



Системный анализ

I. Объект и его описание

Объектом системного анализа выступит информационный продукт (Peas) собственной разработки, реализующий децентрализованную загрузку и раздачу данных по сети интернет, предоставляющий CLI.

Технологический stack: Java 17, GSON, Picocli, JUnit, Java NIO

Области применения:

1. Оптимизация механизма репликации баз данных (например в Cassandra, там есть только режимы single, ring)
2. Ускорение отправки snapshot версий на несколько backup серверов (например результаты аналитики big data)
3. Выпуск большого обновления клиентской части какой-либо игры (например вышло обновления на halloween)
4. Быстрая загрузка первичного ПО на только что купленные хосты с ansible (например для установки защитного обеспечения)

Статус релиза: на момент написания этого документа (24.04.2023) продукт в статусе разработки

Команда продукта и распределение ролей:

Артём Царюк @funcid - solution architect, product owner, system analyst

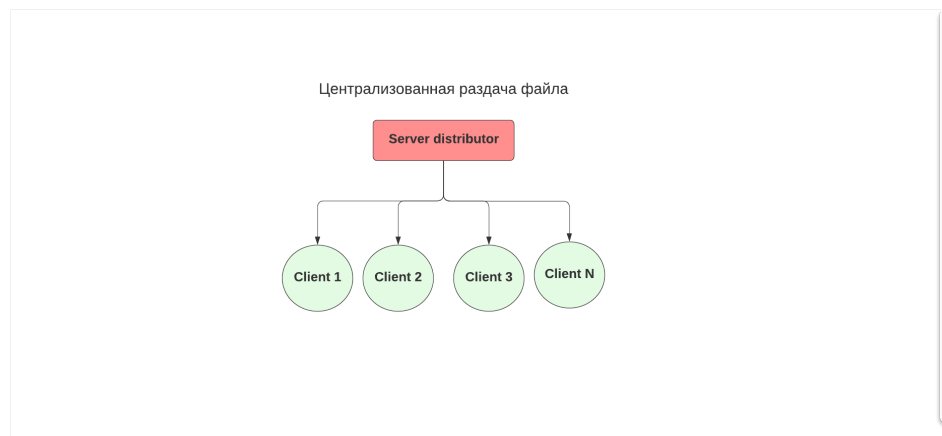
Владислав Иванов @anyuser9999 - software engineer, QA engineer

Исходный код продукта: <https://github.com/funcid/peas-cli>

Лицензия распространения: Apache License 2.0

II. Проблематика

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, endpoint 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

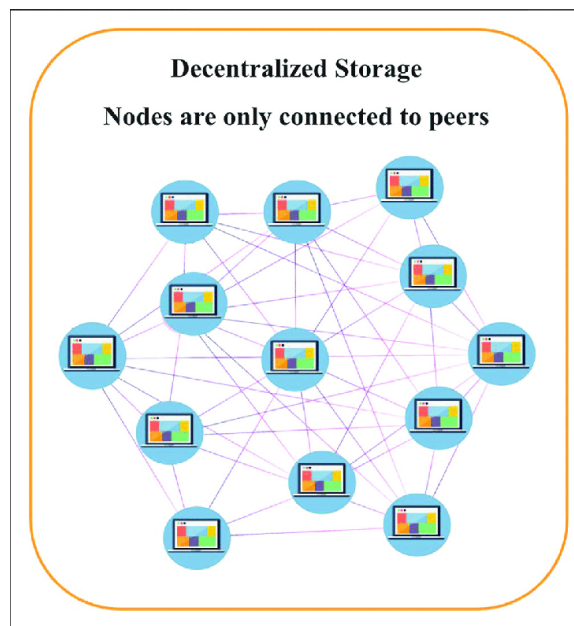
Цель: решить проблемы standalone дистрибьютора путем реализации открытого программного продукта

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 reas с продуктом через CLI.

Технические особенности задач:

1. Вопрос: откуда скачивающий узнает о всех трекерах?
 - a. Дистрибьютор создает .reas файл, указывая свой IP-адрес, хэш-сумму файла и различные мета-данные
 - b. Клиент, как либо, получает .reas файл и переходит по указанному адресу
 - c. Сервер записывает адрес нашего клиента в этот же .reas файл и уведомляет всех трекеров
 - d. Клиент, наблюдая в .reas файле, таких же клиентов, начинает запрашивать у них партии

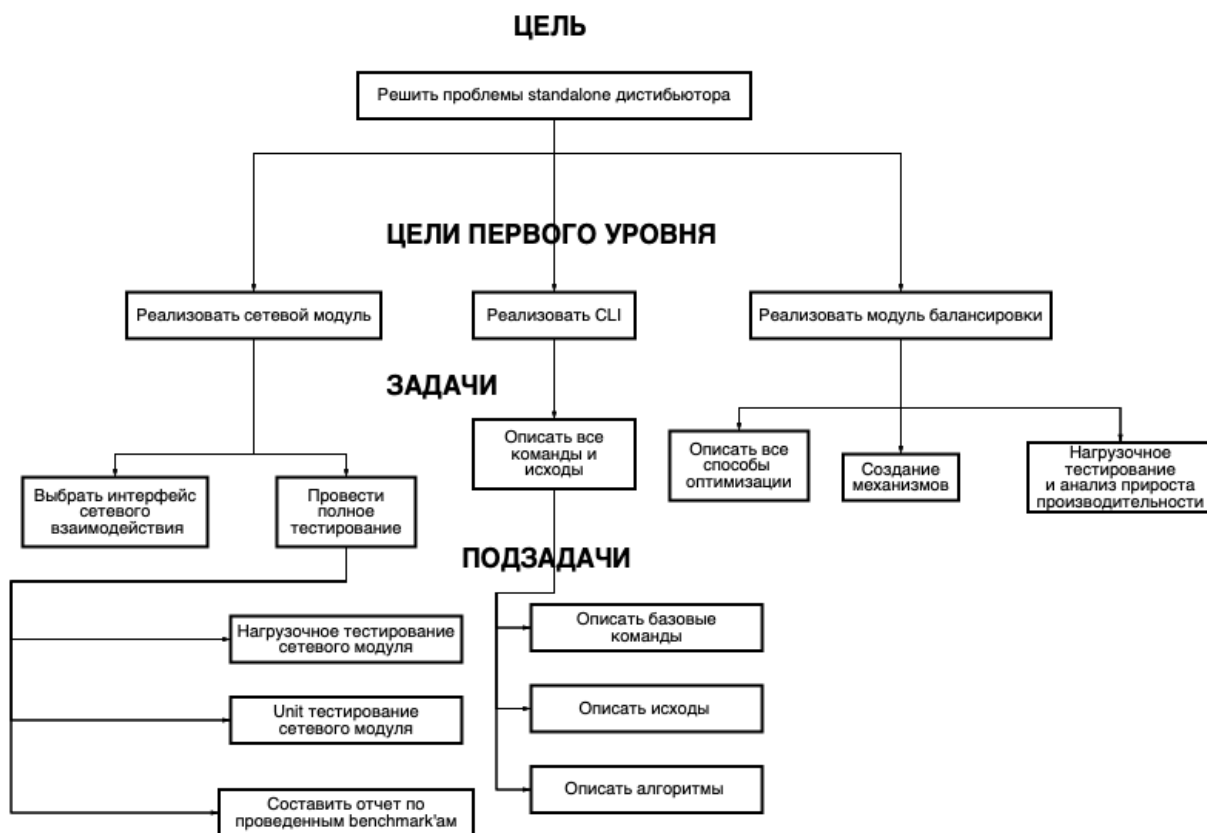


2. Что отдаем взамен:
 - a. Приходится доверять .reas файлу
 - b. Все участники знают адреса друг друга

- с. 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 peers 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.

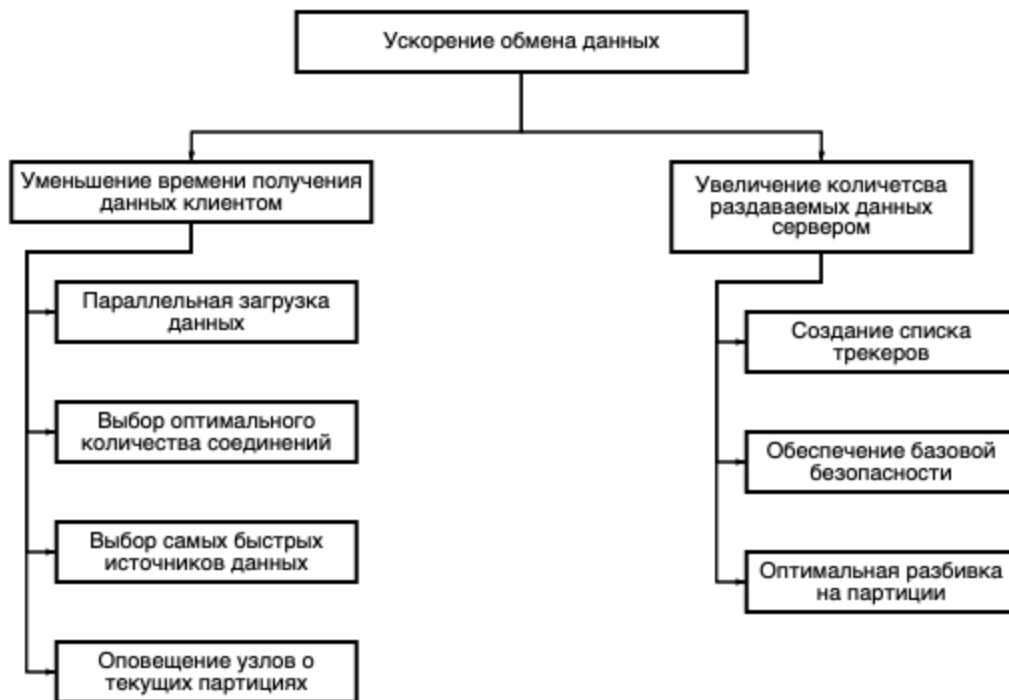
IV. Граф “Цели - Задачи”



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>

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

$$S = \langle A, Q, V, Z \rangle$$

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 реас описательных файлов

Q = <q1, q2, q3>

q1 - система открыта

q2 - система доступна

q3 - система формальна

V = <v1, v2>

v1 \subseteq a1 \times a2 \times a3 - связь между участниками сети

v2 \subseteq a2 \times 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>

