

第一组:郑姝昕

关浩 卢弘毅 周祥瑞







原理

根据 Git 仓库记录的信息建立数据库,将函数的每一个改动与提交日期和作者联系起来。



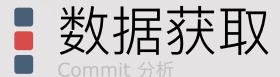
作用 1

辅助 Call graph 分析,用来快速查找某一文件及函数



作用 2

把时间和作者这些额外信息也当作综合因素纳入判断



对代码进行分片,获取函数 建立起行到函数的映射关系

- 使用 CTags CTags 可以分析出代码文件里的函数定义及其所在行,可以依此建立索引。
- 2 利用二分搜索对每行分片 根据上一步建立的行数索引,对于每一行都可以用二分迅 速查找出其所在的函数范围。

分析 Git 仓库,获取提交信息 获取每一行被更改时的相关信息

- 处理 git blame 结果 git blame 可以输出代码文件的某一行最近是在哪次 commit 被修改过,并输出其时间和作者,将其整理成表
- 2 与分片结果(函数)进行整合 讲每一行的修改情况对应到分片后的索引,即可得出每个 函数都经过了哪些修改。

数据处理

1. parser.py

格式化 ctags 输出和 git blame 输出

2. gen_data.py

遍历 Git 仓库,对代码文件分别调用 ctags 和 git blame 进行处理,生成输出信息

3. pydbc.py

连接数据库,插入新的信息条目。

https://github.com/hguandl/Tencent-Mini-Group1-New/tree/master/script/commit_info_database

部分数据结果展示

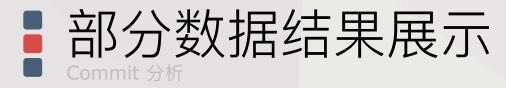
文件表

id	file_path
1	src/misc/ngx_google_perftools_module.c
2	src/misc/ngx_cpp_test_module.cpp
3	src/core/ngx_thread_pool.c
4	src/core/ngx_log.c
5	src/core/ngx_queue.c
6	src/core/ngx_cpuinfo.c

Commit 表

id	hash	time	
1	85dd8fc5b6f033fadd3c21e058a90d63140ae223	2008-03-18 18:36:27+08	0
2	86f791e09092b52063cc42ce9ca4e5304d5fba49	2010-03-27 05:17:26+08	٥
3	e24b57ad9f06e5e48eda02f0d79a922797a68a87	2008-08-06 03:32:50+08	0
4	305fc021db799c87d751f0f1f5e99afee7bb2b3b	2015-03-14 22:37:07+08	٥
5	cb43696e0cb15cf063e475e8bd07a53a52775e2e	2015-03-23 22:51:21+08	٥
6	14b1b6e10a75994a6001a3480901e62f43e05696	2016-08-15 20:52:04+08	٥
7	e87a565aab4da7ade1f016c2bab4371a70998974	2015-03-23 22:51:21+08	٥
8	c34368715f9e6639f11e9c1a70272668085d6886	2015-03-20 04:20:18+08	٥
9	20d07074e3d1cd9069fd786aa4861a51a44f45b5	2015-03-20 04:19:35+08	0

Ī	id	name
	1	Igor Sysoev
	2	Valentin Bartenev
	3	Piotr Sikora
	4	Ruslan Ermilov
	5	Maxim Dounin
	6	Vladimir Homutov
	7	Sergey Kandaurov
	8	Andrei Belov



函数表

id	signature
1	ngx_google_perftools_create_conf(ngx_cycle_t *cycle)
2	ngx_google_perftools_worker(ngx_cycle_t *cycle)
3	ngx_cpp_test_handler(void *data)
4	ngx_thread_pool_init(ngx_thread_pool_t *tp, ngx_log_t *log, ngx_pool_t *pool)
5	ngx_thread_pool_destroy(ngx_thread_pool_t *tp)
6	ngx_thread_pool_exit_handler(void *data, ngx_log_t *log)
7	ngx_thread_task_alloc(ngx_pool_t *pool, size_t size)
8	ngx_thread_task_post(ngx_thread_pool_t *tp, ngx_thread_task_t *task)
9	ngx_thread_pool_cycle(void *data)

部分数据结果展示

以 Nginx 的源码仓库为例,共统计了 228 个源文件。

涉及了 2470 个函数、50 位作者、 3702 次提交。

共统计出 14626 条更改信息。

更改信息条目

id	commit_id	author_id	signature_id	file_id
1	1 →	1 →	1 →	1 →
2	2 →	1 →	1 →	1 →
3	1 →	1 →	2 →	1 →
4	3 →	1 →	3 →	2 →
5	4 →	2 →	4 →	3 →
6	5 →	2 →	4 →	3 →
7	6 →	3 →	4 →	3 →
8	4 →	2 →	5 →	3 →
9	7 →	2 →	5 →	3 →
10	8 →	4 →	5 →	3 →
11	5 →	2 →	5 →	3 →



代码解释 Dot 外理及统计计算

1. `eficient_search`函数

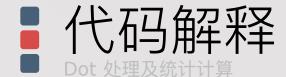
获取一个行中可能含有的信息,结果分为两种,当对应的行是node定义时,返回node的编号与标签,当对应的行是node间关系时,返回起始node与终止node。

2. `scan_path`函数

对一个路径下所有.dot文件进行遍历,并且调用`efficient_search`函数生成总体的call graph文件。返回全局的node定义以及node间关系。

3. `stack_expansion`函数

对一个堆栈信息进行展开,生成所有可能的fail trace,并且生成各个函数在对应的fail trace中的深度,用于后续统计学中的深度分析。



4. `stack_process`函数

根据上面函数生成的信息进行整合,生成所有可能的traces以及对应的深度。

5. `statistic_process函数

对所有可能的traces,按照CrashLocator的公式对每个函数计算怀疑度,返回一个函数对应怀疑度的字典。





https://crash-stats.mozilla.com/

Mozilla 的崩溃信息反馈平台。在此使用爬虫下载 Firefox 64.0.2 的崩溃信息。

https://github.com/hguandl/Tencent-Mini-Group1-New/tree/master/script/bug_reports_retriever

项目仓库位置

https://github.com/hguandl/Tencent-Mini-Group1-New

包含更详细的组员分工及参与情况、对于代码的更多解释、数据库 dump 内容、处理脚本及初步结果。

