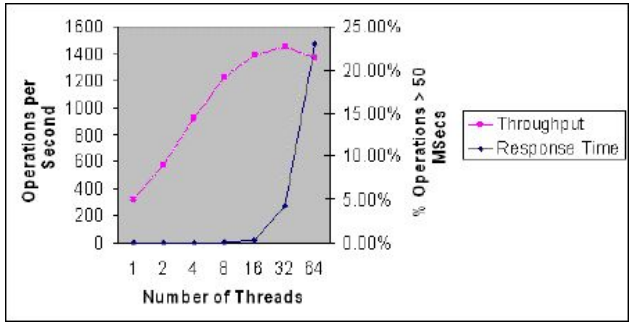
a.)



b.)



c.)



Stack Exchange computer science

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

Thank you

Alexander, Lasse, Svend, Yiwen, and Yuan

Department of computer science, University of Copenhagen

Academic year 2018-2019, Block 2