5DV129: Examensarbete för
kandidatexamen i datavetenskap, 15.0 hp

Planning

Carl-Evert Kangas
dv14cks@cs.umu.se
March 31, 2016

Background

A company making games have noticed that data integrity in some rare cases is compromised in the back end systems. These systems are web applications written in Java and hosted by the Google on their App Engine platform. Data is stored in the so called Datastore which is a schemaless, NoSQL database. A memory cache, also provided by the App Engine platform is used for speed and to keep costs down. An open source library called Objectify is used for convenient access to the datastore.

Problem description

The customer have observed entities where data integrity is compromised. An object { dead: false, alive: true } describing the state of a cat seems in rare cases end up having impossible states like { dead: false, alive: false }[1].

Updates to the datastore are transactional so the problem probably lies somewhere else. One hypothesis is that the cache is updated field-by-field and that the cache update may be interrupted before it is finnished. Another possible explanation to the behaviour could be related to the fact that values may be evicted from the cache if the cache is low on memory.

The questions to be researched in the thesis are as follows:

1. Can the alleged observations be reproduced?
An essential question is whether this problem can be reproduced in a systematic way that support the hypothesis. While data integrity compromised entities[2] have been observed, the cause of the problem may be something completley different than discussed above.

2. Describe the implementation of Objectify from the perspective on how transactions are handled
The notion of transactions seems to be used mostly in the context of databases. In this case, the database itself claims to handle transactions. Can the concept of trasactions be extended to include the caching service and what would this mean in terms of additional computations and database accesses?

3. Which apparent solutions could be devised?
Could some well known ways of performing transactions be reused in this context? In case the patterns aren't applicable, then why not?

[1] The actual objects are both larger and more complex
[2] Data objects in the Datastore are known as *entities*

Methods

Since the companys issues comes from a live system it's data cannot be used for experiments. Conequently, to adress the first question a small system will be built to research the problem. A positive demonstration will acknowledge the problem. There are some obvious variables that needs to be included in the experiment: the execution time (the execution is shut down after 30 seconds in the platform) and the size of the objects to be stored. A more specific method for performing the experiment and handling the statistics will be discussed in the actual thesis.

The second question appears to be a litterature study of ways of doing transactions. While not a very big area in itself when only classical databases are concerned, I am looking forward to see what studies there are on globally distributed databases.

Lastly, can a regular transaction-pattern be applied to make Objectify transactional? This may or may not be doable within the given timeframe. If a complete solution appears to be out of reach, then at least some conclusions regarding such an implementation from the study described above should be drawn.

Other aspects and clarifications

Ethics and social considerations

None.

Regarding the company mentioned

I am not in any way formally associated with the company mentioned. I am not so to speak "doing my thesis there" and I'm not formally supervised by them.

This is an issue that needs to be addressed somehow. Either by ignoring the source of the problem (and lose some context) or formally make an agreement about referring to their enterprise.

Main focus

The main focus is on the Objectify library. There are alternative API:s for accessing datastore as well as the native API. A full review of them does not seem to add valuable information to the questions as they are formulated at the moment. Still, a brief review of the most common alternatives could be done in the Discussion chapter.

Planning

It is roughly seven weeks till the nearly completed manuscript should be handed in at May 20. Making a complete and meaningful plan for the whole project is not possible, so I have deliberatly not put any effort into planning the last weeks.

Week 14, 4/4–8/4
High priority to making a somewhat working program that can be used for answering question one and design the experiment. Write an introdction and read up on relevant things.

Week 15, 11/4–15/4
Need to get into how to deal with question two, i.e. what kinds of notions to use. Contact developers working with Objectify regarding their design choices. Write method-chapter. Introduction should be finalized.

Week 16, 18/4–22/4
Since the mid seminar is scheduled the monday after week 16, at least questions 1 and 2 stated in the problem description earlier needs to either have an answer or a defined type of answer. Otherwise they need to be reformulated urgently. Method-chapter should be more or less final.

Litterature

There are loads of litterature regarding databases and my starting point is the textbooks used at the database courses held by the department: Henry F. Korth – Database System Concepts and Elmarsi Ramex – Fundamentals of Database Systems. The latter cover in its latest edition both distributed databases and NoSQL systems.

Also there seems to exist a lot of high quality papers by major companies in this field.

The source code for Objectify seems well documented and should investigated and some contact with the community around it needs to be established.

Typesetting and source code

I'm planning to write the thesis in LaTeX mainly because of the BibTeX reference management. A public Git repository at `https://github.com/carlevert/bt` has been set up. The repository so far contains mostly boilerplate LaTeX, but will later be accompanied by source code for the projects.

Risks and risk management

One obvious risk is that this project fails at the first question, i.e. it's not possible to demonstrate the inconsitencies. To make the impact of this happening a demonstration of the problem is of top priority, that is, scheduled as soon as possible.

Another risk is that the theory regarding this problem is too straightforward. While not an optimal solution because it may cause more breadth than depth is extending the area of research can be a solution. A total shift in focus is the ultimate but unwanted solution.