CSC 560	Modern DBMS Architectures	Alexander Dekhtyar

Course Project Overview
Team Formation

Due date: Thursday, September 24, in-class

Course Project Overview

In most graduate courses I taught at Cal Poly to date[1] we always had a quarter-long team-based research project, with each team selecting its own project topic.

In this course, we will take a different approach. The overarching goal of this course is for you (and for me) to understand how non-relational (NoSQL) DBMS operate, why they are needed, and how to build them. As such, our course project is to build a non-relational DBMS[2]. To make things more interesting (this *is*a graduate course, after all), each team will select a specific type of a non-relational DBMS it wants to build. So, while we maintain a certain level of freedom of choice and autonomy for the teams, the project will much more regulated than before.

For a class of about 20-22 students, I would like to see four or five teams formed. See my notes on team formation below.

Nature of the Project

The project will consist of a number of stages and assignments and will proceed roughly as follows:

- **Team formation.** During the first week of classes we form the teams.

[1] The sole exception being a CSC 570: Bioinformatics Algorithms course which had outside customers for the project assignments.

[2] Even this is a bit of a misnomer - at least one of the possible options for the project is a relational DBMS but witha non-traditional approach to data storage.

- **Topic selection.** Each team will select, from a given list of DBMS architectures (and archetypical representatives), the one DBMS architecture it would like implement as a prototype.
- **Brainstorming.** Early in the class (week 2, most likely), each team will put together an architectural diagram of the DBMS it is trying to build and will present the diagram to the rest of the class.
- **Background work.** For the rest of the quarter, work on the project will proceed in the following way. For each component of the system identified on the architectural diagram of the DBMS, the team will research how it is implemented in the archetypical systems of its kind, will make decisions on how it needs to be implemented in the project and present the design (complete with explanation of why it was chosen) to the class.
- **Implementation.** The team will then proceed to implement the specified component, while starting the background work on the next component in the system.

This approach to the project resembles somewhat the *just-in-time knowledge* approach of learning with a key major difference. Each team will present its decisions to the full class, and will participate in discussion of the decisions of other teams (thus learning more than is strictly necessary to complete the project).

The **brainstorming** part of the project will yield a list of components each team needs to design and develop. While each DBMS architecture is unique, all DBMS architectures have a basic set of components that need to be built.

Tentatively, we reserve one day a week (on most of the weeks) for presentations of component designs, with the same/similar types of components discussed on the same day (the expectation is that each team gets to present its challenges and solutions on that day, unless the team does not have a specific type of component).

While I will be glad to provide starting points for literature search, each team is expected to conduct its own literature searches when necessary and create and maintain a project bibliography. We will eventually combine all materials used by the teams into a single "mega" bibliography for the future generations to enjoy and use[3]

We are building distributed DBMS

One key wrinkle of the project is that the DBMS each team is building needs to be distributed, i.e., it needs to support shared nothing data partition and

[3] On the bright side, except for the first assignment asking you to read some "State of the database research" papers, you won't be assigned to read and prepare reports on anything else. I may distribute some papers to you as part of lecture materials though.

quering of data speard over multiple servers. Each team will be provided with five *basic* VM instances on the department's new NetApp computing appliance. The VM instances are *on purpose* not very powerful - this allows us to justify the need to building a distributed DBMS to deploy over these resources.

Whether each team is expected to present a solution that utilizes all five machines is subject to negotiations (depending on the project selected and the needs of the particular DBMS architectures), however, the use of at least two machines is required for all projects.

We are building prototypes

To put it bluntly, when compared to the actual NoSQL DBMS systems (BigTable, MongoDB, Memcached/CouchBase, Cassandra, Dynamo, MarkLogic, and so on), your sysyems will be very inefficient. No team is expected to be able to build a fully working highly efficient DBMS in a matter of less than 10 weeks. But this is not the point of this project.

The key reason for the project, is to get you to see what it takes to build a modern DBMS, to observe the decisions that need to happen and to learn how to implement a complete system.

We are building from scratch

.... which is why it is very imortant for you to build your system **from scratch.** Many of the NoSQL and relational DBMS are open source projects, their code is widely available. It may seem tempting to use the codebase of an open source project to aid in your system development, but in this class, it will be considered **cheating** to do so.

You will be allowed to use some supplementary tools (libraries) in developing your code, but each team is responsible for its own implementations of all major algorithms and design decisions.

Team Formation

Our first order of business is to build teams. The constraints/requirements are:

- **Size.** Between three and five people. I prefer teams of size *four* or *five*. Larger teams proved to be inefficient in the context of a graduate course. Smaller teams may yield more work than necessary for each team member.
- **Structure.** I am releasing a list of DBMS architectures as a separate document. A natural way to organize a team, is to do so around a shared common goal - i.e., all students who would like to work on implementing a specific DBMS architecture can form a team.

- **Chemistry.** You will be doing a lot of different things, small, and not so small, as a team. When forming teams, please make sure each teammate is ready to work with others on the team. You should be willing to pull your weight, actively contribute to the team discourse, and, yes, on occasion, when things don't go as expected, cover for your teammates, as your teammates would cover for you. Time is too precious in this course to spend it dealing with personality issues.

Deliverable. By classtime *Thursday, September 24* one person from each team needs to send me an email at dekhtyar@calpoly.edu with the team name (**pick a team name!**) and the list of team members and their email addresses.