Solving DEBS Grand Challenge with WSO2 CEP

Srinath Perera, Suhothayan Sriskandarajah, Mohanadarshan Vivekanandalingam,
Paul Fremantle, Sanjiva Weerawarana
WSO2 Inc.

Outline

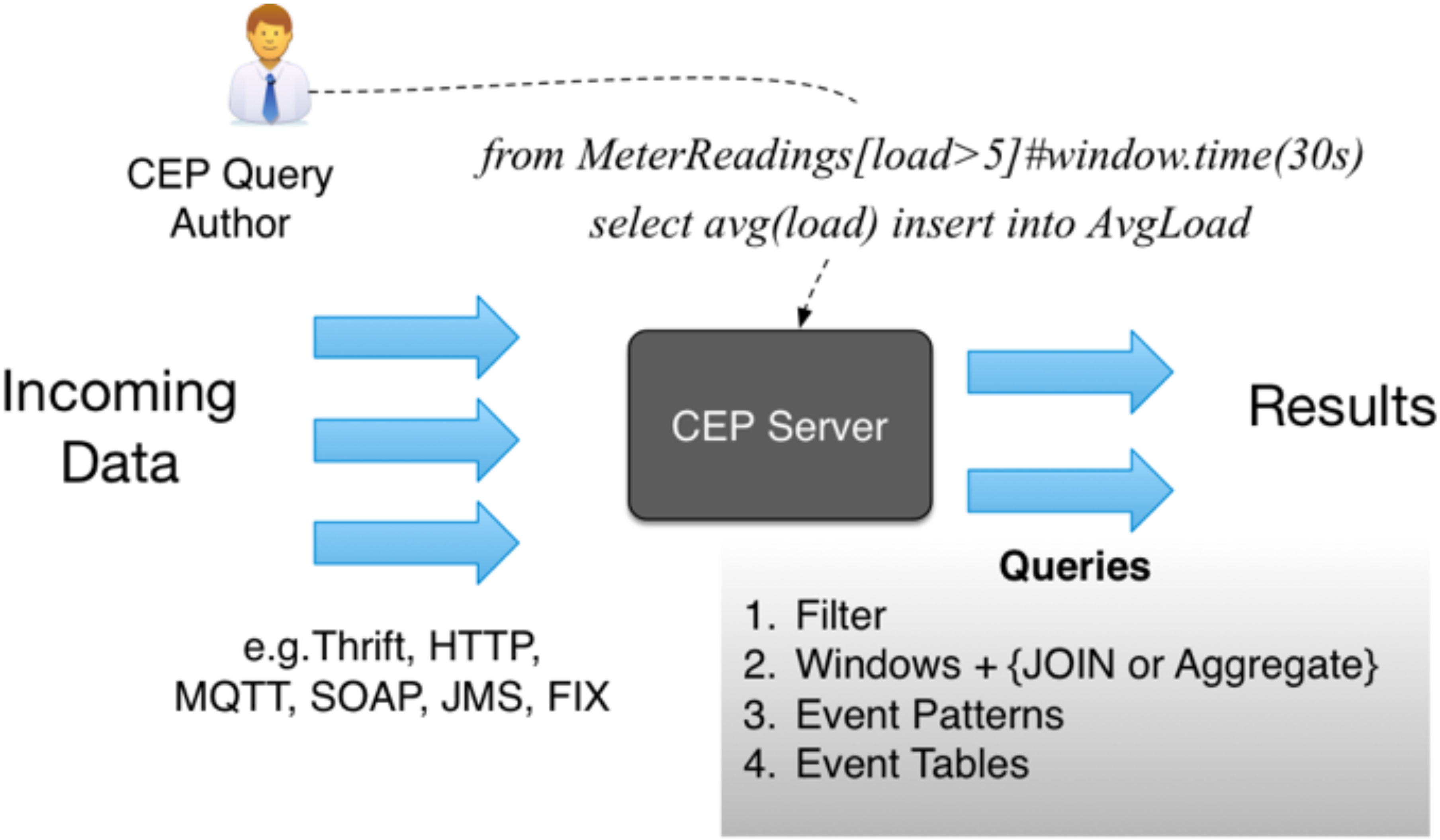
- Grand Challenge
- CEP
- Basic Solution
- Optimizations
- Beyond Grand Challenge

Photo by John Trainoron Flickr http://www.flickr.com/photos/trainor/2902023575/, Licensed under CC

Grand Challenge Problem

- Smart Home electricity data: power and load
- 90 days, 40 houses, 120GB, 2125 sensors
- Event frequency per plug
  - load values: once 2s.
  - Work values: once 10-50s. Value sent through when the difference is more than 1Wh
- Queries
  - Load prediction
  - Outlier detection

Complex Event Processing



from MeterReadings[load>5]#window.time(30s)
select avg(load) insert into AvgLoad

e.g.Thrift, HTTP,
MQTT, SOAP, JMS, FIX

Queries

1. Filter
2. Windows + {JOIN or Aggregate}
3. Event Patterns
4. Event Tables

WSO2 CEP Operators

- Filters or transformations (process a single event)

  -
    ```
    From MeterReadings[load>10] select .. insert into ..
    ```
- Windows + aggregation (track window of events: time, length)

  -
    ```
    from MeterReadings#window.time(30s) select avg(load) ..
    ```
- Joins (join two event streams to one)

  -
    ```
    From MeterReadings#window.time(30s) as b join Cricket as c on ..
    ```
- Patterns (state machine implementation)

  -
    ```
    From e1 = MeterReadings-> e2 = MeterReadings[load > e1.load + 10]
    within 1h
      select ..
    ```
- Event tables (join events against stored data)

  -
    ```
    Define table ReadingSummary(v double) using .. db info ..
    ```
- WSO2 CEP Performance: (core i5 processor)
  - 2-9 million events/sec on same JVM
  - 300k events/sec over the network

How we did this?

- Basic version using CEP EQL Queries
- Does it use machines fully? (load average > 2X number of cores)
- Find and fix bottlenecks (Write extensions if needed)
- Repeat until the deadline ☺️

“About 97% of the time: premature optimization is the root of all evil"

From http://seedtofeedme.blogspot.in/2013/09/watering-plants.html

Correcting Load Values

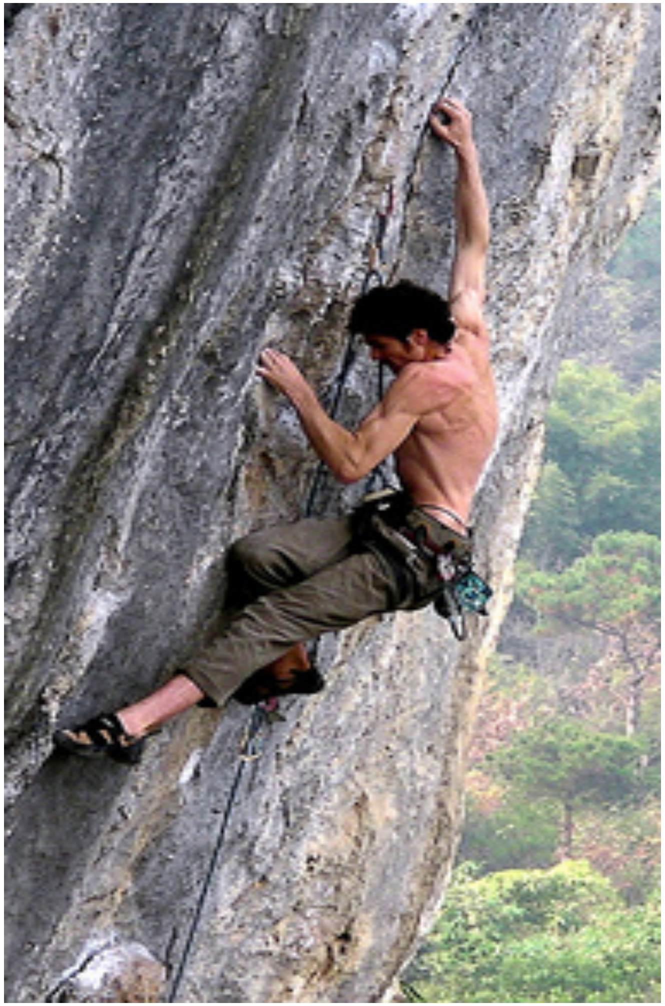
- We receive load values roughly once every 2 seconds from each plug. Sensors send work values only when the difference is more than 1Wh: this happens roughly every 10 to 50 seconds.

$$load(t) = \frac{1000}{3600} * min(\frac{1}{t+1-t_2}, w_2 + (t - t_2) * \frac{w_2 - w_1}{t_2 - t_1})$$

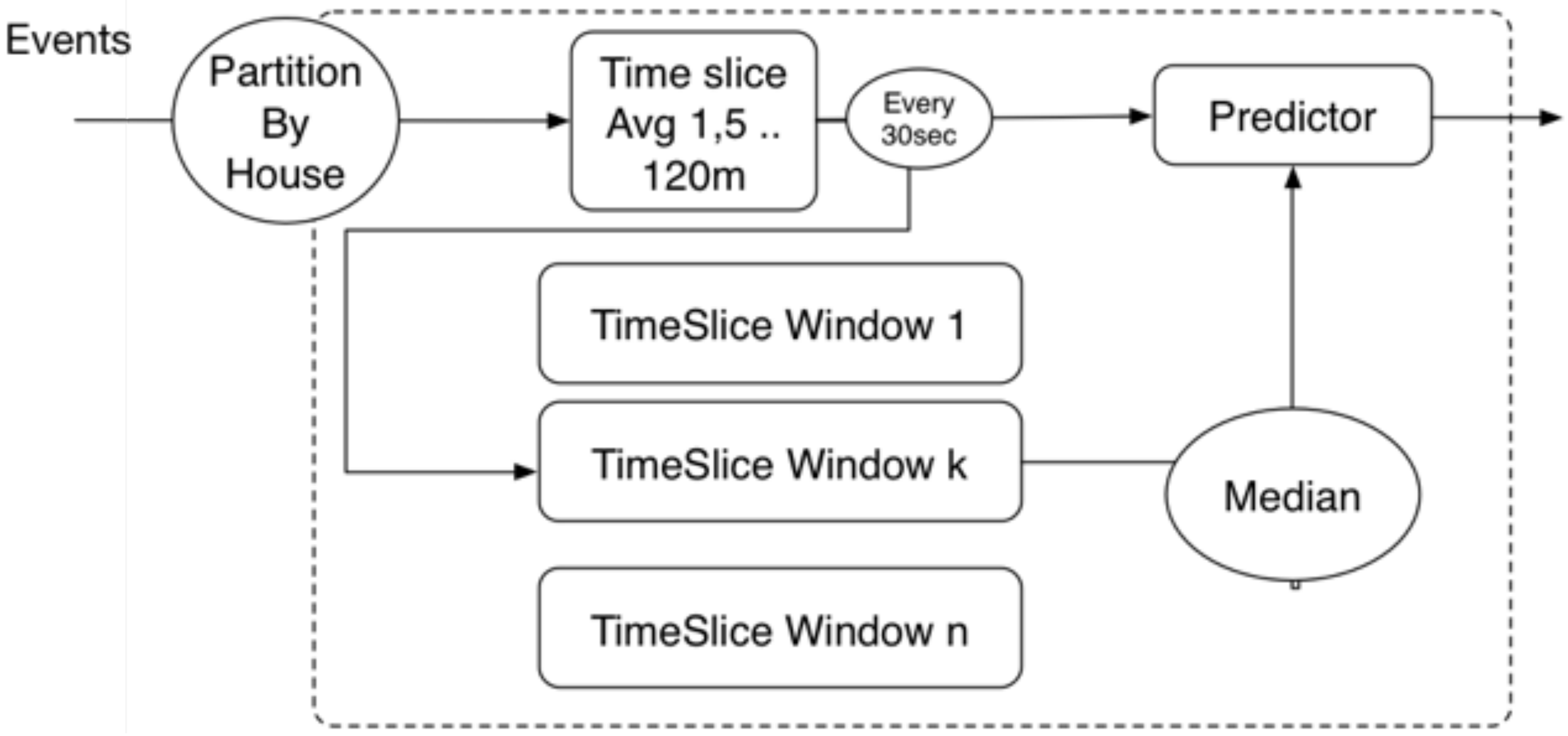
- Correcting function
  - Simple linear fitting using last two work values and the calculate load using that value
  - Correct using the fact that work from t2 and t < 1kWh as we have not seen the work value.
- Micro-benchmark with 100 million events.
  - Errors < 16% of the load value 75% of the time.
  - Error Percentiles 25%=1.9 50%=6.39 75%=15.41

Q1 Challenges

- CEP engine is single threaded, hence cannot use a multi-core machine fully.
- Avoid calculating each timeslot (1,5 .. 120m) separately
- Batch windows. Median calculations only keep < 129,600 events
  - No memory problem
  - Median calculated once for 30 sec, Min-Max Heap is good enough

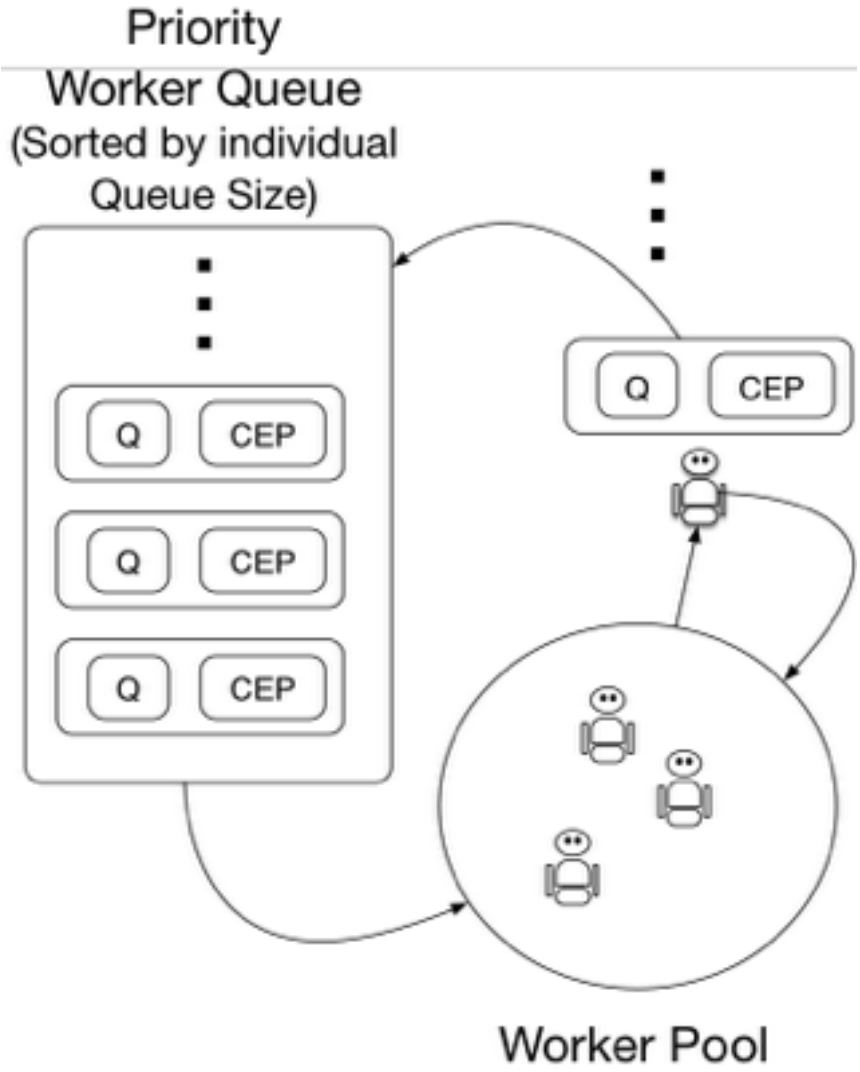


Query 1: Single Node Solution



Optimizations: Improve Parallelism

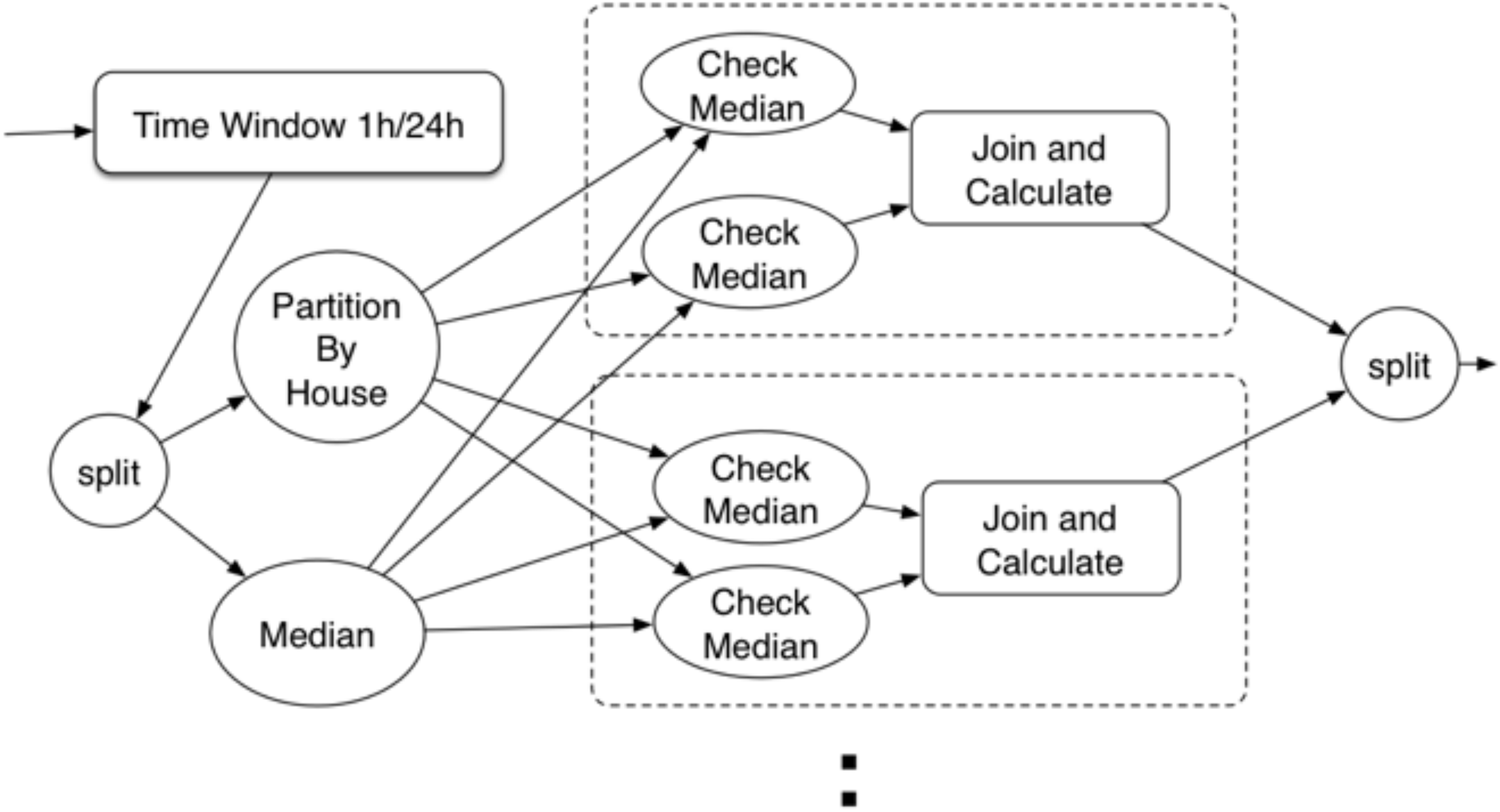
- Do we have enough parallelism? Does load average > 2X number of cores
- Scale, parallelism => data partitions
- Partition data and assign to different CEP engines
  - Partition by House (let us go up to 40 threads)
- Not all houses would have the same workload
  - We use CEP engines and threads independently and use work stealing



Q2 Challenges

- Sliding window has to remember each event in the window to expire them.
  - (1h=3M events and 24h=74M events)
- Calculating mean over window. Q2 is limited by the Global median
- Per plug windows and the global windows will keep two copies of events

Query 2: Single Node Solution



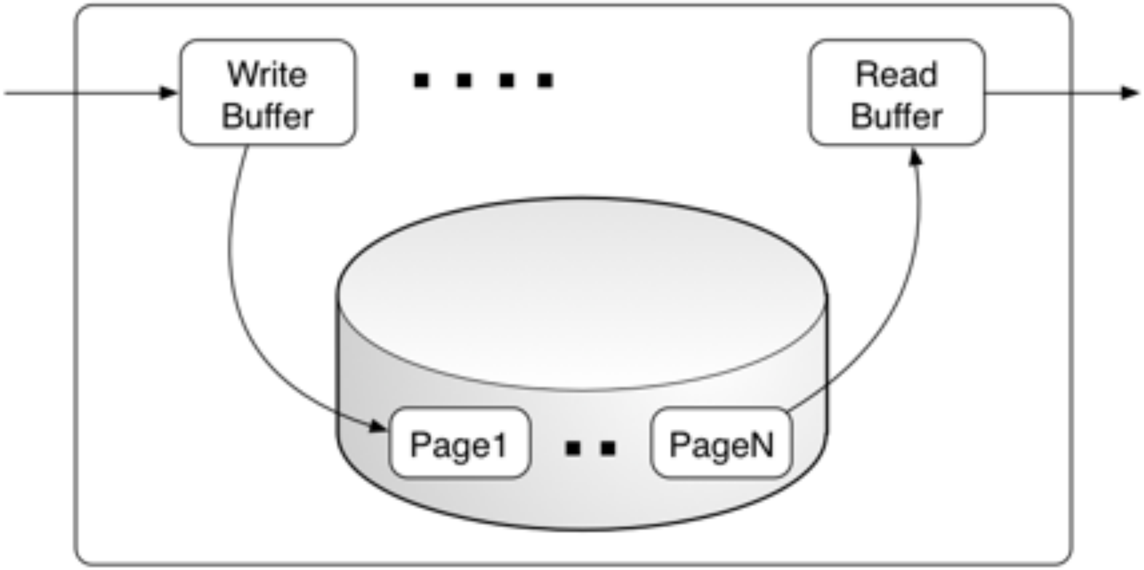
Calculating Median

- MinMax Heap
- Buckets – track how many fall on each bucket
- Buckets with variable resolution
  - Does not work when median shift
- Reservoir Sampling
- Re-Median

Micro-benchmark: (1 million values)

- Min-Max Heap 1000ms
- Reservoir Sampling 4ms
- Bucketing <1ms.

Disk backed Window

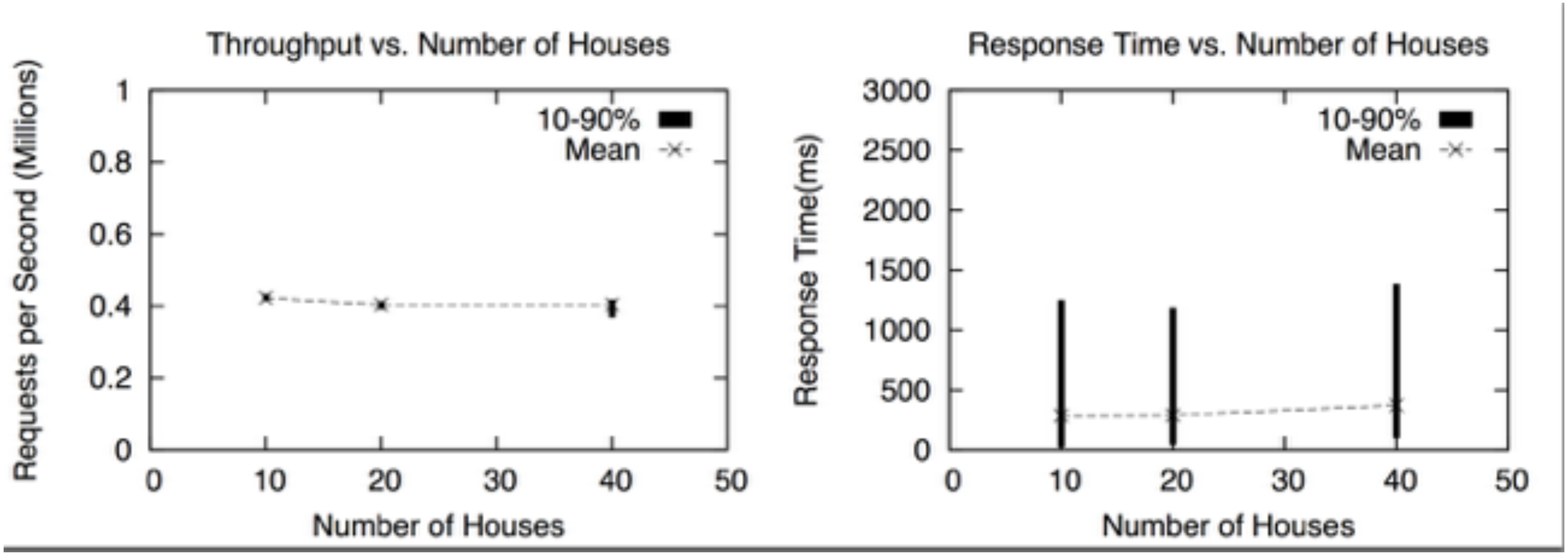


- Sliding window reads from one end and writes to the other
  - Write in batches
  - Pre-fetch data in batches
  - Micro-benchmark shows almost no latency addition

More Optimizations

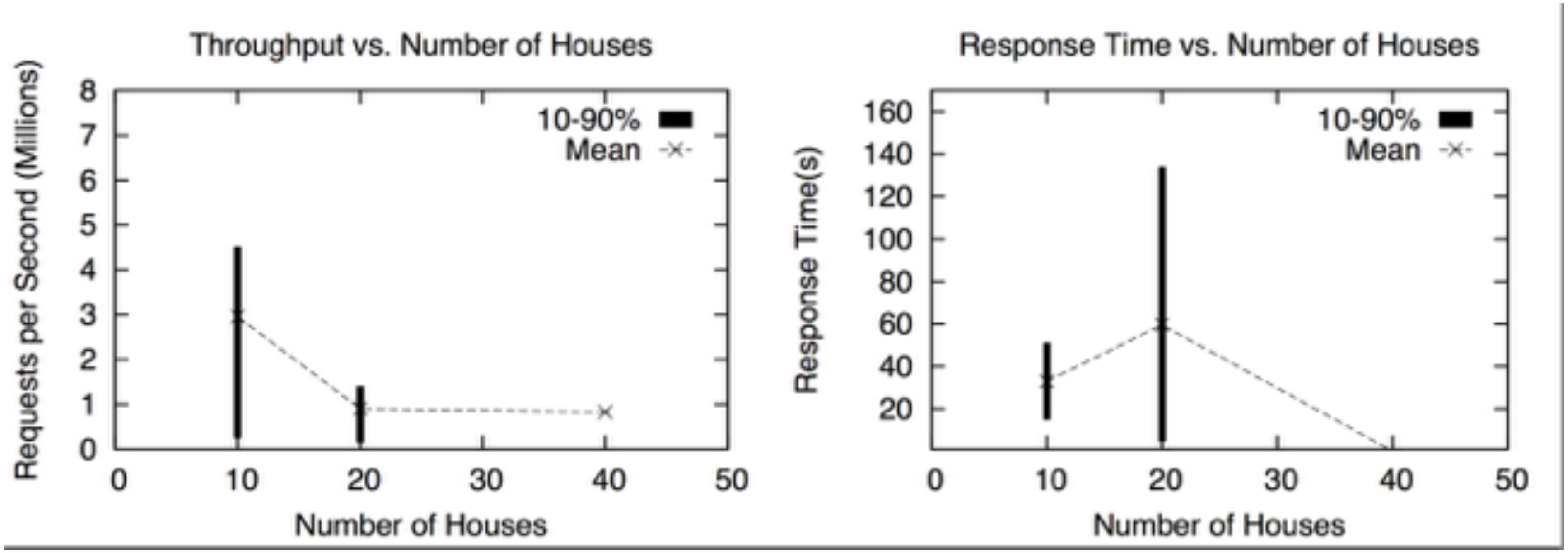
- 1 Sec Slide
  - Windows slide by 1second
  - Rather than tracking all events in a window, we can aggregate events in each second
  - Then adjust the Buckets accordingly when they are added or removed.
- Q2 only asks for how many plugs are greater or less than the global median
  - If you know the global median, then you can check plug median < or > the global median without calculating plug median.

Q1 Single Node Results



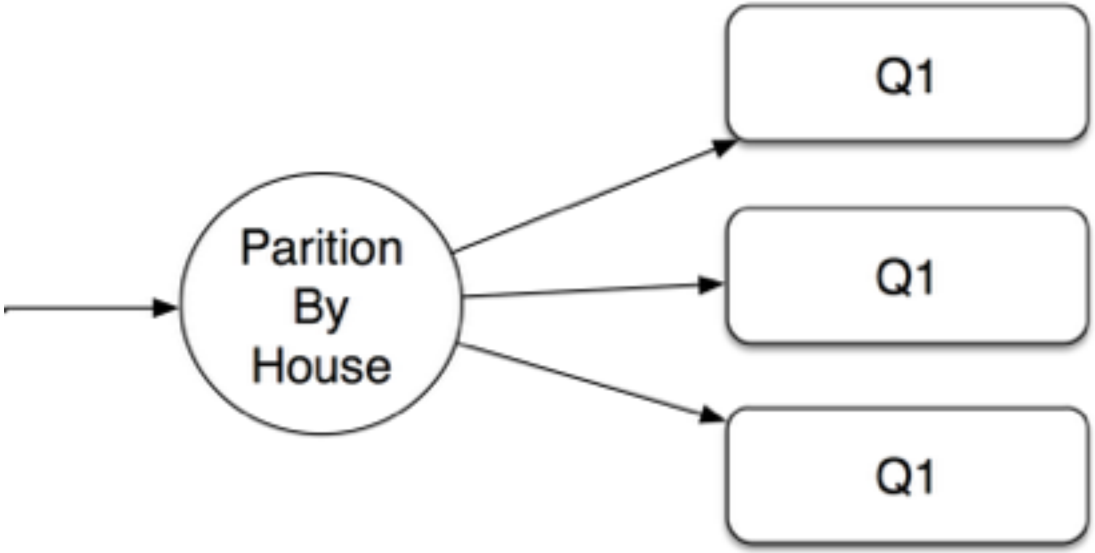
- 16 core, 24GB machine
- Fast (0.4M events/sec) with with 1s latency
- Stable with different number of houses

Q2 Single Node Results



- 16 core, 24GB machine
- Fast (3M to 1M events/sec) with < 100ms latency
- Sensitive to houses (number of events), because that increases the number of events in windows

Query 1: Distributed Solution



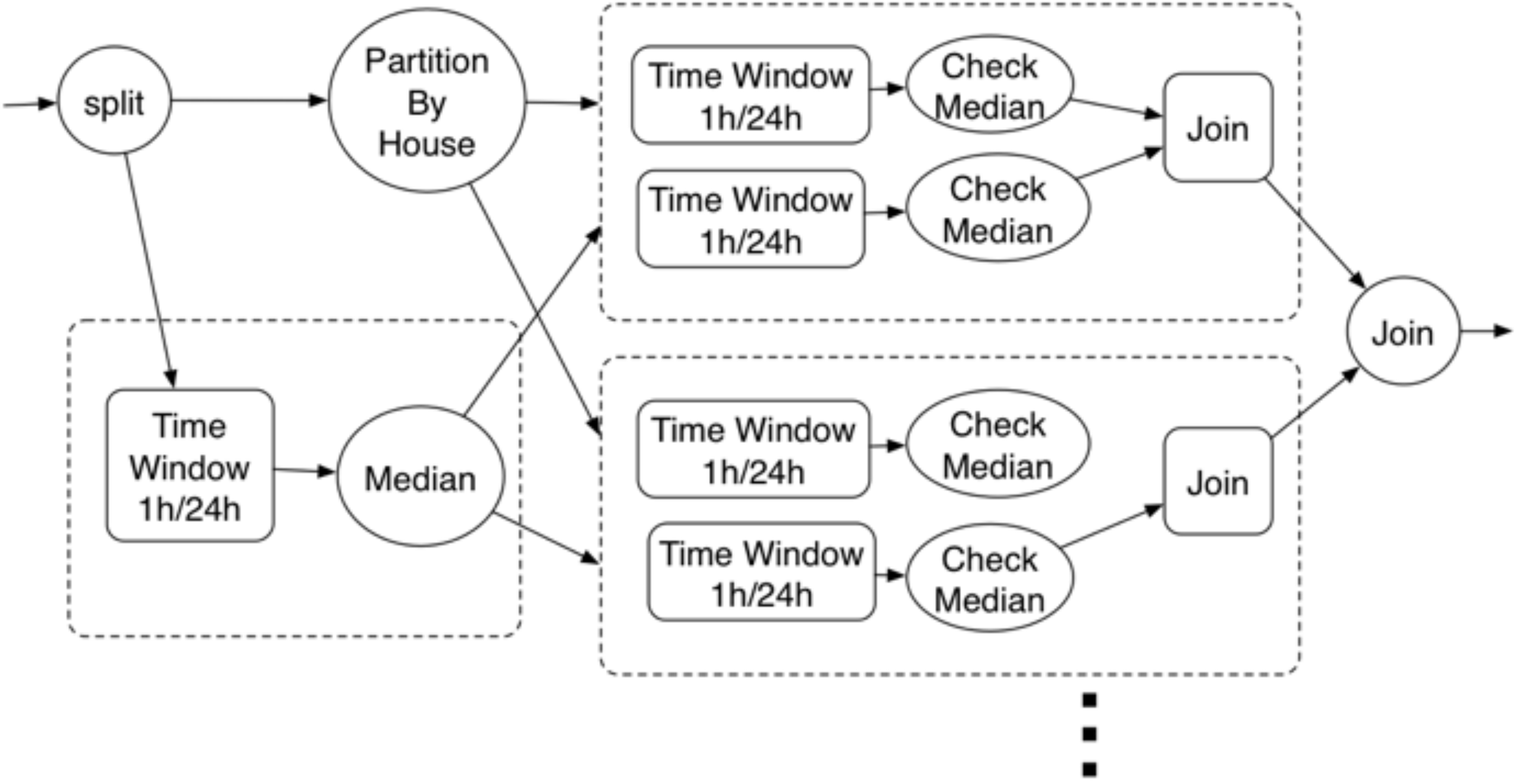
- Embarrassingly parallel by house
- Partition by house

Distributed Communication

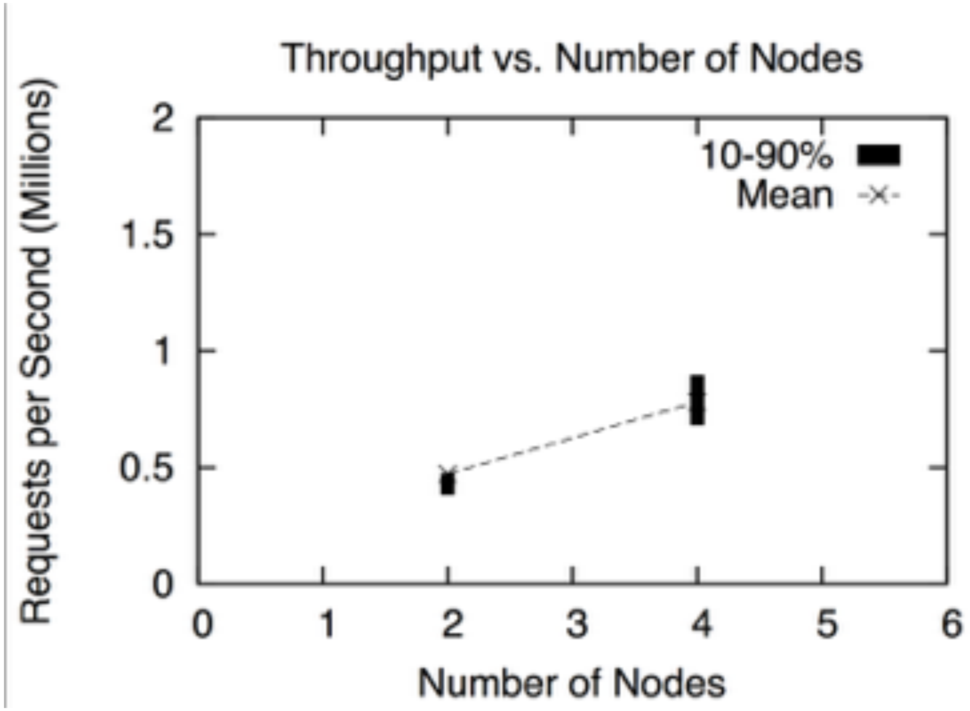
- WSO2 CEP uses a thrift based binary protocol (can handle variable size messages) that can do 0.3M events/sec
- Too slow to scale up (only got 0.18M/sec with thrift)

- We did a custom fixed message size solution (using java ByteBuffer) that can do up to 0.8M events/sec
- Then we got close 1M events/sec for Q1 distributed
- Also tried ZeroMQ, which was bit slower

Query 2: Distributed Solution



Q1 Distributed Results

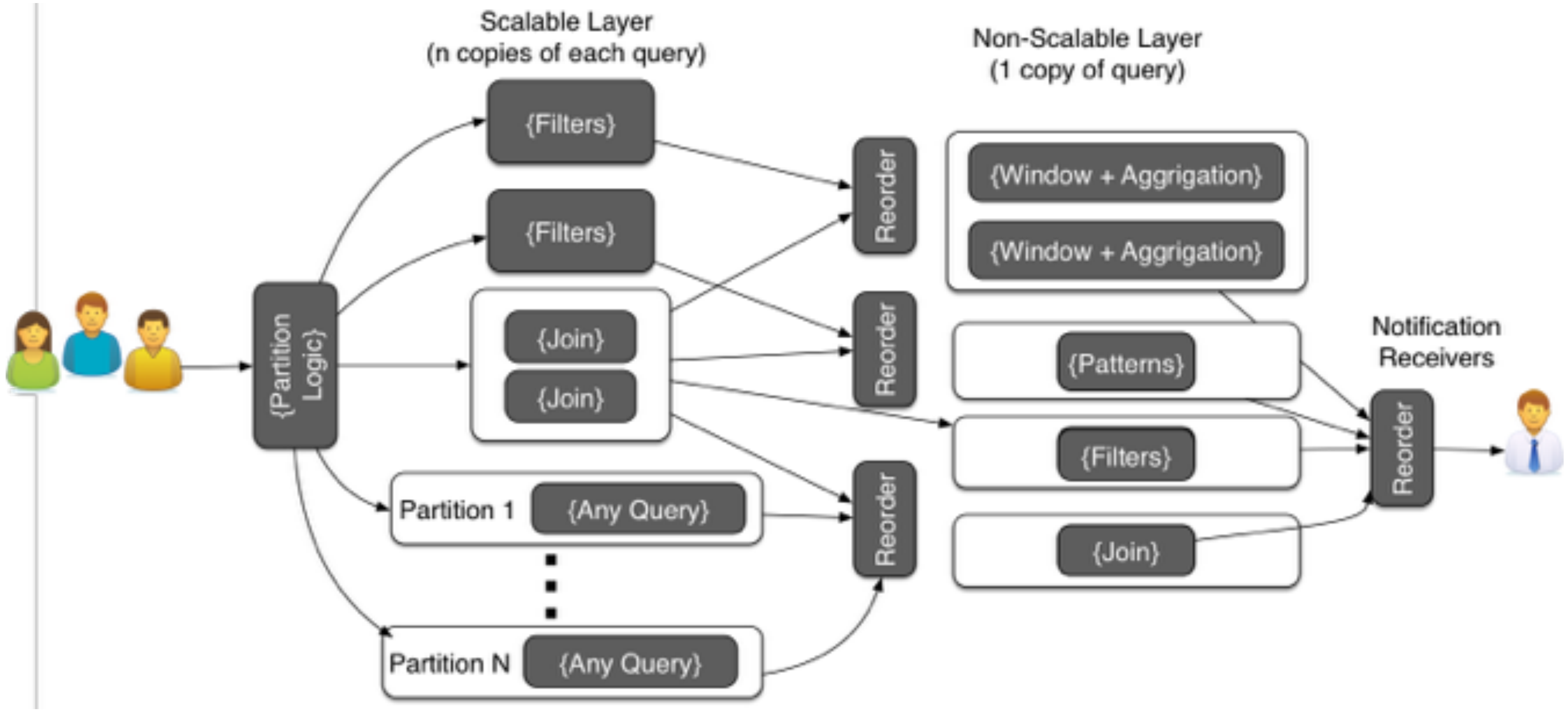


- Using Amazon c1.xlarge, 1 client to 2 CEP nodes ratio
- Got close to 1M msg/sec 2X speedup

Beyond Grand Challenge

- Support for automatic parallelization on partitions with WSO2 CEP
- Adding some custom functions as inbuilt operators
- Describing distributed deployment via queries and automating the deployment
- All new features (e.g. disk backed window, median, scaling) released with WSO2 CEP coming 2014 Q4.

Scaling CEP



- Think like MapReduce! ask user to define partitions: parallel and non parallel parts of computations.
- Each node as Storm bolt, communication and HA via storm

WSO2 CEP

- WSO2 is an opensource middleware company
  - 350+ people
  - Customers: Ebay, Boeing, Cisco …
  - Funded by Intel Capital, Quest, Cisco
- WSO2 CEP is available opensource under apache license from https://github.com/wso2/siddhi
- We welcome contributions (both code and ideas)!! and collaborations
- If you want to build your research on top of WSO2 CEP, let us know. (architecture@wso2.org)

Conclusions

- Main Ideas
  - Load correction via linear prediction
  - Partitioning data via house for parallelism and distribution
  - Did an study for median calculation options
  - Disk backed window with paging
  - Fixed message sized protocol
  - About 400k single node throughput and close to 1M distributed throughput
- Try out WSO2 CEP/ Siddhi: Open source under Apache License.

Questions?