**1. Schema Design**

**Tables:**

* **IPDR Table:** Stores network traffic logs (timestamps, IPs, ports, domains, bytes).
* **AAA Table:** Tracks user sessions (login/logout events with IPs, status, session info).

**Why separate tables?**

* Different data types and use cases (IPDR is high-volume, write-heavy; AAA is smaller and asynchronous).
* Avoids duplication, simplifies ingestion pipelines, and enables independent tuning.
* Allows optimized partitioning and indexing per table.

**Partitioning:**

* Partition **both tables by day** (e.g., toDate(timest0)), enabling efficient time-based queries and automatic data cleanup after 180 days.

**Sharding:**

* Shard by **ip\_src** (source IP), not subscriber\_id, to distribute incoming data uniformly and avoid hotspotting.

**Primary Keys:**

* **IPDR:** (timest0, ip\_src, domain) → Optimizes filters by time, IP, and domain.
* **AAA:** (framed\_ip, timest) → Accelerates session lookup and session-to-traffic joins.

**2. Design Choices**

* **Sharding:** Balances write and query load, keeps same ip\_src traffic on same shard.
* **Partitioning:** Simplifies retention (drop old partitions) and accelerates time-range queries.
* **Primary Key:** Matches query patterns like “get subscriber activity between T1 and T2” or “top domains for IP X.”

**3. Denormalization**

**Pros of separate tables:**

* No data duplication.
* Easier independent optimization (e.g., separate partitioning, storage policies).
* Better compression due to columnar storage specialization.

**Cons:**

* Joins required for queries linking IPDR and AAA (e.g., subscriber activity reports).
* Mitigation: Use **Materialized Views** to pre-join common patterns and speed up dashboards.

**4. Storage Estimates**

* **IPDR:**
  + 10M records/sec → **~172 TB/day** (compressed with ClickHouse: ~10-15x compression).
  + For 180 days retention: around **31 PB raw, 3-5 PB compressed** (not 10-12 TB).
* **AAA:**
  + 10K records/sec → **~86 GB/day** → **~15 TB** over 180 days (small compared to IPDR).

**5. Scaling to 20M IPDR/sec**

* Double the nodes (e.g., from 20 to **40 nodes**).
* Use **Kafka** for scalable, real-time ingestion.
* Separate storage tiers:
  + **Hot (SSD)** → Last 7 days (for dashboards and frequent queries).
  + **Cold (HDD or S3)** → 173 days (for compliance and deep reporting).

**6. Late-Arriving AAA Events**

* Use **time-window joins** during queries (match IPDR timest0 within AAA session start/stop windows).
* Retain AAA data slightly longer (e.g., 185 days) to catch delayed arrivals.
* Optionally run nightly batch jobs to backfill late AAA records into pre-joined views.

**7. Batched Inserts vs. Real-Time**

* **Real-time (Kafka):** Best for live dashboards and alerting.
* **Batched inserts:** Useful for **backfilling** historical data or late arrivals.
* Avoid small batches → ClickHouse prefers **larger insert batches** (e.g., 100,000 rows) for efficiency.

**8. Node Failures in ClickHouse**

* Use **replication** (at least 2 replicas per shard).
* On node failure:
  + Queries automatically route to replicas.
  + Failed node **auto-replicates** missing data when back online.

**9. Alternative Databases**

* **Apache Pinot** or **Druid**: Good for real-time analytics but less mature at petabyte scale.
* **ClickHouse**: Offers better compression (10-15x), proven scalability, mature ecosystem.

***-- IPDR Table***

CREATE TABLE ipdr (

timest0 DateTime64(3),

timest1 DateTime64(3),

ip\_src IPv4,

ip\_dst IPv4,

port\_src UInt16,

port\_dst UInt16,

application\_id UInt32,

domain String,

bytes\_in UInt64,

bytes\_out UInt64

) ENGINE = MergeTree()

PARTITION BY toDate(timest0)

ORDER BY (timest0, ip\_src, domain);

***-- AAA Table***

CREATE TABLE aaa (

subscriber\_id String,

framed\_ip IPv4,

status UInt8,

session\_id String,

timest DateTime64(3)

) ENGINE = MergeTree()

PARTITION BY toDate(timest)

ORDER BY (framed\_ip, timest);