QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

开发工具的云端化

Insight.io 赵扶摇 @

主办方 Geekbang. 极客邦科技 InfoQ

Jeff Dean: Setup Interview [13]

- 现有环境：For writing code, I use emacs and Google's internal **distributed build system** (a version of which was open sourced as Bazel) and our **version control system**, plus Google's **internal code searching** tools that allow me to quickly search over Google's whole code repository.
- 仅有不足：A view of the ocean would also be nice.

大纲

- Google开发工具概述
- 重点分析：
  - 代码仓库
  - 云端构建
  - 代码智能

Google开发流程概述

Google开发流程概述

- 代码管理
  - 所有的代码在一个代码库中
  - 近百TB代码瞬间完成checkout
- 编辑
  - 云端工作空间
- 构建
  - Build from source
  - 全部云端进行
  - 10分钟内完成，1，2个文件秒杀
  - 共享构建/测试结果

Google开发流程概述

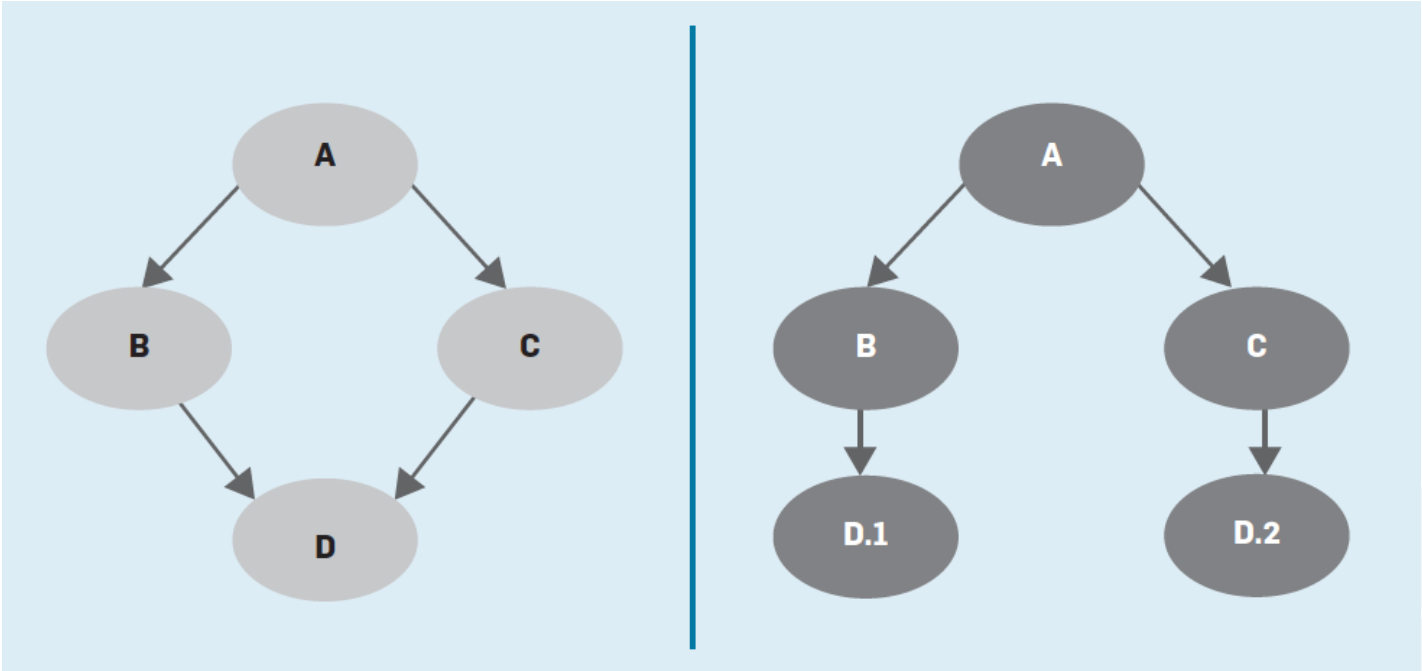
- 探索/理解
  - 用IDE的体验在网页端浏览所有代码
  - 语义化的搜索
- 代码审查
  - 强制代码审查
  - 提供上下文浏览环境
  - 自动查错，自动修改

单根代码树（Monolithic Repo）[1]

- 所有代码统一放在一个库中
- 不同项目存放在不同目录
- Self contain（工具和版本绑定）
- Trunk开发（不需要branch）
- **项目之间源代码依赖**

单根代码树（Monolithic Repo）

- 简化的版本和依赖控制
  - 没有maven中常见的依赖混乱
  - 左：Google的依赖树，右：Maven可能的依赖树



单根代码树（Monolithic Repo）

- 简化地依赖关系方便了
  - 代码重用
    - 配合构建系统，方便地引用其他包的代码
  - 代码重构
    - 可以一次性地修改所有的引用
      - 少量代码：codesearch
      - 大量代码：clangMR
  - 强制使用新版本
    - 不会出现由于升级太晚导致的种种问题

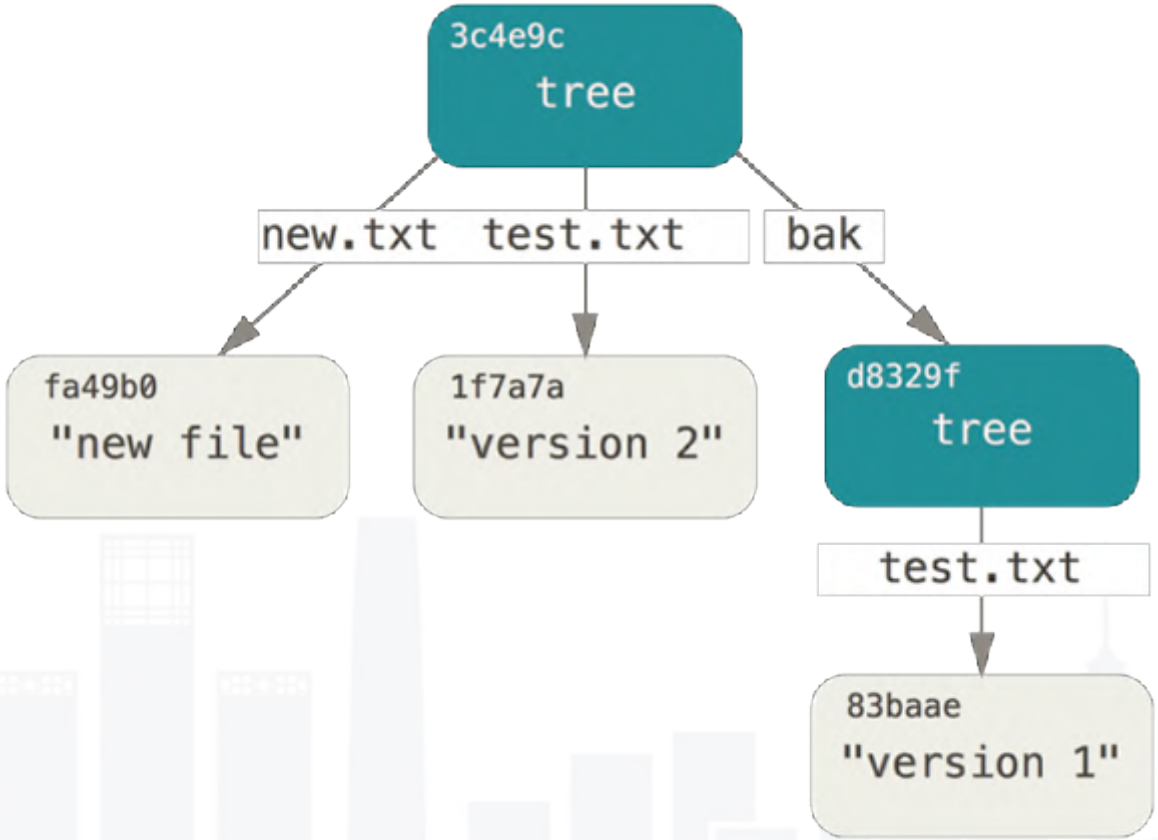
代码规模（2015）

- 10亿文件
- 900万源文件
- 20亿行代码
- 3千5百万commit （每天4万5千，每两秒一提交）
- 86TB
- Google之前一直在用Perforce

Linus @ Google [14]

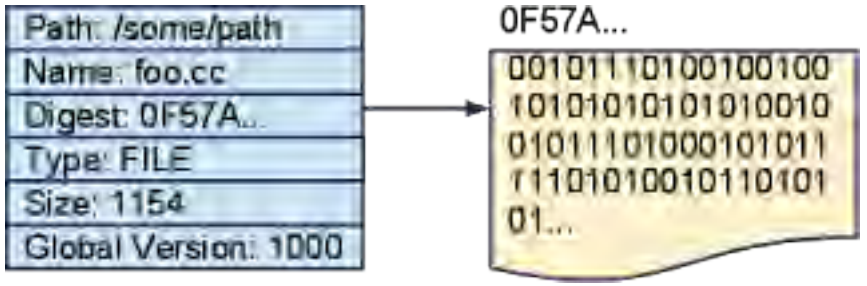
Git

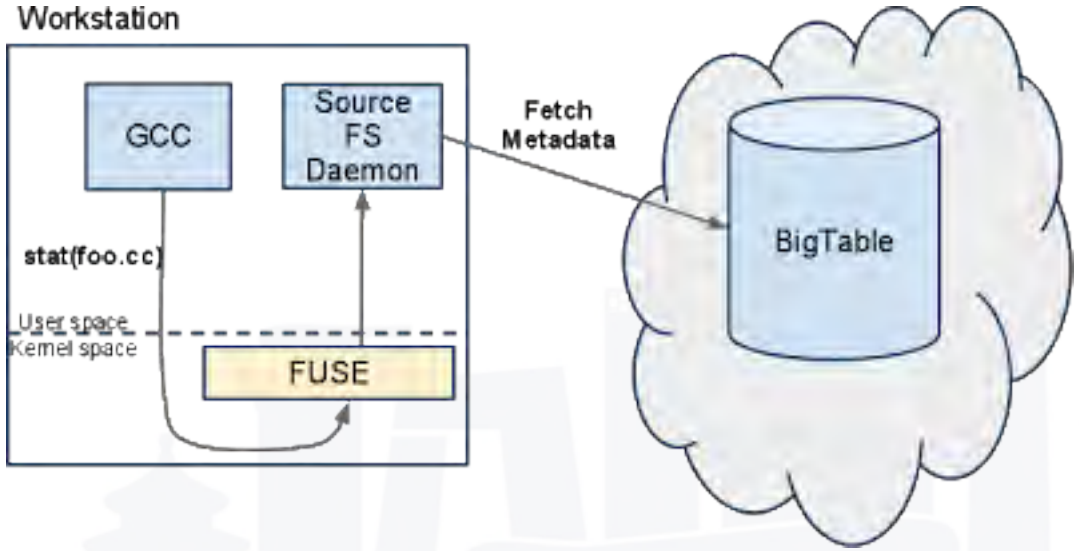
- 可以看成一个KV store
- Git objects在checkout时被copy进文件目录
- 但是无法处理大规模的库
- 一个可能的git云端化的思路?



Piper

- 每个文件/目录有一个Metadata
- Metadata到内容的映射放到Bigtable
- 每个Revision指向一系列的Metadata





其他方案

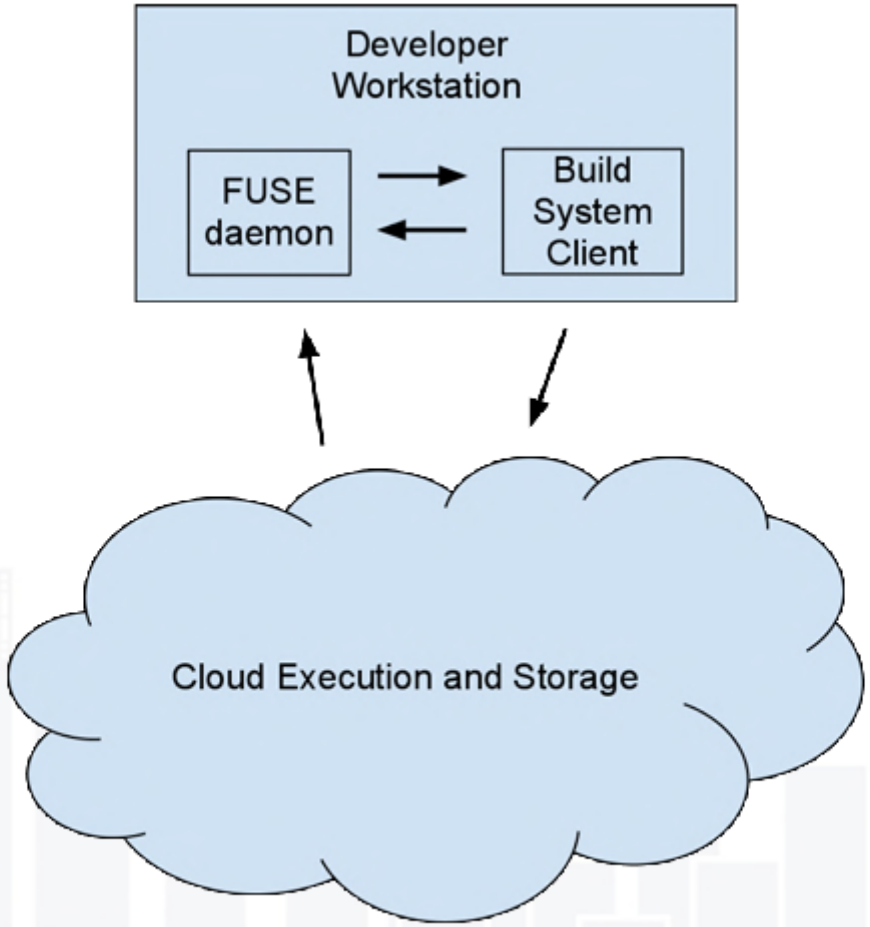
- Facebook fbsource [5,6]
- AWS Codecommit
- Microsoft GVFS (Git Virtual File System) [11]

云端构建

- 问题
  - 代码库大
  - Build from source
- 需要
  - 合理的语言
  - 足够的性能

云端构建（Overview）

- 默认全部为云端构建 [4]
- 组件
  - 前端：Blaze / Bazel
  - 后端：Forge
  - 基于FUSE文件系统

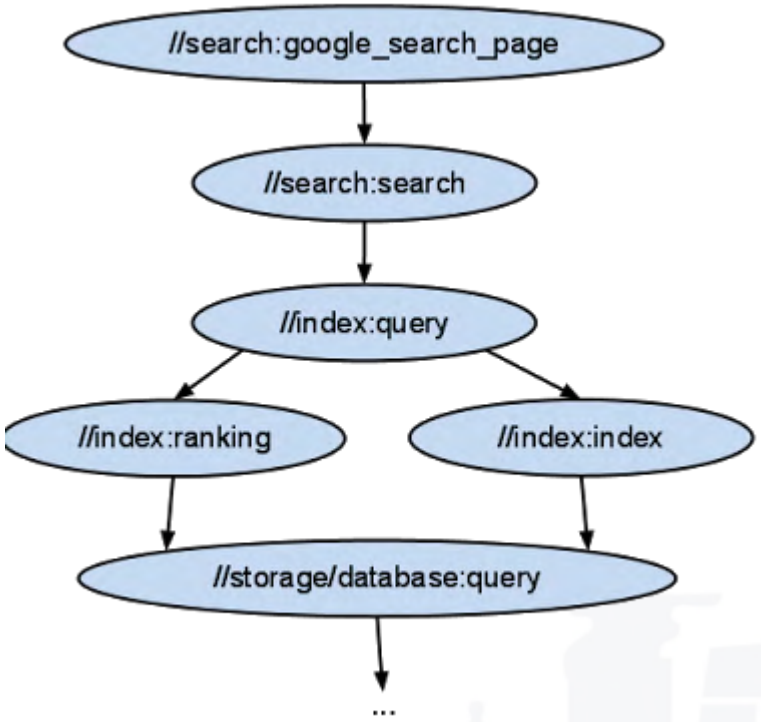


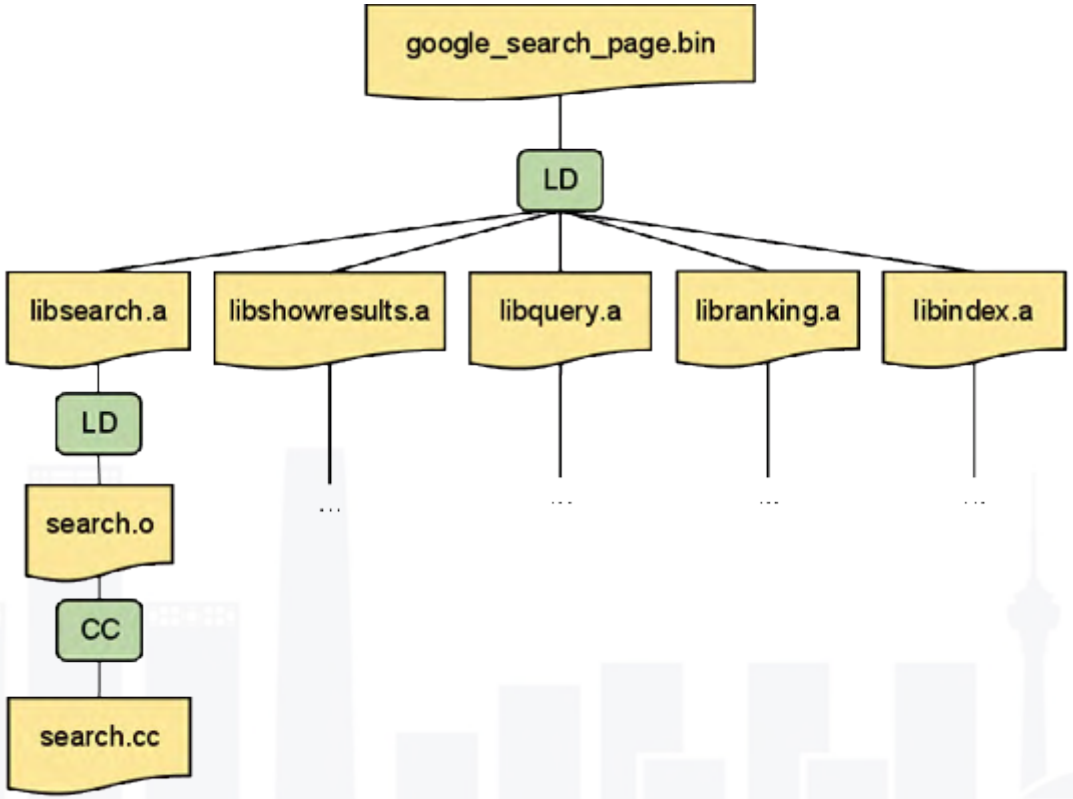
云端构建（Blaze）

- 声明式，python子集
- 全员，全语言，全项目统一抽象
- 以target为最小单元
- 确定性，可重现
  - 同样的版本，一致的环境，同样的源文件
  - 产生同样的结果
- 为云端构建提供了优化空间
- 被buck，pants，blade等工具借鉴

依赖关系

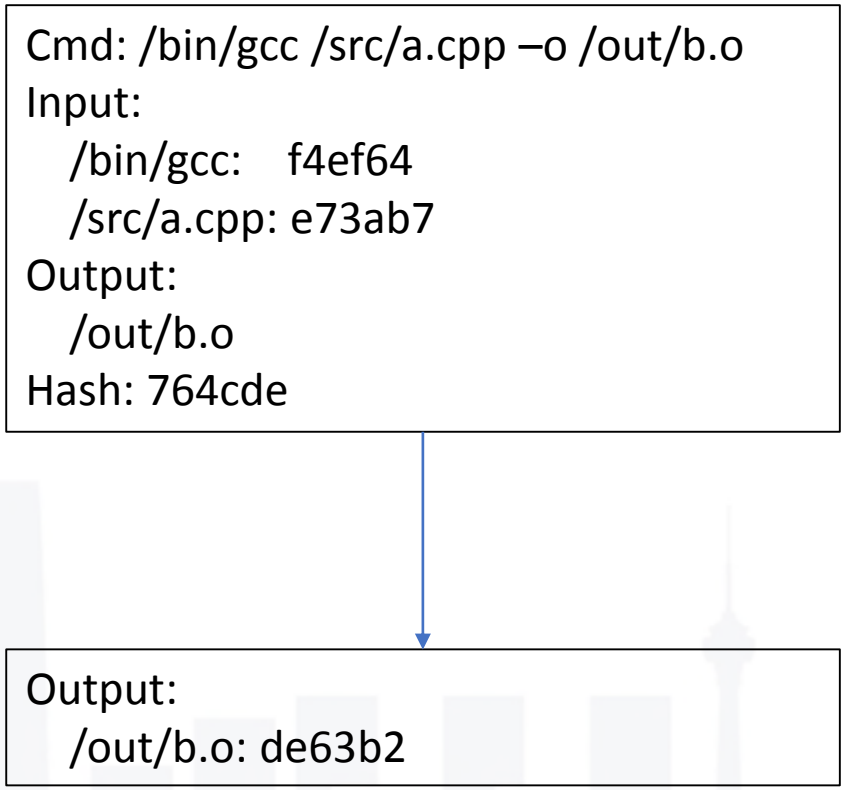
每个Target生成一个或者多个Build Step（Action）





Action详解

- Action包括
  - 命令:
  - 输入文件地址及hash
  - 需要收集的输出地址
- Action输出result:
  - 输出文件的hash值
- build(action) = result
- 输入/输出文件 Content Addressable



处理Action

- 前端Action缓存
  - In memory key-value store (类似redis)
- 构建节点
  - 获取输入文件到tmpfs
  - 在容器内执行命令
  - 收集输出文件
- 文件处理
  - Content Addressable Storage：Big table
  - 输出结果给用户：FUSE

云端构建（系统性能）

- 每天跑7千万测试集
- QPS：百万级
- 缓存命中率：94%（15年）

云端代码智能

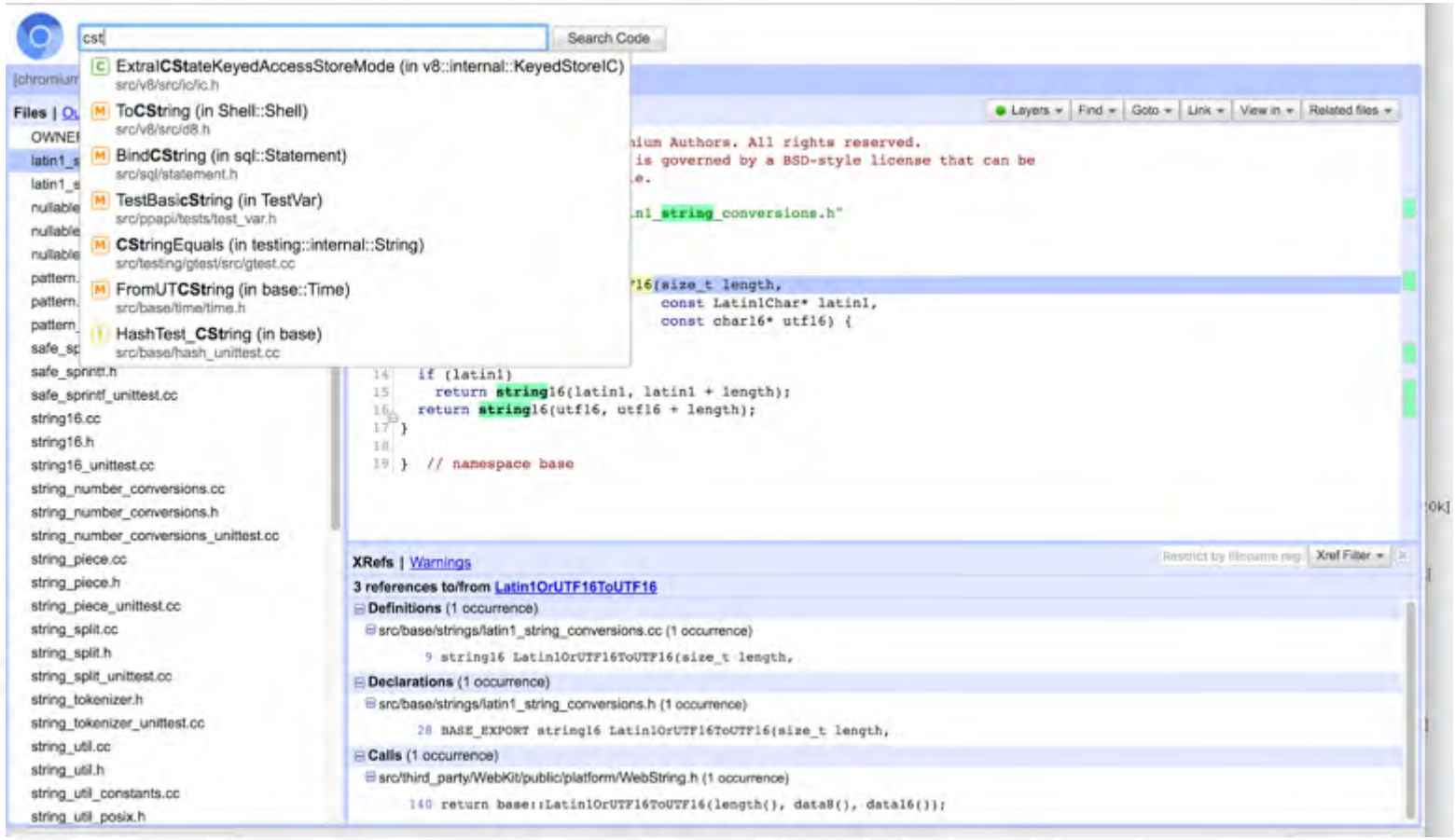
- 涉及产品：代码探索，审查
- 问题：在网页上看代码，能有IDE一样的效果么？
  - 代码库大（不方便下载，代码不全）
  - 不想配置IDE （能要配上小时）
  - Code review，协作
- 传统解决：
  - RE based：LXR, opengrok，etc
- Google的解决：
  - 代码智能

云端代码智能

- 什么是代码智能
- 本地IDE（大部分定型于90s）
  - 跳转定义
  - 交叉引用
  - 重构
  - …
- 主要工具
  - Exploration：Grok / Kythe [15]
  - Check：Tricoder [3]
  - Refactor：clangMR [8]

应用：Code Search

- 语义化的代码浏览
- 高精度搜索
- 每人每天12次搜索



应用：CLI

- CLI
  - 类似SQL
  - 结构化查询
- 例子
  - 找出所有和bar方法一起被调用的foo的用例

```
bar()
....
foo()
```

其他应用

- 日志/异常系统
- 代码审查
- …

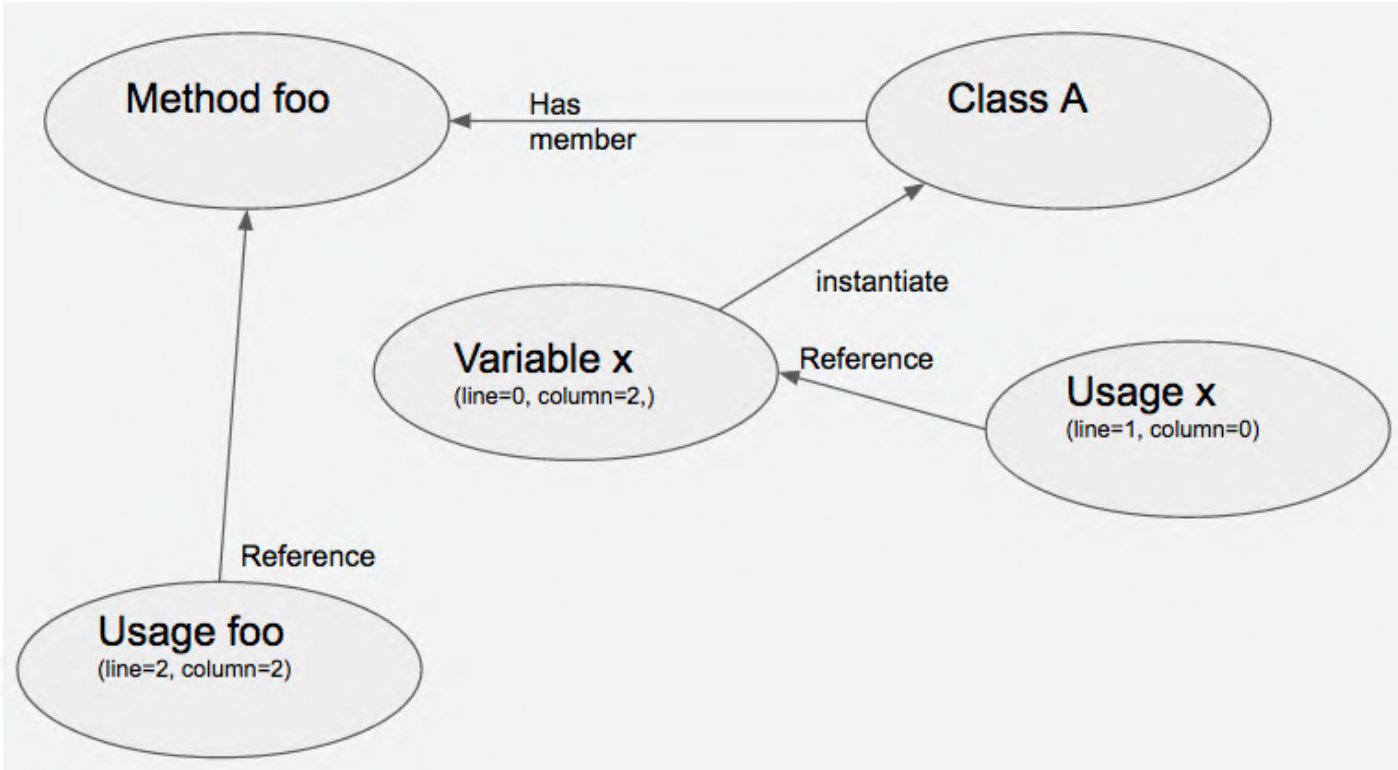
代码智能：Grok/Kythe

- Language Agnostic Graph
- 代码的知识图谱
  - 跨语言
  - 可扩展
- 组成
  - 节点：代码的某一片段（token）
    - e.g.: class，method，const
  - 边：节点直接的关系
    - e.g.: 继承，重载，引用
  - 元数据：描述节点或边的额外属性
    - e.g.: 位置，修饰符

代码智能：Grok/Kythe

```
- A.java
Class A {
  Int foo() {}
}

- Main.java
A x = new A()
x.foo()
```



代码智能：Grok/Kythe

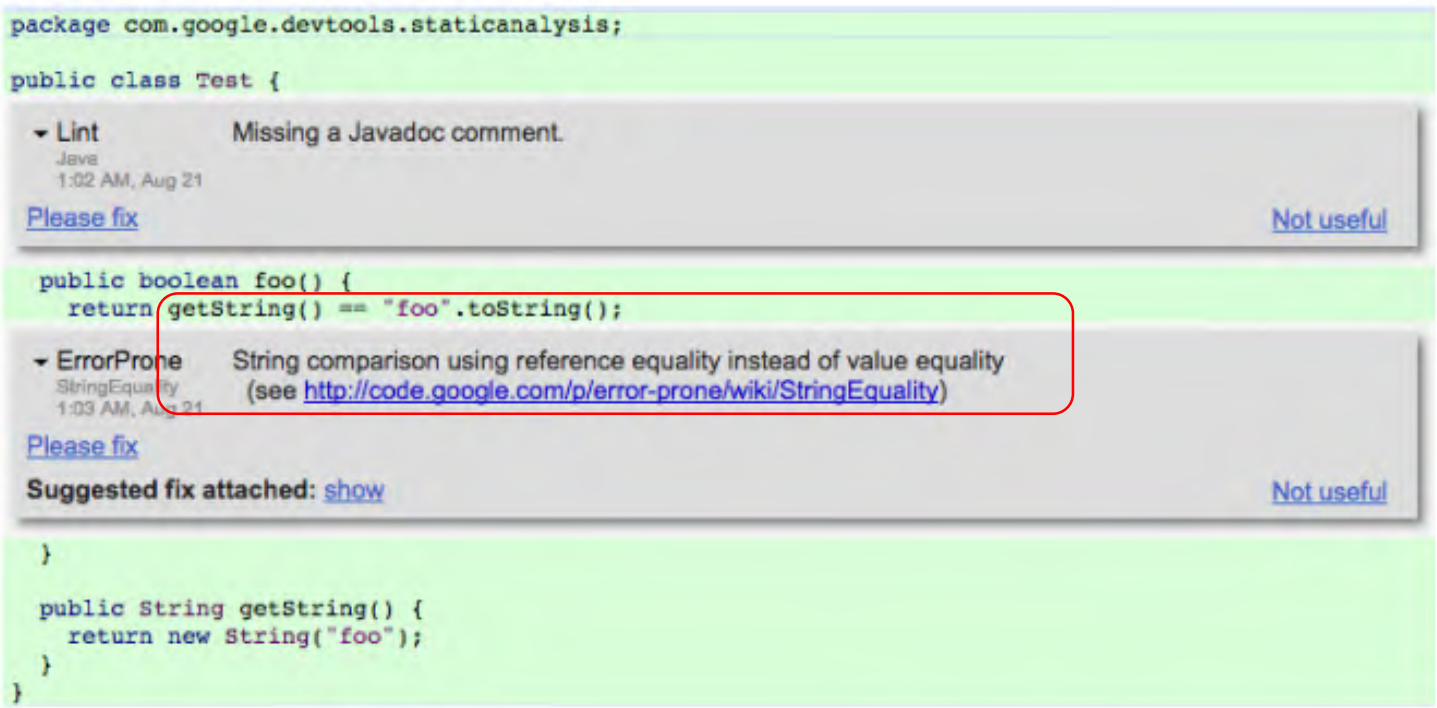
- 统一的标识符系统：
  - 全局可定位：std::algorithm::compare
  - 文件内可定位：uri://file_location:[offset]
- 跨语言
- 对于不同的语言：
  - 增加不同的节点或者边
  - 归纳类似的概念
    - const vs final vs immutable vs unmodifiable
    - namespace vs package vs module vs library
    - public vs global vs default vs exported
    - …

Grok/Kythe实现范例

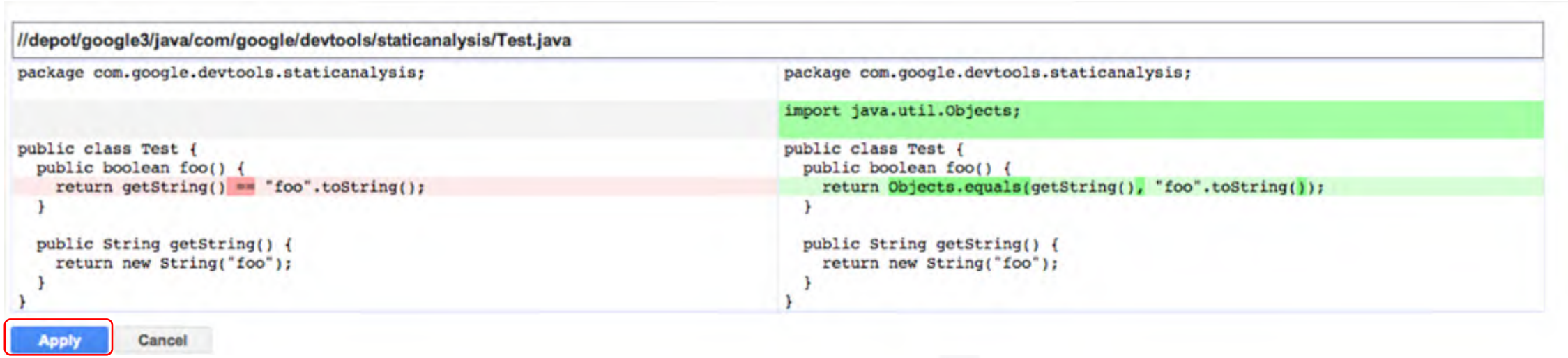
- 静态语言：
  - 触发一次完整构建
  - 提取dependency
  - 在OpenJDK/Clang加一个pass
  - 将信息转换成Grok/Kythe格式
- 动态语言：e.g. python，javascript
  - 没有现成的编译器（只有解释器）
  - 实现类型推导（e.g. Hinder Miller）
  - 不如静态语言，但是好于RE

云端代码智能：代码审查

- Tricoder
- 每次Request Request触发
- 基于Pattern（Errorprone）
  - Style
  - bugs
- 强于findbugs
- 弱于专业分析工具



代码智能：代码审查



One click to fix (apply to workspace)

总结

- 优点：
  - 加强协作
  - 统一环境
  - 增加效率
- Google的经验
  - 代码管理
  - 构建
  - 代码智能
- 挑战：
  - 技术难度
  - 工作流程的整合

Q & A

开发工具的云端化

赵扶摇
fuyaoz@insight.io

Insight.io

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ

References

- [1] Why Google Stores Billions of Lines of Code in a Single Repository: https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext
- [2] How Developers Search for Code: A Case Study: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43835.pdf
- [3] Tricorder: Building a Program Analysis Ecosystem: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43322.pdf
- [4] Google Engineering blog: http://google-engtools.blogspot.com/
- [5] F8 2015 - Big Code: https://www.youtube.com/watch?v=X0VH78ye4yY&t=10s
- [6] Scaling Mercurial at Facebook: https://code.facebook.com/posts/218678814984400/scaling-mercurial-at-facebook/

References

- [7] Development at the Speed and Scale of Google: https://qconsf.com/sf2010/dl/qcon-sanfran-2010/slides/AshishKumar_DevelopingProductsattheSpeedandScaleofGoogle.pdf
- [8] Large-Scale Automated Refactoring Using ClangMR: https://qconsf.com/sf2010/dl/qcon-sanfran-2010/slides/AshishKumar_DevelopingProductsattheSpeedandScaleofGoogle.pdf
- [9] Continuous Integration at Google Scale: http://eclipsecon.org/2013/sites/eclipsecon.org.2013/files/2013-03-24%20Continuous%20Integration%20at%20Google%20Scale.pdf
- [10] Still All on One Server: Perforce at Scale: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/39983.pdf
- [11] Announcing GVFS (Git Virtual File System: https://blogs.msdn.microsoft.com/visualstudioalm/2017/02/03/announcing-gvfs-git-virtual-file-system/

References

- [12] How Google Code Search Worked: https://swtch.com/~rsc/regexp/regexp4.html
- [13] The Setup Interview: https://usesthis.com/interviews/jeff.dean/
- [14] Linus Torvalds on git: https://www.youtube.com/watch?v=4XpnKHJAok8
- [15] An Overview of Kythe: http://www.kythe.io/docs/kythe-overview.html