# try 1

разделение execution и storage

Цель

привнести инновации и новые технологии в технологические процессы в СМ

Разделение execution и storage слоёв в больших data-платформах, и действительно, многие компании на Западе и в мире давно двигаются в этом направлении.

Что раньше: монолитный Hadoop (HDFS + YARN/Spark на одном кластере)

Ранее архитектура выглядела так:

HDFS — хранилище (storage)

YARN + Spark/MapReduce — исполнение (execution)

Всё это было на одних и тех же физических узлах.

🧨 Проблемы:

Если выполняются тяжёлые ML-задачи — execution нагружен, а storage простаивает.

Если кластер используется как "холодное хранилище" — HDFS забит, а вычисления стоят.

При масштабировании — непонятно, что масштабировать: диски или CPU?

Что делают сейчас: разделение execution и storage

🔄 Разделение означает:

| Компонент | Что это | Примеры |
| --- | --- | --- |
| Storage Layer | Где лежат данные | S3, Ceph, HDFS, ClickHouse, Delta Lake |
| Execution Layer | Где происходит вычисление | Spark, Dask, Flink, Ray, Trino, Presto |

Примеры современных архитектур:

| Хранилище (разделено) | Исполнение (отдельно) |
| --- | --- |
| Ceph | Apache Spark |
| S3 | Dask, AWS EMR |
| ClickHouse (read-only node) | Presto/Trino |
| Redis (для fast access) | ML pipeline (в Spark или Python) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* У вас execution слой — Spark (затянули выполнение туда).
* Storage — это Ceph, Redis (как кэш), ClickHouse (для аналитики).
* Сторож изолирован, не мешает вычислительным задачам → вы избегаете конфликтов.

Это современный, разделённый подход (disaggregated architecture), аналогичный тому, что используют:

* Uber — Spark на k8s + HDFS в S3
* Netflix — Spark + S3/Parquet
* Snowflake — полностью разделённый execution и storage
* Meta (Facebook) — Presto работает поверх HDFS, при этом compute-пулы изолированы

🧠 Почему это правильно

### Преимущества:

* Горизонтальное масштабирование по нужде: только CPU или только storage
* Отказоустойчивость — storage живёт отдельно
* Экономия: compute можно арендовать "по требованию" (например, как в AWS)
* Гибкость: один storage может обслуживать Spark, Presto, ML и другие движки

✏️ Итого

Ты описал очень зрелую и правильную архитектуру:

* Вы отделили storage и execution
* Используете Spark для вычислений
* Ceph — как надёжное хранилище
* Redis — для быстрого доступа к "горячим" данным
* ClickHouse — как аналитическую БД (read-optimized, columnar)

Это полностью соответствует тому, что сейчас делают на передовой в data-инженерии.

Если хочешь, могу помочь:

* Нарисовать архитектурную схему (в текстовом виде или визуально)
* Предложить инструменты мониторинга, отказоустойчивости
* Рассказать, как сделать "data mesh" с такой архитектурой

Хочешь?