Разпределени системи за управление на код

Съдържание

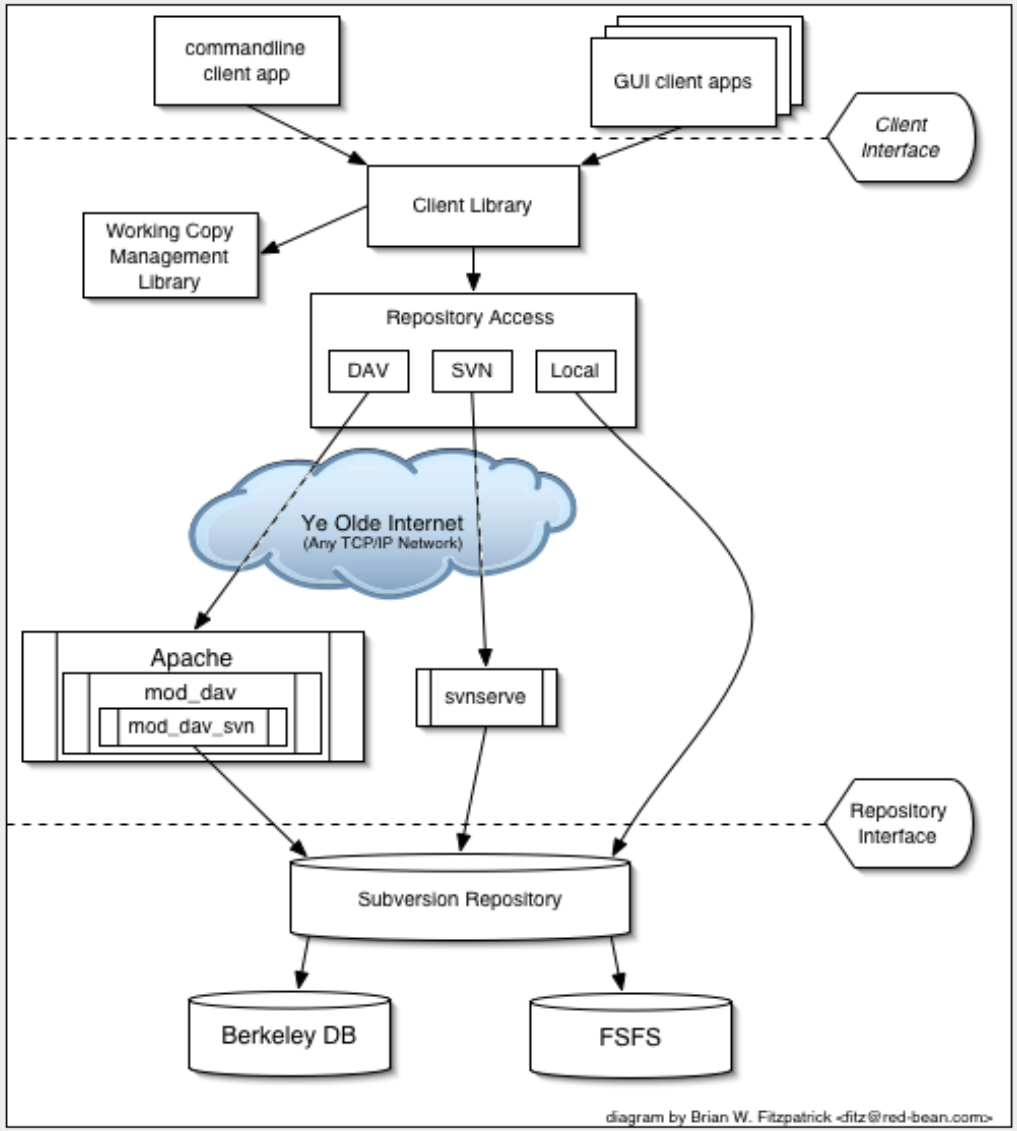
- история
- архитектура на централизирани системи
  - SVN
- архитектура на разпределени системи
  - Mercurial
  - Git
- плюсове и минуси на двата подхода
- заключение

Историческо развитие

- локално следене на файлове
  - SCCS (proprietary, Bell Labs 1972, interleaved deltas)
  - RCS (free software, Walter Tichy 1982, reversed deltas)
- client-server
  - CVS (shell scripts + RCS - multiple files, 1990 - 2008)
  - SVN ("CVS done right", 2000)
- peer to peer
  - BitKeeper (proprietary, 1998)
  - Mercurial (Matt Mackall, 2005)
  - Git (Linus Torvalds, 2005, the fastest out there)
  - Darcs, Camp (Haskell, patch-based)

Архитектура на централизираните системи

- централен сървър пазещ информация за всички ревизии на проекта
- различни модели за достъп до хранилището в зависимост от инфраструктурата
- разнообразни клиенти -- графични, конзолни, разширения на съществуващи IDE



Архитектура на централизирани системи (SVN)

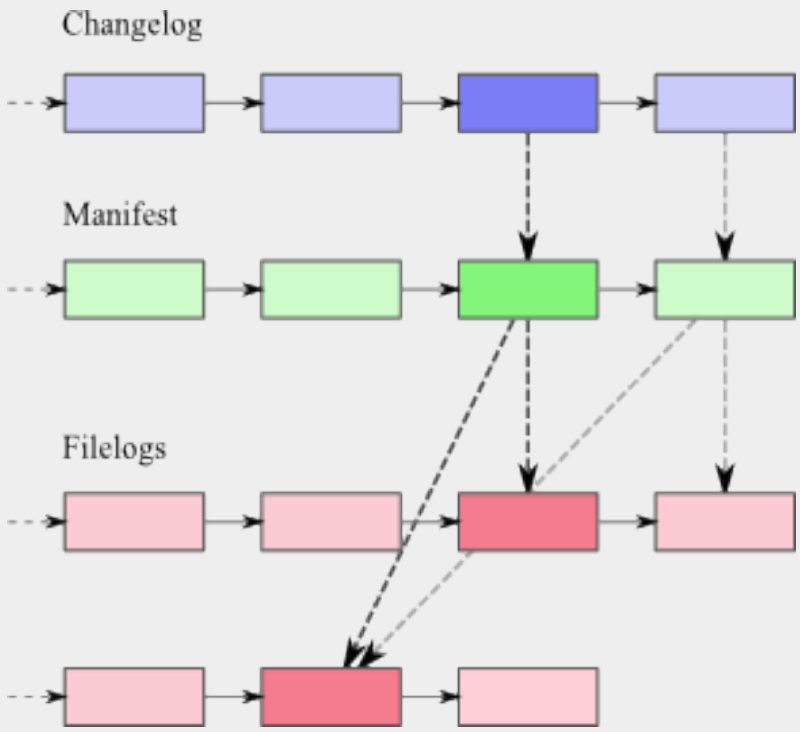
- всеки клиент пази при себе си минимална необходима информация за създаване на нови ревизии -- това най-често е последната запазена ревизия
- най-добре се поддържат линейни ревизии -- всяка зависи от предходната
- има поддръжка на разклонения и сливания, но на практика са тромави и се използват само в краен случай; в по-стари версии има проблеми с имплементацията
- повечето клиентски операции изискват връзка към централното хранилище
- клиента комуникира със сървъра посредством delta -- не се изпраща съдържанието на цялото хранилище за всяка версия

Обща архитектура на разпределените системи

- всеки клиент притежава пълна информация за развитието на целия проект
- проектът представлява DAG (насочен ацикличен граф):
  - върховете са commit-и. Всеки от тях може да се разглежда като някаква промяна на вече съществуваща ревизия (в Mercurial се нарича changeset)
  - ребрата са връзки между ревизиите и описват върху коя съществуваща версия трябва да се приложат промените описани в новата ревизия
- в тази структура разклоненията и сливанията са по-лесни и по-естествени
- произволни две хранилища могат да се свържат и да обменят информация за наличните commit-и и евентуално да си ги разменят -- това просто ще обогати графа на едната (или двете) страни

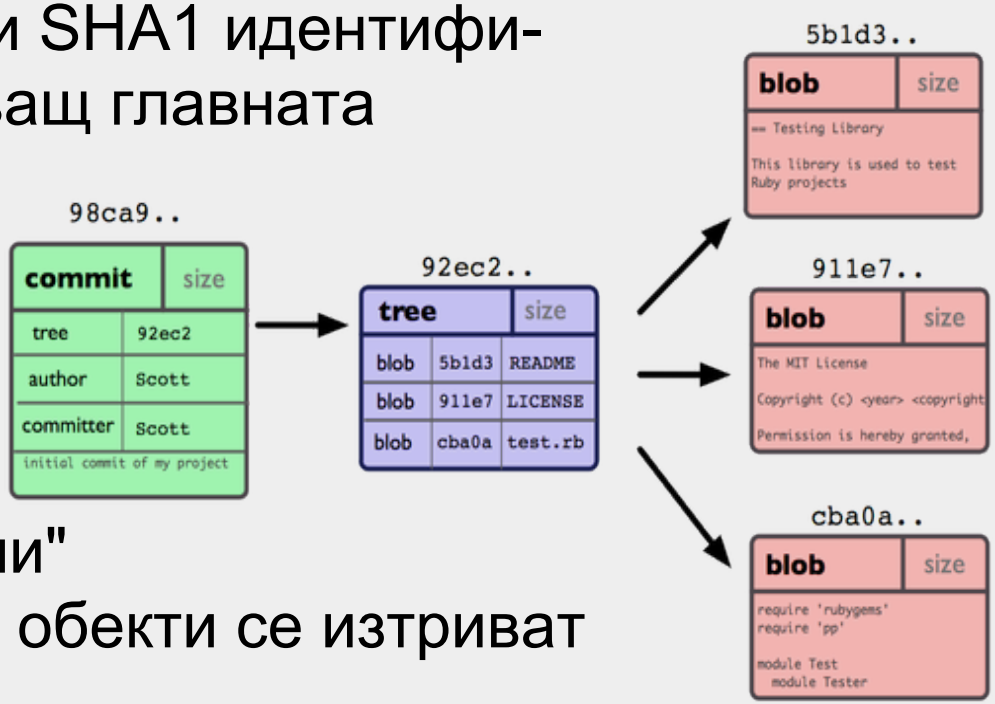
Архитектура на Mercurial

- историята на всеки файл се записва в **revlog** -- това е ефективен начин за запазване на нови и възстановяване на стари ревизии чрез запазване на целия файл или разлика с предишната му версия
- **manifest** файла съдържа версиите на отделните файлове за всяка една ревизия различни могат да сочат към една и съща версия на файл, ако той не е променян
- **changelog** файла пази meta информация за всяка ревизия и версията на manifest файла за тази ревизия
- changelog и manifest се пазят в revlog формат



Архитектура на Git

- всяко Git хранилище е съвкупност от следните обекти:
  - blob -- съхранява произволно съдържание достъпно по ключ, образуван от SHA1 сумата му
  - tree -- съхранява информация за директория -- списък от име, тип (tree / blob) и SHA1 идентификатор
  - commit -- съхранява информация за ревизия -- това включва автор, коментар и SHA1 идентификатор на tree обекта описващ главната директория
- при нормална работа обектите се пазят компресирани по един във файл
- няколко места пазят SHA1 идентификатори на "интересни" обекти -- недостъпните от тях обекти се изтриват след определено време



Предимства на централизираните системи

- проста и лесна за научаване концепция
- по-разпространени са в днешно време, заради което
  - мнозинството програмисти могат да работят с тях
  - имат много добра поддръжка сред:
    - операционни системи
    - IDE програми
    - хостинг услуги
    - консултантски услуги
  - съществуват множество учебни материали за начинаещи и напреднали
- многократно доказани на практика в разработването на proprietary приложения в софтуерни компании

Предимства на разпределените системи

- възможен е централизиран модел на употреба, особено подходящ за софтуерни компании
- позволяват ефективна работа без връзка към интернет
- програмистът има голяма свобода в управлението на разклонения на локалната си машина, което позволява по-лесни експерименти, които не е задължително да бъдат публикувани в централното хранилище
- операциите за работа с хранилището са много бързи, дори при връзка с отдалечен клиент
- създаването на ново хранилище е изключително бързо и лесно, което ги прави идеален кандидат за малки проекти с 1-3^{ма} участници
- включването на външен човек в проекта за кратко време става лесно -- идеален кандидат за проекти с отворен код в които всеки може да допринася без предварителна уговорка

Заключение

- Разпределените системи за управление на код са естествената следваща стъпка в еволюцията на този род системи. Те са базирани на дългогодишен опит с централизирани системи и внимателно отсяване на добрите от лошите им черти.
- Те дават много повече възможности в ръцете на потребителя без отнемат функционалноста съществуваща в централизираните системи, като с това допълнително улесняват прехода към тях.