Initialization of iZeneLib Report–Access Methods Library

Kevin Hu, Yingfeng Zhang

October 31, 2008

Contents

Abstract

This report is as a start of AM-lib (Access Methods library) in iZeneLib(the project name is under discussion of Yingfeng and me, if Yeogirl gets a better idea, we will change it). The goal of writing this report is to make sure we have same understanding of the purposes and approaches of building a well encapsulated C++ library for frequently used modules in aspects of access methods. This report describes the reasons, the initial designs of AM-lib, the probable usages of the future AM-lib and the works and schedules need to be done. At the end of report, some questions are listed as a feedback answer sheet, in order to get a basic understanding of your feedback. And some following detail designs will come soon.

1 Why do we need AM-lib?

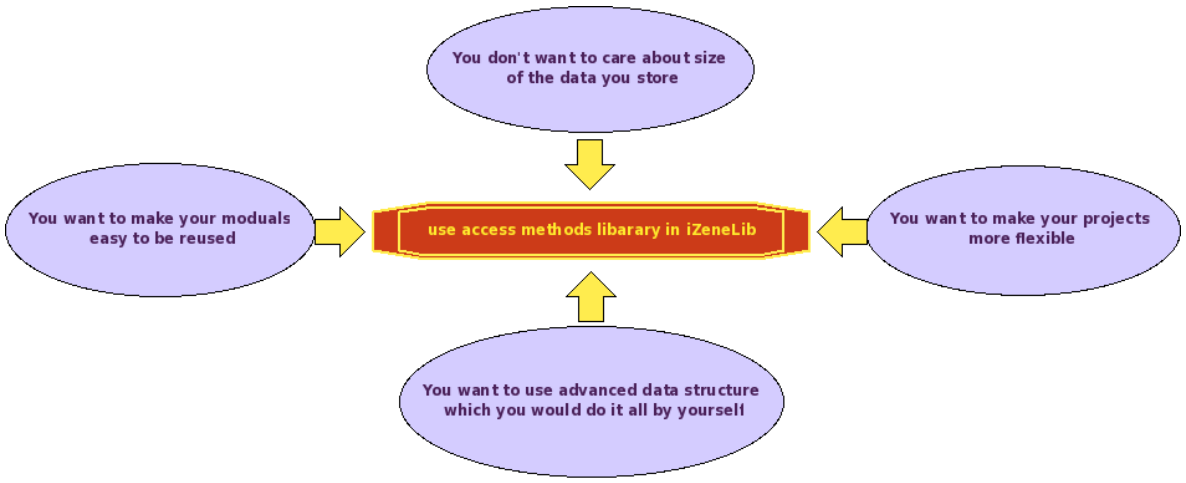

Figure 1: Advantages of AM-lib

2 Initial Architecture Design of Classes in AM-lib

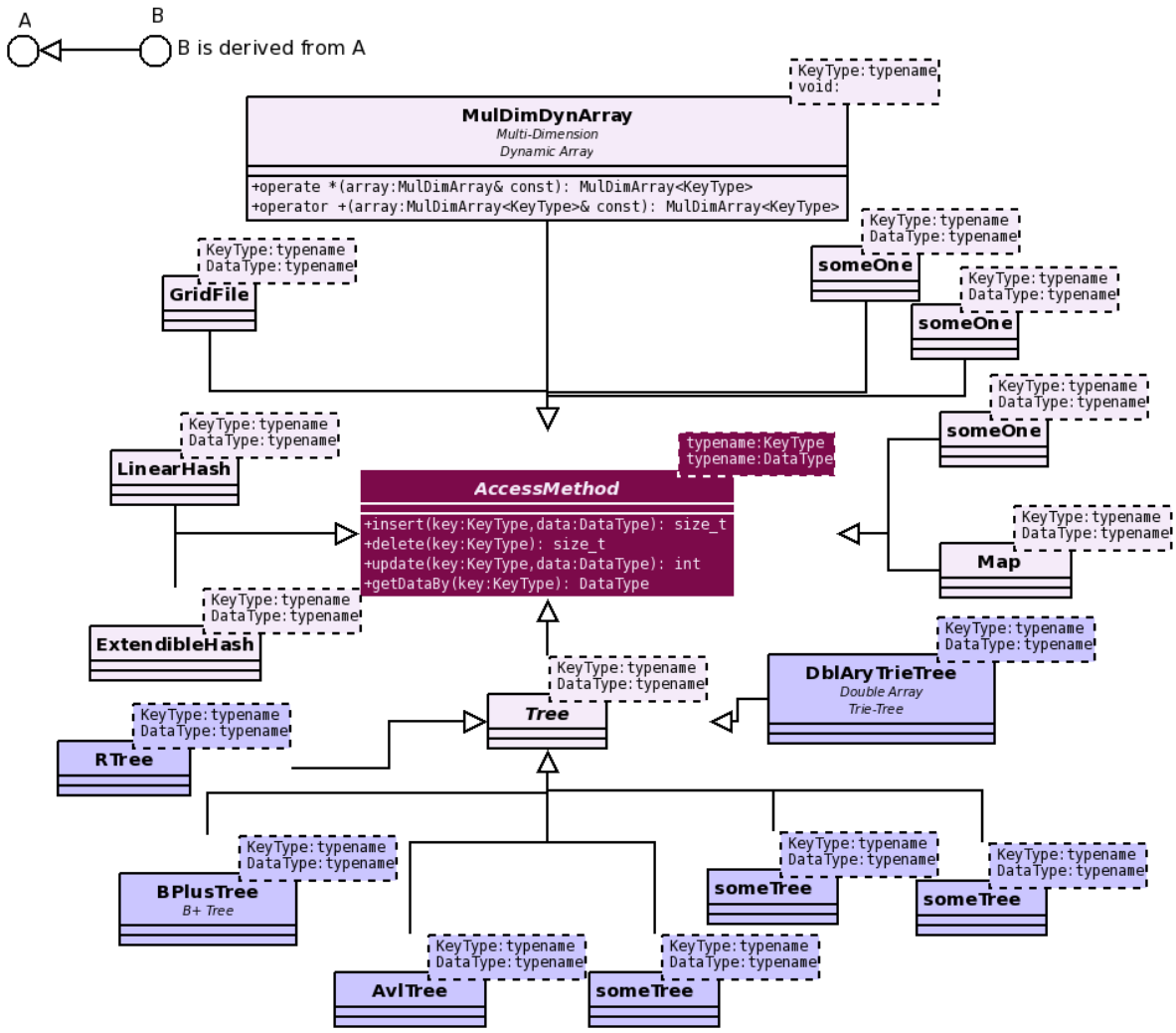

Figure 2: Initial design of AM-lib classes

3 Future Probable Usages

The future probable usages of AM-lib can be illustrated in several ways.

- In your project, you may want to compare the performance of using several different data structures. AM-lib makes this easy anyway.

```
#include "am/am.hpp''
class MyClass {

public:
        MyClass(AccessMethods<int, string>* pAm): pAm_(pAm){}

        void myFoo()
        {
                pAm_->insert(128, ``Hello! AM-lib'');
                pAm_->getDataBy(128);
                . . .
        }

private:
        AccessMethods<int, string>* pAm_;
}
```

```
////////////////////////////////////
#include "myclass.hpp''
#include "am/btree.hpp''
#include "am/rtree.hpp''
#include "am/am.hpp''
int main()
{
        AccessMethods<int, string>* pAm = new BTree<int, string>(...);
        MyClass test1(pAm);
        test1.myFoo();
        delete pAm;
        . . .
        pAm = new RTree<int, string>(...);
        MyClass test2(pAm);
        test2.myFoo();
        delete pAm;
        . . .
}
```

- You can make your modules more reuseable.

```
template<typename AmType>
void foo(AmType& am)
{
        am.insert(128, ``Hello! AM-lib'');
        am.getDataBy(128);
}

////////////////////////////////////
#include "am/btree.hpp''
#include "am/rtree.hpp''
int main()
{
        BTree<int, string> btree;
        foo(btree);
        . . .
        RTree<int, string> btree;
        foo(btree);
        . . .
}
```

- You don't need to worry about the size of data, cause AM-lib will use disk when data size comes very large.

```
void foo(AmType& am)
{
        MulDimDynArray largeArray(100000000, 100000000, 1000000);//initialize a 3 dimensions dynamic array
        MulDimDynArray smallArray(10, 10, 10);
        smallArray.append(largeArray);
        . . .
}
```

4 How to make this come true?

Under discussion of Yingfeng and me, we generally break the iZenelib into two main parts, access methods which I'm in charge of and pre-processing methods that Yingfeng is about to handle. And the scheme in this report is only for access methods part. From some existing library like Ylib, SML and SF1lib, we can get a lot of modules for AM-lib. But, still, there're some advanced data structure and algorithms we don't have, and some of algorithms is outdated, some better ones have came up, and we need to add them into new library. Figure 3 is a genesis of access methods and their modifications.

Of course, we don't have to add all these methods into the new library. But, some of them must be added. Here's a raw list of methods we are about to add. And all the methods support disk/cache operation if data size becomes huge.

Trees B-Tree, B+-Tree, AVL-Tree, Double Array Trie-Tree, R-Tree.

Hashing Linear Hashing, Extendible Hashing, Grid File, lz-Hashing.

Others Multidimensional Dynamic Array, Map, Large List, Large Vector, Skip List.

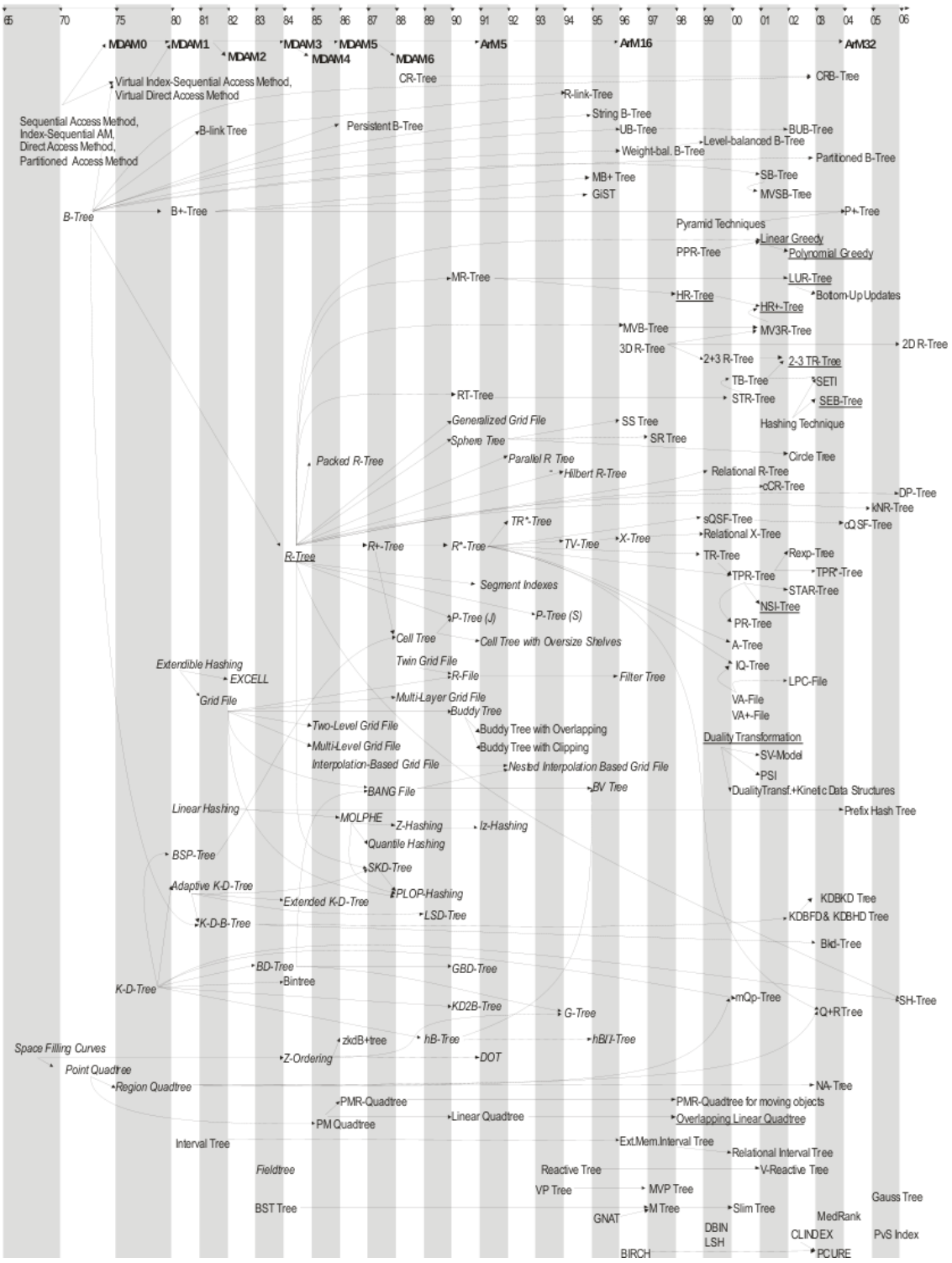

Figure 3: Genesis of access methods and their modifications

From my interviews to other engineers, they would like to have a file system operating library which have some advanced modules and is also a basic sub-library of AM-lib. And in corresponding to array, we would like to have some linear computing modules. Here's a list of modules to come.

File System Operating new files, copy files, read, write, delete, move, rename, get current path, get file name, get extension of file, get all file name in a folder or sub-folder.

Linear Compute array multiply, array summation, array subtraction, array transpose.

The future works include building a framework of this library, making every algorithms and data structures come true, testing of those algorithms and documentations of this library. When we confirm the mutual understanding, the schedule of this project will come soon.

5 Feedback sheet

Here are some questions I need to know for further developing and mutual understanding. If you have some different or extra ideas, please let me know.

- Do you have an general idea of what this report is about? ( )
  - A. Yes, I totally understand.
  - B. Yes, generally.
  - C. No, terribly writen report.

- What's about the advantages of having this AM-lib? ( )
  - A. Just exactly as what Kevin pointed in the first section.
  - B. Kevin have mentioned most of advantages, but, still, missed some.
  - B. Kevin have mentioned most of advantages. But some parts, I disagree.
  - C. Kevin totally misunderstood the purpose of iZeneLib.

- What's about the future usages of this AM-lib? ( )
  - A. Just exactly as what Kevin pointed in the third section.
  - B. Kevin have mentioned most of usages, but, still, missed some.
  - B. Kevin have mentioned most of usages. But some parts, I disagree.
  - C. Kevin totally misunderstood the future usages of iZeneLib.

- What's about the raw architecture design of this AM-lib? ( )
  - A. Kevin have a good start of building this lib.
  - B. Some parts need a correction.
  - C. What a pity! To be frankly, a terrible design.

- What's about module list Kevin's about to add into this AM-lib? ( )
  - A. Kevin could start with these, we can add some later if we need.
  - B. Kevin have missed some important parts.
  - C. Kevin totally don't need to do that.