RISING WAVE: THE SQL-NATIVE STREAMING DATABASE

MILESTONE 2
GROUP 5

MEMBERS:
ANICA REMIN FERNANDEZ
ASWIN SIVAKUMAR
ATHULYA JAYAN
BETSY VARGHESE
NAIJEL ROY
FRIDOLIN FRANKLIN
SAJIN SAHADEVAN
ADVAITH HARISH

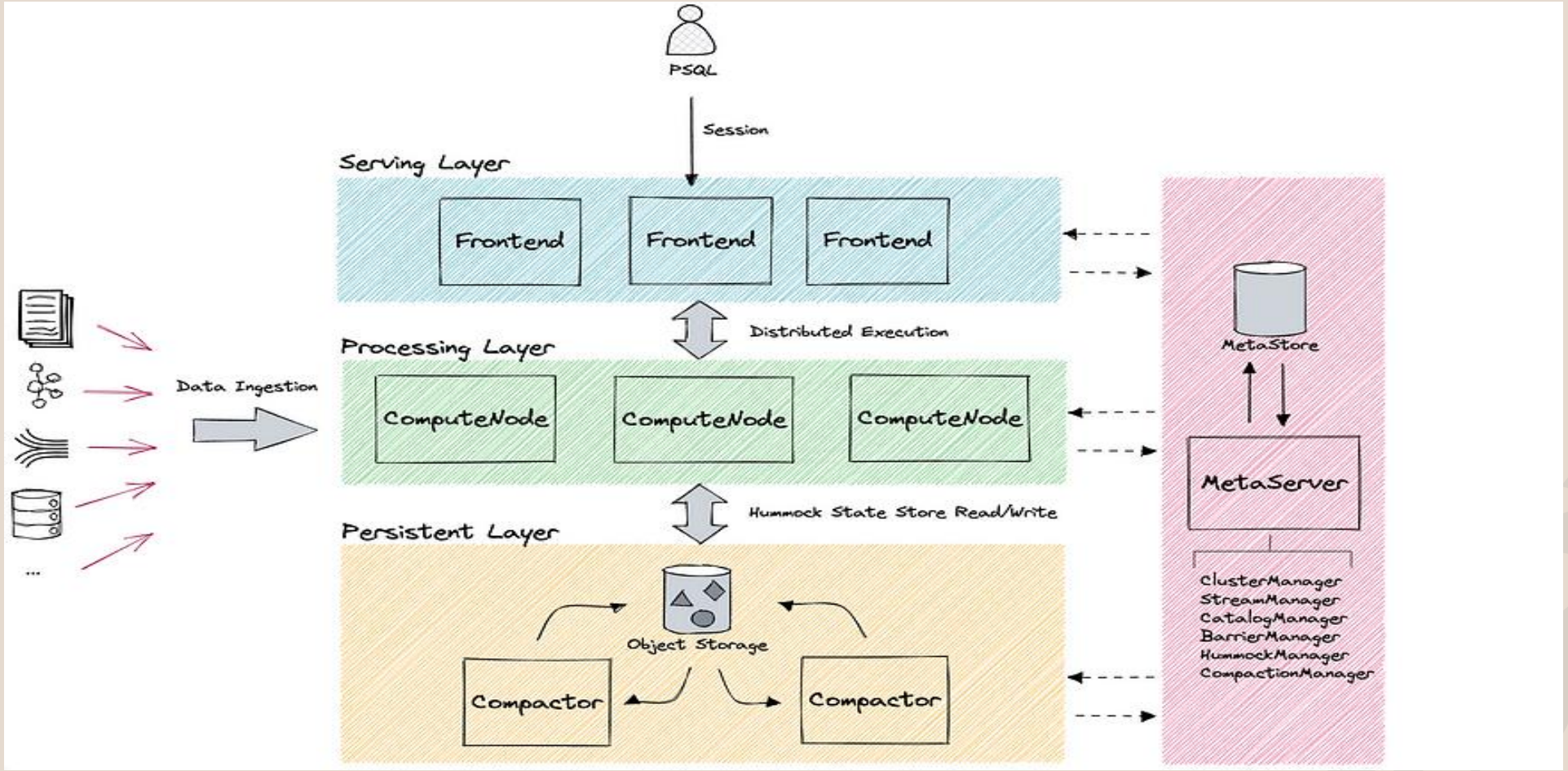
agenda

- INTRODUCTION
- CHALLENGES WITH TRADITIONAL STREAMING STACKS
- INTRODUCING RISINGWAVE
- CORE INNOVATIONS
- ADVANCED FEATURES
- REAL-WORLD USE CASES
- LIMINATIONS & CONSIDERATIONS
- CONCLUSION

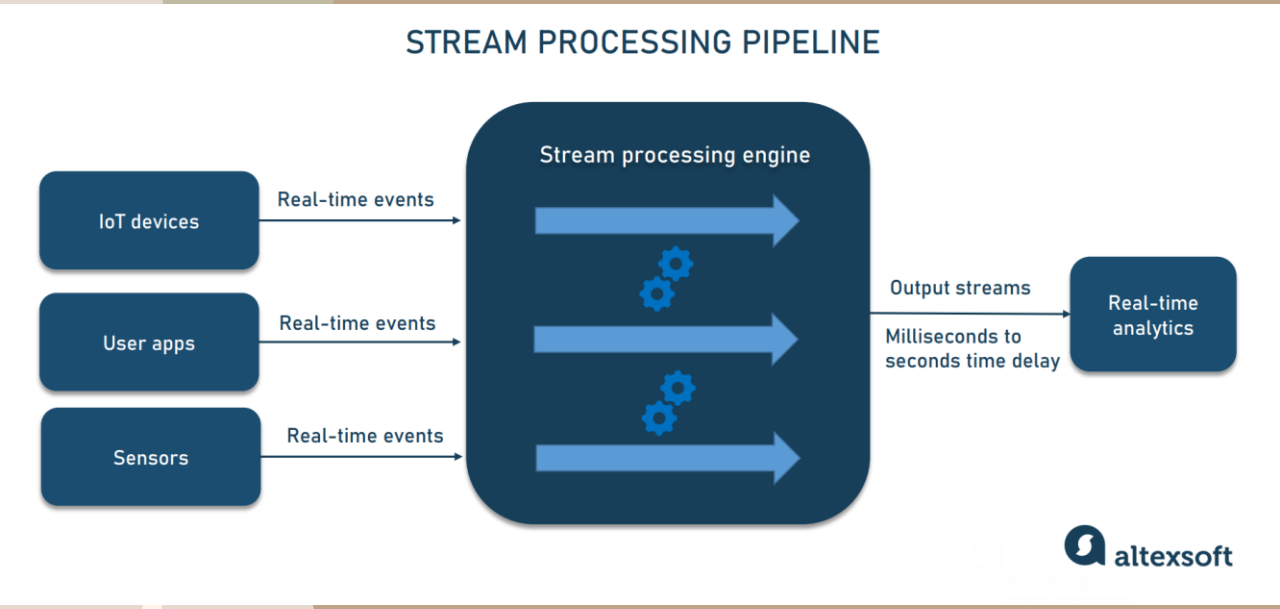
What is rising wave ?

- RISING WAVE IS A SQL NATIVE STREAMING DATABASE BUILT FOR MODERN REAL-TIME ANALYTICS.
- PROVIDES A UNIFIED PLATFORM COMBINING INGESTION, PROCESSING, STORAGE, AND SERVING.
- USES STANDARD POSTGRESQL WIRE PROTOCOL FOR EASY INTEGRATION.WRITTEN IN RUST FOR PERFORMANCE AND SAFETY.
- ENABLES ELASTIC SCALING BY SEPARATING COMPUTE AND STORAGE LAYERS.
- AIMS TO REDUCE COMPLEXITY AND OPERATIONAL COSTS COMPARED TO TRADITIONAL STACKS.

RISING WAVE ARCHITECURE



What is streaming data?



- It's continuous, real-time data generated from multiple sources. Like IoT devices, social media feeds, financial trancations,nand many more.
- Its time-sensitive, requires immediate ingestion and processing.
- Along with batch data its process stored data at intervals.
- Use cases: fraud detection, real-time dashboards, monitoring and alerts.
- Data rates ca be thousands to millions per second.

STREAM PROCESSING VS BATCH PROCESSING

Feature	Batch	Stream
Data Input	Finite datasets	Infinite, real-time flows
Latency	High (minutes–hours)	Low (ms–seconds)
Storage	Data stored, then processed	Processed immediately, minimal storage
Use Cases	Reports, historical analysis	Monitoring, real-time insights
Complexity	Moderate	High (ordering, fault tolerance)
Failure Handling	Re-run entire job	Checkpoints, recovery
Programming Model	SQL, scripts	Event-driven APIs
Examples	ETL, reports	Sensors, fraud detection

TRADITIONAL STREAMING DATA STACK

- Message Broker :Kafka, Pulsar for data ingestion and buffering.
- Stream Processor: Flink, Spark streaming for computation .
- State Store: RocksDB, Cassandra for maintaining state.
- Challenges include :
- Complex maintenance across multiple tools.
- Synchronization and state consistency issues.
- High operational overhead and specialized skills requirements.
- Latency introduced by network hops between components.
- Difficult to implement and scale efficiently.

Storage vs compute decoupling

Aspect	Traditional Systems	RisingWave
Storage/Compute	Tightly coupled (local storage)	Fully decoupled (object store)
Scalability	Joint scaling needed	Independent scaling
Cost Efficiency	Overprovisioning common	Pay-as-you-grow model
Fault Tolerance	Local state, complex recovery	Persistent snapshots, easy recovery
Resource Use	Less flexible	Dynamic and efficient

Fault Tolerence & Exactly-One

- Asynchronous checkpointing = minimal query pauses
- Watermarks + snapshots ensure state consistency
- Exactly-once processing
- Resilient to failures with auto recovery
- Enables continuous ops even during scaling or maintenance

SQL as primary interface

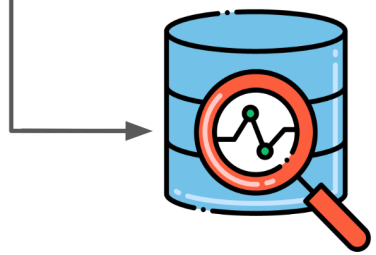
•SQL-first design simplifies stream processing
•Supports:
•**Materialized Views**
•**Window Functions**
•**Temporal Joins**
•**Event-time semantics**
•Works with PostgreSQL clients, BI tools
•Reduces need for Java/Scala code
•Extensible via **User-Defined Functions (UDFs)**

Streaming SQL Highlights

- **Continuous Queries**: Instant updates as data flows
- **Windowing**: Tumbling, sliding, session windows
- **Joins**: Stream–stream & stream–table joins
- **Aggregations**: Real-time, low-latency updates
- **Temporal Tables**: Time-travel queries
- Handles **late-arriving data** with watermarking

Traditional SQL

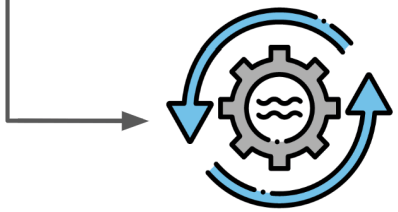
```
SELECT * FROM YOUR_TABLE WHERE...
```



SQL applied to ***static data***

Streaming SQL

```
SELECT * FROM YOUR_TABLE WHERE...
```



SQL applied to ***real-time data stream***

REAL-TIME USE CASES WITH RISINGWAVE:

- FRAUD DETECTION
  - Monitor transactions in real time at scale (millions/sec).
  - Detect anomalies using SQL-based pattern matching.
  - Instantly flag/block suspicious behavior.
  - Enhances compliance and reduces financial loss.
- REAL-TIME AD MONETIZATION
  - Count impressions, clicks, and conversions instantly.
  - Power dynamic bidding and budget control.
  - Feed live dashboards for advertisers/publishers.
  - Maximizes revenue with sub-second insights.

- MARKETING ANALYTICS
  - Real-time segmentation with session windows.
  - Evaluate A/B tests instantly.
  - Live ROI and customer behavior tracking.
  - Integrate web, mobile, and social data seamlessly.
- Production Monitoring & IoT
  - Stream sensor data from industrial equipment.
  - Enable predictive maintenance using anomaly detection.
  - Real-time alerts to prevent downtime/defects.
  - Use event pattern detection with plain SQL.

LIMITATIONS & CONSIDERATIONS

- Requires upfront compute & storage setup
- Trade-offs: **latency** ↔ **throughput** ↔ **cost**
- Some **streaming complexities remain** (e.g., state mgmt)
- Best for **low-latency use cases**
- Ecosystem still maturing compared to Flink/Spark

WHEN TO USE RISINGWAVE:

- You need real-time insights, not hourly/daily reports.
- You want to use SQL, not complex Java/Scala streaming code.
- You want a single system, not a complex stack.
- You need elastic scaling and cloud-native deployment.
- You want materialized views that update in real time.
- You want exactly-once processing guarantees.

thank you