Системный анализ

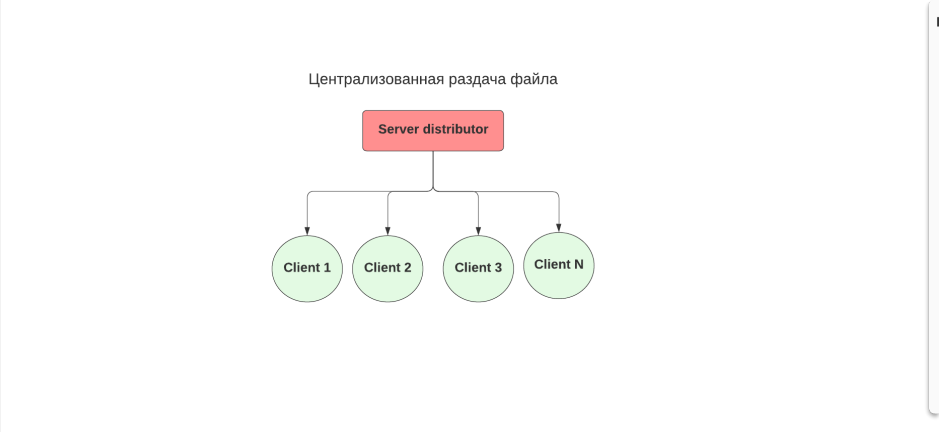
I. Объект и его описание

Объектом системного анализа выступит архитектура файлообмена в моих локальных сетях. Существуют различные реализации файлообмена, параллельный вариант, в котором, загрузка происходит сразу из пары мест или централизованный (линейный), в котором файл попадает к получателям по очереди.

II. Проблематика

В моем случае, нужно выбрать децентрализованную архитектуру, так как приватность мне не важна, так как сеть локальная, а главным требованием является скорость передачи, так как сейчас я не пролезаю по пропускной способности сети. Существующие реализации мне не подходят (например BitTorrent), так как обладают WEB интерфейсом, а это накладные расходы и не достаточно удобны для интеграции с другими автоматическими системами, например, создание резервной копии базы данных. Приведу еще пару аргументов в пользу децентрализованной архитектуры:

1. **Critical:** Во время множественного скачивания тяжелого файла с одного или нескольких серверов, может произойти отказ в обслуживании из-за высокой или неравномерной нагрузки
2. **Critical:** В случае завершения работы дистрибьютора, entrypoint для загрузки будет утерян
3. **High:** Сервер может находиться в другом регионе, что сильно повышает задержки
4. **High:** Протокол загрузки полностью зависит от дистрибьютора, поэтому может быть использован неоптимальный



III. Цель и задачи

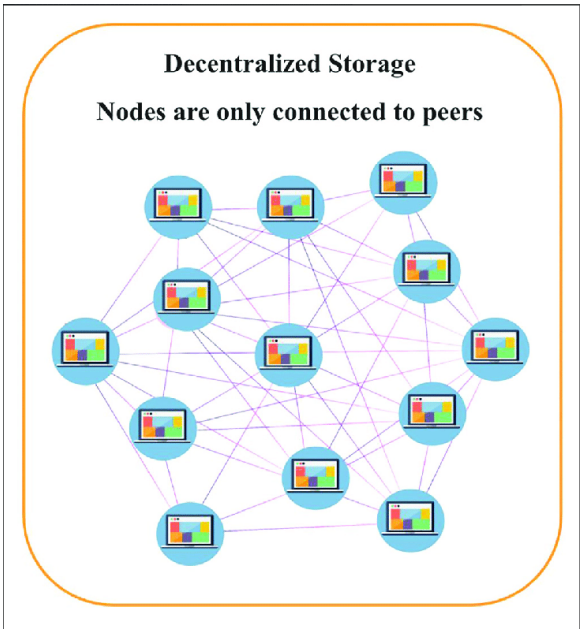
Цель: решить проблему задержек в сети путем создания удобного программного продукта

Задачи: необходимо создать несколько источников раздачи файла, однако тогда получатель будет знать все хосты, которые уже получили фрагменты файла (далее будет использовано слово - **партиция**). Это необходимый trade-off для достижения параллельного получения фрагментов. По сути, каждый клиент становится сервером раздачи (**трекером**). Так же нужно разработать механизмы балансировки нагрузки, например: выбор оптимального количество соединений на каждом трекере, выбор трекеров с наибольшей пропускной способностью сети. Нужно описать и создать модель общения клиента peas с продуктом через CLI.

Технические особенности задач:

1. Вопрос: откуда скачивающий узнает о всех трекерах?
    a. Дистрибьютор создает .peas файл, указывая свой IP-адрес, хэш-сумму файла и различные мета-данные
    b. Клиент, как либо, получает .peas файл и переходит по указанному адресу
    c. Сервер записывает адрес нашего клиента в этот же .peas файл и уведомляет всех трекеров

d. Клиент, наблюдая в .peas файле, таких же клиентов, начинает запрашивать у них партиции



2. Что отдаем взамен:
    a. Приходится доверять .peas файлу
    b. Все участники знают адреса друг друга
    c. Trade-off между ускорением на мультисессионной загрузке и вариацией хэш-сумм (поэтому на маленьких файлах, скорость может только замедлиться)
    d. Слабая защита на бесконечное добавление IP-адресов владельцев в peas файл (нужно придумать механизировано актуализации владельцев, например по времени)

Если потребителю продукта важна безопасность и анонимность, то ему следует рассмотреть задачи и с пониманием выбрать подходящее технологическое решение.

IV. Граф “Цели - Задачи”

Файл для редактирования на https://programforyou.ru/block-diagram-redactor

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/881f35b8-d8cd-49d7-bde4-886423ddd6a9/diagram.json

V. Граф функций системы

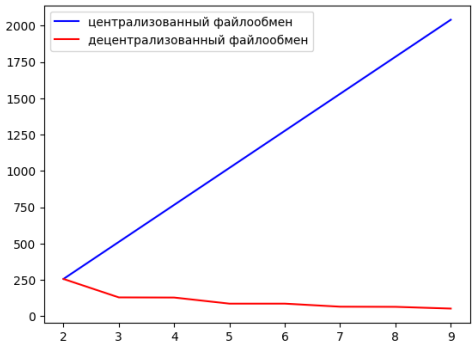
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bade67ab-8809-4f4f-aff2-06f11f78fab0/diagram-2.json

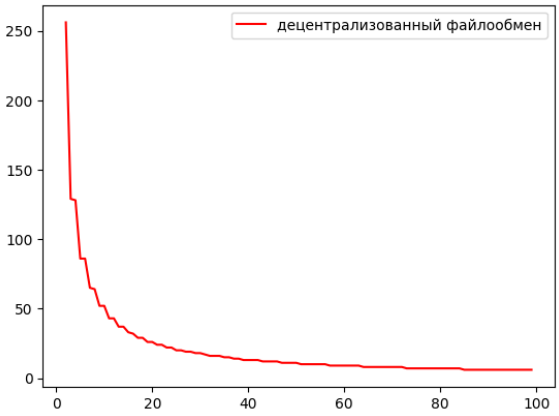
Был проведен опыт, чтобы сравнить производительность двух видов архитектуры файлообмена.

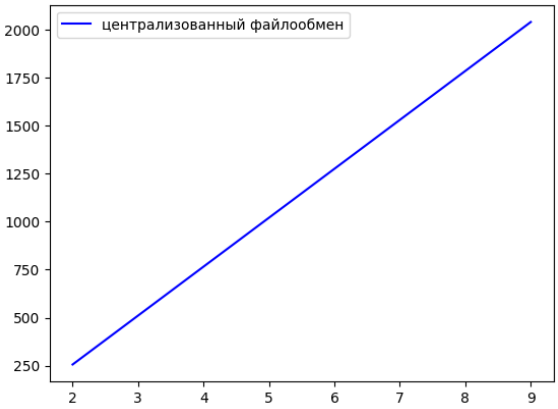
Подробнее: https://github.com/funcid/data-science/blob/master/misc/Сравнение архитектур файлообмена для оценки PeasCLI.ipynb

Результаты:

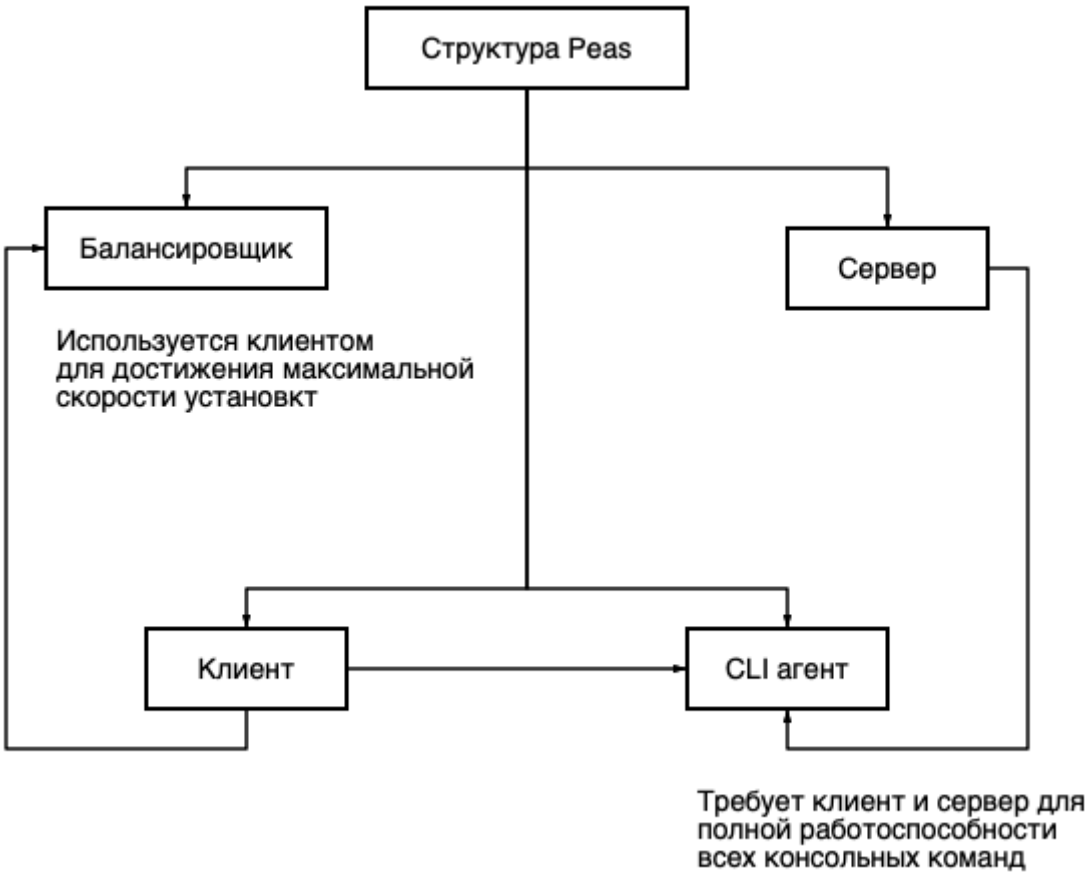
Время загрузки с ростом числа участников сети для обоих вариантов, эмпирическим путем получено, что централизованный файлообмен - линейный, а децентрализованный - экспоненциальный и выигрывает по скорости передачи в разы.







VI. Граф структуры системы



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5a829e7a-d36a-4924-8311-7610d39d1a45/diagram-5.json

VII. Теоретико-множественная модель системы

Общий вид теоретико-множественной модели представляет собой

S = <A, Q, V, Z>

A – компоненты системы

Q – свойства системы

V – связи между компонентами системы

Z – цели системы

A = <a1, a2, a3, a4>

a1 - множество тех, кому нужно данные получить

a2 - множество тех, кто данные раздает

a3 - множество всех сессий

a4 - множество всех peas описательных файлов

Q = <q1, q2, q3>

q1 - система открыта

q2 - система доступна

q3 - система формальна

V = <v1, v2>

v1 $\subseteq$ a1 × a2 × a3 - связь между участниками сети

v2 $\subseteq$ a2 × a4 - связь между владельцами и адресными описаниями данных

Z = <z1, z2, z3>

z1 - для N > 2 участников сети достигается экспоненциально большая приемная способность по сравнению со standalone загрузкой

z2 - в случаях одновременной загрузки, дистрибьютор имеет в N раз меньшую нагрузку на сеть

z3 - при недоступности оригинального дистрибьютора, клиенты могут установить данные у N-1 непроверенных трекеров

Итоговая модель:

S = <a1, a2, a3, a4, q1, q2, q3, v1, v2, z1, z2, z3>