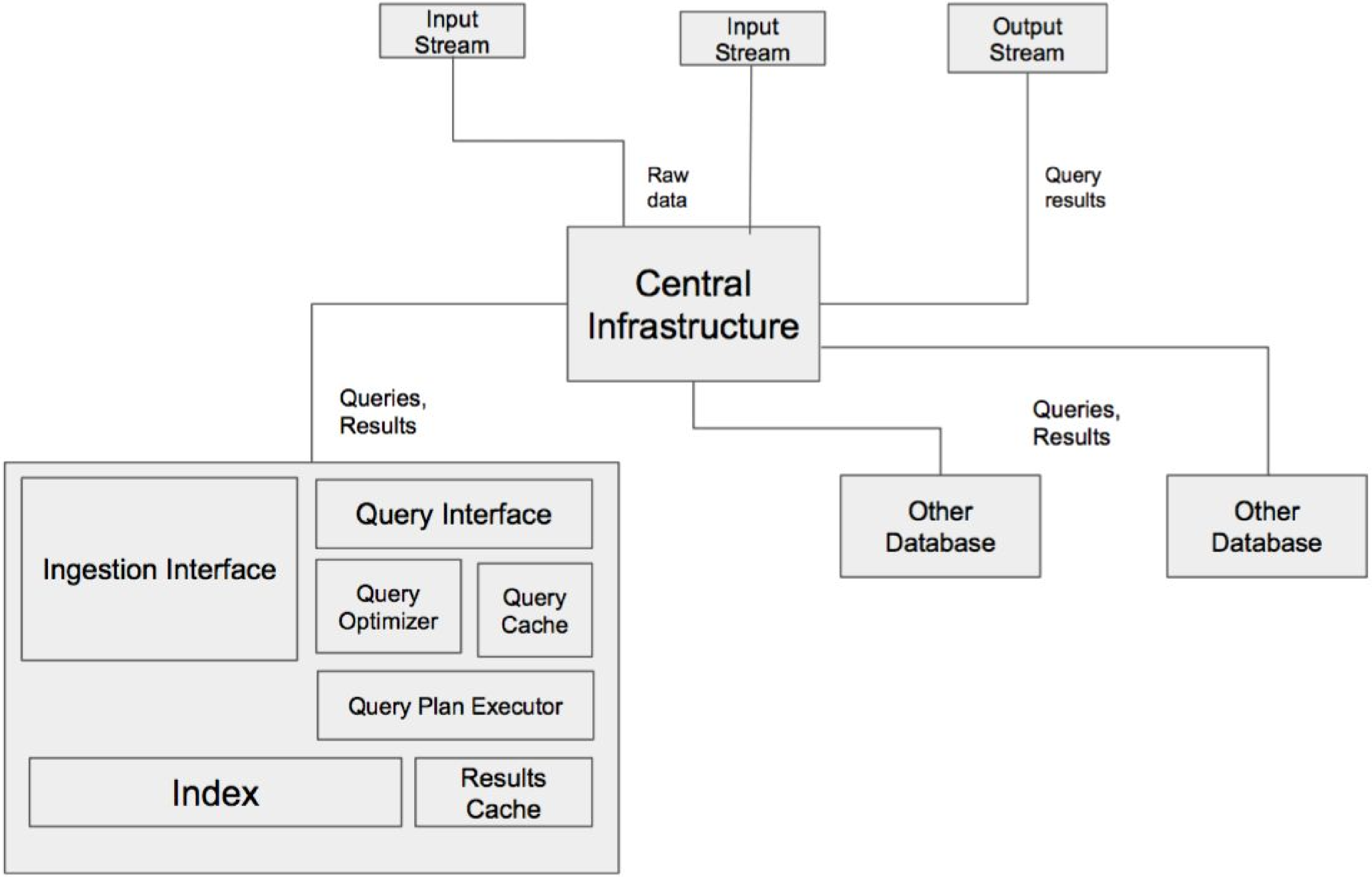
Lightweight Runtimes (Galileo IoT)

Team Sparkle

Dhinesh
Shiva
Keno
Guru


Input Stream
Input Stream
Output Stream
Raw data
Query results
Central Infrastructure
Queries, Results
Queries, Results
Other Database
Other Database
Ingestion Interface
Query Interface
Query Optimizer
Query Cache
Query Plan Executor
Index
Results Cache

Central Infrastructure

- Web Interface
  - Queries in form of `HTTP GET/POST`
  - Most queries decided apriori
  - Insert:
    - `http://128.205.39.183/insert?timestamp=1446175861&temp=17.4&room=2`
  - Query Patterns:
    - `http://128.205.39.183/query?room=1&start=1446175932&end=1446176532`
    - `http://128.205.39.183/query?room=2&windowsize=900`
    - `http://128.205.39.183/query?sql=SELECT+%2Aroom%2C+temperature+FROM+tempdb+WHERE+timestamp+%3E+1446175932%3B`

Data Stream

- Data generated by ITG3200 (Gyroscope)
- Temperature sensor for calibration
- timestamp(`uint64_t`) temperature(`double`) room(`uint8_t`)
- 4 samples/second
  - Each sample assigned to a ‘room’
- ~68bytes/second -> ~240KB/hour
  - Easily stored in main memory

Data Stream

- Room readings almost equal
- Looking into alternate temperature sources
  - `lm-sensors`
  - Multiple sensors

Ingestion Interface

- Sync vs Async
- Everything is in memory
- ArrayDeque of fixed size.
- Frequency of inserts calculated
- Formula for windowSize

Size of Records = Insert freq * Time Window

Data Structure Used

- ArrayList<LeafValue> -----> Tuple;
- ArrayDeque<Tuple> -----> WindowedTable;
- ArrayList<WindowedTable> -----> TableList;

```
insert_tuple(tuple1):

    queue.add(tuple1)

    if queue.size() > threshold : queue.remove()
```

Server Client

SparkeDB is the server listening for any query requests

The Central Infrastructure is the client.

JSON is used for data communication.

JSON

```
{
  “type” : “request” || “response”,
  “queryType” : “insert” || “query",
  “query” : “select ...” || “insert ...",
  “status” : “success” || “failure”,
  “timeTaken” : 10
}
```

Conclusion

Given two streams,

- Run query requiring a join over both streams
- Evaluate percentage of expected results we produce over given window
- Repeat with increasing stream frequencies
- Repeat with increasing window sizes