JUNO数据库接口设计的现状与规划

2017.06.05-2017.06.06

黄文昊
黄性涛

概要

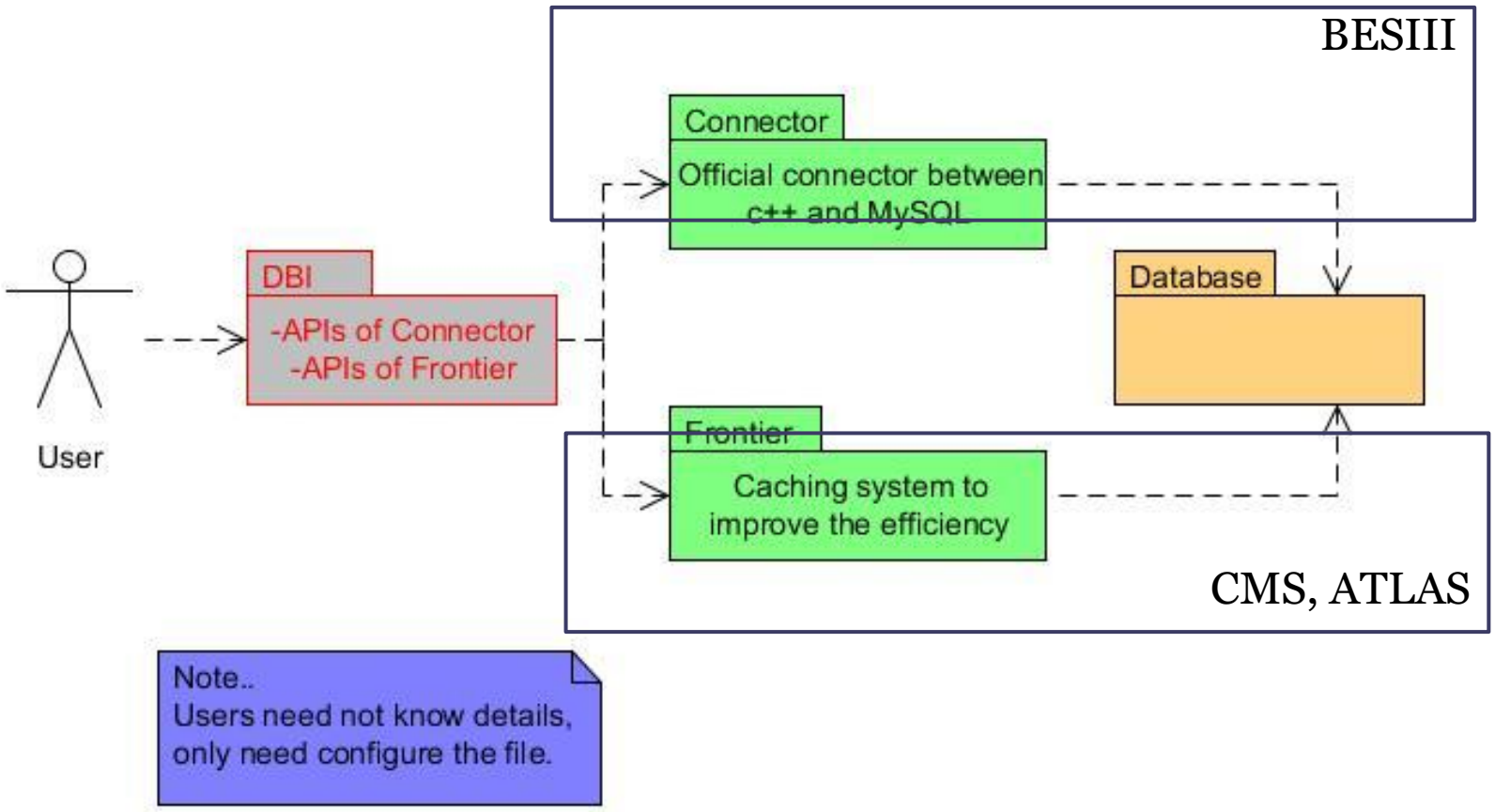
- 数据库接口(DBI)的简单介绍
  - 为什么需要使用DBI
  - DBI的主要构成和使用方式
- 数据缓存系统Frontier
  - 概念和优势
  - DBI的使用现状
- 总结&展望

DBI的简要介绍

- DBI是用户代码和数据库的中间层。

- Database用来保存大量数据:
  - 探测器状态、pmt参数、光学参数等
- 不同类型数据库拥有不同API

DBI的整体设计


BESIII
Connector
Official connector between c++ and MySQL
Database
DBI
-APIs of Connector
-APIs of Frontier
User
Frontier
Caching system to improve the efficiency
CMS, ATLAS
Note..
Users need not know details, only need configure the file.

DBI的使用-----配置文件

- 用户需要配置”my.conf”文件来确定链接数据库的信息。DBI会从该文件中获取链接数据库所需信息。以下为”my.conf”的示例：

```
# The section name is the name of ConName in run.py
[mysql_one]
# Put the url of database here
# If the port of database is not 3306, just add the
# For DataBaseSvc, the Url below is the same to 202.
Url = junodb1.ihep.ac.cn
# Put the username of the database here
User = juno
# Put the password of the user here
Password = [redacted]
# Put your database here
Database = offline_db
~
```

DBI的使用-----Table存在形式

- 对于大部分用户来讲，可以在代码中通过table row来访问数据库：只要按照相应的格式完成对class的定义，剩余的工作（例如生成SQL语句）由DBI自动完成。
- 例如假如需要操作如下table

```
mysql> desc MaterialProperty;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| sftVer  | varchar(16)  | YES  |     | NULL    |       |
| name    | varchar(256) | YES  |     | NULL    |       |
| keyId   | int(11)      | YES  |     | NULL    |       |
| notes   | varchar(256) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

DBI的使用-----Table对应类的定义

- 用户需要如下定义该类。（类的定义格式在附录中有详细描述）

```
class MaterialProperty:public DBITableRow
{
 MaterialProperty(char* sftVer, char* name, int keyId, char* notes);
 virtual ~MaterialProperty();

 char* fsftVer;
 char* fname;
 int fkeyId;
 char* fnotes;

 void SetsftVer(char* sftVer){fsftVer = sftVer;}
 void Setname(char* name){fname = name;}
 void SetkeyId(int keyId){fkeyId = keyId;}
 void Setnotes(char* notes){fnotes = notes;}

 char* GetsftVer(){return fsftVer;}
 char* Getname(){return fname;}
 int GetkeyId(){return fkeyId;}
 char* Getnotes(){return fnotes;}

 ClassDef(MaterialProperty, 1)
};
```

DBI的使用-----Table对应类的定义

- 由于定义该类时重复工作较多，现正考虑采用XOD工具来简化类的定义。
  - XOD工具是JUNO软件组开发简化数据模型的工具，可由xml文件生成c++代码。
- 完成该类的定义后，即可按照如下方式使用。

```
MaterialProperty matpro1("1.0", "name", 0, "comments");
```

DBI的使用方式

- 假如用户希望自己写SQL语句来对数据库进行操作，可以按照如下方式来使用。

```
//this one is to test lookupquery function
DBIRequest request("select * from offline_db.test");

DBIResultPtr dbptr("DatabaseSvc", request);
dbptr.Session();
//dbsvc.Session(request);
int row = dbptr.GetMaxRowcount();
for(int i = row; i >= 0; i--)
{
dbptr.GetResByRowNum(i);
std::cout << "id= " << dbptr.GetInt("id") << std::endl;
//std::cout << "LocalName= " << dbptr.GetString("LocalName
//std::cout << "Capital= " << dbptr.GetString("Capital") <
}
```

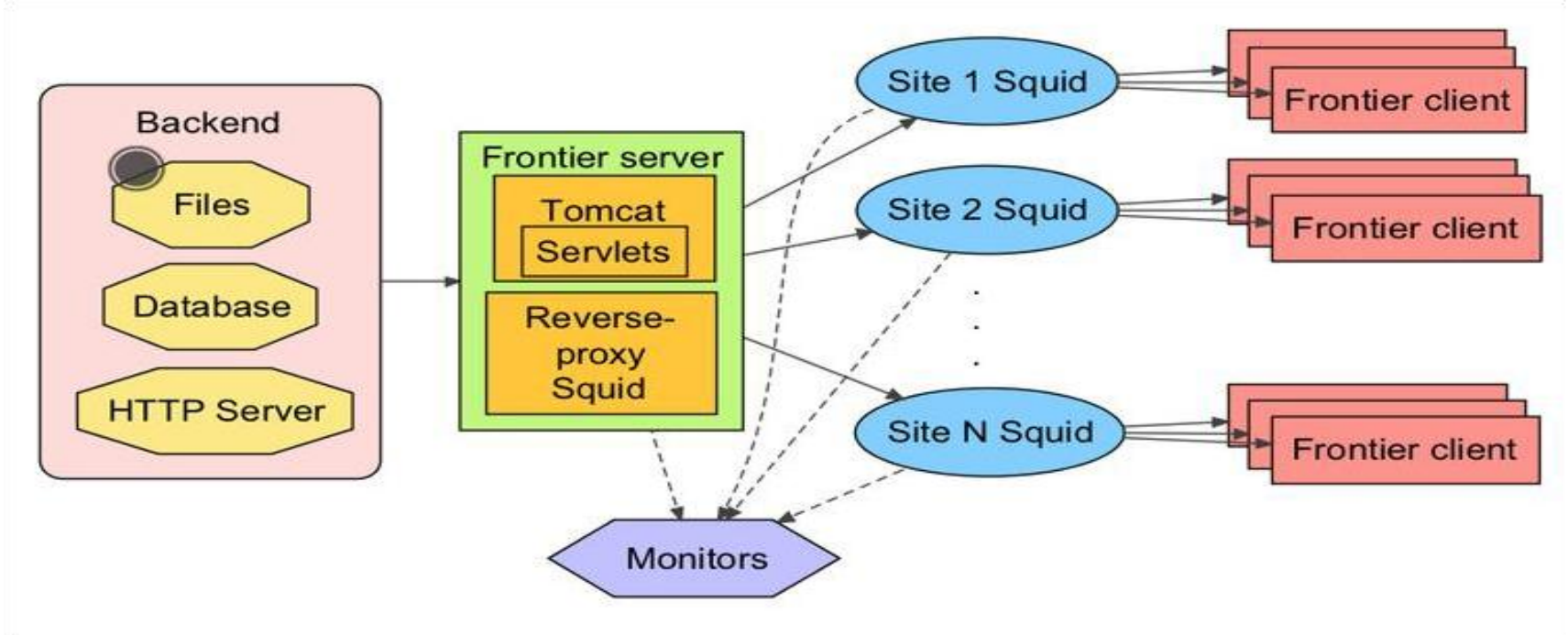
数据缓存系统Frontier

- 构成
- 优势
- DBI对Frontier的采用

Frontier的构成

- Frontier server:包含Frontier tomcat（必需）和Frontier squid（可选），负责通过JDBC直接操作数据库。
- Squid(Frontier squid):负责缓存数据，提升性能。
- Frontier client:负责提供API链接Frontier server和Frontier squid.

Frontier的构成和工作原理



Frontier在以下情况下具有较大优势：

- 有大量的广泛部署的client
- 短时间内有大量相同的数据访问请求

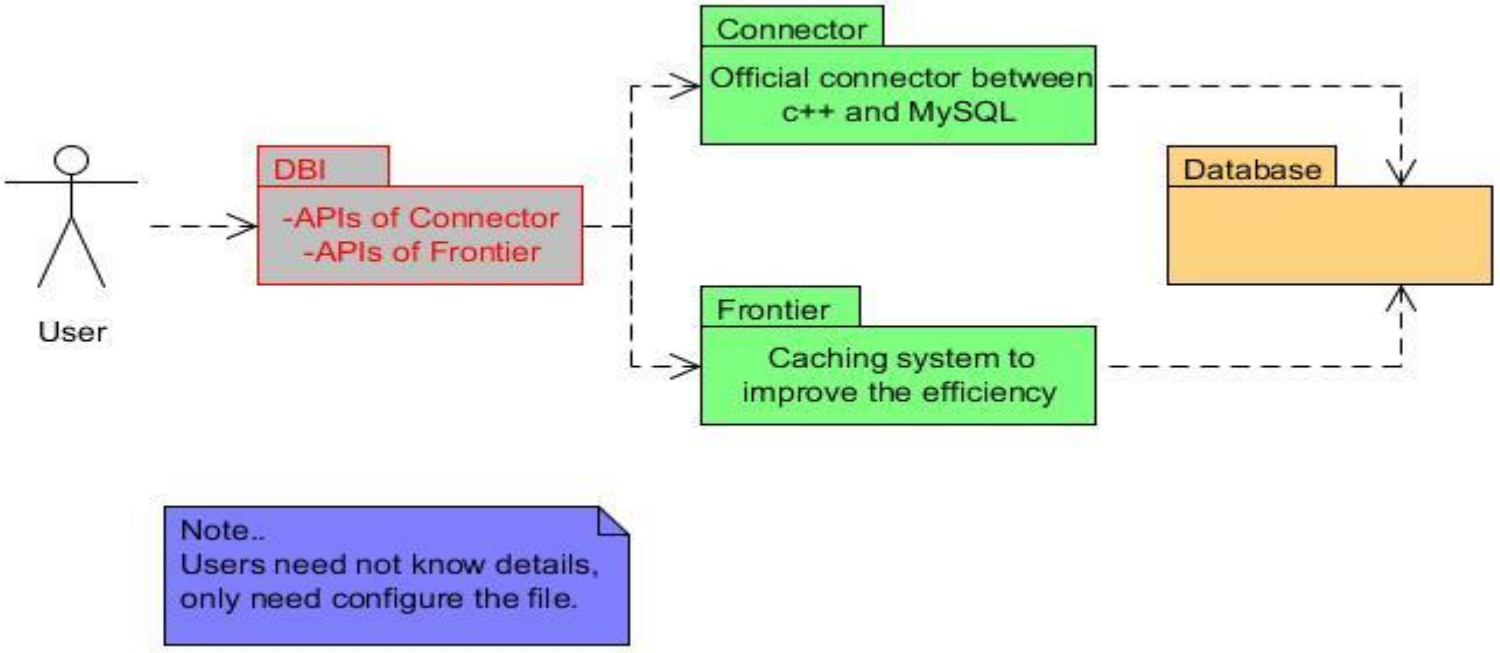
JUNO实验对数据库的需求

- JUNO的现状：
  - 相较于其他的实验，JUNO有更多的pmt（五万多个，是大亚湾的数倍），所以会有更加庞大的数据量。
  - 世界各地的用户需要访问中心数据库。
- Frontier在CMS和ATLAS实验中已经有数年的应用，工作状态良好且Fermilab有专人维护。

所以我们决定将Frontier应用到DBI中

DBI对Frontier的采用

- 目前，在已经安装了frontier-tomcat, frontier-squid和frontier-client的机器上已经将Frontier的API加入了DBI，目前工作正常。



总结&展望

- 介绍了DBI整体设计、构成和使用方式。
- DBI的两种链接数据库的方式（普通和采用缓存机制）。
- 简要说明了数据缓存系统Frontier和目前的采用状况

- 接下来的版本发布中将优化用户接口，提供更多功能（roll back功能正在测试）。
- 将Frontier加入DBI以运用数据缓存机制。

附录1：数据类定义格式

Here gives the format of the class, you can compare these with the example:

1. The name of the class must be completely the same of the name of the table.
2. The class must be inherited from the class 'DBITableRow'.
3. The member variables, either public or private, must be corresponding to the columns of the table, which means the numbers are the same and the name of the member variables must add a letter 'f' before the name of the column.(A column name 'XXX' need a member variable name of 'fXXX')
4. The functions used for getting or setting data must be existed and have the format of 'SetXXX' or 'GetXXX'.(Of course you must make sure they can work.)
5. When you need a variable which type is 'string', please use 'char*' instead.
6. Add a macro named 'ClassDef' just like the above.

附录2：XOD工具

- XOD工具是JUNO软件组开发的一种工具，它可以帮助用户完成书写大量重复代码的工作，以便于减少代码错误和用户的工作量，提升用户工作效率。