Scalable Stream Processing and Map-Reduce

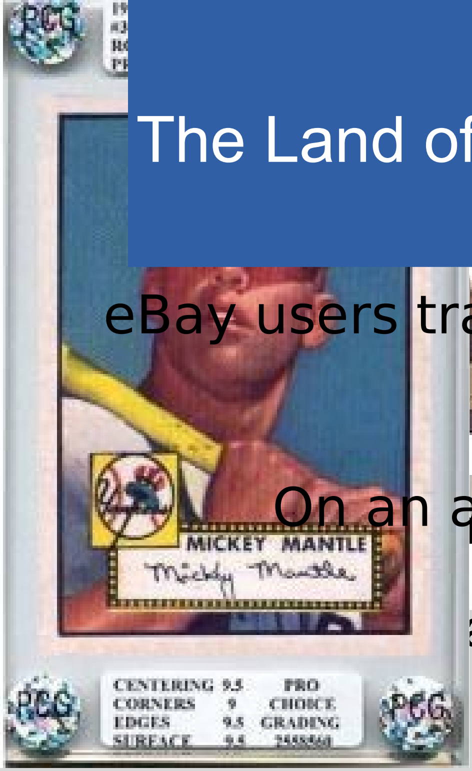
Neel Sundaresan, Evan Chiu, Gyanit Singh

eBay Research Labs

About eBay Research Labs

- Who we are
  - eBay Research Labs was formed in July of 2005. The group's charter is to conduct forward looking research and deliver innovative solutions to business and product
- What we do
  - Search & IR
  - Machine learning
  - Analytics and Optimization
  - Reputation, Trust and Safety
  - Distributed and Grid Computing
  - Social and Incentive Networks
  - Large Scale Visualization
  - Scalable Martix and Graph Computing
  - …
- Basically we “Dig” into data!

The Land of Large Numbers



eBay users trade about $1,400 worth of goods on the site every second.

On an average day on eBay…

nb

an

shoe

sells

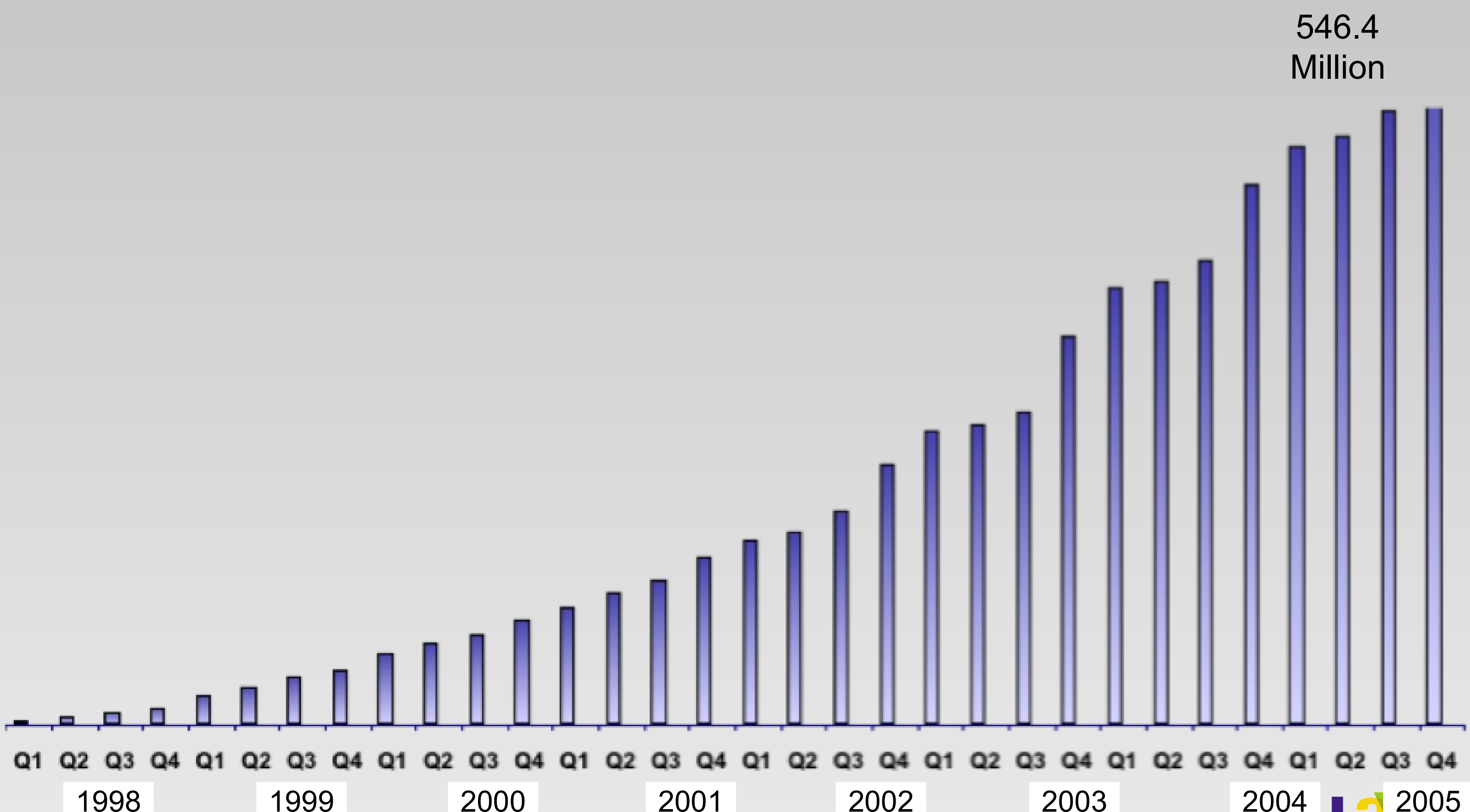
ever

y 10

minu

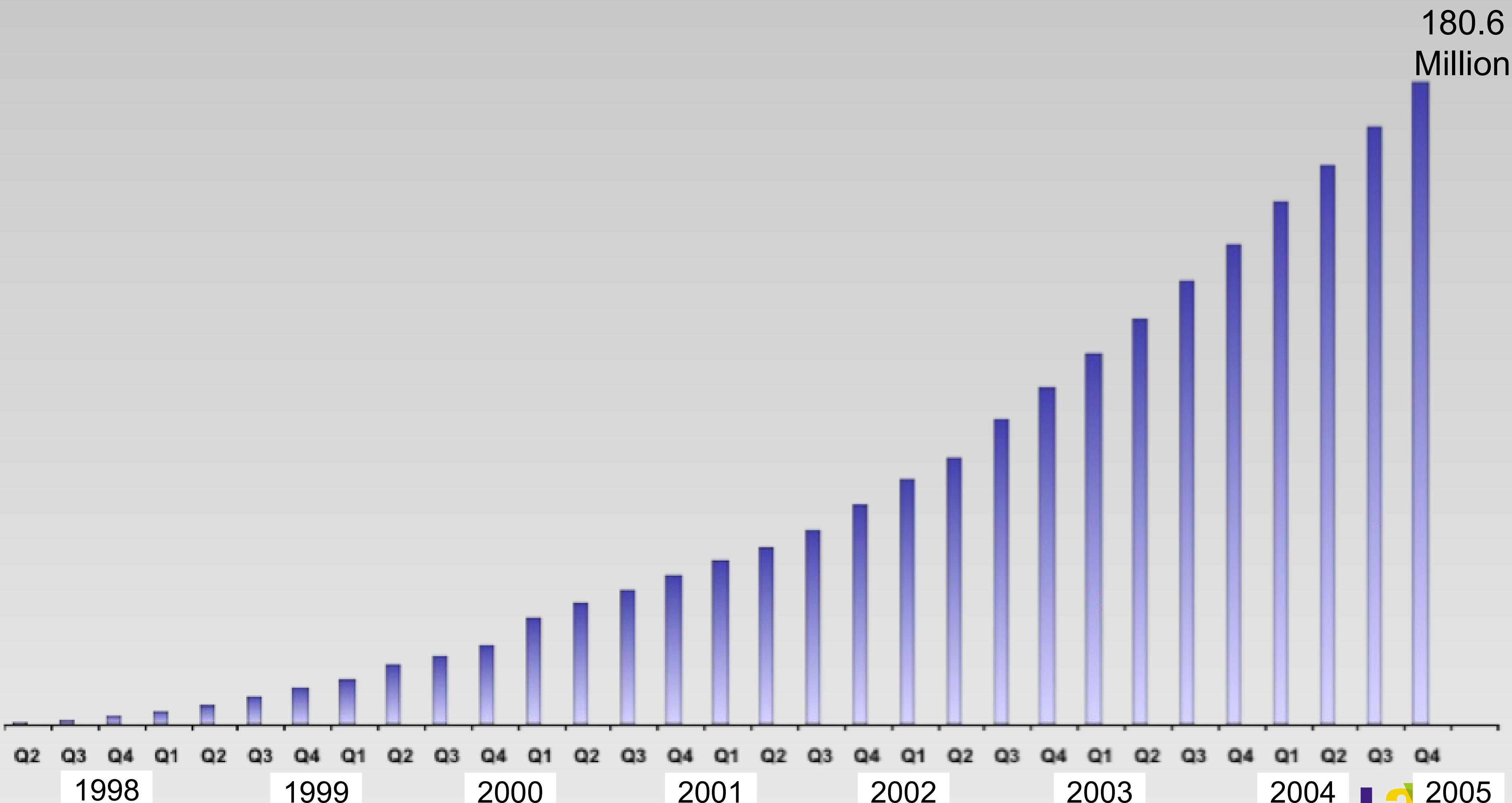
tes

Listings



546.4
Million
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
1998
1999
2000
2001
2002
2003
2004
2005

Registered Users



League of Long Tail

Of Needles, Haystacks, and Magnets…

- Transaction Data and session data
  - Terabytes per day
- Data mining researcher’s source of truth
- Nature of session data
  - Sessionized streams
  - Semi-structured
  - Constantly changing schema
- Examples:
  - View item click through rate for search algorithms.
  - Browsing pattern of users performing searches in different categories.
    - For e.g. computer vs clothing.
  - Purchase rate for various recommendation algorithms – A/B experimentation

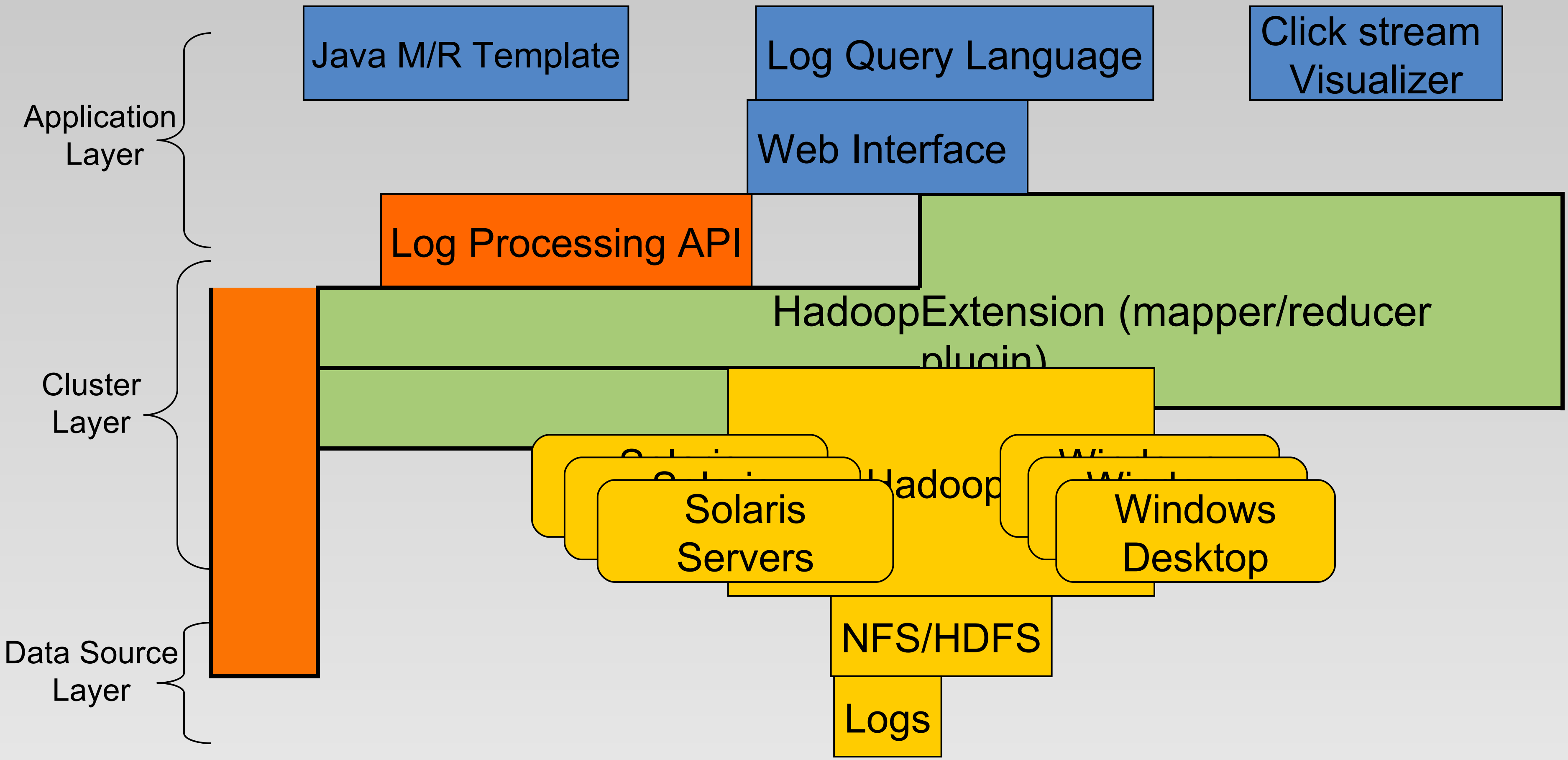
The Log Challenge

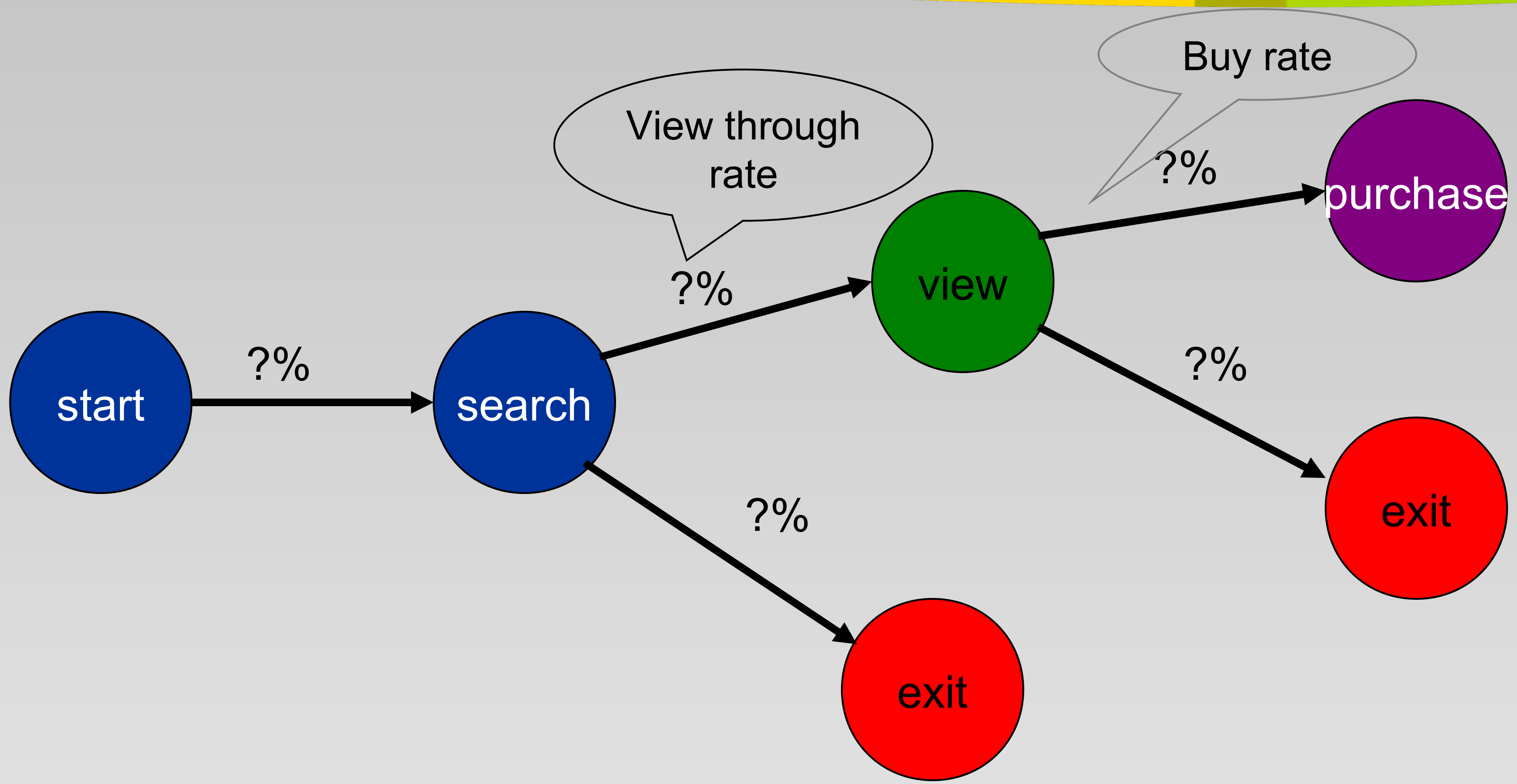
- The amount of the data is huge.
  - TB+ / day
  - Need to perform analysis on weeks or even years worth of data.
- Analysis takes time.
  - Making it distributed will help.
- Text streams are not indexed.
  - Difficult to query
  - Fields often change
- Difficult to perform sessionized analysis
  - E.g.: Study the session paths of session in which a given search algorithm is used.
  - E.g.: Differences in UK user and US users.
- Large Number of session paths (in millions)
  - Visualization is difficult
- Scaling up to potential large user base

What we need is..

- An analytic tool:
  - Easy to perform session analysis.
  - Large scale stream processing.
  - Quick turnaround on analysis.
  - Effective query language.
  - Visual analytics that caters to intuition and provides extensive analytics.
  - Highly customizable processing.
  - Provide interfaces at different levels.

Architecture - Mobius




Buy rate
View through
rate
?%
purchase
?%
view
?%
start
search
?%
exit
?%
exit

User session information

- Questions
  - What percentage of searches done receive clicks?
  - Out of those clicked results, how many are abandoned?
  - How many viewed results are followed to bid?
- Data
  - Session activity grouped together as a stream.
  - A session is a bag of events.
  - Each event is a tuple with various fields.
- Process:
  - Extract session with searches.
  - Compute view through rate.

What do we need in a Stream Query Language

- Detect patterns from the stream over a window
  - window
    - time-based, count-based, event/trigger-based
    - Sliding vs Landmark windows
  - Structure
    - Repetition (*, +), Sequence (//), condition
  - Naming patterns
- Integrate Relational and Pattern operators
- “Inner” Queries

Mobius Query Language (MQL) - Structure of a Query

Input stream → FROM → START

↓

END

↓

FILTER

↓

PATTERN

↓

SELECT

SELECT *
FROM source
WHERE where_condition
PATTERN pattern WITH condition
START start_condition
END end_condition

Simple Example

- Input: Stream of events.
- Event is modeled as tuples.
- Stream is modeled as ordered bag.

inputStream

(1,11:00AM)
(5,11:01AM)
(2,11:02AM)
(6,11:04AM)
(10,11:05AM)
(3,11:06AM)
(7,11:07AM)
(2,11:08AM)
(-1,11:10AM)
.
.
.

Simple Example Contd…

(1,11:00AM)
(5,11:01AM)
(2,11:02AM)
(6,11:04AM)
(10,11:05AM)
(3,11:06AM)
(7,11:07AM)
(2,11:08AM)
(-1,11:10AM)
.
.
.

→ FROM →

(1,11:00AM)
(5,11:01AM)
(2,11:02AM)
(6,11:04AM)
(10,11:05AM)
(3,11:06AM)
(7,11:07AM)
(2,11:08AM)
(-1,11:10AM)
.
.
.

→

(5,11:01AM)
(2,11:02AM)
(6,11:04AM)
(10,11:05AM)
(3,11:06AM)
(7,11:07AM)
(2,11:08AM)
(-1,11:10AM)
.
.
.

→

(5,11:01AM)
(2,11:02AM)
(6,11:04AM)
(10,11:05AM)
(3,11:06AM)
(7,11:07AM)
(2,11:08AM)
(-1,11:10AM)

Tuple in a bag

({(2,11:02AM)}, (6,11:04AM))
({(3,11:06AM)},(7,11:07AM))

```
SELECT size(A), B
FROM inputStream
WHERE $0 > 0
PATTERN ( ($0 < 5) + AS A[ ] //
                    $0 > 5 AS B )
WITH A[0].time  + 5 min > B.time
          && size(A) < 3
START $1 > 11:01AM
END $0 < 0
```

Filter

Pattern

Start-end Sub-stream

Output:

(1, (6,11:04AM))
(1, (7,11:07AM))

Other Systems

- PIG and Hive
  - Patterns are not available.
  - Other stream processing operator also not available. E.g. Start, End.
- CEP based Stream Processing Languages. (STREAM, Streambase, Cayuga)
  - Have flat data model.
    - Can only store few features of patterns.
  - Degree of parallelism is restricted to 1 due to inability to represent substreams.
    - In some cases splitting is done but that splitting operator has 1 degree of parallelism.
- Active Databases
  - ECA (events-conditions-actions) and triggers

Pattern Query

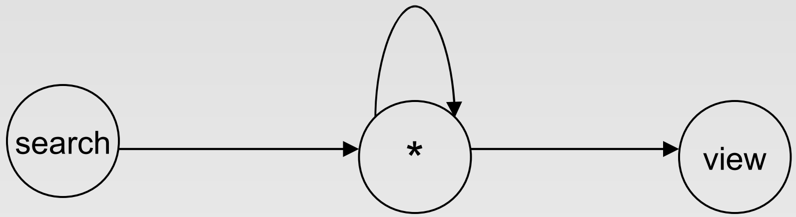
- Problem: For each user identify a set of “search” followed by “view” (click-thru) events
- Input Stream : (uid,session(name,time, itemID,…))*
- Output Stream : (uid,views(S,V)*)*

- DATASET clicks = SELECT uid, {

```
            SELECT S, V
            FROM session
            PATTERN (name == “search” AS S //
                     ** //
                     name == “view” AS V)
            WITH V.itemID belongsto S.impressions
        } AS views
    FROM logs
    WHERE size(session) > 1;
```

Pattern Query

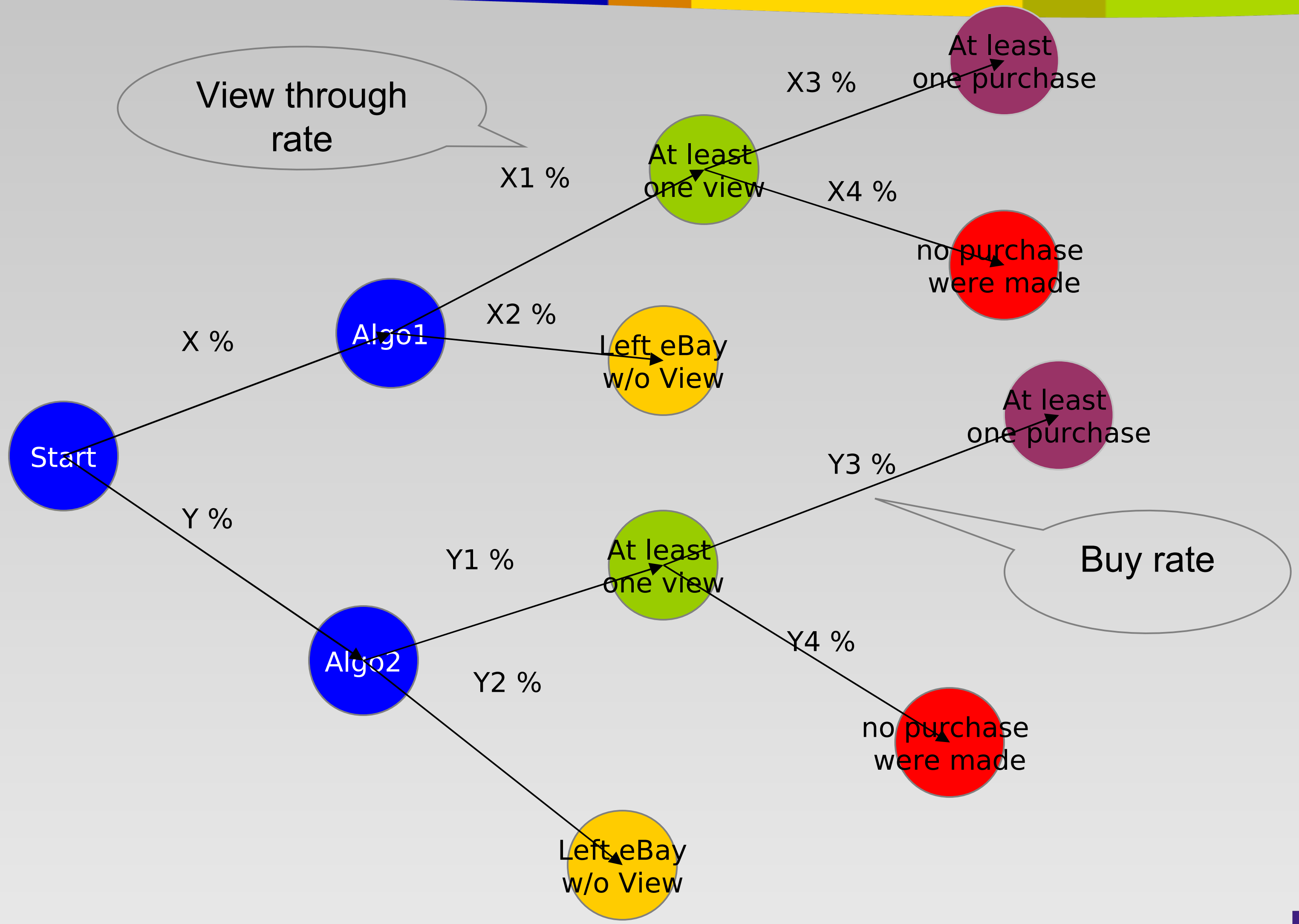
Pattern defined in the pattern query is



Recommender Systems

- Products are recommended to users on various pages.
  - How many clicks does the recommendation gets?
  - Do those clicks result in purchase?
  - Performance of different recommendation algorithms?
- Data
  - User session containing events.
  - Events are of different types
- Process
  - Extract session with recommendations.
  - Group them by algorithm used.
  - Calculate the view and purchase through rate.

Sessionized Analysis 1



Step1 : Extract all sessions with “view”s

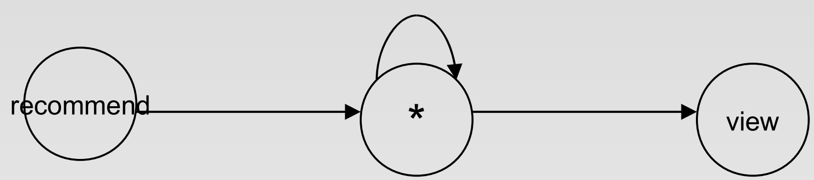
Bag named session

- Input Stream: merch_logs is (uid, sessionid, (name,time,itemID…)*)*
- DATASET merchView = SELECT uid, sessionid, {

```
SELECT S.algorithm AS algo, V.itemID AS itemID
FROM session
PATTERN ( name == “recommend” AS S//
          ** AS B[ ] //
          name == “view” AS V)
WITH (S.itemID == V.itemID) AND
     (B[i].itemid != S.itemID)
} AS viewedRecos
FROM merch_logs ;
```

A

- The PATTERN part defines pattern



- Schema of merchView is (uid, sessionid, (algo,itemID)*)*

Bag named viewedRecos

Step 2: Extract all instances of recommendations made…

```
• DATASET merchShown = SELECT uid, sessionid, flatten({
                            SELECT algorithm AS algo
                            FROM session
                            WHERE name == "recommend"
                              })
                  FROM merch_logs;
(uid, sessionid, (name,time,itemID…)*)*  => (uid,sessionId, algo)*
```

B

Research Labs

Step 3: Produce a flattened view

- DATASET unnestedmerchView = SELECT uid, sessionid, flatten(viewedRecos)

  FROM merchView;

(uid, sessionid, (algo,itemID)*)* => (uid, sessionid, algo, itemId)*

C

Step 4: Group the data by algorithm type and compute counts

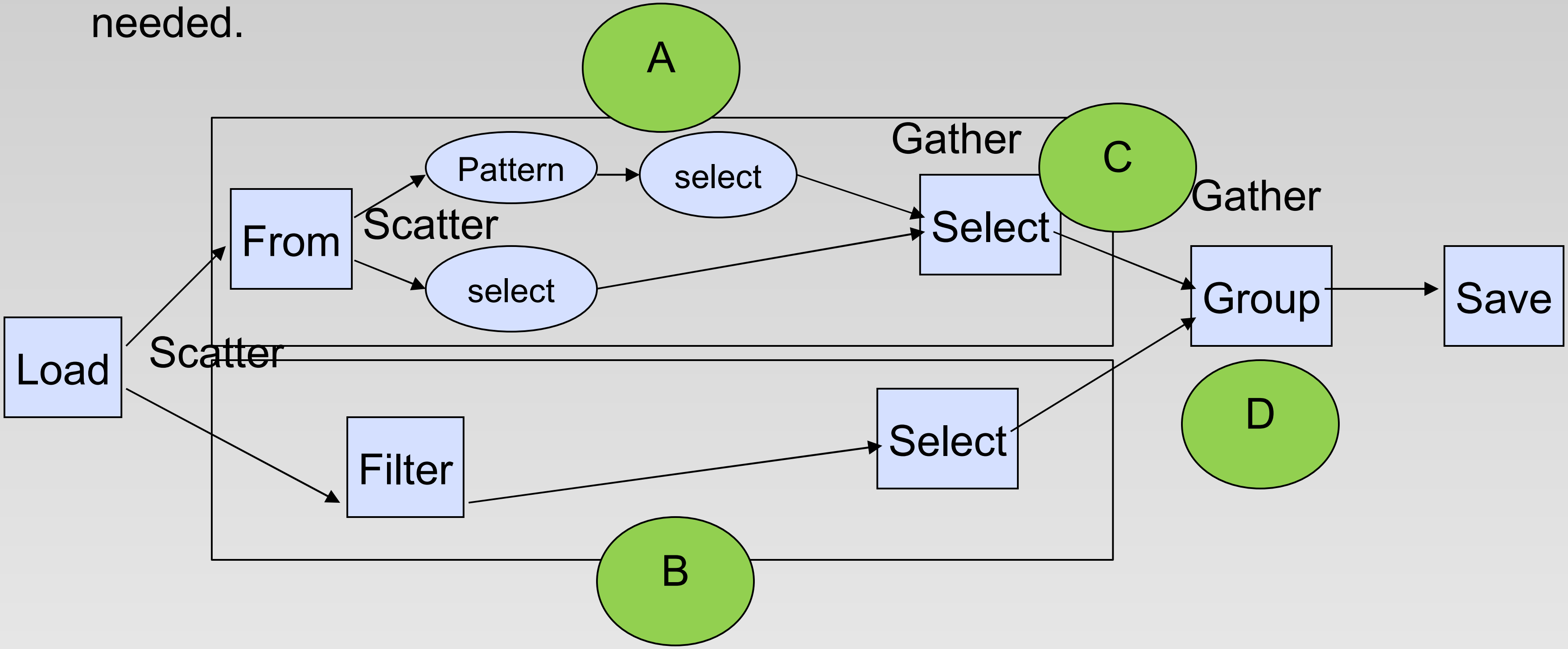
- DATASET merchData = SELECT groupid AS algo, size(unnestedmerchView) AS clicks,
  size(merchShown) AS impressions
  FROM unnestedmerchView, merchShown
  GROUP unnestedmerchView BY algo
  ALSO merchShown BY algo

D

(uid, sessionid, algo, itemId)* , (uid,sessionId, algo)* => (algo, #clicks, #impression)* -- stream/bag of length #algos

Parallel Implementation

- MQL compiling engine compiles queries in to a DAG. (similar to PIG)
- Then the DAG is compiled into one or more map-reduce jobs.
- When ever a grouping, sorting, union operator is seen a reduce phase is needed.

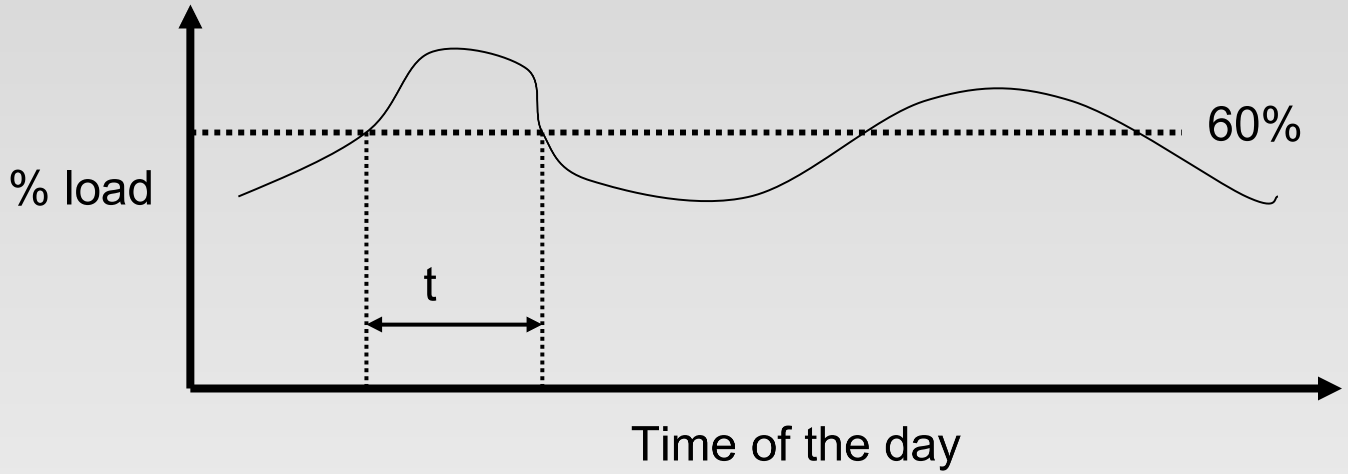


eBay Research Labs

Understanding System Health

- Data
  - System logs containing many beacon streams.
  - Each machine beacon stream comprising of single beacon message.
  - Message contains various state data.

- Problem
  - Contiguous time period load is more than 60%.
  - Time period only interesting if it is more than delta mins.

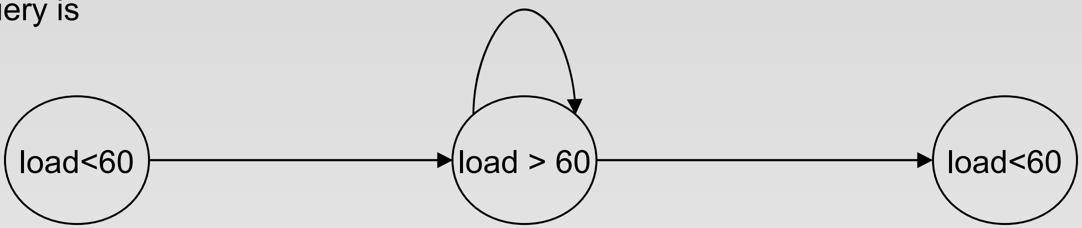


eBay Research Labs

- Input Stream: Schema of systemlogs is (systemname, beacons(load, time, …))
- Output Stream: (systemname, (load,time)*)*
- DATASET system = SELECT systemname, {

```
SELECT LE AS load
FROM beacons
PATTERN ( load < 60  AS SEVENT //
         (load > 60)+  AS LE[ ] //
          load < 60 AS EEVENT)
WITH LE[size(LE)].time – LE[0].time > 10mins
      } AS LoadTimes
FROM systemlogs;
```

- Pattern defined in pattern query is



Ongoing and Future Work

- Optimization mechanism.
  - Filter cannot be pushed ahead of user defined functions and pattern queries.
  - Inferring projection is also limited.
- Compilation to Map-Reduce Jobs
  - Inferring the best strategy to split work between map-reduce in case of multiple queries.
- Degree of parallelization when stream is not split by the users into substreams.
- Near real-time stream processing engine.
- Reporting mechanism.