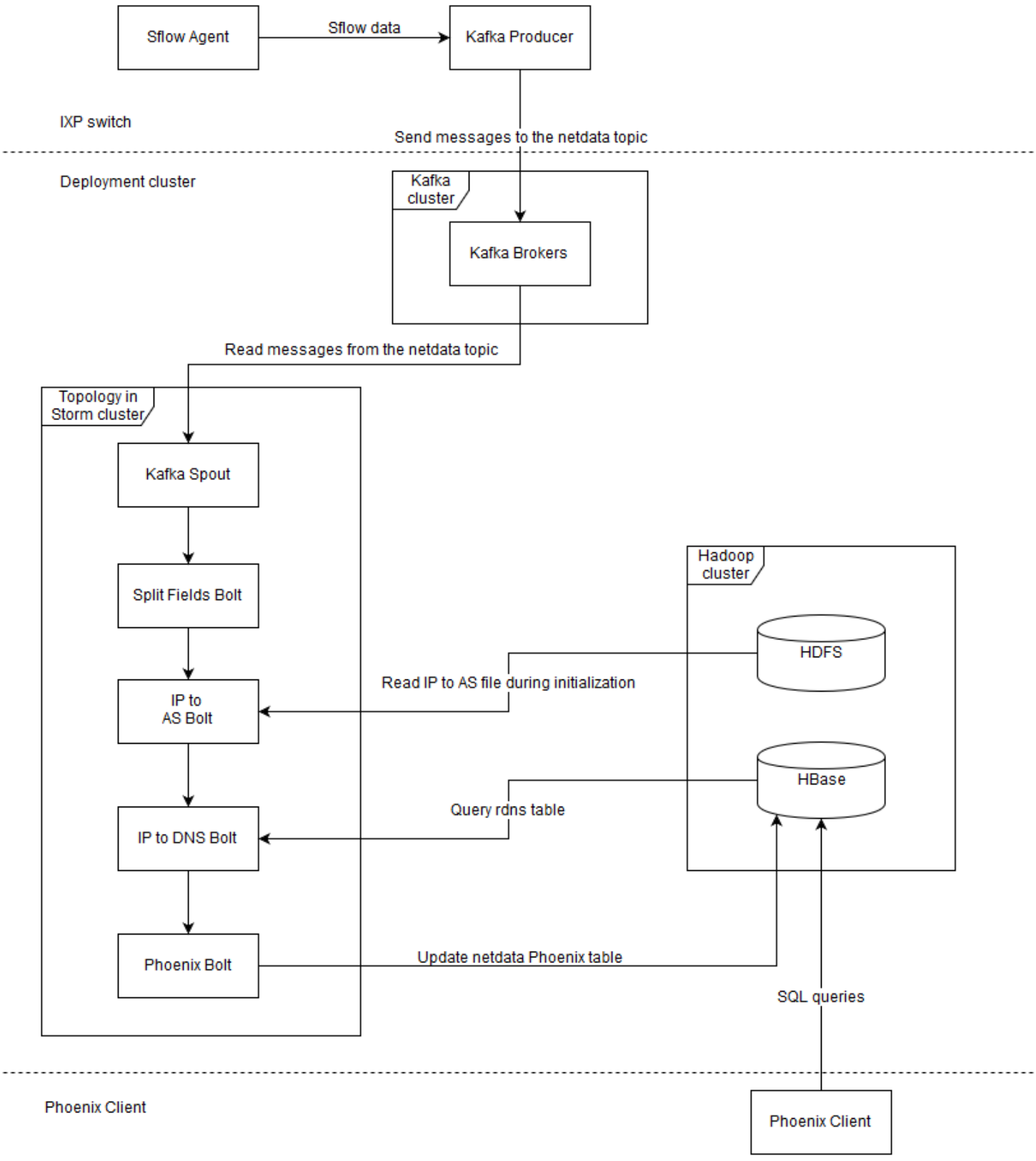
Datix-Realtime architecture



Description of the system

IPX Switch

- **Sflow Agent:** Samples packets from network interfaces.
- **Kafka Producer:** Extracts the useful information from the sflow packets and sends messages to the "netdata" Kafka topic (message format: "sourceIP, destinationIP, protocol, sourcePort, destinationPort, ipSize, dateTime", where dateTime is the Unix Timestamp in microseconds since the packet was processed).

Kafka cluster

- **Kafka Brokers:** Store messages in the topic "netdata" in replicated partitions.

Storm topology

- **Kafka Spout:** Reads messages from the Kafka topic "netdata" and emits them as storm tuples.
- **Split Fields Bolt:** Splits the fields of each message and emits them to the next bolt (fields: sourceIP, sourceIPInt, destinationIP, destinationIPInt, protocol, sourcePort, destinationPort, ipSize, dateTime).
- **IP to As Bolt:** Reads the IP to AS file from HDFS and creates a TreeMap on initialization. Calculates the sourceAS, destinationAS for each message using the TreeMap and appends them to the emitted values.
- **IP to DNS Bolt:** Calculates the sourceDNS, destinationDNS for each message using Get() on the "rdns" hbase table and appends them to the emitted values.
- **Phoenix Bolt:** Inserts the tuple fields calculated by the previous bolts (sourceIP, sourceIPInt, destinationIP, destinationIPInt, protocol, sourcePort, destinationPort, ipSize, dateTime, sourceAS, destinationAS, sourceDNS, destinationDNS) into the "netdata" Phoenix table.

Hadoop cluster

- **HDFS:** Stores the IP to AS file.
- **HBase:** Stores the tables "rdns" (contains IP to DNS name information) and "netdata" (Phoenix table that stores the output of the storm topology).

Phoenix Client

- **Phoenix Client:** Executes SQL queries on the "netdata" Phoenix table.

HBase optimizations

Most optimizations here aim to increase the amount of rows that can fit into the cache. This allows queries on bigger datasets and also improves query latency, since less data has to be transferred to cache.

Enable Snappy compression: Snappy favors compression speed over compression ratio.

Enable HDFS short-circuit local reads: It is possible for the DFSClient to read directly from the disk instead of going through the DataNode when the data is local. RegionServers can read directly off their machine's disks instead of having to open a socket to talk to the DataNode, the former being generally much faster.

Enable compressed BlockCache: When compressed BlockCache is enabled data and encoded data blocks are cached in the BlockCache in their on-disk format, rather than being decompressed and decrypted before caching.

Disable BlockCache on "rdns": Only random reads are performed on the table "rdns", which does not fit into the cache, therefore caching does not provide any benefit. Disabling the BlockCache on "rdns" allows the "netdata" table to fully take advantage of the cache.

Disable Data Block Encoding on all the tables: Since we have enabled compressed BlockCache, Data Block Encoding does not reduce much further the size of data when stored into the cache. In this case disabling Data Block Encoding improves Get performance.

Enable parallel-seeking: Enables StoreFileScanner parallel-seeking in StoreScanner, which reduces response latency.

Phoenix table optimizations

```
CREATE TABLE "netdata" (
    "time" BIGINT PRIMARY KEY,
    "d"."ipS" VARCHAR,
    "d"."ipSI" BIGINT,
    "d"."ipD" VARCHAR,
    "d"."ipDI" BIGINT,
    "d"."proto" SMALLINT,
    "d"."portS" INTEGER,
    "d"."portD" INTEGER,
    "d"."size" INTEGER,
    "as"."asS" VARCHAR,
    "as"."asD" VARCHAR,
    "dns"."dnsS" VARCHAR,
    "dns"."dnsD" VARCHAR
)
SALT_BUCKETS=4,
DEFAULT_COLUMN_FAMILY='d',
DATA_BLOCK_ENCODING='NONE',
COMPRESSION='SNAPPY';
```

Separate column families for AS/DNS: One common query type on the table "netdata" is selecting the top-N AS or DNS pairs over a period of time. Since only the column families needed for the query are cached, having separate column families containing AS, DNS and other information helps by reducing the data that have to be cached, thus reducing query latency.

Small column family and column qualifier names: Having large column names causes a performance penalty each time the data is accessed and increases the data transferred from RegionServer to Phoenix Client.

Phoenix data types: Usage of the appropriate data type for each column (BIGINT for dataTime, SMALLINT for protocol etc) reduces the size of each row, which leads to faster queries.

Salting: Since most queries on the table will be constrained over a period of time, the rowkey should be "time" in order to benefit from table scans. However a monotonically increasing rowkey can be a problem because subsequent writes are performed on the same RegionServer. Region hot spotting can be mitigated by salting, which adds a prefix to the rowkey that spreads writes evenly. In addition to write throughput, read throughput can also increase, because Phoenix scans the salted sorted data in parallel and merge-sorts them in the client.