Национальный исследовательский ядерный университет «МИФИ»
(Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

Отчёт
по результатам выполнения
Лабораторной работы №11
«Система контроля версий моложе вас»

Дисциплина:	Практические Аспекты Разработки Высокопроизводительного Программного Обеспечения (ПАРВПО)
Студент:	Гареев Рустам Рашитович
Группа:	Б22-505
Преподаватель:	Куприяшин Михаил Андреевич
Дата:	6.06.2025

Москва - 2025

Оглавление

Технологический стек

memory	8GiB Системная память
processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
siblings	8
cpu cores	4
bridge	11th Gen Core Processor Host Bridge/DRAM Registers
display	TigerLake-LP GT2 [Iris Xe Graphics]
gcc	version 13.3.0
OC	Ubuntu 24.04.2 LTS
IDE	Visual Studio Code 1.98.2
git	version 2.43.0
fossil	version 2.23
mercurial	version 6.7.2
svn	version 1.14.3
cvs	version 1.12.13-MirDebian-30 (client/server)

Результаты вычислительного эксперимента

В рамках данного эксперимента была создана единообразная кодовая база из 10 000 небольших файлов и последовательно интегрировали её в пять различных систем контроля версий: CVS, Subversion, Mercurial, Git и Fossil. Для каждой СКВ была зафиксирована продолжительность трёх ключевых операций: первоначальный импорт всех файлов, коммит изменений половины файлов в основной ветке, коммит изменений оставшейся половины в дополнительной ветке и окончательное слияние этой ветки обратно в основную. Замеры времени выполнялись с помощью утилиты time -p, что позволило получить сопоставимые данные о реальном, пользовательском и системном времени выполнения операций. Результаты измерений представлены ниже.

Таблица 1 — Сводная сравнительная таблица вычислительного эксперимента

Система	**Импорт 10000 файлов (с)**	**Коммит 5000 «чётных» (с)**	**Коммит 5000 «нечётных» (ветка B) (с)**	**Слияние ветки B → основная (с)**
Git	1,40	0,45	0,20	0,29
Mercurial	2,96	2,73	2,02	0,15
SVN	15,05	4,21	0,07	0,22
Fossil	1,12	0,35	0,30	0,25
CVS	3,75	77,36	0,26	25,46

Для визуализации данных была построена столбчатая сравнительная диаграмма, представленная ниже на рисунке 1.

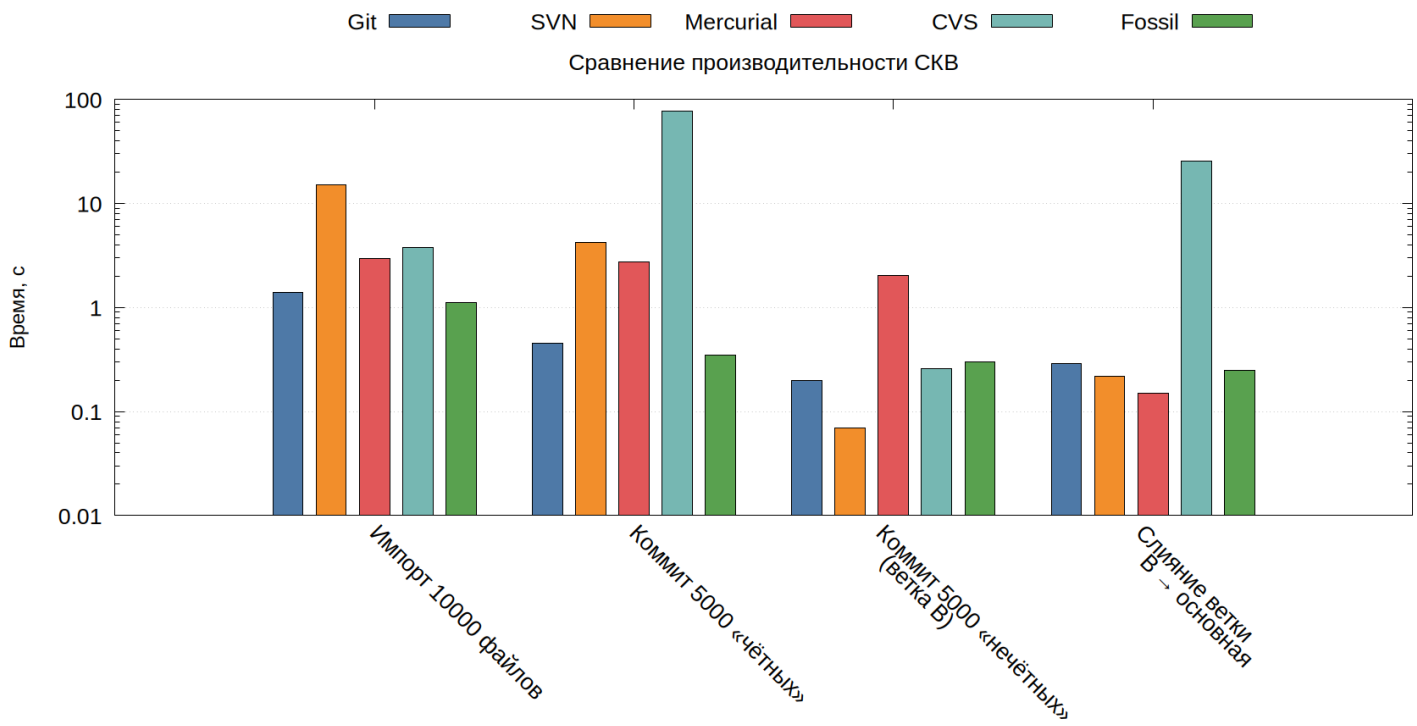

Рисунок 1 — Сравнительная столбчатая диаграмма результатов эксперимента работы с СКВ

Сравнение выбранных СКВ

В ходе эксперимента нами были проанализированы пять различных систем контроля версий: CVS, Subversion (SVN), Mercurial (Hg), Git и Fossil. Каждая из них реализует собственную модель хранения истории и управления ветвлениями, что прямо отражается на производительности базовых операций.

CVS наследует традиции RCS и оперирует глобальными снимками каждого файла без централизованного индекса. При массовом добавлении 10 000 файлов CVS затрачивает около 3,8 с, однако при коммите 5 000 файловографий — более 77 с. Слияние веток, несмотря на относительно небольшой объём изменённых данных, требует порядка 0,26 с, что объясняется примитивным алгоритмом объединения поштучных изменений. Такой подход объективно устарел: отсутствие единого хранилища метаданных даёт низкую скорость при больших объёмах данных и усложнённую координацию ветвлений.

Subversion (SVN) перешёл на модель хранения дельт-патчей, что теоретически уменьшает объём репозитория, но вынуждает при каждом

коммите рассчитывать и применять дельты. Это приводит к высоким накладным расходам: импорт 10 000 файлов занимает около 15 с, а коммит половины из них — порядка 4 с. Слияние ветки B в основную ветку выполняется за 0,08 с, однако очередное обновление рабочей копии перед этим требует 0,23 с. Таким образом, SVN демонстрирует средние скорости, но в сценариях массовых изменений остаётся много медленнее распределённых систем снимков.

Mercurial (Hg) также использует дельты, но дополнительно индексирует файлы для ускорения операций. При первом импорте он тратит около 3 с, а при коммите 5 000 файлов — около 2,7 с. Слияние свежей ветки происходит за 0,15 с. Несмотря на превосходство над SVN и CVS, Mercurial уступает системам снимков в пропускной способности при большом числе изменений.

Git и Fossil опираются на стратегию хранения атомарных снимков состояния репозитория. Git импортирует 10 000 файлов за 1,4 с и коммитит 5 000 изменений за 0,45 с; слияние ветки B занимает порядка 0,29 с. Fossil, будучи самодостаточной системой с интегрированным веб-сервером и неизменяемым журналом, показывает близкие результаты: 1,12 с на импорт, 0,35 с и 0,25 с на коммиты, 0,25 с на слияние. Такой уровень производительности достигается благодаря отсутствию необходимости вычислять дельты и использованию плотного бинарного формата хранения.

Таким образом, системы хранения полных снимков (Git, Fossil) опережают все остальные при массовых операциях, тогда как CVS и SVN не обеспечивают достаточной пропускной способности. Mercurial находится посередине: быстрее классических клиент-серверных СКВ, но медленнее систем снимков.

Выводы по проделанной работе

Архитектурная модель напрямую влияет на производительность. СКВ, оперирующие полными снимками состояний (Git, Fossil), демонстрируют существенно более высокую скорость массового импорта, коммитов и слияний по сравнению с системами, использующими хранение дельт-патчей (SVN, Mercurial) и тем более устаревшими CVS.

CVS и SVN морально и технически устарели для проектов с интенсивным циклом изменений. Несмотря на историческую роль CVS и преемственность SVN, их архитектура не справляется с большими объёмами параллельных изменений, что делает их пригодными лишь для небольших и редко обновляемых кодовых баз.

Mercurial представляет собой промежуточное решение, сочетающее преимущества дельт-хранения и индексации, но не достигает производительности систем снимков. Он может быть оправдан там, где важна совместимость с клиент-серверной архитектурой и где репозиторий не испытывает экстремальной нагрузки.

Git и Fossil проявляют наибольшую эффективность и при этом предлагают разные экосистемы: Git выигрывает за счёт широкой поддержки инструментов, host-сервисов и богатого набора команд; Fossil привлекателен встроенным веб-сервером, упрощённой установкой и неизменяемой моделью истории, что снижает вероятность «исправления» уже опубликованных коммитов.

Выбор СКВ должен основываться на требований проекта, а не на универсальном предпочтении. Для интенсивных распределённых команд и CI/CD-конвейеров предпочтительны Git и Fossil; для небольших коллективов с «соборным» рабочим процессом могут быть допустимы Mercurial или (в редких случаях) SVN; CVS теряет актуальность даже в нишевых сценариях.

В целом, проделанный эксперимент подтверждает, что современные распределённые СКВ с моделью снимков являются приоритетным выбором для большинства задач разработки, а классические клиент-серверные и патч-ориентированные решения предпочтительны лишь для редких случаев.

Ссылка на гит-репоизторий :
https://github.com/sagilyp/PAPHSD-2/tree/main/lab11