《开源软件设计与开发》课程总结

姓名：杨东东　学号：51194507017

1、开源理解

关于阅读源代码方面的心得：

通常情况下，我们不会无缘无故拿到一份源代码，我是说，当想要阅读源代码时，一定是抱着某种目的进行下去的，这个目的会贯穿整个研究过程，比如：

想研究某个东西的实现；

想学习作者的代码风格和项目组织；

想参考实现并用其他语言移植项目；

等等。所以在最初的时候，明确自己阅读源代码的目的非常关键。

在明确了目的之后，就可以正式开始深挖代码了。可以用来写代码的工具有很多，如包括 Vim、Emacs、SublimeText、Atom 等在内的 Text Editor，还有包括 Visual Studio 和 JetBrains 全家桶在内的 IDE。Text Editor 在编写代码时有无与伦比的优势，方便的快捷键、一流的反省速度、高度可配置的编辑环境等等，无疑能极大程度上提高程序员的手速，然而这些亮点都不是在阅读代码时候所必须的；阅读代码时，很多时候需要搞清楚函数之间的调用关系，这时候就需要有强大的代码静态分析工具和一个便捷的断点调试工具，这也正是 IDE 的优势。

接下来，就真的可以开始深挖代码了。

第一步：自顶向下理清代码组织关系。

代码组织结构，很多时候这一点根本就不是问题，因为项目的构建无非是通过构建工具、框架或最佳实践来做的，实在不行还可以查阅开发文档。当然，项目代码里的结构，不仅仅受到了实际需求的影响，也受到了构建工具和设计模式的影响，比如能够应用在全局的 MVC 模式，或者应用在局部场景的单例模式、状态模式等等，在整理这部分内容的时候，通常会在脑海中形成清晰的思路，但如果可以的话，请把这些东西用纸笔画出来。如果想要研究的是具体的功能实现，就需要在上面的基础上更进一步，着眼于功能性的代码，对每个功能实现画出活

动图和时序图。

总之这一步里包含了不小的工作量，但搞清楚这些东西，是继续进行下去的基础。

第二步：有针对性地深挖代码。

这是阅读代码的核心步骤，是最复杂的一个步骤，因为每个人阅读代码的目的不尽相同，每个人研究问题的习惯也有很大差异。研究问题的时候，通常会有大量自己改代码、打断点、做编译的机会，请尽情享受解决自己目标问题的思维过程，并且不要忘记以下重要的辅助资源：开发文档、开发日志（Issues, Mails）、主要开发成员的博客（如果有的话）、相关领域的资料，包括书籍和论文，与同类开源项目产品的实现相比较，与原作者、开发组进行沟通等。

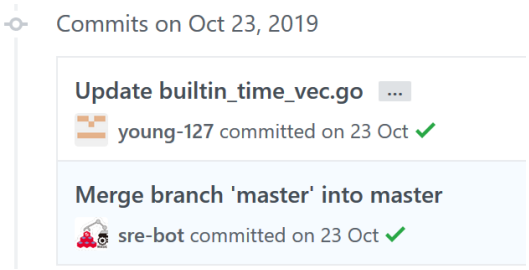
第三步：自底向上理解局部代码在全局中的重要性

不论是把眼光聚焦在对问题的解决方案上，还是在代码组织、设计模式上，能针对结果做进一步抽象，才是最好的。一旦理解了作者的解决方法，对自己理解这种解决方案本身和背后的业务模型都有提升。一旦发现了更好的解决方案，自己就有机会提 PR。尽管要做好这一步，需要较强的领域技术背景，但如果能坚持尝试这么做的话，会受益匪浅。

2、开源贡献

我参加的开源项目是 Tidb。

涉及的工作是内容是 Tidb 向量表达式的构建，TiDB 的向量化计算是在经典 Volcano 模型上的进行改进，尽可能利用 CPU Cache，SIMD Instructions，Pipeline，Branch Predicatation 等硬件特性提升计算性能，同时降低执行框架的迭代开销。最近我们扩展了 TiDB 表达式计算框架，增加了向量化计算接口，初期的性能测试显示，多数表达式计算性能可大幅提升，部分甚至可提升 1~2 个数量级。为了让所有的表达式都能受益，我们需要为所有内建函数实现向量化计算。



项目提交的 PR 链接：https://github.com/pingcap/tidb/pull/12886

3、课程反馈

开源作为信息时代的一种巨大变革，它是一种互联网文化趋势，是一种程序员间潜移默化的观念。以前我很难想象开源会走上课堂，作为一门选修课程来授课。这门课与我以前上的所有 code 课程都不一样，课上请了很多业界大咖，与我们分享最前沿的项目技术。此外，配套的开源项目是最大亮点，使我在 GitHub 上提交了第一个 PR。

希望以后的课程能增加更多有影响力的开源项目供同学选择。

4、参考文献

十分钟成为 Contributor 系列 | 助力 TiDB 表达式计算性能提升 10 倍