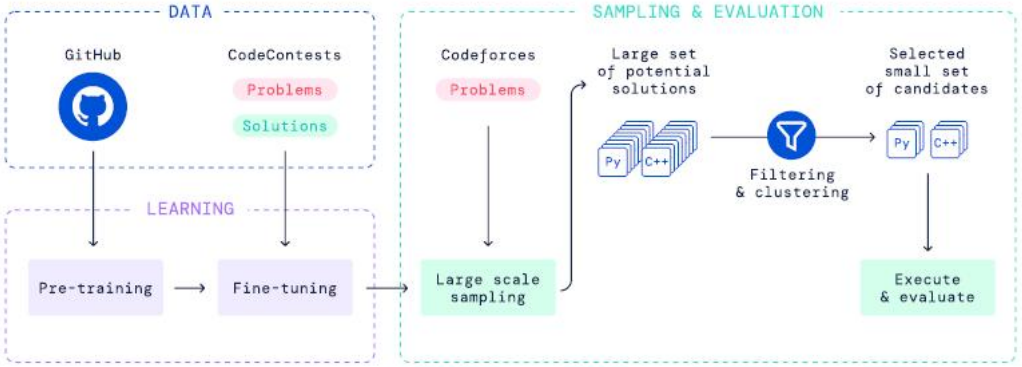
难题1：基于大模型的代码生成技术

出题组织： PaaS技术创新Lab | PaaS开发服务域　　接口专家：马宇驰 mayuchi1@huawei.com

技术背景

2021年6月，微软推出了 AI结对编程助手Copilot，2022年1月谷歌发布了面向程竞赛题编码的AlphaCode，2022年12月，OpenAI发布了具有代码特性的 ChatGPT，2023年1月国际计算机领域的顶刊 CACM发表了文章“编程的终结”。这一系列进展引发了业界广泛的讨论甚至争论。

2022年6月，微软CEO Satya宣布：Copilot结束测试正式发布，收费100美元/年，展示了这个领域商业模式的可行性。



Model	Size	n_{ctx}	n_{vocab}	Data (GB)	Train Tokens	HumanEval (%) pass@1	pass@10	pass@100
GPT-Neo [10]	125 M	2 K	50 K	1,254	300 B	0.75	1.88	2.97
Codex [16]	300 M	4 K	50 K	729	400 B	13.17	20.37	**36.27**
AlphaCode [46]	302 M	1.5 K+768	8 K	715	354 B	11.60	18.80	31.80
CodeGen Multi [50]	350 M	2 K	50 K	1,595	512 T	6.67	10.61	16.84
CodeGen Mono [50]	350 M	2 K	50 K	1,812	665 T	12.76	23.11	35.19
PanGu-Coder	317 M	1 K	42 K	147	211 B	**17.07**	**24.05**	34.55
Codex	679 M	4 K	50 K	729	400 B	16.22	25.70	40.95
AlphaCode	685 M	1.5 K+768	8 K	715	-	14.20	24.40	38.80
AlphaCode	1.1 B	1.5 K+768	8 K	715	590 B	17.10	28.20	45.30
GPT-Neo	1.3 B	2 K	50 K	1,254	380 B	4.79	7.47	16.30
Codex	2.5 B	4 K	50 K	729	400 B	21.36	35.42	**59.50**
PanGu-Coder	2.6 B	1 K	42 K	147	387 B	**23.78**	35.11	50.57
CodeGen Multi	2.7 B	2 K	50 K	1,595	1,012 T	14.51	24.67	38.56
CodeGen Mono	2.7 B	2 K	50 K	1,812	1,331 T	23.70	**36.64**	57.01
GPT-Neo	2.7 B	2 K	50 K	1,254	420 B	6.41	11.27	21.37
GPT-J [67]	6 B	2 K	50 K	825	402 B	11.62	15.74	27.74
CodeGen Multi	6.1 B	2 K	50 K	1,595	1,012 T	18.20	28.70	44.90
CodeGen Mono	6.1 B	2 K	50 K	1,812	1,331 T	26.13	42.29	65.82
InCoder [27]	6.7 B	2 K	22.6 K	216	52 B	15.20	27.80	47.00

2012诺亚语音语义Lab与PaaS技术创新Lab联合发布的PanGu-Coder至今多项指标业界第一

技术挑战

- 代码表征与模型架构挑战：目前的大模型技术主要基于自然语言逻辑构建，当前预训练大多采用raw text形式直接输入，未能充分考虑代码特性，如何探索更优的代码表征与模型架构，以提升大模型针对代码生成能力的准确率是当前的核心挑战。
- 十亿/百亿级高效模型挑战：当前的生成模型的正确性和可用性都有较大的提升空间，特别是十亿/百亿级参数量的模型，如何提升十亿/百亿级参数量模型的生成能力，能够以更低的生成成本达到高质量的代码生成是一项长期技术挑战。
- 知识引入挑战：如何让模型更好地利用已有的代码知识，已有的小领域的代码库来提升特定领域或者场景的代码生成能力，提升整体代码生成的可用性，特别是在小领域或者长尾复杂问题的生成上提升效果。
- 编缉交互挑战：当前的生成能力为从左到右的生成，能否基于上下文对现有代码进行交互式的编辑，修改功能或者错误。

当前结果

- 构建了TB级的多语言代码训练库以及数据清洗工程，为多种语言代码的生成训练提供了数据服务的支撑，可满足各类代码语言模型训练的需求。
- 完成了3亿 与 26亿 2个参数量级的代码大模型的构建，实现基于函数描述生成函数代码的基本能力，部分指标业界领先。

技术诉求

- 探索更优的代码表征与模型架构， **26亿、100+亿参数规模模型PASS@1在HumanEval与CoderEval上超越PanGu-Coder，争取业界领先。**
- 探索设计更高效的方法来增强现有模型，例如在V100推断环境下，**挑战26亿**参数规模模型实现**10ms**单token推断，**挑战100亿+**参数规模模型**15ms**单token推断。
- 探索设计可以基于已有模型使用例如已有代码进行辅助增强的方式，提升整体生成能力。
- 探索设计可以基于已有模型扩展出编辑，注释生成等功能的模型能力。

参考文献：
[1] Chen, Mark, et al. "Evaluating large language models trained on code." arXiv preprint arXiv:2107.03374 (2021).
[2] Christopoulou, Fenia, et al. "PanGu-Coder: Program Synthesis with Function-Level Language Modeling." *arXiv preprint arXiv:2207.11280* (2022).
[3] Li, Yujia, et al. "Competition-level code generation with AlphaCode." *arXiv preprint arXiv:2203.07814* (2022).

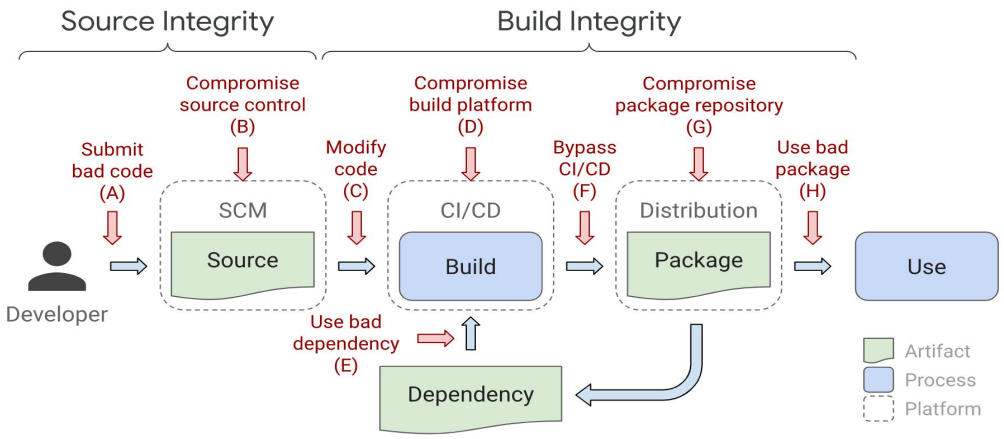
难题2：开源供应链风险识别与修复技术

出题组织：PaaS技术创新Lab | PaaS安全服务域　　接口专家：梁广泰 liangguangtai@huawei.com

技术背景

研究显示：目前应用软件中80%~90% 的代码来自于开源组件，平均每个应用使用的开源软件数量达 100多个，平均每个软件应用存在 50多个开源软件漏洞。开源供应链为企业提供了大量“免费午餐”，但开源漏洞与恶意开源成分的存在也给企业带来了严峻的安全风险。

据2021年7月发布的《2020年中国网络安全报告》称，软件供应链攻击已成为2020年影响最大的威胁之一。开源漏洞从“初始被发现”到“被用于针对企业项目进行漏洞预警与治理”之间仍存在较大的空档期。该空档的存在，已经成了黑客利用开源漏洞作案的绝佳时机。



Google软件供应链威胁模型 SLSA

技术挑战

- **CVE开源漏洞分析与修复技术：**CVE报告以人工描述为主，结构化程度欠佳，CVE所影响漏洞包范围尚未精准描述，系统无法直接消费；三方库版本演化过程中会存在重命名、废弃、fork等情况，需要在海量三方库元数据中进行版本映射追踪，较为繁琐。
- **隐匿漏洞补丁识别技术：**开源社区每天存在大量版本提交，快速甄别出哪些提交正在修复一个安全漏洞，哪些属于高危安全漏洞的补丁非常困难；针对不同类型的漏洞修复补丁，如何提炼出修复特征并利用该特征帮助定位出更多类似的隐匿漏洞补丁具有较大技术挑战。
- **未知漏洞及恶意成分的检测技术：**如何基于已曝光漏洞进行特征学习，举一反三，检测出更多相似的未知漏洞；针对大量疑似未知漏洞，如何高效完成POC验证，快速识别出可利用高危漏洞。

当前结果

- **开源组件漏洞高效感知：**初步完成面向Java、Python的CVE公开漏洞智能精化技术孵化与落地，Java漏洞库成功收录17万+条精准漏洞信息（定位到库版本）。
- **开源漏洞精准极速修复：**支持Java场景库升级、替代方案推荐，支持库移植场景多类api自动化适配能力。

技术诉求

- **开源组件漏洞高效感知与修复：**对于公开漏洞及隐匿漏洞补丁，探索高效的公开漏洞影响包版本范围智能分析算法、隐匿漏洞补丁智能检测算法等，准确率高于80%。
- **未知漏洞及恶意成分的自动检测技术：**基于各种现有的基础技术(例如：静态、动态、克隆等)，探索一系列的**未知漏洞自动挖掘技术，**提供更加完备的风险库修复辅助技术等，准确率高于70%。

参考文献：
[1] Stefanus A. Haryono, Hong Jin Kang, Abhishek Sharma, Asankhaya Sharma, Andrew E. Santosa, Ming Yi Ang, David Lo: Automated identification of libraries from vulnerability data: can we do better? ICPC 2022: 178-189，2020
[2] Bozhi Wu, Shangqing Liu, Ruitao Feng, Xiaofei Xie, Jingkai Siow, and Shang-Wei Lin. Enhancing Security Patch Identification by Capturing Structures in Commits. IEEE Transactions on Dependable and Secure Computing (TDSC), 2022. (accepted for publication).
[3] Kaifeng Huang, Bihuan Chen, Congying Xu, Ying Wang, Bowen Shi, Xin Peng, Yijian Wu, Yang Liu: Characterizing usages, updates and risks of third-party libraries in Java projects. Empir. Softw. Eng. 27(4): 90 (2022)

难题3：代码缺陷的精准静态检测与修复技术

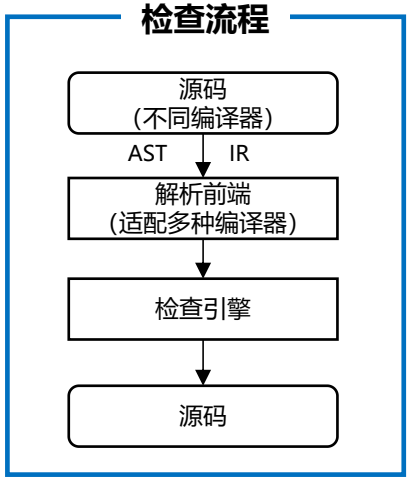
出题组织：PaaS技术创新Lab | PaaS安全服务域　　接口专家：边攀 bianpan@Huawei.com

技术背景

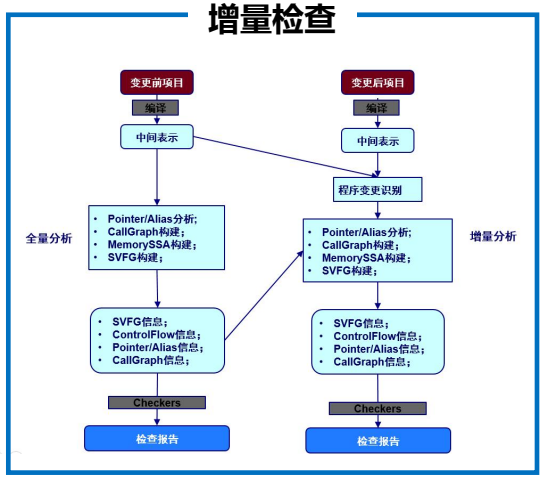
随着软件规模的逐步增加，代码缺陷的静态检测面临着巨大挑战：检查时间长、准确率低、工程适配难等正逐步成为制约其广泛应用的主要问题。

对于大规模软件，往往告警量巨大，人工修复成本高。此外，在实际修复过程中，开发人员可能因缺乏经验和知识，存在分析困难和不知如何解决的情况。迫切需要探索相应的缺陷自动修复技术，能够自动修复缺陷或者提供有针对性的修复建议。

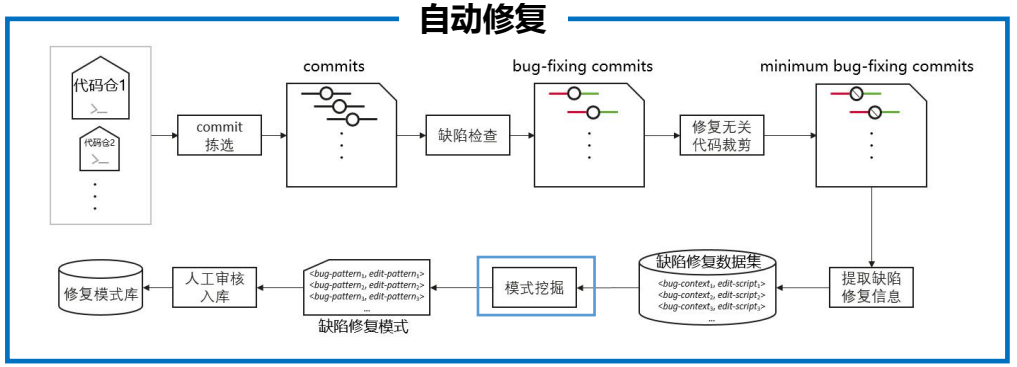
检查流程



增量检查



自动修复



技术挑战

- **精准深度缺陷检查技术**：面向缓冲区溢出、资源泄露、释放后使用、空指针解引用等缺陷类型的检测往往依赖数据流分析及符号执行等深度程序分析技术。业界已对此研究多年，但仍然面临严重依赖编译环境、检查准确率低等多种挑战。增强代码解析前端健壮性、提高分析精度是亟待突破的程序分析基础难题。
- **高速的深度分析技术**：如何综合利用函数摘要、增量分析、并行分析、分布式分析等技术提高深度分析引擎的效率，属于高难度的基础算法方面的研究工作。
- **面向静态缺陷告警的自动修复技术**：如何利用开源社区及华为内部缺陷修复历史数据，有机融合软件分析、机器学习、数据挖掘等技术，有机结合语法树分析、数据流分析、形式化验证等技术，孵化出多层次的代码缺陷自动化修复技术，属业界前沿研究工作，技术挑战高。

当前结果

- **基于形式化描述方法的检查与修复**：定义一种DSL描述语言，能够描述缺陷规则的检查和修复逻辑；实现DSL语言的解析引擎，将DSL描述的检查逻辑转换成clang-tidy和CSA检查器逻辑。此外，目前也在探索基于CPG的形式化描述语言。
- **面向稀疏值流图的增量分析技术**：目前正在探索VFG的增量构建方法。目前已经完成了算法设计，且初步实现了流不敏感的增量分析算法，在大型软件上分析效率提升1倍以上。

技术诉求

- **精准缺陷检测技术**：研究不依赖编译的针对缓冲区溢出、资源泄露、释放后使用、空指针解引用等缺陷类型的精准检测技术，可以在编译环境不完整的情况下进行精准检测，在大规模工程（超千万行）上误报率和漏报率控制在15%以内。
- **高速缺陷检测技术**：重点探索函数摘要、增量分析、并行分析、分布式分析技术，深度检查引擎（数据流分析、符号执行）效率达到700万行/小时。
- **缺陷修复模式挖掘技术**：基于现有的基础技术，从代码提交历史识别安全类缺陷实例及对应修复映射关系，自动提炼修复模式并进行自动化修复。

参考文献：

[1] Bader, Johannes, et al. "Getafix: Learning to fix bugs automatically." *Proceedings of the ACM on Programming Languages* 3.OOPSLA (2019): 1-27.
[2] Ismail, Usman. "Incremental Call Graph Construction for the Eclipse IDE." *University of Waterloo Technical Report*, 2009.
[3] Liu, Bozhen, Jeff Huang, and Lawrence Rauchwerger. "Rethinking incremental and parallel pointer analysis." *ACM Transactions on Programming Languages and Systems* (*TOPLAS*) 41, no. 1 (2019): 1-31.

难题4：面向API的复杂场景测试生成技术

出题组织： PaaS技术创新Lab | PaaS测试服务域　接口专家：万锐媛wanruiyuan@huawei.com；刘喜临 liuxilin3@huawei.com

技术背景

现代化的应用软件大量使用已有的库与服务，以提升开发效率。这些库与服务基本都是以API的方式供外部调用。例如华为云已发布商用的云服务达到220+。以单个服务500+接口数量计算，已商用的云服务的接口总数高达10W+，每个接口的参数量平均超过20个。

现有的主流服务接口测试自动生成工具只能较好地解决单接口测试生成，对于多服务接口的功能交互、协议交互时序、状态相关等场景，生成有效性很差，误报高，效率低下，导致用户E2E业务场景看护不足，易遗漏功能交互场景、异常调用等测试场景，导致现网问题时有发生，甚至服务中断。

当前的API测试生成流程

预期的API复杂场景测试生成流程

技术挑战

- **功能组合测试的生成技术**：目前的API测试生成技术只支持单接口测试，不能处理接口间依赖关系。如何利用接口定义文件和现网用户请求数据，挖掘复杂场景下API接口上下文时序依赖、参数约束依赖和测试Oracle，进而生成API组合功能测试，更全面地看护业务场景。
- **性能压力测试的生成技术**：目前的API性能测试设计效率低下且没有业务依据。如何基于现网及类生产用户请求数据，挖掘测试序列和测试数据时域及频域特征，智能生成接近用户真实行为模式的压测模型，构建用户高频序列并发模式。
- **可靠性测试的生成技术**：如何通过分析高频操作、异常检测和故障传播影响分析，识别高风险接口对象和异常场景，发掘产品和服务的可靠性缺陷，减少现网问题。

当前结果

- **生成有效性**：基于语义感知和搜索技术，生成API接口场景级测试；API请求成功率达到70%以上，误报率低于10%，生成的场景测试序列长度可达10+。
- **测试覆盖率**：接口级覆盖100%，入参组合覆盖100%，响应参数覆盖>93%。
- **缺陷发现能力**：可以发现深层次业务逻辑（数据前后处理不一致、组件级交互场景遗漏）、单接口的参数有效性检测、内存溢出、数据库读写、接口性能等多类问题。

技术诉求

- **功能组合测试生成技术**：基于接口定义文件和现网用户请求等多维度数据，探索生成符合业务逻辑的组合功能测试，提升功能组合测试覆盖率50%。
- **性能压力测试生成技术**：基于现网用户请求数据，实现API组合性能模型自动生成，减少压力模型设计工作量70%以上。
- **可靠性测试生成技术**：自动检测现网异常场景，研究通过结构参数变异等，实现异常场景的复现和挖掘，减少现网问题30%。

参考文献：
[1] RESTTESTGEN: Automated Black-Box Testing of RESTful APIs，ICST2020
[2] Improving Test Case Generation for REST APIs Through hierarchical clustering, ASE 2021
[3] MOREST: Model-based RESTful API Testing with Execution Feedback，ICSE2022

难题5：大规模代码仓库的高速访问技术

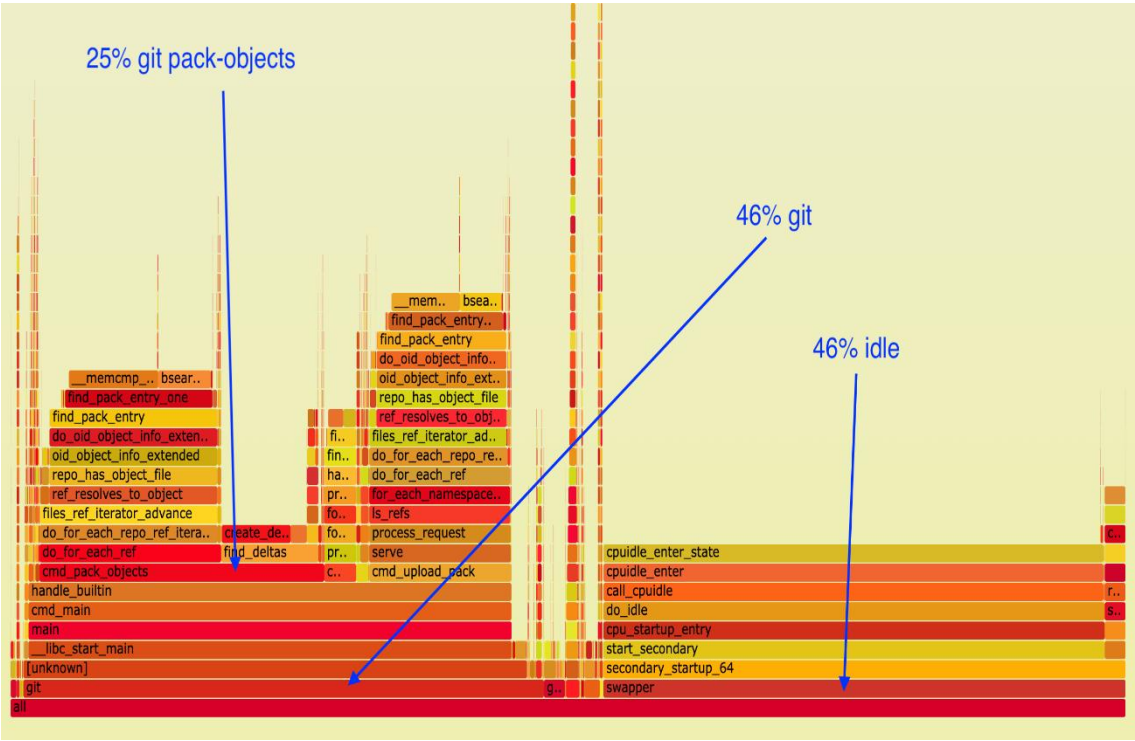
出题组织： PaaS技术创新Lab | PaaS研发协同服务域　接口专家：李飞 lifei91@huawei.com；郝月婵 haoyuechan@huawei.com

技术背景

使用 Git 作为代码仓库托管工具的平台目前普遍面临大规模代码仓的挑战：对于超过100GB或者超过800W对象的代码仓，目前的 Git 存在严重的性能问题，响应时间增加很快。

Git 对代码的基本管理包括上传、下载、后台打包压缩等操作。大规模的代码仓库需要更长的时间才能完成数据遍历与分析，尤其在下载与打包压缩过程，打包流程会导致时间变长，耗时严重。

研发流程中，代码仓的传输性能与响应速度是关键的一环，找到更快的数据搜索遍历、压缩与传输技术，对于节省研发成本，提升生产效率具有重大的意义。



上图来自 gitlab 平台公开的一个 Git 服务器 CPU 负载分析，其中发现 25/54 = 46% 的时间都消耗在 Git 打包进程中

技术挑战

- 大仓库**索引**算法挑战：超大规格的仓库，一般存在超多对象数量和超大存储空间的情况，Git 原生的索引均以对象形式存储于文件中，索引时会对这些数据对象进行加载和遍历，包括获取元数据和必要的文件内容，这会导致时间过慢。
- 大仓库**压缩**算法挑战：Git 会进行仓库内众多对象的计算和对比，然后进行压缩。在获取文件对象时又会进行重新解压和计算，包括类似对象的差分计算等。如何参考存储领域在压缩方面的进展，提高这些计算过程会有非常高的技术挑战。
- 大仓库**传输**算法挑战：Git 存储使用 deflate 算法进行数据压缩和传输，特定情况下，这并不是一种表现很好的算法，如果能够通过更优异的算法来实现新的 Git 下载协议，减少客户端与服务侧的连接时间与传输时间，对时间成本会产生极大优化。

当前结果

- 通过文件对象的**系统缓存**减少文件系统的查找读取消耗，但当仓库规格超过 VM 文件系统缓存大小时，性能就会急剧变差。
- 业界通过增加仓库**只读副本**，增加下载容量，但对于超大规模的仓库，副本无法解决高频写与读并发的场景，每次下载的idle时间也无法降低。

技术诉求

- 大仓库的**极速存储与搜索遍历**的方法与技术，减少数据获取计算耗时**30%**。
- 大仓库的**极速数据压缩**及其中**差分计算的方法**与技术，提升数据获取效率**30-50%**。
- 大仓库的**数据打包传输算法**，减少传输带宽消耗，网络传输的时间优化**50%**。

参考文献：
[1] Tooling for Time- and Space-efficient git Repository Mining, MSR '22: Proceedings of the 19th International Conference on Mining Software Repositories.
[2] Vitalis Salis and Diomidis Spinellis, RepoFS: File system view of Git repositories: SoftwareX, Volume 9, p. 288-292. 2019.
[3] Proposal: packfile cache for Git clone/fetch responses: https://gitlab.com/gitlab-com/gl-infra/scalability/-/issues/688
[4] Proof of concept for git pack-objects cache: https://gitlab.com/gitlab-org/gitaly/-/merge_requests/2832