eXascale Infolab

System-oriented Programming, Prof. Philippe Cudré-Mauroux, Natalia Ostapuk, Abdelouahab Khelifati, Zeno Bardelli

Handout

Project P01:

Fast Small In-Memory Triple Datastructures

1. Introduction

The goal of this project is to get accommodated with the C language and different targets it can be compiled to.

Learning goals are:

- Creating a C project from scratch.
- Working with C language concepts. (Arrays, Pointers)
- Incorporating resources from the internet.
- Autonomously figuring out how to compile C code to a different target (WebAssembly).

Your task is to implement a triple data structure in memory (Triplets always have exactly three values *S*, *P* and *O*). This structure shall be optimized for fast access with arbitrary matching patterns ({*S*}, {*P*},{*O*}, {*SP*}, {*PO*}, {*SO*}, {*SPO*}).

Finally we will test your data structure as WebAssemly Code on any modern webbrowser.

Deliverables to be ready before the final course on Wed, 27. May 2020:

- The runnable code, including testing scripts provided on GitLab with working automated build scripts.
- One individual page report per participant (Approach chosen, Problems encountered, Lessons learned.).
- A short two pages documentation per group.
- Preparation of a final presentation per group (3min/Person).

2. Organization / Groups

For the project you can build up groups with 4 students. If you wish you can have a group with only 2 or 3 people, but you still have to accomplish all tasks as if you were 4.

Announce your group name and teammates in the discussion "Formed groups" in the forum on Moodle. If you need to find teammates, create a new discussion in that forum. By April 7 at the latest all students must be part of a group. Those who are not will be automatically assigned to a group.

Create also your **Group** on *diuf-gitlab.unifr.ch* with the name *SOP2020_P01_<group_name>*. Add everybody to your Group as **Owners**. Inside the groups **Namespace** you can create the project *SOP2020_P01* which will hold the code.

3. Minimal Feature Set

- Add new Triplets through a simple function with the following signature.

  ```
  int insert(char* s, char* p, char* o)
  ```

  The provided variables need to get copied over to memory allocated by your code. The return value of the function is `0` if there was no problem. (Errors can be specified at will.)

- Allocate the memory dynamically with malloc. (Every value in {*S*, *P*, *O*} can hold at least 1024 chars.)

- Provide a query function to find Triples in which one or multiple values can be fixed:

  ```
  int match(char* s, char* p, char* o, long result)
  ```

  `char* s,p,o` Specifies pointers for the query variables and the return variables. If we look for examples for all triples where *S* is set to a specific string, we only provide `s` and initialize the other strings with `0`s. (see example)

  `long result` Specifies which result shall be returned, counting from `0` upwards.

  The result is returned by writing the answers in the initialized variables.

  Then answer of the function is `0` if everything is working correctly and `1` if there is no result found. (Other errors can be specified at will.)

- Implement a feature to delete triples.

- Integrate the provided benchmark with your code and measure the performance.

For the WebAssembly project:

- Show that your code works as *WebAssembly* in the web browser.
- Render the content as a Graph Drawing. (E.g. with d3js).
- Create the same functionality in JavaScript and compare the speed.

Example

```
insert("SOP2020", "has teacher", "PCM")              => returns 0
insert("SOP2020", "topic is", "C")                   => returns 0
insert("PCM", "lastname", "Cudre-Mauroux")           => returns 0

match("SOP2020",p,o,0)     => returns 0, sets p to "has teacher", sets o to "PCM"
match("SOP2020",p,o,1)     => returns 0, sets p to "topic is", sets o to "C"
match("SOP2020",p,o,2)     => returns 1 (no more matches found)
match(s,"topic is","C",0)  => returns 0, sets s to "SOP2020"
```

4. Additional Features (propositions)

- Optimize for speed of inserting.
- Allow dynamic size of values.
- Optimize for size in memory.
- Load data from N-Triples (https://en.wikipedia.org/wiki/N-Triples) files. (e.g. http://classifications.data.admin.ch/municipality/2196?format=nt)
- Insert additional values (for example the date and time of insertion) into the database.

5. Implementation decisions

- Which data structure to hold the pointers / data. Array, Linked-List, …
- Values inline or separated structures?
- How to access/query the data. Keeping indexes, multiple?
- The architecture of your code, structure of files.
- If there are multiple results for one match, how to assure that the order stays the same.

Report the advantages and disadvantages of your implementation decisions regarding write/read performance and memory usage.

6. Resources for the different target.

Webbrowser: WebAssembly

http://webassembly.org/getting-started/developers-guide/
http://kripken.github.io/emscripten-site/docs/getting_started/Tutorial.html

System-oriented Programming, Prof. Philippe Cudré-Mauroux, Natalia Ostapuk, Abdelouahab Khelifati, Zeno Bardelli

7. Evaluation Key

The following aspects of your work will be taken into account for the final mark with the according weights.

Implemented Features	**60%**
Minimal Feature Set	25%
Compiles and Runs on other Target	10 %
Testing program / script	10%
Additional Features	15%
Code Quality	**20%**
Simple and Concise Code	10%
Architecture and Helperfiles (Makefile)	10%
Documentation	**20%**
Final 3 min/Person Presentation	10%
The time limit is respected.	
Content (show something exciting!)	
Short Documentation (max. 2 pages per Group)	10%
Usage of Store and Testing	
Architecture and Decisions	