НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Database mapped storage

Kupriyanov Mikhail

Udalov Ilia

Software Engineering, Faculty of Computer Science

Containers overview

- Different types of data structures:
  - map, set
  - vector
  - list
  - ...
- Template types of elements - user defined data storing
- Specific access to data
  - direct access: *operator*[]
  - iterative approach: *begin*(), *end*(), *operator*++
  - ...

State the problem

Data domain:

- homogenius data
- Large amount of data(much more that RAM)
- Simple data operations are needed

Wishes:

- Native C++ syntax
- High level of abstraction

Solution idea

Container stores data into Database

- Native C++ statements
  - no additional DB logic
- Conditional access to data
  - batch-based data access
- Iterative data proccessing approach
  - handle value by value

Container creation

```
Datapack<std::string, int, double> pack("MyPack", ...);
```

1. Variadic template to describe stored data
2. Container's "name" to get access to data in the future

Data inserting

```
pack.Push("MyString", 10, 2.3);
```

1. Strict parameters order
2. Simple objects placing

Data batch requesting/removing

```
pack.Request(pack[1] == 10);
```

- Upload the batch of data using specific condition
- Use *operator*[] to determine field
- To remove data just use *Remove* instead of *Request*

Conditions:

!=, ==, >=, <=, >, <, …

Use “like” for strings!

Logic operators:

&, |, !

Data field requesting

```
int value;
while(pack.Get(NULL, &value, NULL))
        std::cout << value;
```

1. Iteratively obtain data field from data batch
2. Use pointers to skip uninteresting values
3. When the batch is ended method returns *false*

Benchmark SQLite

- 10k adding operations(INSERT)
  - AVG: 0.02s, Total: 115.4s
- 3 type of reqeuest(SELECT)
  - simple request(SELECT *)
    - 0.016176
  - simple request that return about half of data(random fields)
    - 0.006836
  - huge request with 100+ conditions
    - 0.011508
  - string “like” operation
    - 0.011865

Benchmark PostgreSQL

- 10k adding operations(INSERT)
  - AVG: 0.0158s, Total: 62s
- 3 type of reqeuest(SELECT)
  - simple request(SELECT *)
    - 0.0335
  - simple request that return about half of data(random fields)
    - 0.009336
  - huge request with 100+ conditions
    - 0.051508

Results

- Open source POSIX compatible library
- Portable and platform independent storage, concurrent access, high-speed processing
- Support SQLite and PostgreSQL
- Extendable by adding new DB wrapper using API
- Benchmarks for (SQLite and PostgreSQL)

Responsibility

- Ilia Udalov:
  - Generic architecture
  - Interaction with external databases API
  - DB Wrappers
  - PostgreSQL server management
- Mikhail Kuprianov:
  - Generic architecture;
  - C++ interfaces
  - Dynamic queries generation
  - Benchmark implementation

Future plans

- Smart batch processing with auto data loading
- Extra database management systems support
- Windows OS support(Windows DBMS as well)
- Extended data types support
- Database performance optimization

Thank you for attention!