Scalable Web Programming

CS193S - Jan Jannink - 1/12/10

Administrative Stuff

- Computer Forum Career Fair: Wed. 13, 11AM-4PM
    - (Just in case you hadn't seen the tent go up)
- Any problems with MySQL setup?
- Review: web coding is complex, power law has ramifications everywhere
- Feedback: time to step on the gas
- Office space: Gates B28
- Website: http://cs193s.stanford.edu

Weekly Syllabus

1.Scalability: *(Jan.)*

2.Agile Practices

3.Ecology/Mashups*

4.Browser/Client

5.Data/Server: *(Feb.)*

6.Security/Privacy

7.Analytics*

8.Cloud/Map-Reduce

9.Publish APIs: *(Mar.)**

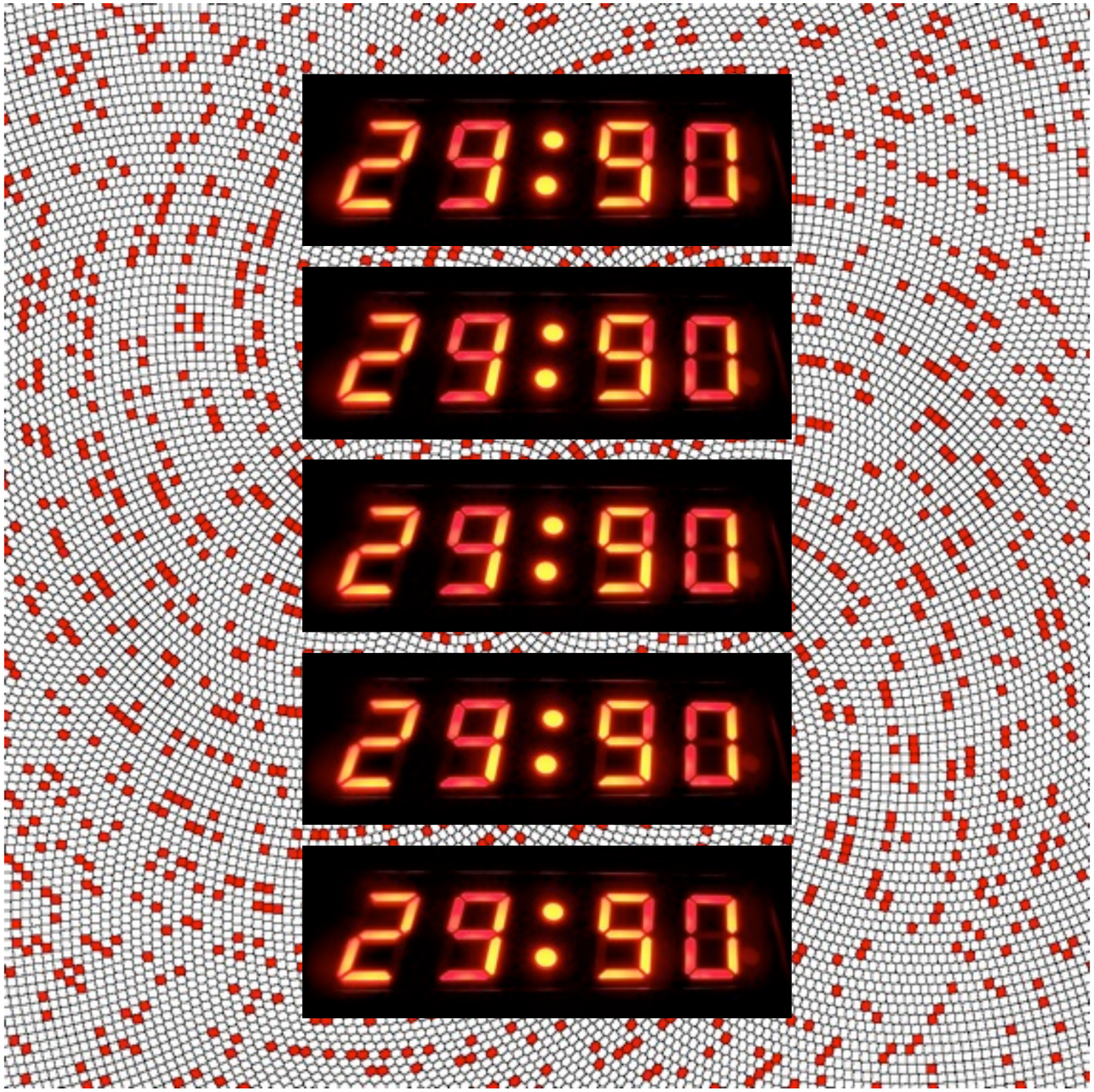
10. Future

* assignment due

Programming Project #1

- Adapt GWT sample application
- Turn existing functionality into an integration test
- Display random 20 digit number in a creative way, e.g.
  - use external captcha code
  - generate funny image from number
- Add several unit tests to validate new code
- Link to a test result display from main page

Sample Mockup



Tests Project Scope

Assignment Motivation

- Grading confined to code & test correctness and test coverage
- Project creativity aids team grouping for larger projects
- Best pages become candidates for demo to angel investor group

- Demo meeting & lunch will take place in March
  - all participants invited to attend
  - 3-6 demos will be presented
  - open discussion with investors follows

Team Formation

- Goal: mirror realistic environment
- Team structure:
    - 1-2 leaders based on creativity of first assignment
    - 2-4 contributors who can work in teams
    - 2-4 consultants who will work on individual projects
- No grade competition:
    - creativity competition for demos

Agile Testing

- Unit tests
    - simple demonstrations of code behavior
- Integration tests
    - show composite behavior of code
- Regression testing
    - essentially application of above tests
- Performance testing

Tests

- Form a living institutional memory of the software
- Communicate developers' intentions and actual accomplishments
- Facilitate greater distribution of coding tasks
- Simplify surgical replacement of code at any level of embedding
- Document APIs and dynamic behaviors
- Maintain code performance over time

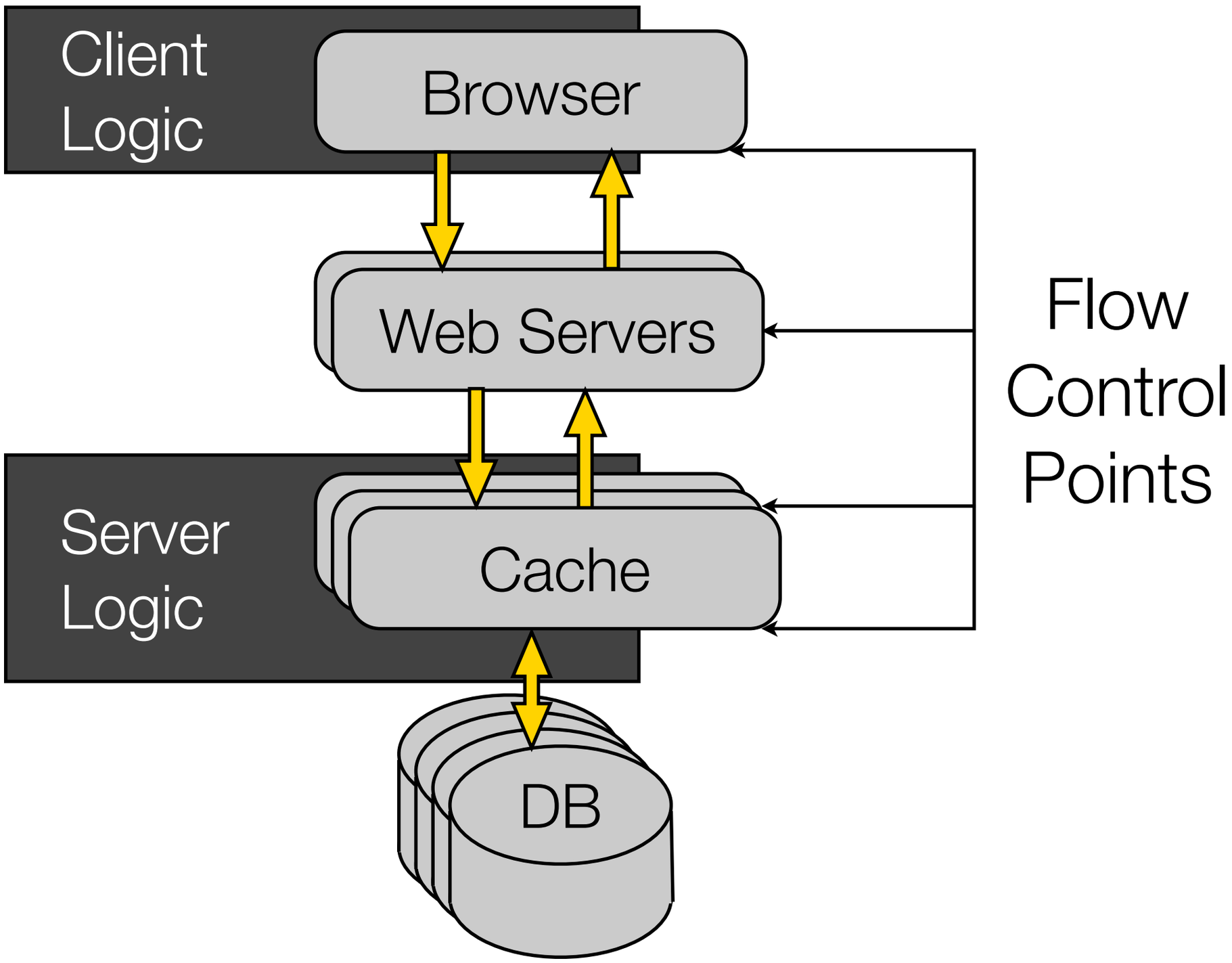
IBM Example

- DB2 relational database first SQL database
- Oracle quickly overtook it in prominence commercially
- During my stint in IBM (‘96-97)
  - DB engine core had been frozen for years
  - R&D engineers were needed to modify large parts of codebase
- Thousands of test cases for the system
  - lack of internal tests turned the query engine into a black box

Back to Software

- So many layers, so little time
- Data flow is the starting point
    - Identify flow control areas
    - Define APIs there
- Use test harnesses to iterate development
    - build back to DB, simulate horizontal scaling
- Start continuous integration

Website Data Path



Website Design Principle

- Flesh out the system front end to back end
    - figure out what people want to see
    - mock up the look, then data links, flesh them out
- Design the data flow back to front
    - use browser view to minimize data satisfying queries
    - keep the schema as simple as possible

Test Design Principles

- Least effort simulation
    - Often results in simpler program logic as well
    - Produces leaner faster test suites
- Save corner cases for after program logic development
    - Avoid guessing what will be important
- Document bug fixes with a new test or test parameters

A Panacea?

- Testing can be overkill in some applications
- UI can change extremely fast
  - overhead of test changes can be prohibitive
  - customers can end up testers (perpetual beta)
- Small, well understood and encapsulated code
  - making code private is a performance guarantee
  - often integration tested by the code that uses it

Cost Benefit Analysis

- Always exhaustively test published APIs
    - distributing sample code teaches use of APIs
- Test based on frequency of code reuse
- Test based on code volatility
- Skip tests only when code verification is enforced
    - either through end user testing, external test suites

Perpetual Beta

- User testing OK
  - does it chase away users?
- Benefits outweigh drawbacks?
- Case study: Cuil
  - you are your first impression
- Case study: Gmail
  - GB’s of storage

Testing in Practice

- Canned data access via a text file
    - Output a simple DB query to text file and use it for a test harness
- Simulate memcached with a hash table
- Use test framework to memorialize DB schema
    - Even use test suite to populate new DB instances
- Server virtualization enables scale out testing

Continuous Integration

- Concept: tests run automatically at every checkin
- Highly effective when combined with development using git
- Enforces simple test development
- Serves as another developer communication tool
- Ant: XML definition of development, test, production builds
  - not 100% trivial to write first scripts
- Cruise Control: well documented, very configurable

Minimal Releasable Code

- Define your customer
    - self, team, die hard fans, casual user
    - enterprise vs. consumer
- Make sure the value outweighs the pain
- Make sure the upgrade process outweighs the pain
- Divide, Conquer, Release, Iterate

Lessons Learned

- Tests can be extraordinarily valuable
    - the longer your code lasts the more valuable it becomes
    - favors group productivity over individual productivity
- Improved communication is the unexpected benefit
    - speeds up integration of new engineers
- Like everything it is amenable to cost benefit analysis
- Continuous integration further speeds development cycle

Scale Out Ideas

- Server virtualization
    - use when not single resource bound
- Automatic server allocation
    - demands simple server setup
- Database replication and partitioning
    - simple key based partitioning is most feasible
    - avoid data loss and increase performance

Worth Checking Out

- Junit
  - http://ww.junit.org/
- The Tipping Point, Malcolm Gladwell
- Ant
  - http://ant.apache.org/
- Cruise Control
  - http://cruisecontrol.sourceforge.net//

Q & A Topics

- Assignment #1
- Project team definition
- Top Down vs. Bottom Up
  - hybrid approach
- 80-20 rule for test development
  - maximize coverage at minimum effort
- Startup development vs. Steady State maintenance