Introduction to Writing a Thesis

Tips, Tools, and Recommendations

Hein Meling

University of Stavanger

hein.meling@uis.no

August 2016

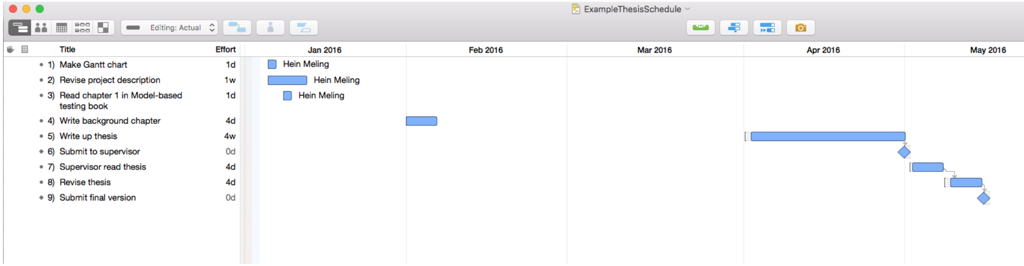
Planning Your Project Work

- Deadline: December 23
- Absolutely no extensions!

Planning Your Thesis Work

- Deadlines:
    - Bachelor: May 15
    - Master: June 15
    - Only the dean can grant extensions (but he never does!)
    - Extending due to sick leave: need sykemelding from doctor;
        - usually only if more than a week of illness
- Not a whole lot of time!
- Need careful planning
- Important to stay focused

Gantt Chart

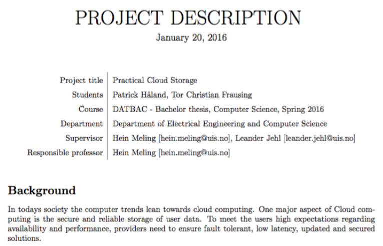


	Title	Effort
1)	Make Gantt chart	1d
2)	Revise project description	1w
3)	Read chapter 1 in Model-based testing book	1d
4)	Write background chapter	4d
5)	Write up thesis	4w
6)	Submit to supervisor	0d
7)	Supervisor read thesis	4d
8)	Revise thesis	4d
9)	Submit final version	0d

Project Description

- Project descriptions are in some cases a bit vague
- Project proposer does not always know what he/she wants
- Write your own version
- Headings:
  - Background
  - Motivation
  - Objectives
  - Gantt Chart with Tasks and Milestones
- Submit project description by **Monday August 29**

Project Description



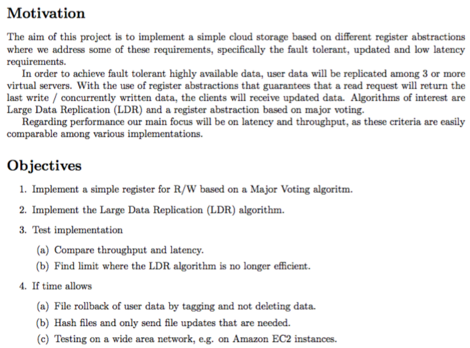
PROJECT DESCRIPTION

January 20, 2016

Project title	Practical Cloud Storage
Students	Patrick Håland, Tor Christian Frausing
Course	DATBAC - Bachelor thesis, Computer Science, Spring 2016
Department	Department of Electrical Engineering and Computer Science
Supervisor	Hein Meling [hein.meling@uis.no], Leander Jehl [leander.jehl@uis.no]
Responsible professor	Hein Meling [hein.meling@uis.no]

Background

In todays society the computer trends lean towards cloud computing. One major aspect of Cloud computing is the secure and reliable storage of user data. To meet the users high expectations regarding availability and performance, providers need to ensure fault tolerant, low latency, updated and secured solutions.



Motivation

The aim of this project is to implement a simple cloud storage based on different register abstractions where we address some of these requirements, specifically the fault tolerant, updated and low latency requirements.

In order to achieve fault tolerant highly available data, user data will be replicated among 3 or more virtual servers. With the use of register abstractions that guarantees that a read request will return the last write / concurrently written data, the clients will receive updated data. Algorithms of interest are Large Data Replication (LDR) and a register abstraction based on major voting.

Regarding performance our main focus will be on latency and throughput, as these criteria are easily comparable among various implementations.

Objectives

1. Implement a simple register for R/W based on a Major Voting algoritm.
2. Implement the Large Data Replication (LDR) algorithm.
3. Test implementation
   - (a) Compare throughput and latency.
   - (b) Find limit where the LDR algorithm is no longer efficient.
4. If time allows
   - (a) File rollback of user data by tagging and not deleting data.
   - (b) Hash files and only send file updates that are needed.
   - (c) Testing on a wide area network, e.g. on Amazon EC2 instances.

Work Process

- Weekly supervision: 20-30 minutes
  - Must be efficient; come well-prepared
  - You are expected to drive the meetings
  - What's been done; what are the challenges
  - Action Points for next meeting

Work Process

- Weekly supervision: 20-30 minutes
  - Must be efficient; come well-prepared
  - You are expected to drive the meetings
  - What's been done; what are the challenges
  - Action Points for next meeting
- After each meeting
  - Minutes from meeting; two sections
    - Short summary of work done (max 5 lines)
    - Action Points for coming week
  - Upload to shared Dropbox folder

Work Process

- Weekly supervision: 20-30 minutes
    - Must be efficient; come well-prepared
    - You are expected to drive the meetings
    - What's been done; what are the challenges
    - Action Points for next meeting
- After each meeting
    - Minutes from meeting; two sections
        - Short summary of work done (max 5 lines)
        - Action Points for coming week
    - Upload to shared Dropbox folder
- Need more assistance
    - Schedule separate meeting with your advisor

Work Process (2)

- Towards the final report
  - Two deliverables and final report
  - (Read and comment on each others reports)
  - Learn a lot from reading what others write

Work Process (2)

- Towards the final report
  - Two deliverables and final report
  - (Read and comment on each others reports)
  - Learn a lot from reading what others write
- Deliverables:
  - October 3
  - November 7
  - December 23 (final report)

Work Process (3)

- Your advisor will read and comment on your report too:
  - Final report
  - Some of the deliverables
- In some cases: review code
- Will help as much as I can by email or Slack

When to Start Writing?

When to Start Writing?

- Today!

When to Start Writing?

- Today!
- You can set up the document

When to Start Writing?

- Today!
- You can set up the document
- Prepare a disposition (fill in the headings)

When to Start Writing?

- Today!
- You can set up the document
- Prepare a disposition (fill in the headings)
- You can always change things later

How to structure the thesis report

- Abstract (300-500 words)
- Introduction (3-5 pages)
- Background, Technology Review, Related Work (10-20 p.)
- Design, Model, Architecture (10-30 p.)
- Implementation (10-20 p.)
  - Technical details of interest, optimizations, algorithms, ...
  - Don't explain every bit of code; only the interesting parts
- Evaluation, Discussion, Insights (10-30 p.)
- Conclusion (300-500 words)

How to structure the project report

- Same as for thesis report
  - Except scaled down to 10-16 pages.
- Double column format.

Tips for the report content

- Name chapters appropriately; other names than those on previous slide is ok.
- Start Chapter/Section with a brief summary of what the reader can expect to learn from reading it.
    - Thus far, we have presented Replacement for a single Paxos instance. In this section, we explain how to apply Replacement to an RSM that executes a sequence of Paxos instances.
- Insert paragraph breaks (avoid too long paragraphs)

How to Work?

How to Work?

- Switch between coding and writing!

How to Work?

- Switch between coding and writing!
- Expect to rewrite everything you write;
    - So don’t spend time making it perfect

How to Work?

- Switch between coding and writing!
- Expect to rewrite everything you write;
  - So don't spend time making it perfect
- Iterate to improve report to make it perfect

How to Work?

- Switch between coding and writing!
- Expect to rewrite everything you write;
  - So don't spend time making it perfect
- Iterate to improve report to make it perfect
- Why work this way?

How to Work?

- Switch between coding and writing!
- Expect to rewrite everything you write;
  - So don't spend time making it perfect
- Iterate to improve report to make it perfect
- Why work this way?
  - Two modes of thinking
  - Makes you think differently about the problem

Specification: Thinking Before Coding

- Write specifications:
  - high-level specification of system
  - low-level specification for each function
- Textual specifications are fine!
- Also: Express ideas in diagrams and figures
- Simplicity

Modeling Languages

- Unified Modeling Language (UML)
  - Use Case Diagrams
  - **Class Diagrams** (Entity Relation)
  - **Sequence Charts**
  - **State Machine Diagrams**
- **Flow Charts**
- Work Flow Diagrams
- Data Flow Diagrams
- Wireframe / UI Mockups
- Infographics

Tools

- git and github
- Dropbox
- LaTeX/bibtex for the report
- Slack for chat rooms (Gophers/general)

Conclusions

- I have high expectations
- Want you to succeed!